ANÀLISIS, DISSENY I CONSTRUCCIÓ D'UN ROBOT EDUCATIU BASAT EN MICROCONTROLADOR AVR-ATMEGA

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Sergi Hernández Arroyuelo
i dirigit per
Joan Oliver Malagelada
Bellaterra, 18 de setembre de 2009

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Joan Oliver Malagelada

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, 18 de setembre de 2009

ALQUIST: Ha estat un crim fabricar robots.
DOMIN: Com?
ALQUIST: Que ha estat un crim fabricar robots.
DOMIN: No, Alquist. No em penedeixo d'això ni tan sols avui.
ALQUIST: Ni tan sols avui?
DOMIN: Ni tan sols avui, l'últim dia de la civilització. Va ser una gran aventura.

KAREL CAPEK, a *R.U.R. (Robots Universals Rossum)*

Agraïments

Voldria donar les gràcies als meus pares per l'esforç d'haver-me donat una bona educació.

Al meu germà per aguantar-me durant el dia a dia i a la Mariona per recolzar-me contínuament, i per la seva paciència durant les absències.

A Joan Oliver per guiar-me i ajudar-me durant l'elaboració del projecte, amb la seva disponibilitat i oferiment constant.

I a tots aquells que, abans de mi, van trobar-se amb reptes semblants i no van dubtar a oferir els seus coneixements lliurement.

ÍNDEX

1 INTRODUCCIÓ

Si el dramaturg Karel Capek aixequés el cap es trobaria que aquella paraula que va popularitzar a la seva obra *R.U.R* (*Robots Universals Rossum)*[1], s'ha estès a la vida contemporània fins a convertir-se en un mot força quotidià. El que va imaginar el famós escriptor txec, un futur on els robots substituirien els homes en les feines més perilloses i mecàniques, és cada cop més real, tot i que difereixi sensiblement del plantejament catastròfic de l'obra.

La robòtica mòbil, tal i com l'entenem avui en dia, té els seus inicis a la dècada dels 60 i des d'aleshores el seu creixement ha augmentat constantment per les seves aplicacions industrials i pel interès que el seu objectiu suscita: realitzar tasques com les fan els essers humans. La possibilitat de dur a terme feines que poden ser difícils per a l'home o fins i tot perilloses ha fet evolucionar notablement aquesta ciència. Dins d'aquest camp, la navegació mòbil i autònoma ha tingut molt a dir, per la possibilitat d'arribar a allà on no es pot arribar, d'aconseguir una precisió incapaç per part de l'ésser humà i per la millora contínua de la intel·ligència dels robots. La implantació de sensors ha estat essencial en aquest sentit, doncs atorguen al robot la informació fonamental per a entendre el seu entorn, prendre decisions i actuar.

Si la robòtica mòbil és interessant pel que fa a aplicacions industrials, també ho és pel que fa a aplicacions educatives, doncs conviu amb la majoria dels camps de l'enginyeria informàtica. És per això que ens permet oferir a l'alumne un exercici atractiu i interessant, amb el qual pot aplicar els coneixements en tècniques de programació i fonaments de computació, i observar les seves conseqüències d'una forma molt experimental. De fet, l'aplicació de robots mòbils a pràctiques educatives no és una idea nova. Al mercat és fàcil trobar ofertes de robots completament pensats per aplicacions educatives a nivell principiant. No obstant, no és tan senzill trobar solucions que

[1] L'obra de teatre *R.U.R.* descriu una societat deshumanitzada per la proliferació del maquinisme en la vida quotidiana. Estrenada el 1921, és coneguda per contenir la primera aparició del terme "robot".

permetin introduir a l'alumne en nivells més exigents. Em plantejo donar resposta a aquesta necessitat i proporcionar una alternativa adaptada als requeriments propis, amb una informació acurada referent tant a la implementació com als costos del projecte, que permeti portar-lo a terme amb garanties d'èxit.

1.1 OBJECTIUS

L'objectiu principal del meu projecte és la construcció d'un robot educatiu adequat per a la formació i aprenentatge tant en assignatures de hardware com de software, basat en el microcontrolador Atmega 128 d'ATMEL. El robot haurà de ser intel·ligent i autònom, disposant d'una sèrie de sensors que possibilitin un ventall prou ampli per a l'experimentació de futurs alumnes en assembler i llenguatge C.

Per portar a terme aquesta idea global de la manera més òptima, cal tenir en compte l'elecció d'una arquitectura adient per tal de proporcionar les funcionalitats proposades i establir una primera idea d'aquestes funcionalitats amb la intenció de millorar-les a mesura que s'avança i els objectius inicials van sent completats. Aquesta primera idea proposa, bàsicament, un robot capaç de detectar i evitar obstacles. Cal dotar-lo de la informació necessària per establir un contacte amb l'exterior i poder controlar certs paràmetres que seran interessants per possibilitar al màxim l'experimentació dels alumnes. És necessari, per tant, realitzar la programació dels drivers de control que atorguen al robot de l'autonomia i relació amb l'entorn abans esmentada, i la integració d'aquests en un sistema operatiu que permeti gestionar-los eficientment.

El robot proposat servirà per establir les bases d'un projecte educatiu per a volums mitjans, en el qual es tractarà de reduir costos i simplificar funcionalitats per tal de que sigui el més viable possible. S'oferirà una descripció detallada dels costos.

Com a objectiu personal, cal que esmenti el fet d'aprofundir en el camp de la robòtica mòbil, així com ampliar els meus coneixements pel que fa a la programació en llenguatge C i assembler, concretament la programació de microcontroladors, i en la comprensió de l'arquitectura i funcionament de l'Atmega 128.

1.2 PLANIFICACIÓ

La planificació del projecte ha estat portada a terme segons l'ordre que em vaig plantejar inicialment i seguint l'orientació donada pel meu director. A la figura 1 es pot veure el desenvolupament durant l'any acadèmic 2008-2009.

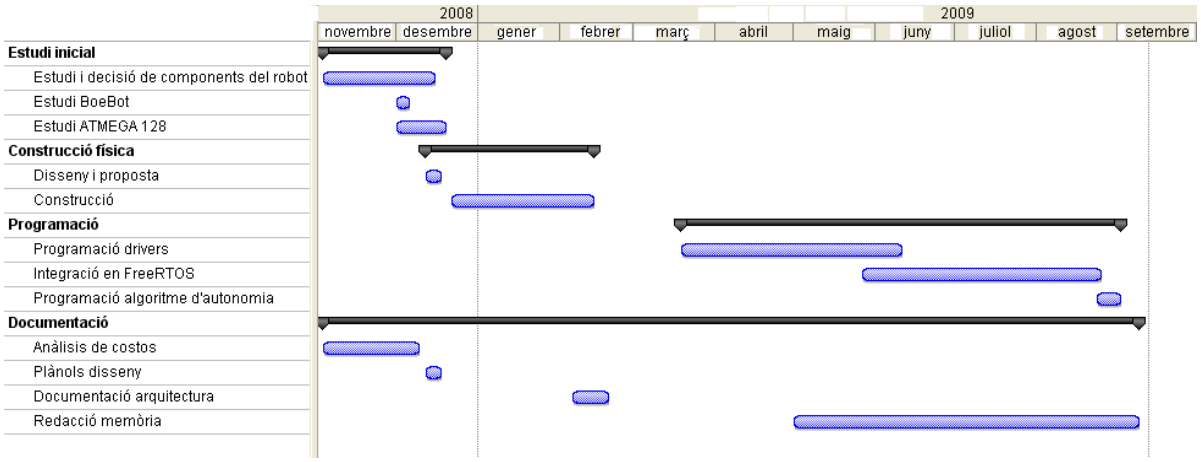


Figura 1

Vaig començar per fer un primer estudi dels components que havia de portar el robot. Va ser una etapa d'anàlisi en la qual vaig anar documentant-me per acabar fent una tria. Després vaig estudiar les alternatives que trobava al mercat, especialment el Boe-Bot de Parallax, i vaig començar a estudiar el microcontrolador sobre el qual es basa el projecte: l'Atmega 128 d'Atmel. Aquest primer estudi va comportar el primer mes de projecte. A continuació vaig realitzar el disseny de

l'arquitectura sobre la qual treballaria. Un cop rebuts els components i sensors necessaris, vaig poder iniciar la construcció física del robot.

A mitjans de març de 2009, ja amb el robot construït, vaig iniciar la programació dels drivers. Aquesta tasca em va portar fins a juny del 2009, moment en el qual em vaig plantejar allargar el projecte fins al setembre per tal de poder integrar el sistema operatiu FreeRTOS. Durant l'estiu vaig dedicar-me a compaginar la integració de FreeRTOS amb l'elaboració de la memòria, que ja havia començat temps abans. Finalment, vaig implementar un algoritme de navegació autònoma de prova.

2 ANÀLISI DE REQUERIMENTS

Començaré per fer un anàlisi dels requeriments del projecte. Explicaré aquelles necessitats que preveig abans de començar i que m'han de servir per definir com vull que sigui el robot. Diferenciaré tres tipus de requeriments: educatius, tècnics i físics.

2.1 REQUERIMENTS EDUCATIUS

M'he plantejat l'elaboració d'aquest robot com un projecte de caire educatiu. És la seva raó de ser i, per tant, hi ha certs aspectes a tenir molt en compte.

- *Eina educativa*: el robot, com a eina educativa, ha d'oferir quantes més possibilitats d'aprenentatge millor i versatilitat per adaptar-se a pràctiques diferents. En aquest sentit cal que em plantegi utilitzar una arquitectura que suporti llenguatges d'ús comú en l'aprenentatge de les enginyeries d'informàtica i telecomunicacions, i evitar aquelles que ofereixen moltes facilitats però només permeten experimentar amb un llenguatge propi, de poc interès per aquell que ha d'aprendre uns fonaments, com és el cas dels robots de Parallax. Per altra banda és important oferir una eina entenedora i lliure de conceptes complexes pels quals l'alumne no ha de perdre temps, podent dedicar-se directament a allò que li interessa.

- *Extensibilitat*: un projecte educatiu de caire tecnològic sempre ha d'estar obert a extensions que el puguin fer millor. És molt positiu donar lloc a possibles ampliacions que allarguin la vida del projecte com a eina educativa. De fet, la millora de la proposta que presento podria ser entesa com la darrera funcionalitat educativa d'aquest projecte.

- *Viabilitat econòmica*: tot projecte requereix una viabilitat econòmica. Tractant-se d'un projecte educatiu és convenient oferir un anàlisi de costos que permeti estudiar si és viable o no. En el mercat de la robòtica, i amb les possibilitats globals que ofereix Internet, és fàcil trobar diferències de preus per un producte similar i, de vegades, per al mateix producte. Per tant, forma part del projecte tractar d'adequar-se a les necessitats sense deixar de tenir en compte els costos del material, cercant sempre la solució més econòmica possible.

2.2 REQUERIMENTS TÈCNICS

La meva proposta consisteix en un robot mòbil amb capacitat per evitar obstacles. Per aquest propòsit cal que em plantegi tres requeriments.

- *Locomoció*: s'ha de dotar al robot d'un sistema de locomoció que li permeti moure's, amb la possibilitat de variar la velocitat i canviar de sentit i direcció.

- *Percepció*: per tal de que interactuï amb l'exterior necessitarà un sistema de percepció format per diferents dispositius.

- *Processament*: la informació rebuda a de ser processada amb certa capacitat de càlcul com per permetre la programació de comportaments intel·ligents. Dins d'aquest requeriment s'inclou la implementació d'un sistema operatiu que permeti gestionar el comportament del sistema.

2.3 REQUERIMENTS FÍSICS

Per últim, exposo dos requeriments físics que caldrà assolir de la manera més ideal possible:

- *Mida reduïda*: quan més compacte i lleuger sigui el robot, millor. Per tant, durant la tria de components i en el plantejament de l'arquitectura, tindré en compte aquest tret per intentar dissenyar un robot el més petit possible.

- *Facilitat de reproducció*: com a projecte amb possibilitats de ser replicat en un nombre mitjà de còpies, facilitaré la seva construcció. En aquest sentit incidiré en una arquitectura senzilla d'elaborar i amb facilitats per ser replicada.

3 ARQUITECTURA DEL ROBOT

L'arquitectura proposada parteix dels requeriments i de les característiques bàsiques del robot. A la arquitectura del robot s'han d'encabir totes les parts pensades inicialment de tal manera que sigui fàcil de construir i de reproduir. Alhora, convé un disseny suficientment modelable, que permeti afegir noves propostes i ampliar el projecte inicial. Per això he decidit construir un xassís de fusta compost bàsicament per tres làmines de diferents mides. He triat fusta de marqueteria de 4 mm, molt fàcil de tallar. Una plantilla amb les mides exactes de les làmines ens permet construir diverses còpies fàcilment amb materials barats com, per exemple, el PVC.

Com es pot observar a la figura 2, el disseny físic del robot queda dividit en tres grans mòduls, tal i com em vaig plantejar originalment: mòdul de locomoció i alimentació, mòdul d'unitat de procés i mòdul de sensors.

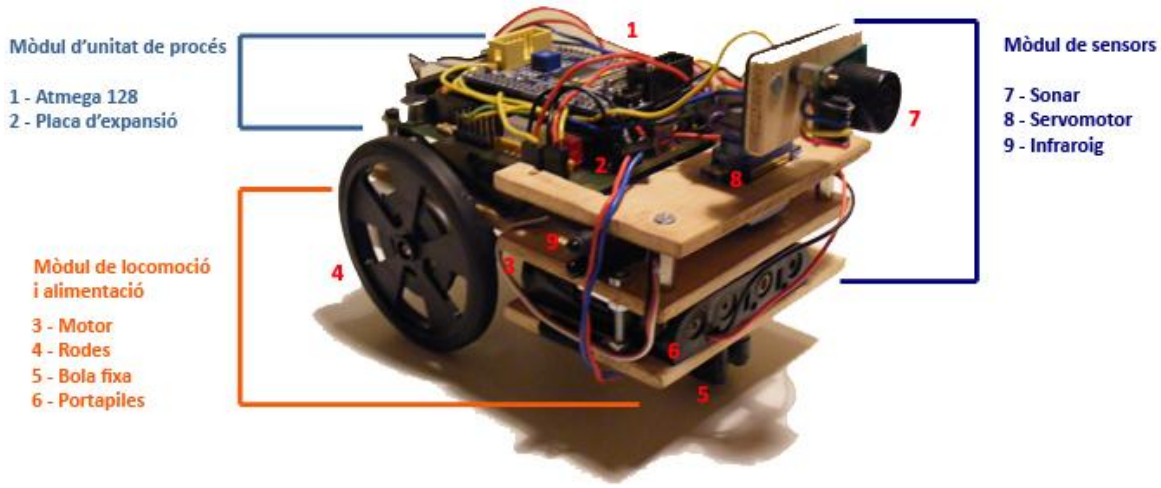


Figura 2

3.1 MÒDUL DE LOCOMOCIÓ I ALIMENTACIÓ

Aquest mòdul està compost per les parts de locomoció i alimentació. La seva col·locació en l'esquelet del robot queda a la part més inferior.

La locomoció del robot està formada per dos motors de corrent continu de 5 V amb les respectives rodes que permeten la propulsió del robot. Estan col·locats sobre un eix imaginari i collats al xassís. Les possibilitats de rotació queden definides per les velocitats aplicades a cadascuna de les rodes, així com el sentit. Podrem realitzar girs de radi ample si apliquem el mateix sentit però diferent velocitat, i girs de radi petit si apliquem, per exemple, la mateixa velocitat però en sentit contrari. El sistema de locomoció queda recolzat per una bola col·locada per davant de les rodes, centrada, que permet estabilitzar el robot i definir la direcció.

L'alimentació està formada per un conjunt de quatre piles de 1,5 V, el que permet tenir un total de 6 V. Cal, per tant, un regulador de tensió per que el microcontrolador funciona a 5 V. Les piles queden allotjades a un portapiles collat al xassís, protegit de l'exterior, i a on s'ha deixat un espai suficient per afegir una pila més mitjançant un mòdul acoblable.

3.2 MÒDUL D'UNITAT DE PROCÉS

El mòdul d'unitat de procés és l'encarregat de suportar el microcontrolador i permetre la seva comunicació tant amb el mòdul de locomoció i alimentació, com amb el mòdul de sensor. Es troba situat per sobre de la fusta que exerceix com a base principal del xassís i està format per la placa del Atmega 128 i per una placa d'expansió de construcció pròpia que permet l'ús correcte del microcontrolador. A la figura 3 observem l'esquemàtic de la versió final de la placa.

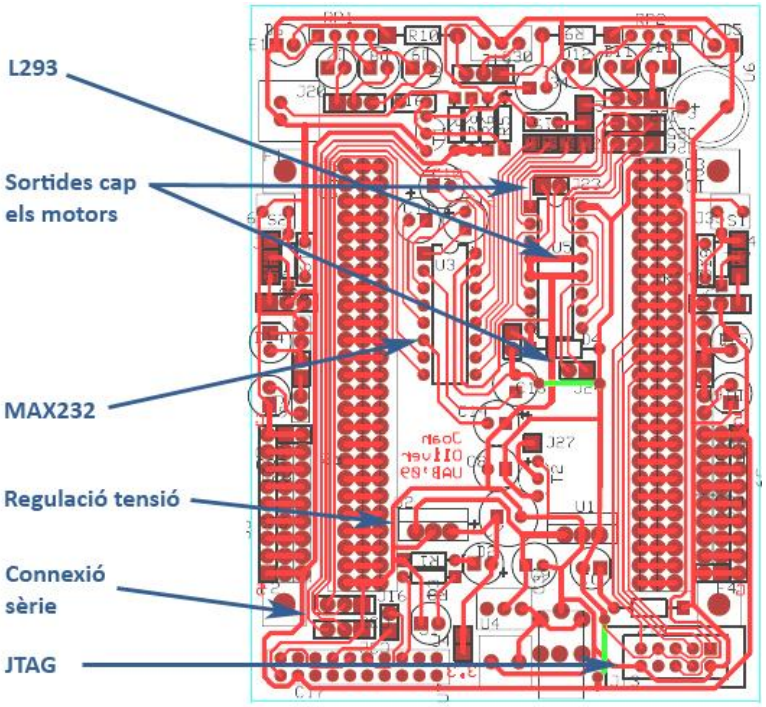


Figura 3

Podem diferenciar com disposa de la comunicació RS-232 mitjançant el circuit integrat MAX232. Remarco també el regulador de tensió, el circuit integrat L293 per al control dels motors i l' entrada JTAG molt útil per la descàrrega del programa des del PC.

3.3 MÒDUL DE SENSORS

El mòdul de sensors és l'encarregat d'aconseguir informació de l'exterior i oferir-la al robot per a que pugui treballar amb ella. Inicialment vaig plantejar aquest mòdul com un tercer pis en el que quedarien distribuïts els diferents sensors. Però a la pràctica ha resultat molt més interessant col·locar-los a la part davantera, a una alçada semblant a la del mòdul de control. Els sensor de que disposa són: un sonar i dos infrarojos.

El sonar és el sensor que ha de donar informació al robot del que tingui davant, per això queda enfocat cap a la part davantera. Queda ubicat en una petita torreta que consta d'un servomotor que permet fer escombrades. El sonar va collat a una petita base col·locada sobre el servomotor.

Els infrarojos estan situats als laterals de la part davantera. Com que tenen un abast de detecció molt més petit que el del sonar actuen únicament per evitar que el robot topi amb obstacles situats als seus laterals.

4 CARACTERITZACIÓ DELS COMPONENTS

4.1 MOTORS

Una de les parts més importants pel que fa al funcionament efectiu del robot és la locomoció, la qual possibilitarem gràcies a dos motors de corrent continu. De les possibles solucions alhora de triar la manera de proporcionar la locomoció, em vaig plantejar dues alternatives: motors i servomotors. Tot i que els darrers faciliten el seu control i acostumen a oferir un torsió més elevat, em vaig decidir pels motors amb la intenció d'aconseguir un ventall de velocitats més ampli i evitar haver de modificar els servos per a la rotació contínua, doncs en la majoria dels casos estan preparats per rotar únicament 180°.

Un motor de corrent continu és una eina que transforma l'energia elèctrica en energia mecànica amb moviment rotatori. El seu principi de funcionament consisteix, bàsicament, en fer passar un corrent elèctric per un conductor que es troba dins de l'acció d'un camp magnètic, de tal manera que dit conductor es mourà en sentit perpendicular a l'acció del camp magnètic.

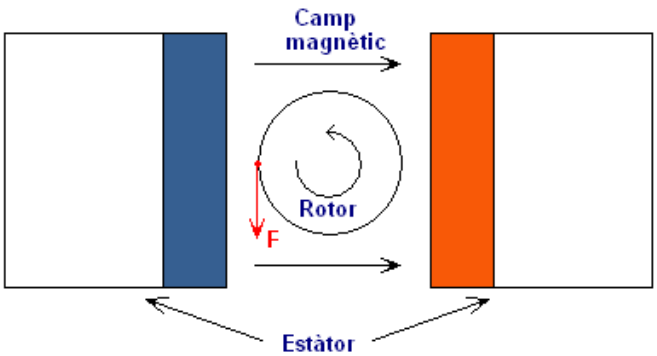


Figura 4

4.1.1 GM8

El motor triat és el GM8 de Solarbotics, un petit motor que ofereix bones prestacions a un preu molt econòmic. M'interessa la seva mida molt reduïda, així com el seu poc pes, que contribuirà a fer el robot més lleuger i compacte. A més, el fet de que Solarbotics ofereix rodes i altres complements que encaixen a la perfecció amb el motor, i que també es comercialitzen a preus baixos, em facilita la feina alhora de configurar tot el mòdul de locomoció.

Solarbotics GM8

Voltatge: 5 V

Torsió: 3.096 kg-cm

Velocitat: 70 rpm

Pes: 37 grams

Dimensions: 55x48x23 mm

Fabricant: Solarbotics

Preu: 3,73€

4.1.2 ARQUITECTURA I CONNEXIÓ

4.1.2.1 EL PONT H

Per poder controlar els dos motors de manera útil, podent establir la rotació en un sentit i en un altre, és necessari l'ús d'un pont H. Un pont H és un circuit electrònic que ens permet controlar un motor en ambdós sentits. Consisteix en fer servir quatre interruptors que, en funció de quins estiguin oberts i quins tancats, farà girar el motor en un sentit o en un altre, parar-lo sobtadament o deixar-lo frenar per pròpia inèrcia. Com podem veure a la figura 5, l'activació dels interruptors S1 i S4, juntament amb la desactivació dels S2 i S3, generaria un moviment circular en un sentit invers al que aconseguiríem amb la combinació oposada.

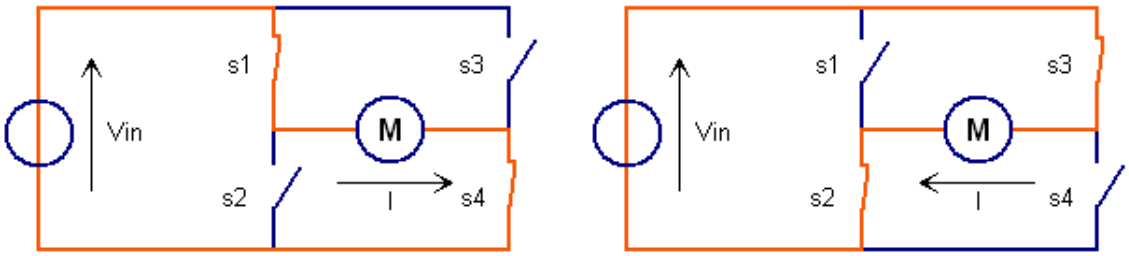


Figura 5

La taula de veritat d'un pont H queda definida de la següent manera:

S1	S2	S3	S4	Resultat
1	0	0	1	Gir en sentit d'avanç
0	1	1	0	Gir en sentit de retrocés
0	0	0	0	Detenció sota pròpia inèrcia
0	1	0	1	Detenció per fre

4.1.2.1 EL L293

A l'hora de construir el pont H podem fer-ho manualment o bé fent-nos servir d'un dels circuits integrats que hi ha al mercat i que ens redueixen la tasca de construcció, facilitant-nos molt la feina. En el nostre cas farem servir el circuit integrat L293, que ens permet controlar dos motors mitjançant un pont H per cadascun d'ells. Així, amb un sol circuit integrat podem fer ús de tots dos motors d'una manera molt senzilla.

Per explicar el funcionament d'aquest circuit cal aclarir que hi ha dues parts ben diferenciades i que a la figura 6 queden delimitades per una línia vermella discontínua. Les dues parts son simètriques i cadascuna governarà un motor de manera independent.

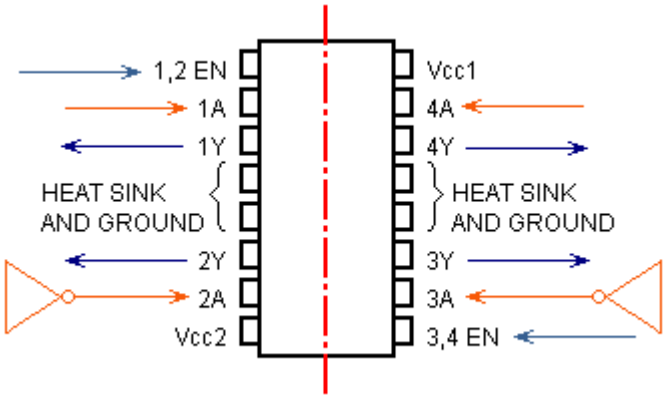


Figura 6

Si ens fixem en la part esquerra del xip, tenim les senyals 1A i 2A per indicar la direcció de gir del motor. Quan 1A sigui 0 el motor girarà en un sentit, quan sigui 1 en el contrari. La patilla 2A ha de rebre el senyal contrari a 1A, per això és col·loca un inversor a l'entrada, evitant que el circuit arribi a fer curtcircuit. Les patilles blau fosc son les sortides cap als motors i aniran directament connectades als borns de cada motor. Les patilles 1,2EN i 3,4EN són entrades d'habilitació del xip i simplement faran funcionar el motor corresponent, de la manera descrita fins ara si estan a 1, i pararan el motor

en el cas d'estar a 0. Queden 6 patilles: 2 de Vcc i 4 de Gnd. Vcc1 és l'alimentació de l'electrònica i va connectada als 5 V, mentre que Vcc2 és l'alimentació dels motors i, tot i que en el meu cas també és de 5 V, en altres casos podria ser un voltatge superior. Les 4 patilles de Gnd aniran connectades entre elles.

4.1.3 DRIVER

El control dels motors mitjançant software es fonamenta bàsicament en l'ús correcte del L293, que és força senzill: només em caldrà activar en el moment adequat els bits de direcció per aconseguir que el motor giri en un sentit o en un altre. Amb això tinc solucionat el control elemental del motor, però no el de la seva velocitat. M'interessa poder establir la velocitat que prengui el robot en determinades situacions, i per això faré servir la modulació per amplada de pols, més coneguda com PWM (Pulse-Width Modulation). Aquesta tècnica consisteix en fer variar el cicle de treball d'una senyal periòdica, mantenint la mateixa freqüència. Així podrem controlar la quantitat d'energia que enviem i, per tant, la velocitat del motor.

Atmega 128 ens ofereix una sèrie de registres que ens faciliten l'ús del PWM. Diferenciarem bàsicament tres tipus de registre: TCNTx, ICRx, i OCRxA/B/.

TCNTx: és el comptador del timer, es va incrementant fins al valor límit (ICRx), moment en el qual començarà a decréixer fins a 0, tornant a començar periòdicament.

ICRx: és un registre de comparació i serveix per establir en quin punt el comptador (TCNTx) començarà a decréixer.

OCRxA/B/C: és un registre de comparació, que definirà el període PWM en els punts en els que es creui amb el valor de TCNTx.

OCxA/B/C: no és un registre, sinó el pin de sortida en el qual s'obté el senyal PWM. És en aquest pin en concret on haurem de connectar el senyal d'enable del motor. La freqüència de sortida quedarà determinada per la següent fórmula:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

On $f_{clk\_I/O}$ és la freqüència del rellotge d'entrada i sortida, i N és el valor del prescalar. Aquests valors són configurables i haurem de triar-los en funció de certs requisits.

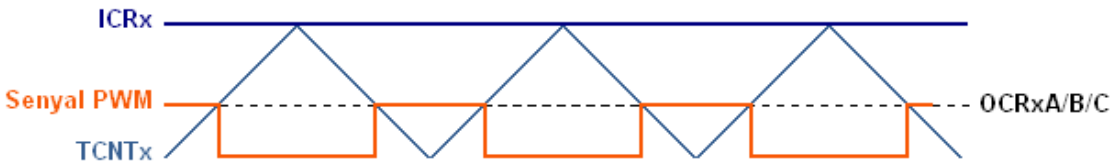


Figura 7

Com podem observar a la figura 7, TCNTx inicia una compta incremental fins arribar a ICRx, moment en el qual començarà a decréixer. Durant aquest cicle es creuarà en dues ocasions amb el valor constant de OCRxA/B/C.

- Quan TCNTx = OCRxA/B/C mentre el comptador estigui en pujada: la sortida OCxA/B/C es posarà a 1.

- Quan TCNTx = OCRxA/B/C mentre el comptador estigui en baixada: la sortida OCxA/B/C es posarà a 0.

D'aquesta manera definim el senyal PWM. Observem que si, per exemple, triem un valor de OCRxA/B/C molt més alt, el cicle de treball serà molt més ample i, en el nostre cas, aconseguiríem una velocitat superior. A la figura 8 podem veure com varia el voltatge final en funció del cicle de treball mantenint la mateixa freqüència.

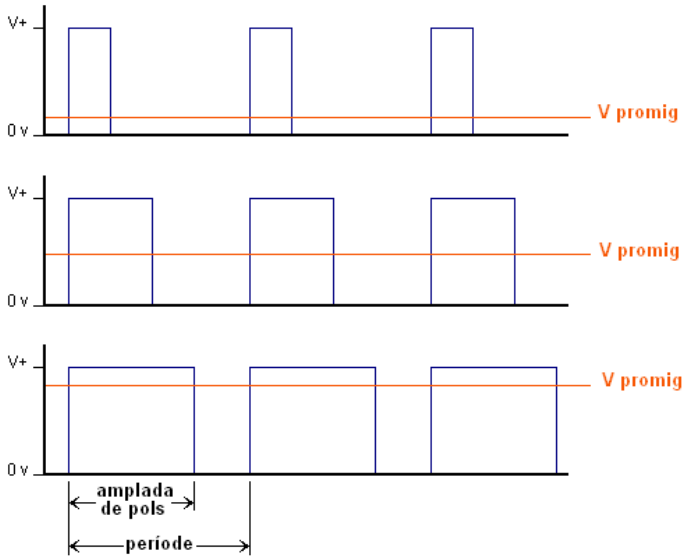


Figura 8

4.1.3.1 FUNCIONS DE CONTROL DEL MOTOR

```
void motorConfig(void);
```

La funció d'inicialització dels motors defineix com a sortides els pins que utilitzarem. Per cada motor, un pin d'enable i dos per establir la direcció. El pin d'enable es posa a 1 per tal de que els motors quedin preparats per funcionar.

```
void Stop(); void motorRStop(void);  void motorLStop(void);
```

Les funcions d'aturada actuen posant a 0 els bits de direcció.

```
void Endavant(int velocitat); void motorEndavantR(int velocitat);
void motorEndavantL(int velocitat);
```

Les funcions de marxa endavant activen els bits de direcció de tal manera que els motors girin endavant (A a 0 i B a 1). El paràmetre *velocitat* especifica la velocitat amb la que volem que girin els motors i estarà comprès entre els valors 47 i 255. Serà el valor assignat al registre OCR1x.

```
void Enrera(int velocitat); void motorEnreraR(int velocitat);
void motorEnreraL(int velocitat);
```

Funcionen de la mateixa manera que les anteriors, però activant els bits de direcció de manera inversa (A a 1 i B a 0).

4.2 TORRETA MÒBIL AMB SONAR

El ultrasò és una ona acústica amb una freqüència per sobre del límit perceptible per l'ésser humà. El funcionament del sonar consisteix bàsicament en emetre una ràfega d'ultrasons que en trobar-se amb algun objecte rebotarà i serà rebuda pel mateix sonar. El temps transcorregut entre l'emissió i la recepció permetrà al dispositiu calcular-ne la distància. Em servirà per captar la presència d'obstacles davant del robot i rebre informació respecte la distància a la qual es troba l'obstacle. Serà per tant un element fonamental pel que fa a la capacitat de navegació del robot, detectant la presència d'obstacles i evitant-los exitosament.

Per millorar la capacitat de percepció, el sonar funcionarà conjuntament amb un servomotor que permetrà moure'l en un radi aproximat de 180 graus. Un servomotor és un dispositiu format per un motor de corrent continu, una reducció i un circuit de control que permet que el motor s'aturi en un punt desitjat, quedant bloquejat en aquella posició. En el meu cas, l'ús del servomotor esdevé molt interessant per tal de donar al sonar capacitat per enfocar en diferents direccions o realitzar

escombrades, aturant-se per realitzar la medició. D'aquesta manera dotaré el robot d'una "visibilitat" molt més amplia que la que tindria amb un sonar fix.

4.2.1 EL SERVOMOTOR

4.2.1.1 MINISERVO DY-S0206

El servomotor triat destaca per la seva mida reduïda i el seu poc pes, característiques molt interessants pel que fa a la compactació del robot. Com està pensat per fer moure el sonar, la seva mida reduïda em facilita encabir-lo al xassís evitant els problemes d'espai que em podria donar un servomotor de mida més gran. El recorregut del servomotor és, com en la majoria dels casos, de 180 graus, tret que ja em resulta convenient tenint en compte que la meva intenció es fer-lo moure sempre en un rang de 0 a 180 graus. Potser no ofereix la mateixa força de torsió que altres servomotors, però en el meu cas no cal que sigui excessiva. També és interessant el seu preu, bastant econòmic en comparació amb altres productes de característiques semblants.

Miniservo DY-S0206

Voltatge: 4.8 - 7,2 V

Força de torsió: 1,5 kg-cm

Velocitat: 100 rpm

Pes: 9 grams

Dimensions: 23x12,5x30 mm

Fabricant: Dong-Yang

Preu: 8,45€

4.2.1.2 DRIVER DEL SERVOMOTOR

El control d'un servomotor ve plenament associat amb el concepte de PWM que ja he comentat anteriorment per als motors. En aquest cas, en comptes de poder modificar la velocitat amb la que gira un motor, el PWM em permet especificar al servomotor en quin punt vull que s'aturi.

La freqüència del senyal PWM del circuit intern d'un servo és de 50 Hz, o el que és el mateix de 1/50 = 20 mseg. Com podem apreciar a la figura 9, la durada del cicle de treball ha de ser d'1 mseg per a que el servomotor es posicioni a l'esquerra, 1,5 mseg per a que es posicioni al centre i 2 mseg per a que ho faci a la dreta.

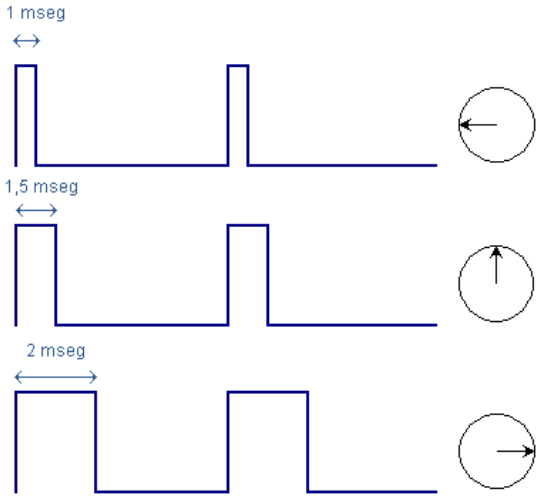


Figura 9

Per tant, cal que trobi el valor que he de fer servir als registres de PWM (TCNTx, ICRx i OCRxA/B/C) per que el funcionament sigui el correcte. Utilitzaré la funció:

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

Obtinc que per a 20 mseg el valor (tenint en compte que fem servir 8 de preescaler) del OCRnA/B/C ha de ser aproximadament 20000. Aquest serà el valor de l'ICRx per tal d'establir la freqüència del pols en 20 mseg. Calculo el valor de OCRxA/B/C per a 1 mseg, 2 mseg i 2,5 mseg i obtinc els valors 2000, 3000 i 4000 respectivament. Així doncs, aquests valors seran els que hauré d'associar a OCRxA/B/C quan vulgui posicionar el servomotor en un dels punts concrets.

4.2.1.3 FUNCIONS DE CONTROL DEL SERVOMOTOR

```
void servoInit(void);
```

Inicialitza el servomotor configurant el PWM segons els paràmetres calculats i posiciona el servomotor cap al centre.

```
void moveServo(int graus, int dir);
```

Mourà el servomotor a la posició indicada. Aquesta posició quedarà definida pels paràmetres *graus* i *dir*, on el primer definirà els graus de gir i el segon la direcció, 1 per anar a la dreta i 0 per anar a l'esquerra.

```
void turn_left(void);
```

Posicionem el sonar a 0 graus, a l'esquerra. En aquest cas el valor de OCRxA/B/C és 2000.

```
void turn_right(void);
```

Posicionem el sonar a 180 graus, a la dreta. En aquest cas el valor de OCRxA/B/C és 4000.

```
void center(void);
```

Posicionem el sonar a 90 graus, al centre. En aquest cas el valor de OCRxA/B/C és 3000.

```
void escombrada(int num);
```

Funció que realitza escombrades des de 0° a 180° i de 180° a 0°. El paràmetre *num* especifica el nombre d'escombrades que es faran. L'efecte d'avançament és fa mitjançant el increment iteratiu de OCRxA/B/C amb un retard entre cada iteració, des de 2000 fins a 4000 i després el decrement des de 4000 fins a 2000.

4.2.2 EL SONAR

4.2.2.1 SRF02

El sonar triat es caracteritza principalment per tenir un mateix transductor per l'emissió i la recepció dels ultrasons. Això el diferencia d'altres que disposen d'un transductor per l'emissió i un altre per la recepció, i possibilita que la distància mínima a la que detecta una presència pugui ser menor. Aquest tret és important, doncs aquest sonar ens permet detectar obstacles entre els 15 i els 600 cm. Disposa de dos modes de comunicació: sèrie i I2C. I2C es un bus de comunicacions dissenyat per Phillips i consta únicament de dues línies: la de dades (SDA) i la de rellotge (SCL). Per al nostre projecte és convenient l'ús del I2C, degut a que el mode sèrie necessita fer ús dels pins de transmissió i recepció del Atmega 128, que necessitarem per a altres usos.

Devantech SRF02

Alimentació: 5 V

Mides: 24x20x17 mm

Pes: 4,6 gr

Fabricant: Devantech

Preu: 17,09 €

4.2.2.2 ARQUITECTURA I CONNEXIÓ

Per poder desenvolupar la comunicació cal afegir unes petites modificacions a nivell hardware. Caldrà que afegeixi les línies SCL i SDA connectant-les a l'alimentació mitjançant una resistència de pull-up de 4,7kΩ per cada línia. Les línees SCL i SDA tenen els seus propis pins a l'Atmega 128, i són el PD0 i el PD1.

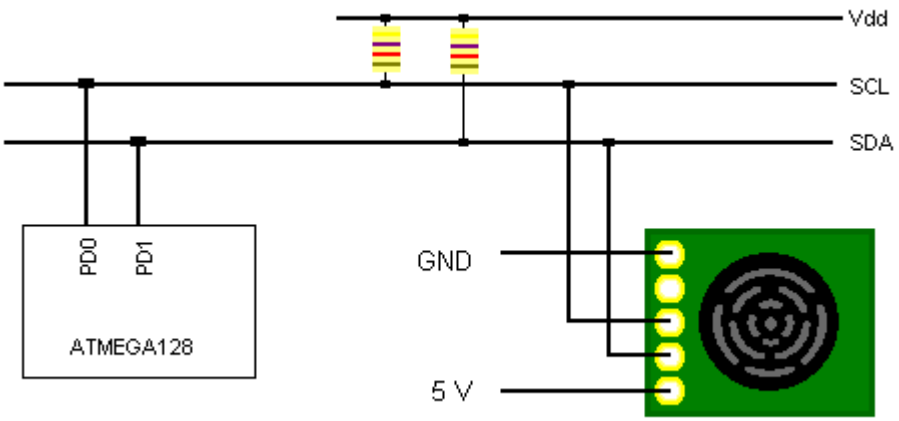


Figura 10

4.2.2.3 DRIVER DEL SONAR

Un cop establertes les connexions necessàries per a la comunicació I2C cal entendre com establirem el diàleg entre microcontrolador i sensor. Cal que en aquest apartat comentem dos conceptes claus: el Two-Wire Serial Interface i el protocol de comunicació I2C.

El Two-Wire és la interfície sèrie proporcionada per l'Atmega 128 per a comunicacions I2C. Permet connectar 128 dispositius diferents utilitzant únicament les línies SCL i SDA. L'esquema de funcionament és el de mestre-esclau, on el mestre és el microcontrolador i els esclaus són cadascun dels 128 dispositius possibles. Hi ha dos conjunts d'especificacions: les de busos amb freqüències

inferiors a 100 kHz (com en el cas del SRF02 que és de 40 kHz) i les de busos amb freqüència fins a 400 kHz. Per configurar la freqüència del rellotge (SCL) faré servir els registres TWBR (rang del rellotge) i TWPS (prescalar) i em guiaré per la següent fórmula, proporcionada pel fabricant:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

Els altres registres que faré servir per la comunicació seran el TWCR i el TWDR. El primer és el registre per al control d'operació del Two-Wire. L'utilitzaré per indicar el tipus d'operació que faré a continuació i ho indicaré segons el valor indicat a les especificacions del SRF02, que es poden consultar a l'annex (10.1.3). El registre TWDR, en mode de transmissió, contindrà el següent byte a ser transmès. En mode de recepció, contindrà el darrer byte rebut. Per tant, serà el registre en el qual escriurem i llegirem les dades.

4.2.2.4 FUNCIONS DE CONTROL DEL SONAR

```
char i2c_writebyte(char addr ,char reg,char value); unsigned char i2c_readbyte(char addr,char reg);
```

Aquestes dues funció venen incloses a la llibreria d'I2C de Peter Fleury. La primera la faig servir per escriure en el dispositiu la comanda a realitzar i la segona per llegir la resposta. En totes dues el paràmetre *addr* especifica l'adreça del dispositiu amb el que ens volem comunicar, i el paràmetre *reg* el registre al qual volem accedir.

```
unsigned int medicio(void);
```

Aquesta funció realitza una medició mitjançant la combinació de les funcions d'escriptura i lectura abans comentades. Primer envia la comanda per obtenir una medició en centímetres i després rep la resposta, retornant-la com a enter.

4.3 INFRAROJOS

Un sensor d'infraroig és un dispositiu electrònic amb capacitat per mesurar la radiació electromagnètica infraroja en el seu camp de visió. És per això que mitjançant la radiació que emeten tots els cossos podrem detectar la presencia d'un obstacle i augmentar la capacitat de percepció del robot. Com que els infrarojos acostumen a tenir un camp de visió reduït, en el robot aniran destinats a captar presències als laterals del robot, zona que queda fora de l'abast del sonar. El principi de funcionament es basa en la combinació d'un emissor, un diode LED infraroig, i un receptor, el fototransistor. Al mercat ens trobarem amb la possibilitat de comprar cada component per separat i fabricar pel nostre compte la parella de comunicació, o bé sensors on la parella ja ve composta en un petit sistema que inclou un circuit integrat.

4.3.1 HITEC DISTANCE SENSOR

El sensor d'infraroig triat s'ofereix amb la parella emissor/receptor ja composta, la qual cosa simplifica bastant el seu ús. Es redueix simplement a la detecció d'objectes, sense donar informació de la distància a la qual es troben. A la meva arquitectura facilita molt la col·locació dins del xassís, doncs es tracta d'un sensor de mida força reduïda. La connexió és igual a la d'un servomotor, amb un senyal que passarà de 1 a 0 quan es detecti una presència.

Hitec Distance Sensor

Alimentació: 5 V

Distància màxima de detecció: 15 cm

Mides: 12.9x25.4 mm

Fabricant: Hitec

Preu: 7,75 €

4.3.2 FUNCIONAMENT

Per la simplicitat del seu funcionament no és necessària l'elaboració de cap driver. Simplement haurem de definir com a entrada el port al qual l'associem. Consultarem aquest port d'entrada periòdicament, i segons el seu estat podrem determinar si hi ha alguna presència. Com en el meu cas els infrarojos estan col·locats als laterals del robot, quan es detecti una presència corregiré la trajectòria del robot per evitar la col·lisió i després tornaré a adreçar la direcció.

5 FUNCIONAMENT GLOBAL DEL SISTEMA

Un cop programats els drivers per controlar els diferents sensors i les altres parts del robot, és necessari plantejar com englobar-ho tot en un mateix sistema. Cal tenir en compte que un processador convencional es limitaria a executar codi de manera seqüencial, i que això resultaria molt poc útil si la idea que pretenc és fer un robot consistent. Aquesta limitació és força important, doncs no em permetria realitzar tasques en paral·lel quan en molts casos la concurrència esdevé necessària. Per exemple, obtenir informació dels sensors alhora que funciona la locomoció o comunicar amb el PC mentre s'executa alguna tasca. Per això em plantejo una nova perspectiva per al control global del sistema: el Sistema Operatiu.

Un sistema operatiu és un software format per un elevat nombre de programes que s'encarreguen d'administrar els recursos d'un sistema. De manera molt genèrica podríem dir que el que tracta de fer és solucionar el problema de la concurrència que abans hem esmentat, permetent l'execució de diferents tasques paral·lelament i de la manera més eficient possible. Actualment hi ha un gran ventall de sistemes operatius i per això cal triar el més adequat per al projecte. M'interessa un sistema operatiu senzill i "petit", doncs tenim certes limitacions bastant obvies, i que a ser possible ofereixi la possibilitat de suportar la nostra arquitectura. Amb aquests requisits en ment, s'ofereix una alternativa que a més presenta un gran nombre d'avantatges: FreeRTOS.

5.1 EL SISTEMA OPERATIU DEL ROBOT: FREERTOS

FreeRTOS és un sistema operatiu en temps real, portable i de codi obert pensat per a sistemes integrats. Quan diem que un sistema operatiu és en temps real estem parlant d'un tipus de plataforma que garanteix l'execució de les tasques en un temps limitat. Per altra banda, la capacitat de portabilitat permet que sigui capaç de ser mogut a altres arquitectures sense gaires problemes. Si

a més afegim la particularitat de tractar-se de software de codi lliure, ens trobem amb un sistema operatiu pràcticament fet a mida amb múltiples avantatges que ens facilitaran la feina.

Destaco les següents característiques:

- FreeRTOS és un sistema operatiu pensat per ser simple, petit i fàcil d'utilitzar. El nucli del kernel es troba contingut en tres únics fitxers i és amplament configurable per part del programador. El codi està escrit majoritàriament en C, la qual cosa facilita la comprensió.

- El seu caràcter lliure ha permès que a més de poder fer ús d'aquesta eina sense restriccions econòmiques, es trobin molts recursos a la xarxa. Hi ha una molt bona documentació i el fet d'haver estat utilitzat i desenvolupat per múltiples usuaris ha permès donar-li consistència, millorar-lo i crear una comunitat molt útil, on és fàcil trobar solucions a problemes i on s'ofereixen recursos de manera lliure. La llibreria de I2C de Peter Fleury seria un bon exemple.

- Com hem comentat abans, FreeRTOS suporta una gran varietat d'arquitectures. Entre elles la família AVR-Atmel. Tot i no trobar-se una implementació especifica per al Atmega128 és relativament senzill adaptar-lo a partir del Atmega323.

- FreeRTOS és un sistema operatiu molt senzill. Però disposa de molts recursos útils per a la comunicació i sincronització de processos: cues, semàfors recursius i binaris, mútex...

- No té restriccions en quant al nombre de tasques que poden ser creades. Tampoc en quan al nombre de prioritats que poden ser utilitzades, ni en l'assignació d'una prioritat a diverses tasques.

5.1.1 COM FUNCIONA?

He parlat anteriorment del problema de la concurrència i de la necessitat que tenim d'executar processos en paral·lel. A continuació explicaré tres conceptes fonamentals per entendre com FreeRTOS realitza aquesta funció.

5.1.1.1 MULTITASKING

Un sistema operatiu multitasca és aquell que és capaç de gestionar els processos de tal manera que produeixi la sensació de que s'executen concurrentment. Es tracta d'executar les tasques de manera seqüencial però alterna de tal forma que sembli que s'executen alhora. Per tant la concurrència no és real sinó aparent. Si observem la figura 11 podem entendre millor aquest concepte. La gràfica superior mostra el resultat aparent: tres tasques en execució durant el mateix interval de temps. A la segona gràfica podem veure com a la realitat el sistema operatiu proporciona una franja de temps per procés que és va alternant, i que permet que les tasques comparteixin el mateix processador produint la sensació d'execució concurrent.

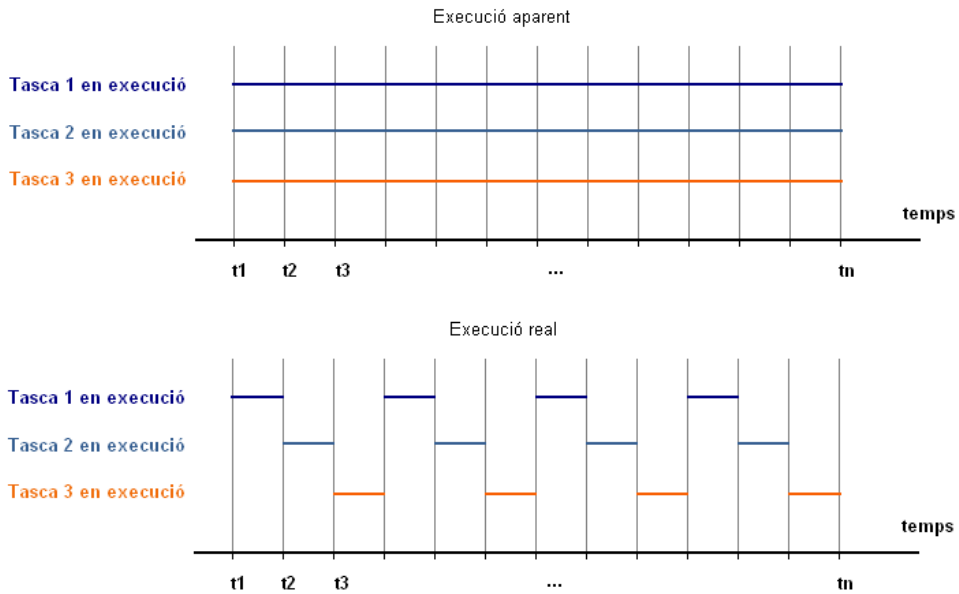


Figura 11

5.1.1.2 SCHEDULING

L'scheduling o planificador és la part del kernel encarregada de decidir quina tasca farà ús del processador en cada moment. Governa, per tant, la concurrència, decidint quina tasca queda suspesa i quina s'executa. Ho fa mitjançant una política de planificació que és l'algoritme que prendrà aquestes decisions. Normalment aquest algoritme s'encarrega d'assignar una franja de temps d'execució durant la qual cada tasca pot restar en aquest estat, i respecta l'ordre d'arribada dels processos mitjançant una cua de peticions. En el nostre cas a més de ser suspesa pel propi kernel, una tasca pot ser suspesa per sí mateixa si, per exemple, necessita que algun recurs estigui disponible o que succeeixi un esdeveniment, evitant consumir temps d'execució de manera innecessària.

Cal esmentar que en tractar-se d'un sistema operatiu en temps real, l'algoritme del planificador ofereix certes particularitats respecte a un sistema operatiu normal. Com hem dit anteriorment, que un sistema operatiu sigui en temps real no vol dir altra cosa que ser capaç de respondre a esdeveniments en un cert temps. El sistema operatiu ha de respectar aquests terminis de temps i ho fa mitjançant valors de prioritat que el programador assignarà a cada tasca. El kernel de FreeRTOS mesura aquests terminis de temps mitjançant una variable que actua com a comptador *(tick counter)* i una interrupció del *timer* que salta cada cert període de temps, regulat pel rellotge del microcontrolador. El *tick counter* serà incrementat cada cop que es generi una interrupció de tal manera que permeti mesurar el temps amb la resolució de la freqüència adequada. Així una tasca pot especificar el període de temps que romandrà suspesa fins que torni a despertar, o el temps màxim d'espera quan estigui bloquejada. Per tant, en base al temps establert per les prioritats, la política del planificador consistirà en que la tasca amb una prioritat més gran i disponible de ser executada, faci ús del processador.

5.1.1.3 CONTEXT SWITCHING

Una tasca en execució fa servir els registres del microcontrolador i accedeix a les memòries RAM i ROM com qualsevol programa. Aquest conjunt de recursos és el que entenem per context. En un microcontrolador AVR està format per 32 registres de propòsit general, un registre d'estat, un *program counter* i dues piles. Quan una tasca queda suspesa, altres tasques passen a estar en execució podent modificar aquests registres. Per tant, una tasca no pot saber que els registres del processador han estat modificats per que en cas de fer servir els valors modificats, el resultat seria incorrecte. Per això és necessari salvar el context d'una tasca cada cop que aquesta entri en estat de suspensió i recuperar-lo just abans de que passi a estar en execució, això es fa mitjançant una pila de memòria on introduirem i extraurem les dades del context. El procés de salvar i restaurar el context és el que anomenem *context switching*, i esdevé clau per mantenir la coherència en el funcionament del sistema operatiu.

5.1.2 IMPLEMENTACIÓ

El sistema operatiu final ha quedat compost per dos mòduls que controlen el principals drivers del robot, més un mòdul d'interfície que utilitzarem per comunicar-nos amb el PC. Els mòduls corresponents als drivers són els de locomoció i el del sonar. Aquest últim controlarà tota la torreta, és a dir: el moviment del servo i les captacions del sonar. En termes generals, podem dir que cada un d'aquests dos mòduls està compost per quatre fitxers, dos fitxers de codi font (extensió .c) i dos fitxers de llibreria (extensió .h). Una primera parella serà la composta pel fitxer on està implementat el codi font del driver i la seva respectiva llibreria. La segona parella estarà composta pel codi font on es duen a terme la crida a les possibles funcions del driver i la interacció amb el PC, amb la respectiva llibreria. Apart caldran altres arxius de suport i llibreries que poden ser necessàries, per exemple, per a la comunicació sèrie o per controlar el bus I2C. Des de el punt de vista del planificador del FreeRTOS, cadascun d'aquests mòduls suposa una tasca a ser gestionada.

5.1.2.1 INICIALITZACIÓ DEL SISTEMA

La inicialització del sistema es troba al fitxer *main.c*. El primer que cal fer, abans de res, és iniciar i configurar la comunicació amb el PC, per això cal crear una tasca per aquest propòsit. Posteriorment es crea una cua para cada una de les tasques a gestionar. Com he dit anteriorment, es correspon una tasca per mòdul i per tant cal crear dues cues: motors i sonar. Seguidament s'inicialitza el bus I2C abans de que sigui utilitzat per alguna tasca. Un cop fet això, es crea cadascuna de les tasques i finalment arranquem el planificador per a que les gestioni i puguin ser executades en paral·lel. Per activar el planificador només cal que cridem la funció *vTaskStartScheduler*().

5.1.2.2 MÒDUL D'INTERFÍCIE

Per la seva diferència respecte als altres mòduls que composen el nostre sistema, vull fer menció apart el mòdul d'interfície. L'objectiu d'aquest és rebre les comandes enviades des del terminal del PC mitjançant el port sèrie, per després repartir la feina.

Per rebre les comandes, aquest mòdul acumula a la seva cua d'entrada els caràcters enviats pel port sèrie. Es llegeix el primer caràcter enviat i si correspon a algun dels mòduls, s'envien els caràcters restants a la cua del mòdul corresponent. Aquest farà una interpretació de la comanda i si és correcta executarà la funció corresponent. En cas contrari mostrarà pel terminal el missatge "E:BadSec", indicant que la comanda enviada no és correcta.

En cas de que el primer caràcter rebut a la cua del mòdul d'interfície sigui "i", la comanda serà executada directament (si és correcta) sense haver de passar per altres mòduls.

5.1.2.3 TASQUES

Les tasques a FreeRTOS equivalen al que anomenem processos a qualsevol altres sistema operatiu. Quan s'inicia el sistema operatiu es crea una tasca pròpia, anomenada *idle task*, que s'executa sempre que no s'estigui executant cap altre tasca. Roman en espera de que s'executin altres tasques i allibera la memòria d'aquelles que ja han estat eliminades. Aquesta tasca sempre està en estat de poder ser executada i la seva prioritat sempre està per sota de la prioritat d'una altra tasca. Per a les tasques creades per nosaltres cal que utilitzem la funció *xTaskCreate()* de l'API del FreeRTOS, on especificarem la prioritat que li assignem, i una funció de recepció que sempre tindrà una estructura semblant.

```
static portTASK_FUNCTION( pdTASK_CODE, pvParameters )
{
    ( void ) pvParameters;
    for( ;; )
    {
          /*Crides a les diferents funcions del driver*/
    }
}
```

Aquesta funció es executada quan es crea la tasca i té com a objectiu rebre la petició i diferenciar la funció a realitzar dintre de les possibilitats de la tasca. Si posem com a exemple el sonar, podem diferenciar entre la funció que mou la torreta 90 graus a l'esquerra i fa una medició o la que genera una escombrada de 180 graus d'abast. En funció de la codificació establerta la funció de recepció executarà la funcionalitat concreta. Per veure la codificació que he establert per cada comanda, podeu adreçar-vos a la guia de comandes de l'annex (10.2).

Els estats en els quals pot romandre una tasca són quatre, alguns dels quals ja he anat citant anteriorment:

- *En execució (Running):* quan la tasca està en execució. És la tasca que fa ús del processador.

- *Preparada (Ready):* quan està disponible per ser executada, però no ho fa perquè alguna altra tasca d' igual o més prioritat està en execució en aquell moment.

- *Bloquejada (Bloqued):* quan roman en espera d'un esdeveniment temporal o extern.

- *Suspesa (Suspended):* quan ha estat cridada la funció *vTaskSuspend*() de l'API de FreeRTOS. No serà alliberada d'aquest estat fins que no sigui cridada la funció *VTaskResume*().

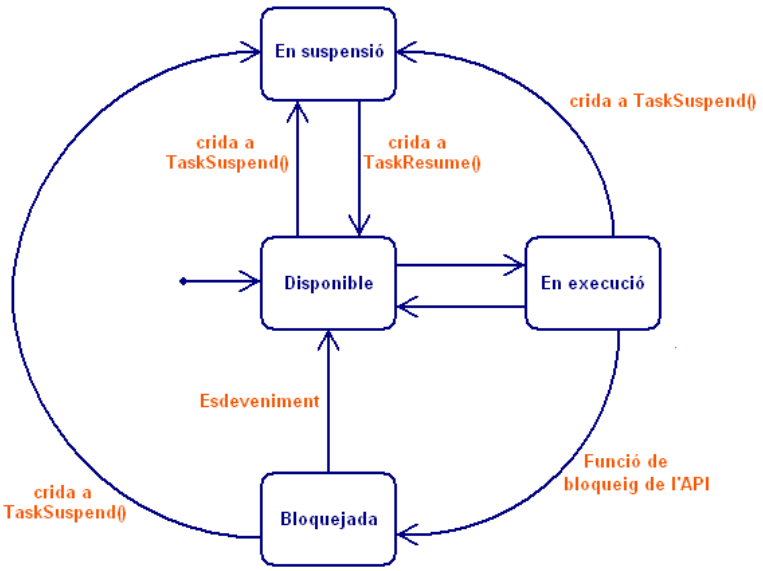


Figura 12

5.1.2.4 COMUNICACIÓ ENTRE TASQUES

FreeRTOS ofereix diferents eines que permeten la comunicació segura i eficient entre les tasques del nostre sistema. Diferenciem tres eines:

- *Cues*: són l'element més elemental de comunicació entre tasques. Permeten enviar missatges entre tasques, i entre tasques i interrupcions. En la majoria dels casos es fan servir de manera semblant a una *buffer* amb política FIFO.

- *Semàfors*: son una eina molt útil per a la compartició de recursos en un sistema operatiu. Actuen donant pas o restringit l'ús d'un recurs per part d'un procés que el necessita per a la seva execució. A FreeRTOS funcionen de manera semblant a com ho fan a la resta de sistemes operatius i hi ha de dos tipus: binaris (orientats a la sincronització de tasques) i comptadors (comptadors d'esdeveniments i de recursos disponibles).

- *Mútex*: semblant a un semàfor binari, inclou una propietat de prioritat que els fa idonis per solucionar problemes d'exclusió mútua. Gestiona l'accés de les seccions crítiques a recursos compartits.

5.1.3 AFEGIR UN MÒDUL AL SISTEMA OPERATIU

Suposem ara que volem afegir un nou mòdul al sistema operatiu implementat. Per fer això, hauríem de seguir l'estructura que he comentat anteriorment:

- *Inicialització del sistema*: com a nova tasca, hauríem de inicialitzar-la dins del *main* especificant el valor de prioritat i la seva respectiva cua.

  Caldria, primer, definir la cua de la manera següent:

  ```
  static xQueueHandle nova_cua;
  ```

  Després cridaríem a la funció per crear-la:

  ```
  nova_cua = xQueueCreate(16,sizeof(portCHAR)*8);
  ```

  Finalment, cridaríem la tasca corresponent:

  ```
  Nova_Tasca(tskIDLE_PRIORITY + 2,nova_cua);
  ```

- *Mòdul d'interfície*: caldria afegir la nova comanda al mòdul d'interfície per tal de que la reconegués des del terminal. Per fer-ho afegiríem un nou estat dins del *switch-case* que fem servir per diferenciar les noves comandes. Dins d'aquest estat, ens encarregaríem de moure el contingut del *buffer* d'entrada, que contindria la comanda, a la cua de la nova tasca, de la forma següent:

  ```
  xQueueSend(nova_cua,&buffer_entrada[1],(portTickType) 100 );
  ```

- *Creació del mòdul*: crearíem els fitxer de codi font i llibreries necessaris. Una primera parella amb el codi font del driver i la seva llibreria. I una segona parella per gestionar la identificació de la comanda i la creació de la nova tasca mitjançant la funció *xTaskCreate().*

Crearíem la tasca tal i com s'indica a continuació:

```
xTaskCreate( Nova_Tasca, ( const signed portCHAR * const ) "Nom", tascaSTACK_SIZE, NULL, uxPriority, ( xTaskHandle * ) NULL );
```

on la mida de la pila queda definida de la següent manera:

```
#define tascaSTACK_SIZE    configMINIMAL_STACK_SIZE+48
```

Finalment definiríem la funció de recepció:

```
static portTASK_FUNCTION( Nova_Tasca, pvParameters )
{
   /* Just to stop compiler warnings. */
   ( void ) pvParameters;

   for( ;; )
   {
         /*Crides a les diferents funcions del driver*/
   }
}
```

6 EINES EMPRADES

6.1 L'ENTORN PER AL DESENVOLUPAMENT

6.1.1 AVR STUDIO 4

AVR Studio 4 és un entorn de desenvolupament orientat a la programació de microcontroladors de la família AVR. Es tracta d'una eina molt útil, d'accés gratuït, que permet simular el funcionament de manera eficaç i que facilita la connexió amb el microcontrolador i la descàrrega del programa. A continuació citaré les característiques més remarcables:

- Entre altres llenguatges, suporta C i assembler. Aquests dos llenguatges són els que m'havia proposat en un principi des del punt de vista educatiu, doncs és fonamental el seu aprenentatge en els primers cursos. Personalment m'inclinaré per l'ús de C com a llenguatge per a la programació de drivers, doncs fer servir un llenguatge de baix nivell com l'assembler podria complicar més el projecte.

- Permet la configuració manual de les memòries SRAM i EEPROM, a més de la memòria flash, registres i entrades/sortides. Especificant el microcontrolador que estem fent servir AVR Studio s'adequa a aquestes característiques i mostra els registres concrets del nostre microcontrolador. La manipulació de registres durant la simulació és molt interessant per poder observar el comportament del programa davant canvis forçats pel programador i solucionar errors ràpidament.

- L'ús del *debugger* és molt interessant. Es pot fer servir simplement com a simulador virtual, o bé simulant sobre el propi microcontrolador. En tots dos casos permet la visualització de l'estat de registres i *flags* facilitant la comprensió, i l'ús il·limitat de *breakpoints*.

6.1.2 WINAVR

WinAVR és un conjunt d'eines executables per a aplicacions Atmel AVR a plataformes Windows. Inclou el compilador de GNU per a C i C++, anomenat *avr-gcc*; el depurador *avr-gdb* i el programador *avrdude*. Serà, per tant, una eina fonamental que treballarà juntament amb el AVR Studio. Com tot el software fet servir en el projecte, WinAVR es tracta d'una altra eina de software lliure i per tant el seu accés és gratuït, està obert al desenvolupament de millores i noves versions, la qual cosa garanteix consistència i suport molt accessible a la xarxa.

6.2 PROGRAMADORS

Un programador és un dispositiu electrònic que s'utilitza per descarregar el programa a la memòria del microcontrolador. Durant l'elaboració del meu projecte n'he fet servir dos: el programador sèrie PonyProg i el ET-AVR JTAG.

6.2.1 PONYPROG

PonyProg és un programador sèrie molt senzill però robust. El principal inconvenient que té és la lentitud en la descàrrega del programa, la qual cosa comporta fer simulacions abans de fer la descàrrega per evitar perdre temps. El seu ús és una mica tediós, enrederint el procés de proves durant la programació dels dispositius. Com a avantatge, es tracta d'un programador molt més econòmic que el JTAG.

6.2.2 ET-AVR JTAG

JTAG és una interfície física a nivell hardware i un protocol a nivell software que, avui en dia, es troba en la majoria de microcontroladors. Podem entendre'l, per tant, com allò que connectem entre el PC i el microcontrolador, sempre i quan aquest últim suporti aquesta interfície. Però també el podem entendre com l'estàndard de comunicació que permet interactuar el PC i el microcontrolador.

El ET-AVR JTAG és un programador que utilitza aquesta tecnologia i que permet treballar conjuntament amb l'entorn de desenvolupament del AVR Studio. Això ofereix la possibilitat de visualitzar els registres del microcontrolador, una facilitat molt interessant que ja hem comentat anteriorment. A més la descàrrega del programa és molt més ràpida, a diferència del PonyProg, i redueix molt el temps de proves i de desenvolupament dels programes. També és molt més fàcil i còmode la descàrrega del programa, degut a la seva integració amb l'AVR Studio, de tal manera que ara només caldrà que polsem un botó per realitzar transferir les dades.

El seu preu és força més elevat, però per les característiques abans esmentades podríem dir que es tracta del programador ideal per a un projecte sobre una aplicació AVR.

7 ANÀLISIS DE COSTOS I PROPOSTA DE VIABILITAT

Com he remarcat anteriorment, l'objectiu del meu projecte és educatiu i per tant cal donar a conèixer un anàlisis dels costos que determinin la seva viabilitat. En els inicis del projecte, a l'hora de cercar i triar cadascun dels components em trobava amb un ampli ventall de possibilitats. En molts casos les diferències de preu eren força elevades i vaig decantar-me per aquells productes que complien els requeriments proposats amb un preu més accessible. Internet és avui en dia una eina que facilita molt aquesta tasca, doncs a més de proporcionar-nos un catàleg molt ample també ens permet accedir a mercats, com l'americà, que ofereixen preus més reduïts que a Europa, per la devaluació de la seva moneda. És per això que m'he decantat pels proveïdors estrangers, tot i que presenten l'inconvenient propi dels enviaments a domicili: temps d'espera que acostuma a ser més llarg del previst i despeses d'enviament.

Un cop avaluat el preu final del prototip, proporcionaré una avaluació de costos més real i precisa, tenint en compte que el meu projecte s'ofereix com a proposta per a realitzar una petita producció de robots. Analitzaré la viabilitat del projecte en funció d'aquesta producció, tenint en compte aquells costos variables i fixos amb els quals m'he trobat durant l'elaboració del prototip i, finalment, decidiré si és viable o no tenint en compte el cost final i les alternatives que trobi al mercat. Podeu adreçar-vos a l'annex per consultar en detall els costos de fabricació de la placa base (10.4) i les despeses d'enviament dels diferents proveïdors (10.5).

7.1 COST UNITARI DEL ROBOT

A continuació mostro la taula de costos dels components del robot per a una única unitat que podem entendre com el prototip.

	Component	Proveïdor	Unitats	€/unitat	Total
Mòdul de sensors	Servomotor DY-S0206	Super Robotica (Espanya)	1	8,45	8,45
	Devantech SRF02	RobotShop (Canada)	1	17,09	17,09
	Hitec Distance Sensor	RobotShop (Canada)	2	7,75	15,5
Mòdul d'unitat de procés	Microcontrolador Atmega128	Futurlec (USA)	1	9,71	9,71
	Mòdul Bluetooth Bluemore 200	Eikon (Itàlia)	1	45	45
Mòdul de locomoció i alimentació	Motor de CC Solarbotics GM8	RobotShop (Canada)	2	3,73	7,46
	Rodes Solarbotics GM Series	RobotShop (Canada)	1	4,22	4,22
	Solarbotics GM Bracket	RobotShop (Canada)	2	0,87	1,74
	Bola d'acer Tamiya 70144	Pololu (USA)	1	4,22	4,22
	SFE Battery Holder	RobotShop (Canada)	1	1,37	1,37
	Kit de dos plafons de PVC	Super Robotica (Espanya)	1	9,05	9,05
		Components de la placa d'expansió + Fabricació			26,41
				SUBTOTAL	**105,22**
				Despeses d'enviament	64,17
				TOTAL	**169,39**

Aquests son els costos amb els que em vaig plantejar el projecte. El preu unitari del robot no és excessivament elevat si tenim en compte el preu sense les despeses d'enviament. De moment podem dir que a nivell unitari és viable, tot i que les despeses d'enviament comporten gairebé la mateixa despesa que la suma de tots els components. Com a tot prototip és necessari assumir una despesa més elevada del que s'espera durant el període d'elaboració del projecte. Un cop acceptat el prototip com a vàlid, cal aprofundir en l'anàlisi final del projecte. Tractaré d'amortitzar les despeses d'enviament i fer una avaluació molt més realista del projecte, tenint en compte la producció final decidida.

7.2 PROPOSTA DE VIABILITAT

Analitzar els costos del projecte tenint en compte un nombre d'unitats de tirada no només em permetrà observar fins a quin punt puc amortitzar les despeses d'enviament i, en conseqüència, tenir més dades respecte de la pròpia viabilitat del projecte, sinó que a més em permetrà aprofitar les ofertes que ofereixen la majoria de proveïdors per a compres d'unitats elevades i que ens ajudaran a reduir el preu final per unitat. Finalment, ens oferirà el cost total del projecte en conjunt, i que al cap i a la fi serà la quantitat que ens haurem de plantejar per tenir en compte si el projecte es pot dur a terme o no.

Primer de tot cal que decideixi quants robots crec que són necessaris per al meu projecte. Tenint en compte el context inicial, basat en la facultat d'enginyeries de la Universitat Autònoma de Barcelona, la tirada adient seria entorn a les 25 unitats. Aquest nombre em permet aprofitar les reduccions de costos abans comentades, que acostumen a efectuar-se per a 10 i 25 unitats (també per a tirades més elevades, però que escapen de les nostres perspectives), sense fer una despesa més gran que la estrictament necessària. Així doncs, proposo una tirada de 25 robots que permetrà assolir de sobres les necessitats educatives i oferir un petit marge de error.

	Component	Proveïdor	Unitats	€/unitat	Total
Mòdul de sensors	Servomotor DY-S0206	Super Robotica (Espanya)	25	8,45	211,25
	Devantech SRF02	RobotShop (Canada)	25	14.81	370,25
	Hitec Distance Sensor	RobotShop (Canada)	50	6,39	319,5
Mòdul d'unitat de procés	Microcontrolador Atmega128	Futurlec (USA)	25	8,86	221,5
	Mòdul Bluetooth Bluemore 200	Eikon (Itàlia)	25	45	1125
Mòdul de locomoció i alimentació	Motor de CC Solarbotics GM8	RobotShop (Canada)	50	3,32	166
	Rodes Solarbotics GM Series	RobotShop (Canada)	25	3,79	94,75
	Solarbotics GM Bracket	RobotShop (Canada)	50	0,79	39,5
	Bola d'acer Tamiya 70144	Pololu (USA)	25	3,97	99,25
	SFE Battery Holder	RobotShop (Canada)	25	1,29	32,25
	Kit de dos plafons de PVC	Super Robotica (Espanya)	25	9,05	226,25
				Components de la placa d'expansió (x25)	490,25
				Construcció placa d'expansió	170
				SUBTOTAL	**3565,75**
				Preu per unitat sense despeses d'enviament	142,63
				Despeses d'enviament	62,84
				TOTAL	**3628,59**
				Preu final per unitat	**145,14**

Com podem observar, el preu final de cada unitat és de 145,14 €, quantitat lleugerament inferior a la del cost del prototip. El cost total del projecte, s'eleva als 3628,59 €. Per establir una comparació amb una solució alternativa, m'agradaria remarcar les diferències entre la meva proposta i l'alternativa més semblant que podem trobar al mercat. El Boe-Bot de Parallax, que es ven com un paquet on s'inclouen tots els elements necessaris per a la construcció d'un robot similar, té un preu de 87,64 € per unitat (preu per a compres de més de 20 unitats), sense tenir en compte les despeses d'enviament. Tindríem, per tant, un cost final aproximat de 2191 €. El cost final del projecte, per tant, és molt més reduït si optem per l'opció de Parallax. Però el que vull remarcar és que aquesta opció reduiria greument la capacitat educativa del projecte, doncs el llenguatge de programació que utilitza Parallax, el PBASIC, és un llenguatge propi de Parallax i té molt poc interès per alumnes d'estudis superiors. El que succeeix amb el Boe-Bot, passa també amb l'Scribbler o el SumoBot del mateix fabricant. És podria dir que, en el context en el que ens trobem, els robots de Parallax no tenen el valor degut com a eina educativa.

8 CONCLUSIONS

Un cop exposat el projecte des del plantejament inicial fins a la pròpia implementació, és moment de que exposi les conclusions extretes i avaluï la proposta.

Des d'un punt de vista tècnic, l'alternativa que proposo gaudeix d'una unitat de procés preparada per a grans volums de càlcul i amb capacitat per suportar un nombre elevat de sensors. Això permet certa agilitat en el sistema i unes possibilitats de moviment i interacció amb el medi força significatives. Per altra banda, i com ja he esmentat en altres capítols, la versatilitat del Atmega 128 permet que sigui programat en llenguatges de programació tan fonamentals com són C i assembler. Per això puc dir que respecte a altres alternatives, la que presento ofereix una millor qualitat de processament, un horitzó molt ampli per a millores i ampliacions, i una important adaptabilitat educativa que no travaríem en cap altre cas. Els cost, tot i ser més elevat que la resta d'alternatives trobades al mercat, no és excessivament superior. Més si tenim en compte els avantatges abans esmentats i el grau de coneixement que arribem a tenir respecte a un autòmat que ha estat construït per un mateix.

Respecte a la idea original que em vaig plantejar en quant a la funcionalitat del robot, penso que ha estat aconseguida en diferents aspectes. S'ofereix un bon control sobre el microcontrolador, a través del sistema operatiu, i es permet l'actuació i percepció dels sensors tal i com era necessari, mitjançant una interfície i a través d'un terminal. En aquest sentit, han estat més difícil dominar o entendre el control del sonar, per exemple, que no pas el del servomotor. Per altre banda també s'ha aconseguit dotar de navegació autònoma al robot i d'alguna manera posar a prova els coneixements adquirits, obtenint un resultat satisfactori.

Per la meva part, l'experimentació amb l'Atmega 128 i els diferents dispositius ha estat molt profitosa, doncs m'ha permès aprendre aspectes molts concrets de la programació d'un microcontrolador, arribant a entendre d'una manera més global els conceptes fonamentals i la metodologia de treball necessària per dur a terme un projecte d'aquesta envergadura. Si la raó de ser d'aquest projecte és la seva voluntat educativa, concloure dient que la seva primera funció docent, el meu projecte final de carrera, ha complert l'objectiu.

8.1 POSSIBLES MILLORES

Per molt que tot plantejament inicial tracti de ser el més perfecte possible, al final del projecte és fàcil observar certes millores que es podrien aplicar.

A nivell tècnic hi ha moltes possibles millores. Pel que fa a la percepció del robot, seria molt profitós augmentar la seva capacitat de detectar presències per la part frontal. Tot i comptar amb la torreta del sonar, que permet aconseguir una navegació força intel·ligent, és fàcil observar que un parell de sensors d'infrarojos col·locats per sota del sonar permetrien detectar obstacles baixos que de vegades queden fora de l'abast de detecció del sonar. En la mateixa direcció, si volguéssim oferir encara més possibilitats, podríem afegir un parell de infrarojos més a la part del darrera, per controlar també la locomoció en aquesta direcció.

A nivell funcional es podria fer una ampliació força interessant que, tot i haver-la plantejat en un principi, va quedar descartada per la dificultat de dur-la a terme en el temps previst del projecte. Aquesta funcionalitat és la de col·locar un sistema seguidor de línies, a la part inferior del xassís, que permetés al robot seguir una ruta marcada al terra. Aquest sistema consistiria, bàsicament, en col·locar un conjunt de sensors de proximitat que fossin capaços d'orientar el robot quan hi hagués algun canvi de direcció a la ruta. Amb aquesta funcionalitat dotaríem el robot d'una major versatilitat i capacitat per adaptar-se a diferents problemes.

Cal dir que totes aquestes millores i ampliacions no repercutirien gaire en els costos del projecte i que, per tant, la seva implementació no comportaria grans canvis en aquest sentit.

9 BIBLIOGRAFIA

Llibres:

1.- Gadre, Dhananjay V. *Programming and customizing the AVR microcontroller*. Nueva York: McGraw-Hill, 2001. 339p. ISBN 0-07-134666-X

2.- Parab, J.S., & Shelake, V. G., & Kamat, R. K., & Naik, G. M., *Exploring C for Microcontrollers*. Dordrecht: Springer, 2007. 157p. ISBN 978-1-4020-6066-3

Adreces electròniques:

3.- Atmel Corporation. *ATMEGA128 (L) Summary.*[En línia]. 2002. Atmel Corporation, 2002, última revisió : 2009. [2008-2009]

http://www.atmel.com/atmel/acrobat/doc2467.pdf

4.- Peter Fleury. *AVR-GCC Software* [En línia]. 2006. Peter Fleury., 2008-2009, [2008-2009].

http://homepage.hispeed.ch/peterfleury/avr-software.html

5.- AVR Freacks. *AVR. The nr.1 AVR Comunity* [En línia]. 2004. AVR Freaks, 2004-2009, [2008-2009]

http://www.avrfreaks.net/

6.- Richard Barry. *FreeRTOS* [En línea]. 2000-2007. Richard Barry., 2003-2007, [2006-2007]. **http://www.freertos.org**

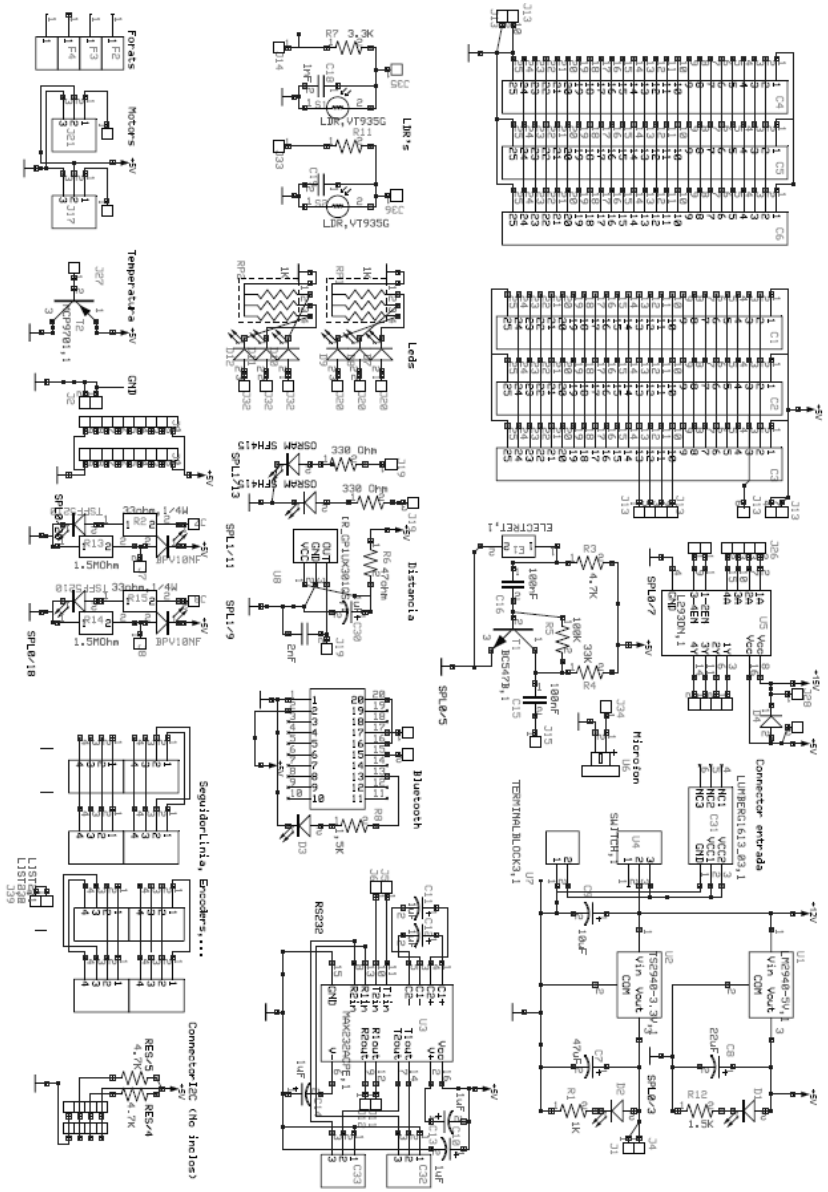
7.- Nand Gupta, K., & Agrawal, P., & Agarwal, M. *Motor driver L293D* [En línia]. 2004. Mobilná Robotika, 2008-2009, [2008-2009]

http://robotika.yweb.sk/skola/H-brigde/IC%20L293%20and%20Its%20avr%20%20interface%20v1.0.pdf

10 ANNEX

10.1 ESPECIFICACIONS HARDWARE

10.1.1 ESQUEMÀTICS DE LA PLACA BASE ACTUAL



Datasheet oficial de Solarbotics.

GM8 Plastic Geared Motor
Offset/inline output shaft
Ratio: 143:1
Dimensions: 53.8 x 47.8 x 22.9mm
Weight: 31.5g

SOLARBOTICS
1-866-276-2687 www.solarbotics.com

GM8 with Default RM3 motor

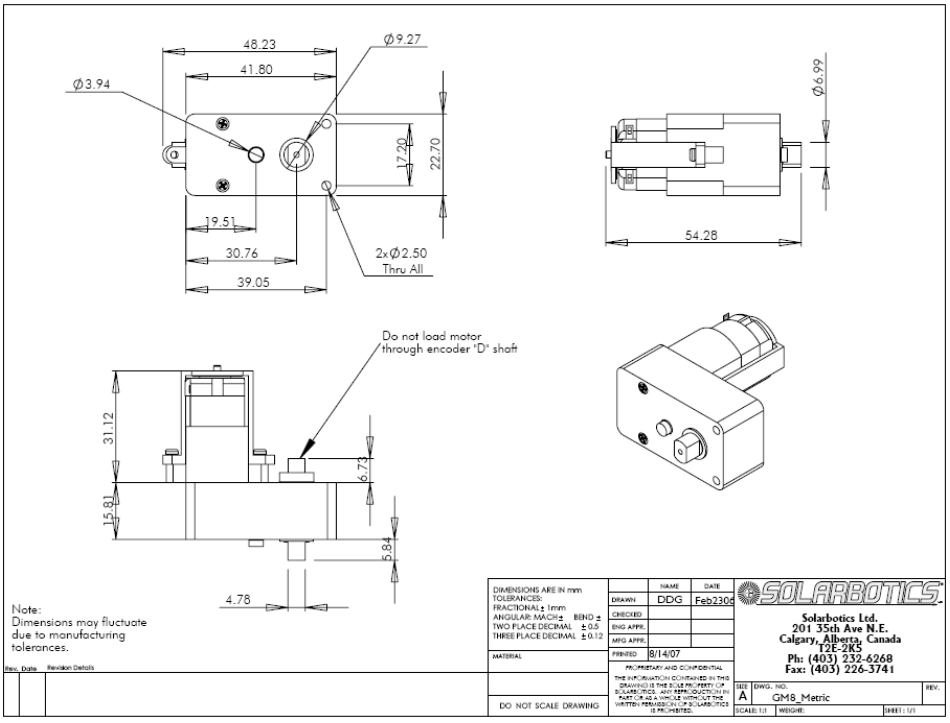
Test Voltage (V)	Unloaded RPM	Unloaded Current (mA)	Stall Current (mA)	Stall Torque (gm*cm)	Stall Torque (oz*in)
3	40	50	400	3200	44.44
6	78	52	700	5500	76.38
9	110	62	1000	5700	79.16
12	135	76	1250	5850	81.24

GM8 with Rev'ed up RM2 motor

Test Voltage (V)	Unloaded RPM	Unloaded Current (mA)	Stall Current (A)	Stall Torque (gm*cm)	Stall Torque (oz*in)
3	145	350	3.2	5575	77.42
6	280	595	4	6000	83.32
9	Not Recommended	Not Recommended	Not Recommended	Not Recommended	Not Recommended
12	Not Recommended	Not Recommended	Not Recommended	Not Recommended	Not Recommended

GM8 with optional 12v motor

Test Voltage (V)	Unloaded RPM	Unloaded Current (mA)	Stall Current (mA)	Stall Torque (gm*cm)	Stall Torque (oz*in)
3	20	45	220	4650	22.91
6	47	60	400	4300	59.72
9	70	65	600	5500	76.38
12	100	72	800	6000	83.32



10.1.3 DEVANTECH SRF02

La informació exposada a continuació ha estat estreta de www.robot-electronics.co.uk.

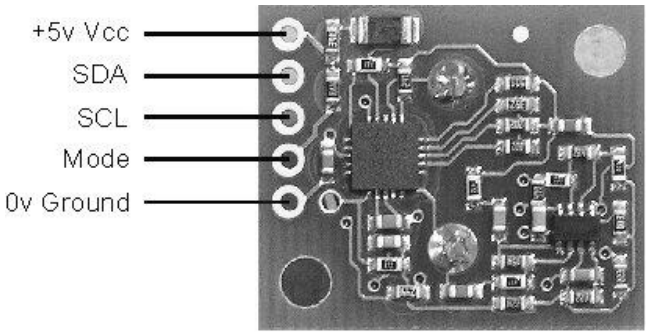
I2C Communication

To use the SRF02 in I2C mode, make sure nothing is connected to the mode pin, it must be left unconnected.

The I2C bus is available on popular controllers such as the OOPic, Stamp BS2p, PicAxe etc. as well as a wide variety of micro-controllers. To the programmer the SRF02 behaves in the same way as the ubiquitous 24xx series EEPROM's, except that the I2C address is different. The default shipped address of the SRF02 is 0xE0. It can be changed by the user to any of 16 addresses E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC or FE, therefore up to 16 sonar's can be used.

Connections

The connections to the SRF02 are identical to the SRF08 and SRF10 rangers. The "Mode" pin should be left unconnected, it has an internal pull-up resistor. The SCL and SDA lines should each have a pull-up resistor to +5v somewhere on the I2C bus. You only need one pair of resistors, not a pair for every module. They are normally located with the bus master rather than the slaves. The SRF02 is always a slave - never a bus master. If you need them, I recommend 1.8k resistors. Some modules such as the OOPic already have pull-up resistors and you do not need to add any more.



Registers

The SRF02 appears as a set of 6 registers.

Location	Read	Write
0	Software Revision	Command Register
1	Unused (reads 0x80)	N/A
2	Range High Byte	N/A
3	Range Low Byte	N/A
4	Autotune Minimum - High Byte	N/A
5	Autotune Minimum - Low Byte	N/A

Only location 0 can be written to. Location 0 is the command register and is used to start a ranging session. It cannot be read. Reading from location 0 returns the SRF02 software revision. The ranging lasts up to 65mS, and the SRF02 will not respond to commands on the I2C bus whilst it is ranging.

Locations, 2 and 3, are the 16bit unsigned result from the latest ranging - high byte first. The meaning of this value depends on the command used, and is either the range in inches, or the range in cm or the flight time in uS. A value of 0 indicates that no objects were detected. Do not initiate a ranging faster than every 65mS to give the previous burst time to fade away.

Locations, 4 and 5, are the 16bit unsigned minimum range. This is the approximate closest range the sonar can measure to. See the Autotune section below for full details.

Commands

The are three commands to initiate a ranging (80 to 82), to return the result in inches, centimeters or microseconds. Another set of three commands (86 to 88) do the same, but without transmitting the burst. These are used where the burst has been transmitted by another sonar. It is up to you to synchronize the commands to the two sonar's. There is a command (92) to transmit a burst without doing the ranging and also a set of commands to change the I2C address.

Command		**Action**
Decimal	**Hex**	
80	0x50	Real Ranging Mode - Result in inches
81	0x51	Real Ranging Mode - Result in centimeters
82	0x52	Real Ranging Mode - Result in micro-seconds
86	0x56	Fake Ranging Mode - Result in inches
87	0x57	Fake Ranging Mode - Result in centimeters
88	0x58	Fake Ranging Mode - Result in micro-seconds
92	0x5C	Transmit an 8 cycle 40khz burst - no ranging takes place
96	0x60	Force Autotune Restart - same as power-up. You can ignore this command.
160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

Ranging

To initiate a ranging, write one of the above commands to the command register and wait the required amount of time for completion and read the result. The echo buffer is cleared at the start of each ranging. The ranging lasts up to 66mS, after this the range can be read from locations 2 and 3.

Checking for Completion of Ranging

You do not have to use a timer on your own controller to wait for ranging to finish. You can take advantage of the fact that the SRF02 will not respond to any I2C activity whilst ranging. Therefore, if you try to read from the SRF02 (we use the software revision number a location 0) then you will get 255 (0xFF) whilst ranging. This is because the I2C data line (SDA) is pulled high if nothing is driving it. As soon as the ranging is complete the SRF02 will again respond to the I2C bus, so just keep reading the register until its not 255 (0xFF) anymore. You can then read the sonar data. Your controller can take advantage of this to perform other tasks while the SRF02 is ranging. The SRF02 will always be ready 70mS after initiating the ranging.

LED

The red LED is used to flash out a code for the I2C address on power-up (see below). It also gives a brief flash during the "ping" whilst ranging.

Changing the I2C Bus Address

To change the I2C address of the SRF02 you must have only one sonar on the bus. Write the 3 sequence commands in the correct order followed by the address. Example; to change the address of a sonar currently at 0xE0 (the default shipped address) to 0xF2, write the following to address 0xE0; (0xA0, 0xAA, 0xA5, 0xF2). These commands must be sent in the correct sequence to change the I2C address, additionally, No other command may be issued in the middle of the sequence. The sequence must be sent to the command register at location 0, which means 4 separate write transactions on the I2C bus. When done, you should label the sonar with its address, however if you do forget, just power it up without sending any commands. The SRF02 will flash its address out on the LED. One long flash followed by a number of shorter flashes indicating its address. The flashing is terminated immediately on sending a command the SRF02.

Address		Long Flash	Short flashes
Decimal	Hex		
224	E0	1	0
226	E2	1	1
228	E4	1	2
230	E6	1	3
232	E8	1	4
234	EA	1	5
236	EC	1	6
238	EE	1	7
240	F0	1	8
242	F2	1	9
244	F4	1	10
246	F6	1	11
248	F8	1	12
250	FA	1	13
252	FC	1	14
254	FE	1	15

Take care not to set more than one sonar to the same address, there will be a bus collision and very unpredictable results.

Note - there is only one module address stored in the SRF02. If you change it, the equivalent Serial Mode address will also change:

0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF2, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC, 0xFE I2C addresses

0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F Equivalent Serial addresses

AutoTune

The SRF02 does not require any user calibration. You power up and go right ahead and use the SRF02.

Internally, there are tuning cycles happening automatically in the background. After the ultrasonic burst has been transmitted, the transducer keeps on ringing for a period of time. It is this ringing which limits the closest range the SRF02 can measure. This time period varies with temperature and from transducer to transducer, but is normally the equivalent of 11 to 16cm (4" to 6"), a bit more if the transducer is warm. The SRF02 is able to detect the transducer ring time and move its detection threshold right up to it, giving the SRF02 the very best performance possible. On power up, the detection threshold is set to 28cm (11"). The tuning algorithms quickly back this right up to the transducer ring. This happens within 5-6 ranging cycles - less than half a second at full scan speed. After this the tuning algorithms continue to monitor the transducer, backing the threshold up even further when possible or easing it out a bit when necessary. The tuning algorithms work automatically, in the background and with no impact on scan time.

The minimum range can be checked, if required by reading registers 4 and 5. This value is returned in uS, cm or inches, the same as the range. It is also possible to make the SRF02 re-tune by writing command 96 but you can ignore this command. It is used during our testing.

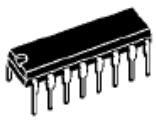
Datasheet oficial de SGS-Thomson.

SGS-THOMSON MICROELECTRONICS

L293D L293DD

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

- 600mA OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (non repetitive) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)
- INTERNAL CLAMP DIODES



SO(12+4+4) Powerdip (12+2+2)

ORDERING NUMBERS:

L293DD L293D

DESCRIPTION

The Device is a monolithic integrated high voltage, high current four channel driver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

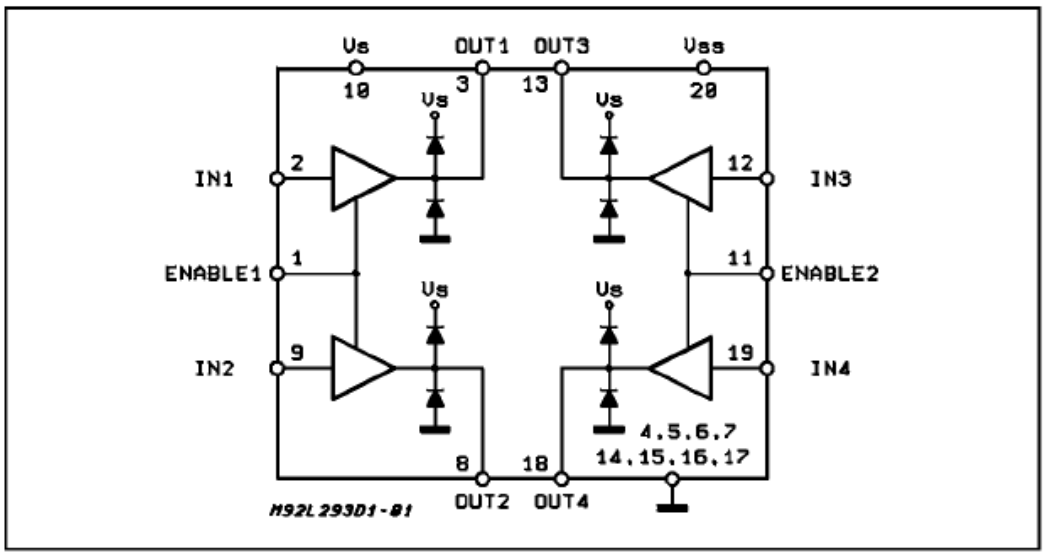
To simplify use as two bridges each pair of channels is equipped with an enable input. A separate supply input is provided for the logic, allowing operation at a lower voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 kHz.

The L293D is assembled in a 16 lead plastic packaage which has 4 center pins connected together and used for heatsinking

The L293DD is assembled in a 20 lead surface mount which has 8 center pins connected together and used for heatsinking.

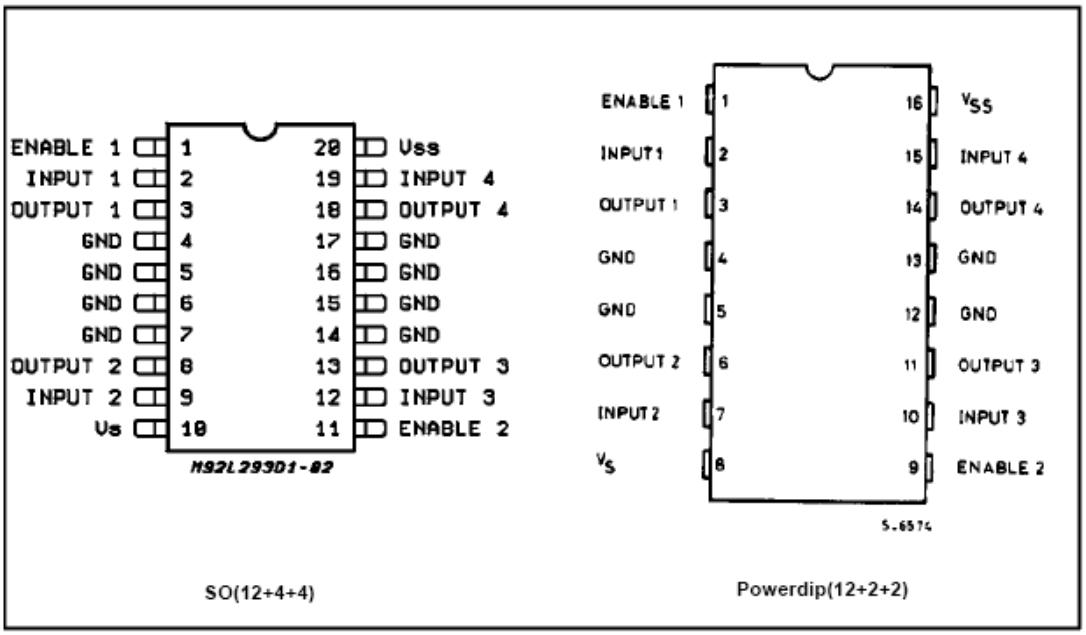
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply Voltage	36	V
V_I	Input Voltage	7	V
V_{en}	Enable Voltage	7	V
I_o	Peak Output Current (100 µs non repetitive)	1.2	A
P_{tot}	Total Power Dissipation at T_{pins} = 90 °C	4	W
T_{stg}, T_j	Storage and Junction Temperature	– 40 to 150	°C

PIN CONNECTIONS (Top view)



SO(12+4+4) Powerdip(12+2+2)

THERMAL DATA

Symbol	Decription		DIP	SO	Unit
$R_{th\ j\text{-}pins}$	Thermal Resistance Junction-pins	max.	–	14	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance junction-ambient	max.	80	50 (*)	°C/W
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	max.	14	–	

(*) With 6sq. cm on board heatsink.

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_I = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_I = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_I = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_I = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	2.3 V ≤ V_{IH} ≤ V_{SS} – 0.6 V		30	100	µA
$V_{en\,L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\,H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\,L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\,L}$ = 1.5 V		– 30	– 100	µA
$I_{en\,H}$	High Voltage Enable Current (pin 1, 11)	2.3 V ≤ $V_{en\,H}$ ≤ V_{SS} – 0.6 V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_I to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_I to 0.5 V_O		200		ns

(*) See fig. 1.

TRUTH TABLE (one channel)

Input	Enable (*)	Output
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance
(*) Relative to the considered channel

Figure 1: Switching Times

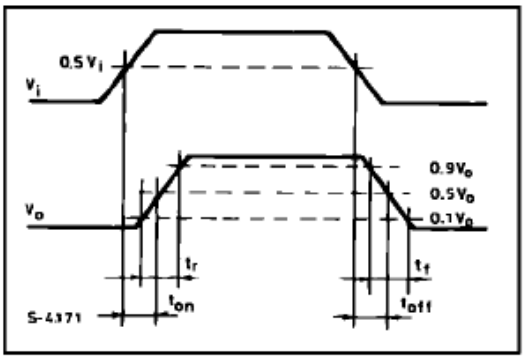
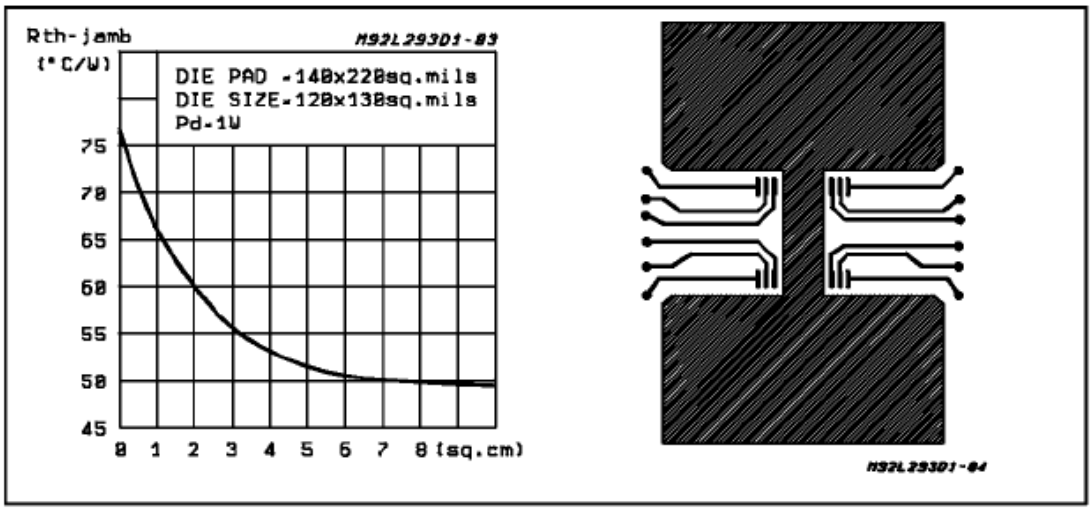
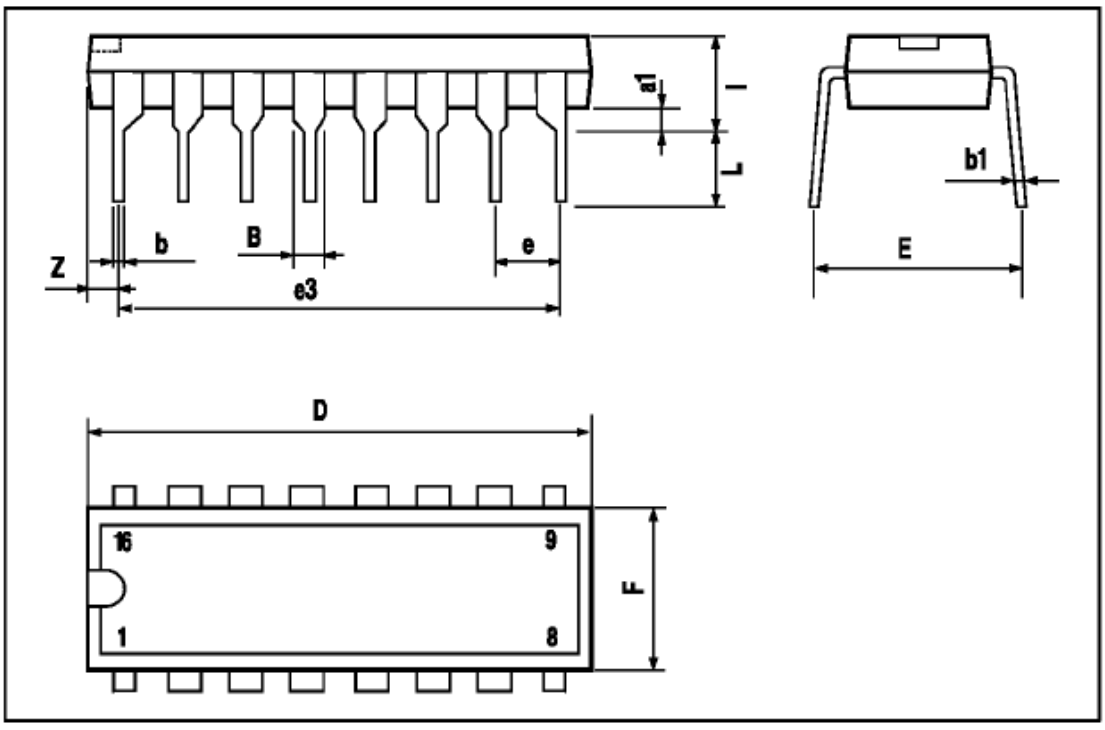


Figure 2: Junction to ambient thermal resistance vs. area on board heatsink (SO12+4+4 package)



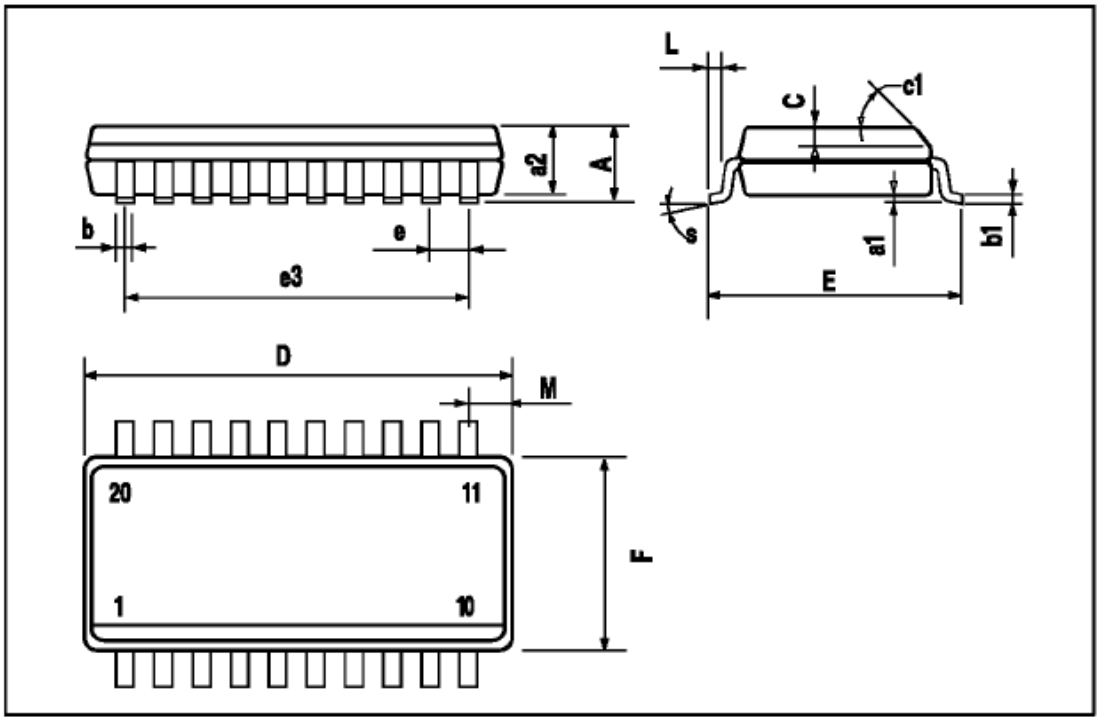
POWERDIP16 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.85		1.40	0.033		0.055
b		0.50			0.020	
b1	0.38		0.50	0.015		0.020
D			20.0			0.787
E		8.80			0.346	
e		2.54			0.100	
e3		17.78			0.700	
F			7.10			0.280
I			5.10			0.201
L		3.30			0.130	
Z			1.27			0.050



SO20 PACKAGE MECHANICAL DATA

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			2.65			0.104
a1	0.1		0.2	0.004		0.008
a2			2.45			0.096
b	0.35		0.49	0.014		0.019
b1	0.23		0.32	0.009		0.013
C		0.5			0.020	
c1		45			1.772	
D		1	12.6		0.039	0.496
E	10		10.65	0.394		0.419
e		1.27			0.050	
e3		11.43			0.450	
F		1	7.4		0.039	0.291
G	8.8		9.15	0.346		0.360
L	0.5		1.27	0.020		0.050
M			0.75			0.030
S	8° (max.)					



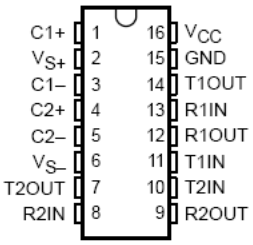
Datasheet oficial de Texas Instruments.

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

- **Meet or Exceed TIA/EIA-232-F and ITU Recommendation V.28**
- **Operate With Single 5-V Power Supply**
- **Operate Up to 120 kbit/s**
- **Two Drivers and Two Receivers**
- **±30-V Input Levels**
- **Low Supply Current . . . 8 mA Typical**
- **Designed to be Interchangeable With Maxim MAX232**
- **ESD Protection Exceeds JESD 22 – 2000-V Human-Body Model (A114-A)**
- **Applications**
  - **TIA/EIA-232-F**
  - **Battery-Powered Systems**
  - **Terminals**
  - **Modems**
  - **Computers**

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube	MAX232N	MAX232N
	SOIC (D)	Tube	MAX232D	MAX232
		Tape and reel	MAX232DR	
	SOIC (DW)	Tube	MAX232DW	MAX232
		Tape and reel	MAX232DWR	
	SOP (NS)	Tape and reel	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube	MAX232IN	MAX232IN
	SOIC (D)	Tube	MAX232ID	MAX232I
		Tape and reel	MAX232IDR	
	SOIC (DW)	Tube	MAX232IDW	MAX232I
		Tape and reel	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

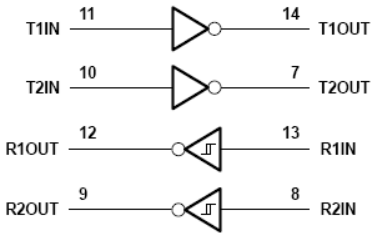
H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) ..... –0.3 V to 6 V
Positive output supply voltage range, V_{S+} ..... V_{CC} – 0.3 V to 15 V
Negative output supply voltage range, V_{S-} ..... –0.3 V to –15 V
Input voltage range, V_I: Driver ..... –0.3 V to V_{CC} + 0.3 V
Receiver ..... ±30 V
Output voltage range, V_O: T1OUT, T2OUT ..... V_{S-} – 0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT ..... –0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT ..... Unlimited
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 73°C/W
DW package ..... 57°C/W
N package ..... 67°C/W
NS package ..... 64°C/W
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds ..... 260°C
Storage temperature range, T_{stg} ..... –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	–40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 3 and Figure 4)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, All outputs open, T_A = 25°C		8	10	mA

‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

DRIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND		5	7		V
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND			–7	–5	V
r_o	Output resistance	T1OUT, T2OUT	$V_{S+} = V_{S-} = 0$,	V_O = ±2 V	300			Ω
I_{OS}§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V,	V_O = 0		±10		mA
I_{IS}	Short-circuit input current	T1IN, T2IN	V_I = 0				200	µA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/µs
SR(t)	Driver transition region slew rate	See Figure 3		3		V/µs
	Data rate	One TOUT switching		120		kbit/s

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

RECEIVER SECTION

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (see Note 3)

PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	R1OUT, R2OUT	I_{OH} = –1 mA		3.5			V
V_{OL}	Low-level output voltage‡	R1OUT, R2OUT	I_{OL} = 3.2 mA				0.4	V
V_{IT+}	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C		1.7	2.4	V
V_{IT-}	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C	0.8	1.2		V
V_{hys}	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V		0.2	0.5	1	V
r_i	Receiver input resistance	R1IN, R2IN	V_{CC} = 5,	T_A = 25°C	3	5	7	kΩ

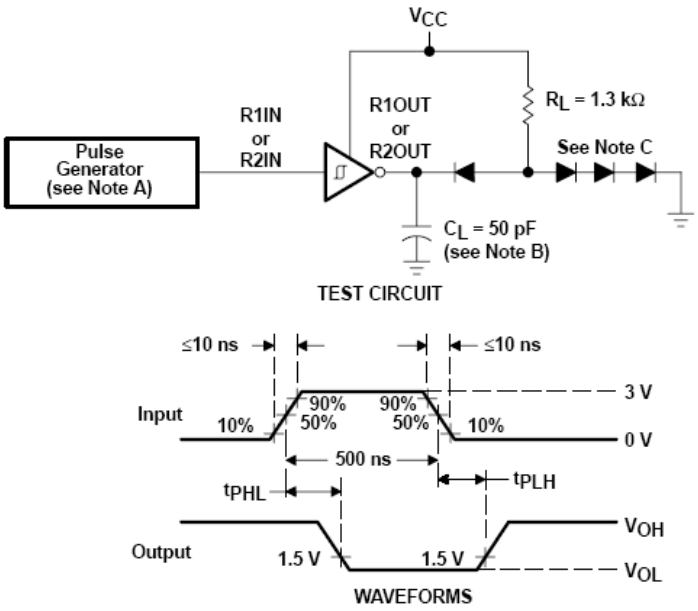
† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Note 3 and Figure 1)

PARAMETER		TYP	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	500	ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	500	ns

NOTE 3: Test conditions are C1–C4 = 1 µF at V_{CC} = 5 V ± 0.5 V.

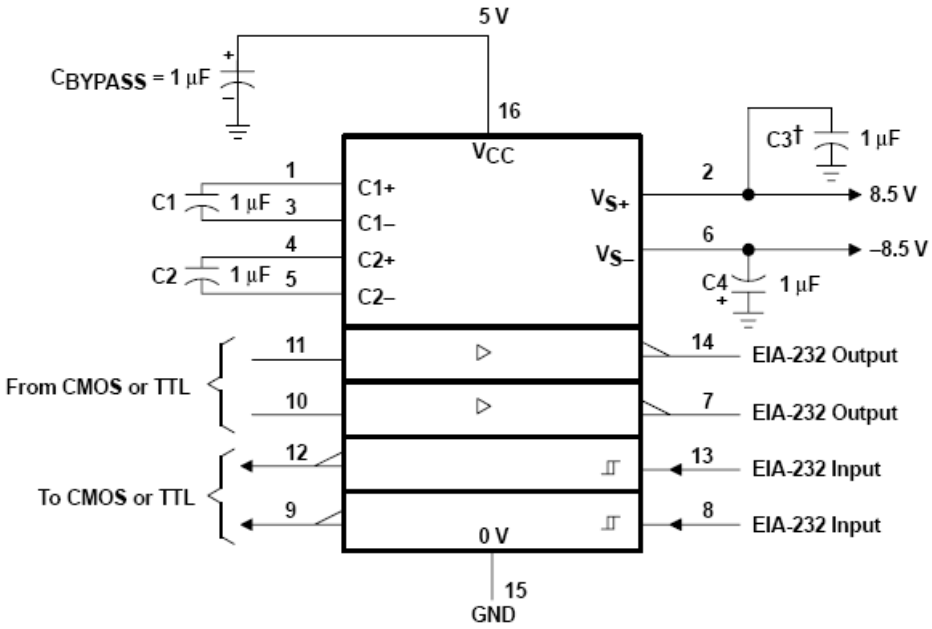
PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.
C. All diodes are 1N3064 or equivalent.

Figure 1. Receiver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements

APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

Figure 4. Typical Operating Circuit

10.2 GUÍA DE COMANDES DELS DRIVERS IMPLEMENTATS

Motors

Les comandes referents als drivers dels motors s'identifiquen pel caràcter inicial "m".

Identificació	***Funció***	***Selecció***	***Velocitat***
m	**f:** endavant **b:** enrera **s:** parada	**r:** motor dreta **l:** motor esquerra **x:** tots dos motors	valor de velocitat, nombre entre 47 i 255 (només per funcions endavant i enrera)
m	**s:** parada	**r:** motor dreta **l:** motor esquerra **x:** tots dos motors	

Exemples:

mfx50: motors en marxa endavant, a velocitat 50.

msr: motor dret aturat.

Torreta amb sonar

Les comandes referents als drivers de la torreta s'identifiquen pel caràcter inicial "s".

Identificació	*Funció*	*Posició*
s	**l:** direcció esquerra, amb medició del sonar **r:** direcció dreta, amb medició del sonar	Valor de posició, en graus.
s	**c:** posicionament al centre, amb medició del sonar **b:** escombrada de 180°	

Exemples:

sl90: torreta posicionada 90 graus a l'esquerra, medició amb el sonar i retorn del resultat.

sc: torreta en posició central, medició amb el sonar i retorn del resultat.

10.3 INSTAL·LACIÓ DE LES EINES UTILITZADES

10.3.1 INSTAL·LACIÓ AVR STUDIO 4 I WINAVR

Descàrrega dels arxius necessaris

Primer de tot cal descarregar els arxius necessaris en un directori local. Els trobarem a la l'adreça http://atmel.com/avrstudio/, i descarregarem la versió més recent.

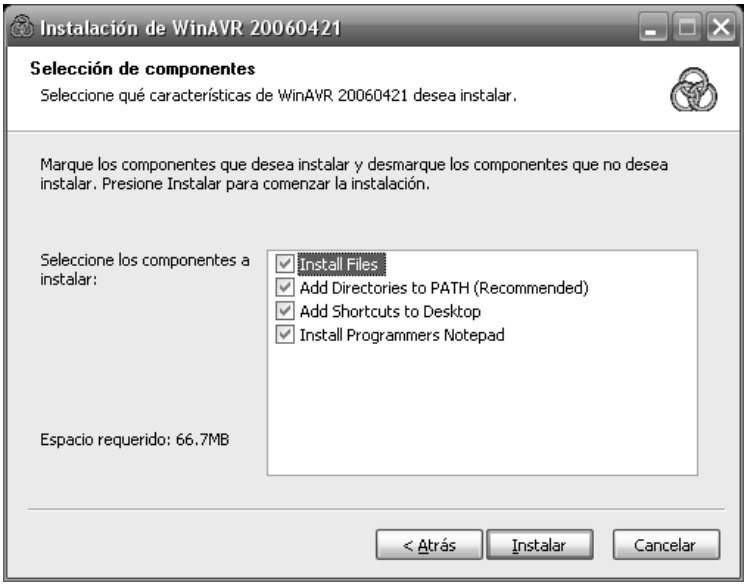
Per descarregar WinAVR ens adreçarem a la següent direcció http://winavr.sourceforge.net/download.html.

Instal·lació Avr Studio 4

Executem el fitxer *AvrStudio417Setup.exe* i seguim els passos. És una instal·lació molt senzilla on triarem el directori de destí. Un cop feta aquesta instal·lació podem procedir a instal·lar el WinAVR, per tal de que reconegui el compilador de C de GNU.

Instal·lació WinAVR

Executem el fitxer *WinAVR-20090313-install.exe*. La instal·lació també és molt senzilla i només cal que ens preocupem de que totes les caselles que es veuen a la finestra de la figura inferior, estiguin marcades.



10.3.2 INSTAL·LACIÓ PONYPROG

Per instal·lar el software necessari pel programador PonyProg, cal que descarreguem, primer de tot, el fitxer m'instal·lació més recent que trovarem a l'adreça web http://www.lancos.com/ppwin95.htm. Executem el fitxer anomenat Setup.exe i ens limitem a seguir les instruccions.

10.3.3 INSTAL·LACIÓ ET-AVR JTAG

Primer descarregarem l'arxiu .zip corresponent al ET-AVR JTAG USB que trobarem a la pàgina http://www.etteam.com/download.html. Descomprimirem el fitxer en un directori local. Després connectem el cable usb al PC. Quan sigui reconegut ens apareixerà un missatge informant que s'ha trobat un nou dispositiu.

Posteriorment ens apareixerà una finestra que demanarà la instal·lació del software necessari per al dispositiu reconegut. En aquest punt especificarem nosaltres mateixos la ruta a on es troben els fitxer descomprimits anteriorment.

10.4 COSTOS DE LA PLACA D'EXPANSIÓ

El preu total mostrat és per una sola placa i per tant, aquests costos seran per cada unitat. Cal que afegim el preu de la fabricació de les plaques. En el meu cas plantejo la fabricació de 25 plaques, que és la tirada de robots plantejada. Això suposa un cost de 170 €, aproximadament uns 7 € per cada placa.

Component	Unitats	€/unitat	Total
L293D	1	2,56	2,56
Led	2	0,10	0,20
Led baix consum	10	0,24	2,4
Capacitat 22 uF	1	0,081	0,081
Capacitat 1 uF	7	0,049	0,343
Bloc terminal 2 pos	1	0,48	0,48
Connector alimentació	1	1,57	1,57
Connector alimentació	1	1,36	1,36
LM2940	1	1,19	1,19
MAX232CPE	1	1,19	1,19
Electret	1	0,41	0,41
BC547C	1	0,0238	0,0238
Diode	1	0,03	0,03
Sòcol 20 pins	2	0,073	0,146
Capacitat 100nF	2	0,022	0,044
Resistència 4,7Kohm	5	0,02	0,1
Resistència 100Kohm	1	0,02	0,02
Interruptor	1	1,37	1,37

Connector doble f 20 pins	3	2,44	4,88
Connector IDC JTAG	1	0,14	0,14
Connector cable pla 10 pins	1	0,134	0,134
Cable pla 20 cm	2	0,40	0,80
Connector AR cable pla 10 pins	1	0,96	0,12
Jumper	1	0,02	0,02
		TOTAL	**19,61**

10.5 DESPESES D'ENVIAMENT

Superrobòtica

Les despeses d'enviament pel distribuïdor Superrobotica són de 7 € més IVA si la compra és inferior a 150 €. Per a compres superiors a aquesta xifra, les despeses d'enviament són de franc.

Futurlec

L'empresa Futurlec ofereix diferents taxes en funció de la despesa de la comanda.

- Comandes de fins a 29 $, despeses d'enviament de 4 $.
- Comandes de 30 a 49 $, despeses d'enviament de 6 $.
- Comandes de 50 a 99 $, despeses d'enviament de 9 $.
- Comandes de més de 100 $, despeses d'enviament de 14 $.

Robotshop

L'empresa canadenca ofereix diferents possibilitats.

Per UPS:

- Expedited: entrega de 2 a 7 dies, a 64,85 $, és la que he tingut en compte en l'elaboració de l'anàlisi de costos.
- Express:: entrega de 1 a 4 dies, a 73,36 $.

Per Canada Post:

- Small Packets – Air: entrega abans de 2 setmanes, amb comanda assegurada si és inferior a 100$. El cost és de 15,96 $.
- Parcel Surface: entrega abans de 1 a 3 mesos, amb comanda assegurada si és inferior a 1000$. El cost és de 31,95 $.
- Xpresspost international: entrega abans de 12 dies, amb comanda assegurada si és inferior a 1000$. El cost és de 58,61 $.

Pololu

Pololu ofereix tres possibilitats d'entrega:

- USPS First Class Mail International, a 9,95 $, que és la opció triada durant l'elaboració de l'anàlisi de costos.
- USPS Priority Mail International, a 31,95 $.
- USPS Express Mail International, a 32,95 $.

RESUM

Aquest projecte presenta el disseny, construcció i programació d'un robot autònom, com a base per una proposta educativa. Per aconseguir aquest objectiu s'ha dotat el robot d'una unitat de procés, un sistema de locomoció i un seguit de sensors que proporcionaran a la unitat informació respecte l'entorn. Per gestionar totes aquestes funcionalitats, s'ha fet servir un sistema operatiu en temps real capaç de gestionar amb efectivitat les tasques que puguin ser executades pel robot. Finalment, s'ha exposat una detallada descripció dels costos per una producció de volum mig i de caire merament educatiu.

RESUMEN

Este proyecto presenta el diseño, construcción i programación de un robot autónomo, como base de una propuesta educativa. Para conseguir este fin se ha dotado al robot de una unidad de proceso, un sistema de locomoción y una serie de sensores que proporcionan a dicha unidad información sobre el entorno. Para gestionar toda esta funcionalidad, se ha utilizado un sistema operativo en tiempo real capaz de gestionar con efectividad las tareas que puedan ser ejecutadas por el robot. Finalmente, se ha expuesto una detallada descripción de los costes para una producción de volumen medio y de voluntad meramente educativa.

ABSTRACT

This project presents the design, development and implementation of an autonomous robot for an educational purpose. In order to achieve this goal, the robot has been embedded with a processing unit, a motion system and a set of sensors that provides information of the environment. To control all these functions, it has been used a real time operating system capable of accurately handling all the tasks that can be executed by the robot. Finally, it has been shown out a detailed description of the costs for a medium production of our educational purpose.