Escola Universitària
Politècnica de Mataró

Enginyeria Tècnica de Telecomunicació: Especialitat Telemàtica

Interacció persona-ordinador amb Arduino.

VOLUM

I

Adrià Capdevila
Pere Barberan i Pere Tusset

TARDOR 2009

“Dedicat a totes aquelles persones que m’han fet costat i han confiat en mi en tot moment”

Resum

L’objectiu del projecte és dissenyar, implementar i avaluar el funcionament d’un sistema de rehabilitació del caminar amb *feedback* multimèdia adaptat a persones discapacitades, principalment amb paràlisi cerebral. El projecte està dividit en dues parts: la primera part es centra en el disseny i la implementació del *hardware* i *software* necessaris per permetre la interacció entre la persona i l’ordinador que actuarà com a *feedback* multimèdia a través d’un projector i uns altaveus. La segona part es centra principalment en la creació de continguts multimèdia que proporcionin una interfície amigable a l’hora d’interactuar amb el sistema.

Resumen

El objetivo del proyecto es diseñar, implementar y evaluar el funcionamiento de un sistema de rehabilitación del caminar con un *feedback* multimedia adaptado a personas con discapacidades, principalmente con parálisis cerebral. El proyecto está dividido en dos partes: la primera parte se centra en el diseño y la implementación del hardware y el software necesario para permitir la interacción entre la persona y el ordenador que actuará como un *feedback* multimedia a través de un proyector y unos altavoces. La segunda parte se centra principalmente en la creación de contenidos multimedia que proporcionen una interficie amigable a la hora de interactuar con el sistema.

Abstract

The project aims to design, implement and evaluate the performance of a system of rehabilitation of walking with a multimedia feedback adapted for people with disabilities, primarily cerebral palsy. The project is divided into two parts: the first part focuses on the design and implementation of hardware and software needed to allow interaction between person and computer to act as a feedback through a multimedia projector and speakers. The second part focuses mainly on the creation of multimedia content to provide an interface friendly when interacting with the system.

Índex

Índex Figures

1. Introducció

Actualment estem en una societat en contínua evolució que cada cop més lluita per la igualtat i integració de les persones i per convertir el concepte “d’ igualtat dels éssershumans” en una realitat i no en una utopia o una promesa electoral com ha vingut sent temps enrere. Per això actualment va en augment el que s’anomena consciència social.

La consciència social es pot definir com el coneixement que una persona té sobre l'estat dels altres integrants de la seva comunitat. L'individu amb consciència social és, justament, conscient de com l'entorn pot afavorir o perjudicar el desenvolupament de les persones.

La consciència social suposa que l'home entén les necessitats del proïsme i pretén cooperar a través de diferents mecanismes socials, ja sigui a través de donacions econòmiques, duent a terme activitats de voluntariat o qualsevol altre tipus d’acció social en general.

Aquesta consciència social va en augment i cada cop és més valorada, per això cada cop són més les empreses i els empresaris que, ja sigui per la seva pròpia consciència o per donar-li un valor afegit a l’empresa, col·laboren amb accions o projectes socials per la integració de persones amb risc d’exclusió social.

Personalment afronto aquest projecte com un repte, però a l’hora també com una oportunitat, una oportunitat de col·laborar en una acció social i promoure un mica més aquesta consciència social.

1.1 Quina és la problemàtica de les persones discapacitades?

S'entén per persona discapacitada aquella que té impedida o entorpida alguna de les activitats quotidianes considerades normals, per alteració de les seves funcions intel·lectuals o físiques.

Per tant un dels problemes que es troben les persones discapacitades és la necessitat d'una ajuda extra , per dur a terme accions de la vida quotidiana (caminar, veure, etc).

Dintre del col·lectiu de persones discapacitades, es poden trobar diferents tipus de discapacitats, per això les necessitats dependran de la persona i del tipus de discapacitat que pateixi.

1.2 A quin col·lectiu va dirigit?

Dintre del col·lectiu de persones discapacitades aquest projecte va dirigit a totes aquelles que pateixen paràlisi cerebral. Per saber quina és la problemàtica d'aquest col·lectiu primer s'ha definir què és la paràlisi cerebral:

La paràlisi cerebral és un terme que engloba a totes les paràlisis no progressives degudes a una lesió cerebral permanent produïda abans, durant o després del naixement. Entre 0,1% i 0,2% dels nois pateixen alguna forma de paràlisi cerebral; en el cas de bebès prematurs o de baix pes, aquesta xifra augmenta a l'1%. La causa específica de la major part dels casos de paràlisi cerebral és desconeguda. La lesió cerebral pot produir-se abans, durant o al poc de temps del naixement. Els factors prenatals que s'han relacionat són les infeccions maternes (sobretot la rubèola), la radiació, l'anòxia (dèficit d'oxigen), la toxèmia i la diabetis materna. Les causes implicades en el moment del naixement són els parts traumàtics, l'anòxia, els parts prematurs i els parts múltiples (en aquest cas és el bebè nascut en últim lloc el que té més risc). El grup de causes post natals inclou les infeccions i els tumors cerebrals, els traumatismes cranials, l'anòxia i les lesions vasculars cerebrals.

Un nen amb paràlisi cerebral té dificultats per controlar els músculs del seu cos. Normalment el cervell diu a la resta del cos exactament què ha de fer i quan fer-ho. Però com la paràlisi cerebral afecta al cervell, depenent de la part del cervell afectada, el nen podria no poder caminar, parlar, menjar o jugar de la manera que ho fa la majoria dels nens.

La teràpia és important per a un nen amb paràlisi cerebral. Els nens amb paràlisi cerebral generalment necessiten fisioteràpia, teràpia ocupacional o de la parla per ajudar-los a desenvolupar habilitats com caminar, sentir, tragar i fer servir les mans.

1.3 Quina és la motivació per realitzar aquest projecte ?

La motivació principal a l'hora de realitzar aquest projecte, és la possibilitat que em brinda de veure i profunditzar en alguns dels conceptes que he vist al llarg de la carrera i dels qual sempre m'havia sentit atret.
Tot i que el fet de saber que realitzant aquest projecte estic aportant una petita ajuda a gent que ho necessita, va ser un incentiu més a l'hora d'escollir el projecte.

1.4 A qui va adreçat el projecte?

El projecte està adreçat a totes aquelles persones que pateixen una paràlisi cerebral que tenen afectada la regió del cervell la qual s'encarrega de controlar els músculs de la part inferior del tronc i per tant tenen dificultats per caminar.

1.5 Qui està implicat ?

"La Fundació privada El Maresme pro persones amb disminució psíquica" és una entitat d'iniciativa social sense afany de lucre que promou i impulsa la integració social i la millora de la qualitat de vida de les persones amb discapacitat intel·lectual de la comarca del Maresme i de les seves famílies.
La Fundació el Maresme va ser constituïda el 22 de desembre de 1994, continuant la tasca iniciada en l'any 1966 per l'Associació Patronat pro persones amb disminució psíquica del Maresme.
Com a centre comarcal de referència, atén i dóna resposta a qualsevol qüestió relacionada amb les persones amb disminució psíquica, ja sigui a nivell informatiu i d'orientació, o d'assistència o tractament.

La Fundació el Maresme procura donar resposta a les necessitats i demandes de la comarca, organitzant una xarxa de serveis que ofereixin una atenció en continuïtat a la persona amb disminució psíquica i la seva família. Per a accedir a un servei de la Fundació, cal que acrediti la seva condició de persona amb discapacitat que atorga el CAD (Centre d'Atenció al Disminuït), organisme de la Generalitat de Catalunya, que és qui atorga el certificat de disminuït.

1.6 Què es vol aconseguir?

Amb aquest projecte el que es vol aconseguir, és ajudar a les persones que pateixen una discapacitat que els afecta al caminar, aportant una nova eina que els faciliti la rehabilitació.

1.7 Com funciona el sistema ?

Tal com es mostra a la Figura 1, el sistema de rehabilitació del caminar amb *feedback* multimèdia està compost per tres elements principals: la placa emissora, la placa receptora i la unitat de *feedback*. A continuació es descriu cadascun dels blocs que formen els elements principals, així com la seva funcionalitat dins del sistema.

1.7.1 Placa emissora:

És la placa encarregada de detectar el caminar de la persona a través dels sensors de pressió, i transmetre les dades a la placa receptora mitjançant un enllaç sense fils. La placa està formada per quatre blocs principals: alimentació, microcontrolador, sensors i comunicacions. Es tracta d'un sistema autònom, és a dir, alimentat amb piles o bateries

1.7.2 Placa receptora:

És la placa encarregada de rebre i processar les dades provinents de la placa emissora i transmetre-les a la unitat de *feedback* a través del port sèrie per tal que siguin interpretades pel sistema de *feedback* multimèdia. La placa està formada per dos blocs principals: microcontrolador i comunicacions. Es tracta d'un sistema no autònom, és a dir, alimentat a través de l'energia subministrada pel propi ordinador.

1.7.3 Unitat de *feedback*:

La unitat de *feedback* està formada per un ordinador personal amb capacitat multimèdia connectat a un sistema audiovisual format per un projector i uns altaveus. L'ordinador executa un programari (Processing[1],Flash) capaç d'interpretar les dades provinents de la placa receptora a través del port sèrie per tal d'oferir *feedback* multimèdia a les persones discapacitades que interactuen amb el sistema, per exemple interactuar amb un conte multimèdia.

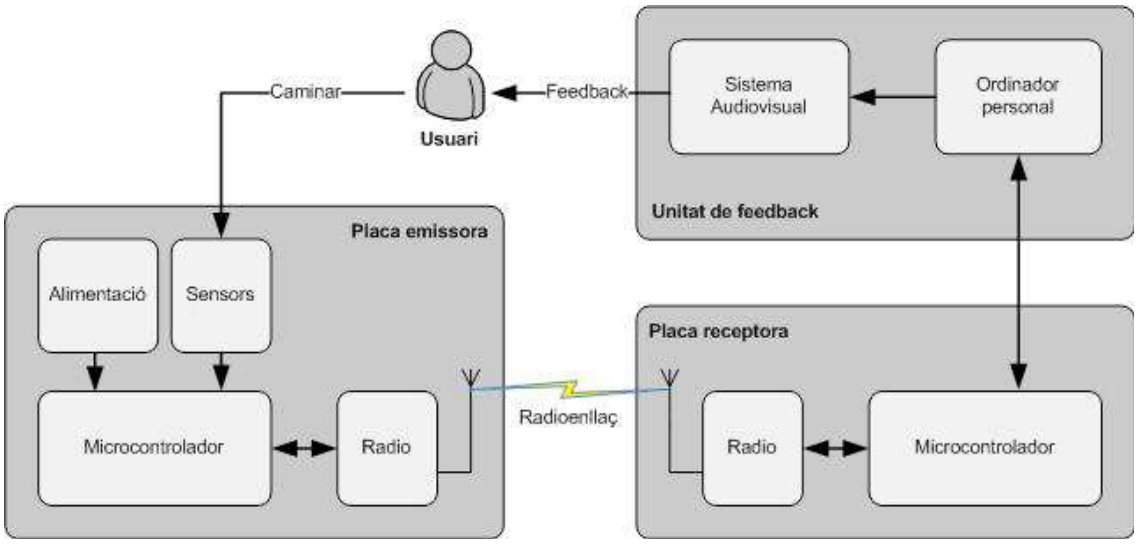


Fig. 1 Diagrama de blocs del sistema.

[1] *Veure Annex I*

1.8 Especificacions i requeriments inicials del disseny

- El sistema ha de ser capaç d'interactuar amb un mínim de 8 sensors (simulant 4 passos per cada cama) amb la resolució adequada en funció del requeriments establerts. D'una banda es pot censar únicament la presència o absència del peu (1bit) o d'altre banda, es poden monitoritzar paràmetres com la força exercida; en aquest cas farà falta una resolució major (per exemple 8 bits) depenent del paràmetre a monitoritzar.

- La placa emissora ha de tenir un disseny de baix consum, ha d'estar alimentada de manera autònoma, per exemple amb bateries o piles, permetent una autonomia de 24 hores de funcionament ininterromput. D'altre banda, la placa receptora ha d'estar alimentada a través del ordinador a la que va connectada i, per tant, l'únic requeriment es que tingui un disseny que en minimitzi el consum.

- La unitat de procés ha de permetre la comunicació amb el subsistema de comunicacions i amb l'ordinador a través del port sèrie. A més a més, la unitat de procés també ha de tenir en compte els requeriments de consum energètic.

- El subsistema de comunicació ha de permetre la transmissió de dades sense fils a la distància típica d'una habitació i a una velocitat de transmissió adequada per tal que el sistema pugui interactuar a temps real amb l'usuari en funció del volum de dades a transmetre, per exemple segons el tipus i quantitat de sensors. És important destacar que la comunicació només ha de ser unidireccional, de la placa emissora a la placa receptora, doncs a l'inrevés no cal transmetre dades.

1.9 Existeixen sistemes similars ?

En quant a sistemes similars, existeixen, però no amb la mateixa finalitat.

L'exemple més clar és l'estora de ball que es pot trobar en diferents plataformes de videojocs.

L'exemple més conegut és el DDR (Dance Dance Revolution) també conegut com a Dancing Stage.

DDR és una prolífica sèrie de videojocs musicals produïda per la companyia japonesa Konami. Dance Dance Revolution és la sèrie pionera del gènere de simuladors de ritme / ball en els videojocs. Els jugadors es col·loquen sobre una "plataforma de ball", també anomenada "pista" i pressionen amb els peus les fletxes disposades en forma de creu sobre aquesta pista, seguint el ritme de la música i el patró visual que apareix a la pantalla. D'acord amb la sincronització que els jugadors mostrin amb el ritme i lo bé que segueixin el patró de fletxes, se'ls atorga una puntuació (que varia d'acord a la versió del joc). El seu primer llançament, en forma d'àrcade, va ser realitzat al Japó el 1998. Des de la seva creació han estat produïdes múltiples variacions, incloent algunes per a ús casolà. És classificat com un joc Bemani (Bemani és una abreviació japonesa de la paraula Beatmania, el nom del primer joc musical de Konami).

Fig. 2 . Plataforma DDR

Actualment aquesta Konami té més de 40 títols diferents repartits entre les diferents plataformes de jocs.

2. Disseny i implementació del sistema

2.1 Disseny

El disseny del sistema està dividit en dos grans blocs, el disseny de la placa emissora i el disseny de la placa receptora, tot i que en les dues plaques el microcontrolador és el mateix, el propòsit és diferent i per tant una placa no tindrà les mateixes necessitats que l'altra.

2.1.1 Placa emissora:

És la placa encarregada de detectar el caminar de la persona a través dels sensors de moviment i transmetre les dades a la placa receptora mitjançant un enllaç sense fils. La placa està formada per quatre blocs principals: alimentació, microcontrolador, sensors i comunicacions. Es tracta d'un sistema autònom, és a dir, alimentat amb piles o bateries.

Microcontrolador:

- **Descripció:**

  Un microcontrolador és un circuit integrat o xip que inclou en el seu interior les tres unitats funcionals d'un ordinador: unitat central de processament, memòria i unitats d'E / S (entrada / sortida).

- **Necessitats:**

  El bloc emissor necessita un microcontrolador que disposi de diverses entrades analògiques que permeti la lectura dels sensor i un port sèrie que faciliti tant la programació del microcontrolador com la comunicació amb l'altre microcontrolador del bloc receptor.

- **Tecnologies**

Dintre de la família dels microcontroladors els fabricants més comuns i alguns dels seus microcontroladors són:

- AVR Amtel *ATmega16x*
- Freescale *68HC12, 68HCS12, 68HCSX12, 68HC16*
- Intel *MCS96, MXS296*
- Microchip *PIC24F, PIC24H i dsPIC30FX*

També es poden trobar projectes com Arduino, que és una plataforma de hardware lliure, que ja porta incorporat un microcontrolador Atmega, un xip senzill i de baix cost que permet el desenvolupament de milers de dissenys

- **Elecció i justificació:**

Finalment l'elecció ha estat la plataforma Arduino, ja que compleix tots els requeriments necessaris que ha de tenir el microcontrolador del bloc emissor i a l'hora proporciona un entorn de desenvolupament d'objectes interactius autònoms.

Alimentació

- **Necessitat:**

Al tractar-se d'un sistema autònom, és necessària una font d'alimentació externa. El microcontrolador i els amplificadors operacionals per funcionar adequadament necessiten una tensió de 5 volts i els sensors una tensió de -5 volts.

Per proporcionar aquests 5 volts al microcontrolador i als amplificadors operacionals s'haurà d'utilitzar un regulador de tensió, ja que no hi ha cap font d'alimentació al mercat que ens proporcioni aquests 5 volts necessaris.

Per alimentar els sensors, el sistema necessitarà un circuit inversor, per convertir els 5 volts positius de la sortida del regulador de tensió als 5 volts negatius necessaris.

- **Tecnologies:**

Per donar resposta a la necessitat d'una font d'alimentació, en el mercat existeixen diverses tecnologies que s'adeqüen a les necessitats del sistema. Les més interessants que calen destacar són les següents:

- Bateries

Nº cel·les:	Vàries
Recarregable:	Sí
Reciclable:	No
Tensió:	Varia depenent del format
Preu:	Varia depenent del format

- Piles AA

Nº cel·les:	Una
Recarregable:	No
Reciclable:	Sí
Tensió:	1,5 Volts
Preu:	~ 0,90€

- Piles AAA

Nº cel·les:	Una
Recarregable:	No
Reciclable:	Sí
Tensió:	1,5 Volts
Preu:	~ 0,75€

En quan a reguladors de tensió, existeixen també diverses tecnologies de les quals cal destacar dues :

- Reguladors de tensió lineals

  - LM7805C → National Semiconductor
  - L7805TT → STMicroelectronics
  - LM7805 → Fairchild Semiconductor

- Reguladors de tensió de bomba de càrrega o "charge pump"

  - LTC1516 → Micropower
  - MAX619 → MAXIM
  - TPS60110 → Texas Instruments

Per passar d'una tensió positiva a una tensió negativa, s'ha dut a terme una recerca exhaustiva d'un circuit inversor que s'adeqüés a les necessitats del sistema, tenint com a resultat els següents circuits:

- MAX660 →MAXIM
- MAX660 →National Semiconductor

- **Elecció i justificació:**

Les piles, que són no recarregables, són emprades amb freqüència, principalment els tipus AA, AAA i botó. Les piles alcalines ofereixen una alta densitat energètica a un preu força assequible. En canvi les bateries recarregables, a causa de la seva baixa densitat i alt cost econòmic són punts en contra que fa que es desestimi l'opció d'utilitzar–les.
Sembla clar que la millor elecció possible passa per escollir una pila no recarregable, de liti. El tipus seleccionat són les de tipus AA. S'han descartat les de botó i les AAA per la poca capacitat que ofereixen les solucions comercials presents al mercat.

En quant a reguladors de tensió, tenint en compte que els reguladors lineals al estar en sèrie amb càrrega, les caigudes de tensió en els seus components provoquen grans dissipacions de potència, per la qual cosa seria necessari un dissipador de calor i això fa que es descarti aquesta tecnologia.
Entre els reguladors de bomba de càrrega, comparant els datasheets, la circuiteria interna de tots tres models és la mateixa, l'única variació és el nom del model que varia per cada fabricant, per la qual cosa l'elecció ha estat purament econòmica (MAX619 de MAXIM)
L'elecció del circuit inversor no ha estat problema, ja que els únics circuits que s'han trobat adequats al sistema són el mateix però de fabricants diferents. L'elecció final ha estat pel MAX660 de la companyia MAXIM per una qüestió purament econòmica.

Sensors

- **Necessitats:**

  Sensar / detectar quan la persona està situada en una regió concreta del sistema.

- **Tecnologies.**

  Dintre de la família dels sensors per mesurar la pressió es poden trobar diferents tecnologies, de les quals cal destacar :

  - Piezoresistius:

Canvi en el potencial:	No
Canvi en la resistència:	Sí
Pes soportat:	Variable
Preu:	~13,5 €

  - Piezoelectrics:

Canvi en el potencial:	Sí
Canvi en la resistència:	No

Pes soportat: Variable depenent del sensor
Preu: ~90 €

- Capacitius:
  Canvi en el potencial: No
  Canvi en la resistència: Sí
  Pes soportat: Variable depenent del sensor
  Preu: ~57,5 €
  .

- **Elecció i justificació:**

Finalment s'ha decidit escollir un sensor de pressió de tipus piezoresistiu, ja que aquest tipus de sensors només causa canvis en la resistència i són més economics (això comporta una reducció del cost de fabricació).
S'ha escollit el sensor de la marca FLEXIFORCE que permet sensar fins un màxim de 50Kg

Comunicacions

- **Necessitat:**

Comunicació amb el bloc emissor, el mòdul de comunicació que s'utilitzi ha de ser capaç de permetre enviar un mínim de 12 bytes [1](8 bytes de dades dels sensors, més 4 bytes de capçalera) pel port sèrie.

- **Tecnologies disponibles:**

Algunes de les tecnologies sense fils i les seves característiques que es poden trobar al mercat:

- Bluetooth :
  Banda : 2,4GHz

Establiment de connexió previ:	Sí
Comunicació:	full-duplex
Modulació:	GFSK
Preu:	~33,5 €

- Wifi:

Banda:	2,4GHz 5GHz
Establiment de connexió previ:	Sí
Comunicació:	full-duplex
Modulació:	BPSK, QPSK, 16 QAM, 64 QAM
Preu:	~45 €

- zigBee:

Banda :	2,4GHz
Establiment de connexió previ:	Sí
Comunicació:	full-duplex
Modulació:	BPSK
Preu:	~17,5 €

- Radio:

Banda:	434MHz
Establiment de connexió previ:	No
Comunicació:	simplex
Modulació:	ASK
Preu:	~ 3,5 €

- **Elecció i justificació:**

Dintre del ventall de tecnologies disponibles de l'apartat anterior, s'ha decidit optar per la ràdio per diversos motius. A diferència de les altres tecnologies, la rèdio no necessita un establiment de connexió previ per

dur a terme la comunicació (facilita la connexió al no haver de buscar un client a qui connectar-se), no necessita que la comunicació sigui full-duplex, ja que només es dedicar a enviar dades i dintre del ventall de tecnologies és la més assequible (això comporta una reducció del cost de fabricació).

2.1.2 Placa receptora:

És la placa encarregada de rebre i processar les dades provinents de la placa emissora i transmetre-les a la unitat de *feedback* a través del port sèrie per tal que siguin interpretades pel sistema de *feedback* multimèdia. La placa està formada per dos blocs principals: microcontrolador i comunicacions. Es tracta d'un sistema no autònom, és a dir, alimentat a través de l'energia subministrada pel propi ordinador.

Microcontrolador

- **Necessitats:**
  El bloc receptor necessita un microcontrolador que disposi d'un parell de ports sèrie per rebre la informació del bloc emissor i l'altre port sèrie per passar-li la rebuda del bloc emissor al PC.

- **Elecció i justificació:**
  Finalment l'elecció ha estat la plataforma Arduino, ja que compleix tots els requeriments necessaris que ha de tenir el microcontrolador del bloc emissor i a l'hora proporciona una connexió sèrie al PC a través del port USB.

Alimentació

Donat que el bloc receptor es connecta al PC a través del port USB, aquest ja proporciona l'alimentació necessària (5 volts) al sistema per fer-lo funcionar.

Comunicacions

- **Necessitat:**

  En el cas del bloc receptor, la necessitat d'enviar les dades cap al PC ja està coberta gracies a que la plataforma Arduino proporciona un enllaç sèrie a través del port USB, en quant a la comunicació amb el bloc emissor i tenint en compte que la necessitat principal és rebre les dades i no enviar-les (ja que aquesta última ens la proporciona el port USB) es necessitarà un receptor, capaç de captar el senyal transmès pel bloc emissor.

- **Elecció i justificació**

  Al haver escollit un mòdul emissor de radio, en el bloc receptor i per tal de que la comunicació sigui possible entre tots dos blocs, s'ha escollit un mòdul receptor de radio amb les mateixes característiques que el mòdul emissor

2.2 Implementació del sistema

A l'hora d'implementar el sistema s'utilitzen dos llenguatges de programació diferents: l'Arduino per programar el microcontrolador de les plaques Arduino, tant del bloc emissor com del bloc receptor i Processing per implementar una solució visual per la unitat de *feedback.*

Els dos llenguatges són molt similars, ja que els dos tenen com a base el Java

2.2.1 Bloc emissor:

En la figura 5 s'observa el diagrama de flux amb el funcionament bàsic del bloc emissor. Quan algun dels sensors rep un estímul, el sensor tradueix l'estímul en un senyal analògic que s'envia cap a l'Arduino. Un cop l'Arduino rep l'estímul converteix el senyal analògic en un senyal digital. Abans d'enviar les lectures afegeix una capçalera que indicarà a la unitat de *feedback* on comença la trama.

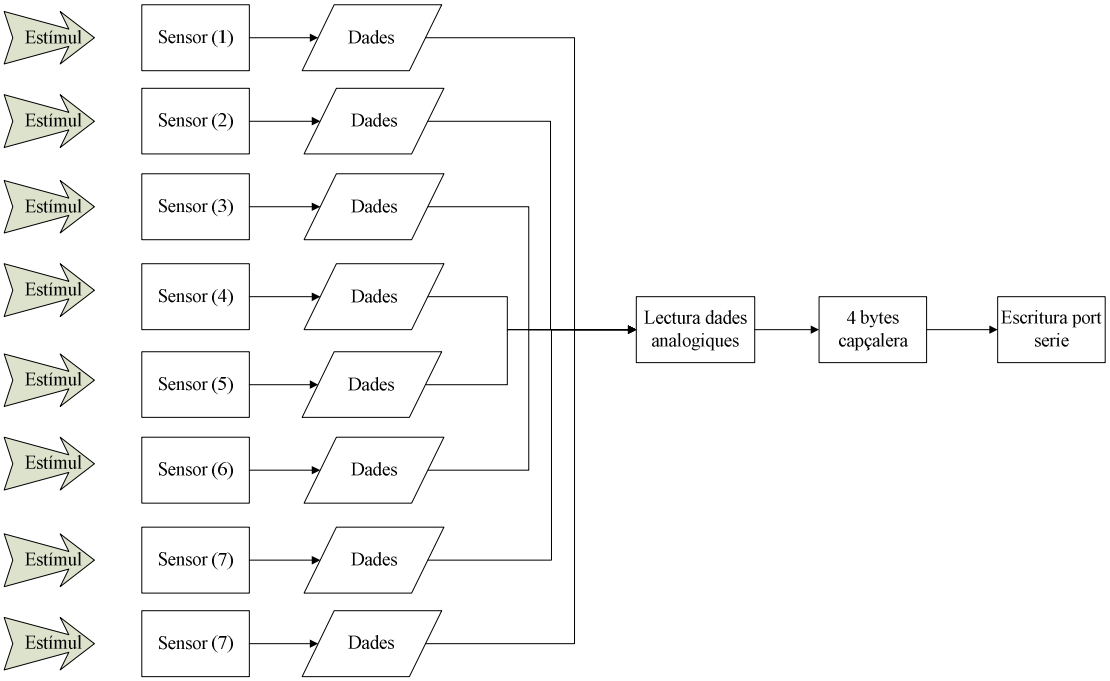


Fig. 3 Diagrama de flux del bloc emissor.

A l'hora dissenyar el circuit s'ha tingut en compte les especificacions del fabricant i el circuit recomanat en cadascun dels casos.

En el cas del fabricant dels sensors, recomanaven el següent circuit:

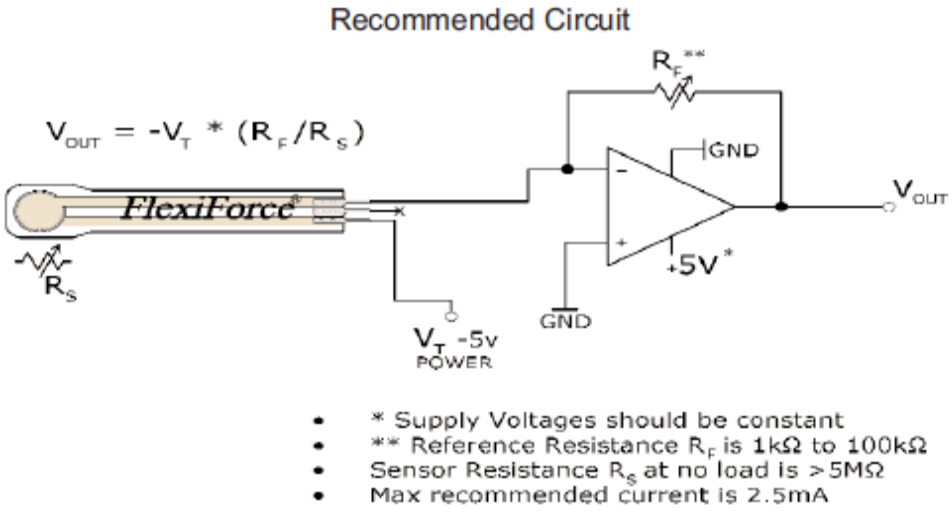


Fig. 4 Circuit recomanat per el fabricant del sensor

En el circuit mostrat, el rang de la força dinàmica del sensor pot ser ajustada al canviar la resistència de referència (RF).

Per aconseguir que sigui el màxim possible s'ha triat com a resistència de referència (RF), una resistència de 100KΩ

$$Vout = -Vt \times \frac{Rf}{Rs}$$

$$Vout = 5 \times \frac{100K\Omega}{Rs}$$

Per tractar d' atenuar el soroll elèctric produït per l'amplificador, s'ha afegit un condensador en paral·lel a la resistència de referència, produint així l'efecte d'un filtre passa baixes.

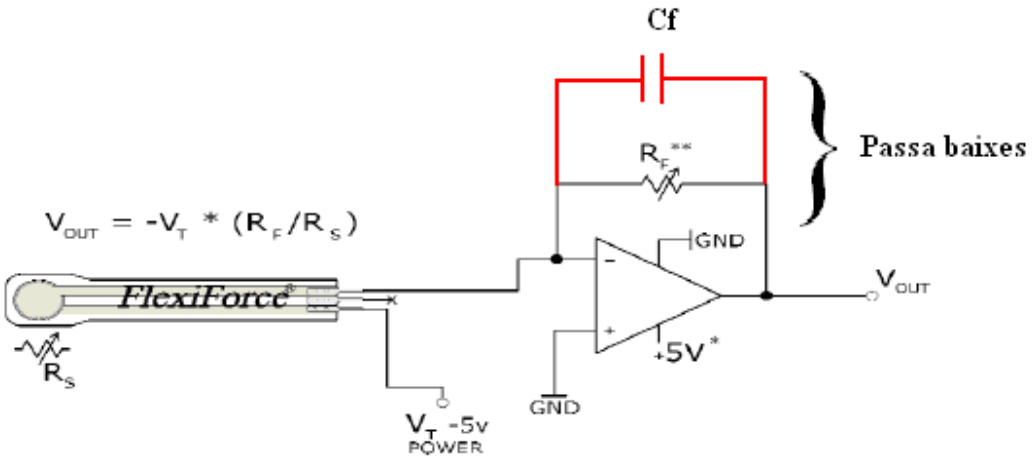


Fig. 5 Circuit amb el soroll filtrat.

$$Fc = \frac{1}{2 \times \pi \times R \times C} \rightarrow C = \frac{1}{2 \times \pi \times R \times Fc}$$

$$C = \frac{1}{2 \times \pi \times 100k\Omega \times 25} = 63,6nF$$

Donat que en el mercat no existeixen condensadors de 63,6nF, s'ha optat per escollir un condensador de 47nF

$$\tau = R \times C$$

Ideal (teòrica)

$$\tau = R \times C \rightarrow 100K\Omega \times 63,6nF$$

Escollida

$$\tau = R \times C \rightarrow 100K\Omega \times 47nF$$

A continuació s'ha calculat la freqüència de mostratge i el tamany que han de tenir les trames que s'enviin pel port sèrie:

A l'hora d'escollir la freqüència de mostreig s'ha de tenir en compte d'agafar una freqüència suficientment ràpida, que sigui capaç de detectar i distingir el caminar d'una persona. Aproximadament una persona pot fer al voltant d'unes 8 passes/segon, anant a una velocitat ràpida. Aplicant el teorema de Nyquist tenim que la freqüència de mostratge haurà de ser:

$$Fm \geq 2 \times fo \quad \rightarrow \quad Fm \geq 2 \times 8$$

$$Fm \geq 16 \frac{mostres}{segon}$$

Amb la finalitat d'afegir més precisió, aprofitar i evitar enviar més quantitat de Bytes redundants per l'enllaç, s'ha sobredimensionat la freqüència de mostratge a 25mostres/segon.

Sabent que l'antena emissora del sistema és capaç d'enviar dades a 2400bps o 4800bps i tenint en compte la quantitat d'informació que s'ha d'enviar, s'ha arribat a la següent conclusió:
Amb la finalitat d'enviar menys bytes de redundància s'ha escollit la velocitat d'enllaç a 2400bps.
Tenint una velocitat de l'enllaç de 2400bps es poden enviar un màxim de 300 bytes/segon

$$Ve = \frac{2400}{8} = \frac{300Bytes}{s}$$

Sabent que s'està mostrejant a 25 passos/segon i que el sistema està format per 8 sensors s'obté la quantitat de dades que s'enviarà per segon.

$$Vx = \frac{25mostres}{segon} \times 8\ sensors = 200 \frac{Bytes}{s}$$

Per portar un control i saber el inici de la transmissió, s'han afegit 4 bytes de capçalera, essent aquests 4 bytes el màxim de bytes de redundància que pot permetre el sistema amb la velocitat de l'enllaç de 2400bp.

$$Vx = \frac{25 mostres}{segon} \times (8\ sensors + 4\ capçalera) = 300 \frac{Bytes}{s}$$

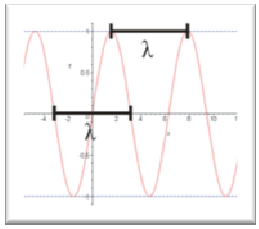
Longitud de l'antena :

A l'hora de triar l'antena pel sistema de radio s'ha de tenir en compte la longitud òptima que haurà de tenir per que el sistema funcioni correctament.
Tenim que la longitud òptima pel cable d'antena es troba de la següent forma:

$$l = \frac{\lambda}{2}$$

On λ representa la longitud d'ona que es pot trobar fàcilment si sabem la freqüència i la velocitat de propagació que en aquest cas serà 3.10^8 (velocitat de propagació en el buit)

$$\lambda = \frac{c}{v}$$



Per tant la longitud òptima de l'antena és de :

$$\lambda = \frac{3.10^8}{434.10^6} = 0{,}969 \rightarrow \quad l = \frac{0.69}{2} = 0{,}34\ metres$$

D'una manera molt similar es calcula la distància mínima que ha d'haver-hi entre el bloc emissor i el bloc receptor per que el funcionament sigui òptim.

$$d = 3 \times \lambda$$

$$d = \ 3 \times 0.69 = 2{,}07\ metres$$

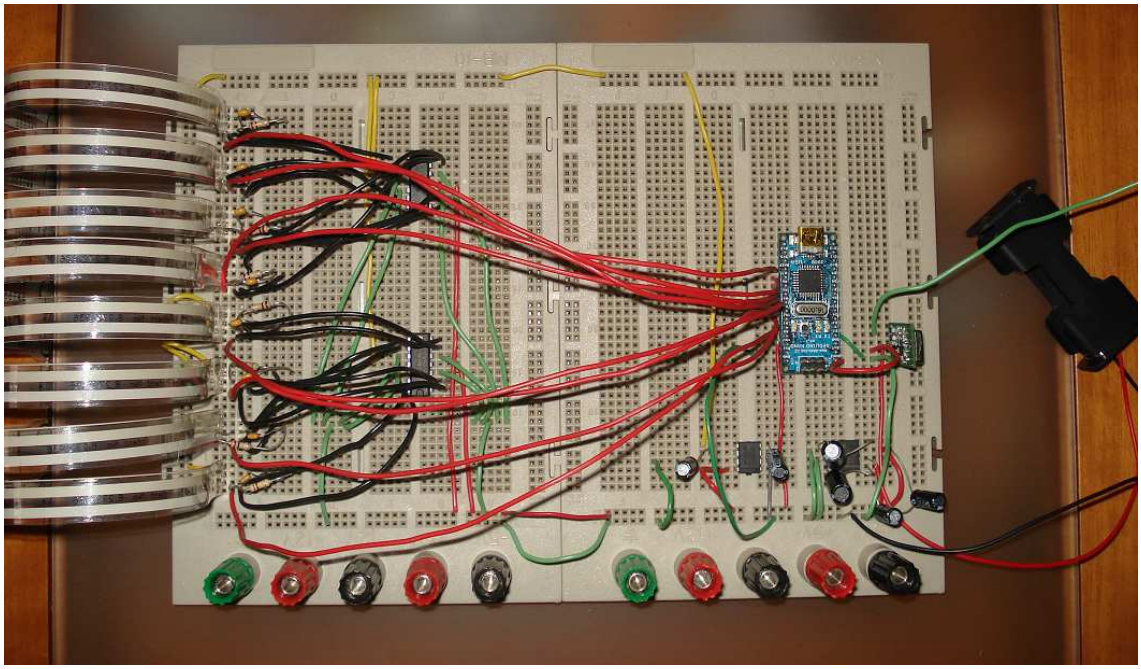


Fig. 6 Foto del bloc emissor.

2.2.2 Bloc receptor

Aquest bloc serveix de passarel·la entre el bloc emissor i la unitat de *feedback*.
Per que la comunicació entre el bloc emissor i el bloc receptor sigui satisfactòria, s'ha d'escollir, la mateixa velocitat d'enllaç, 2400bps .

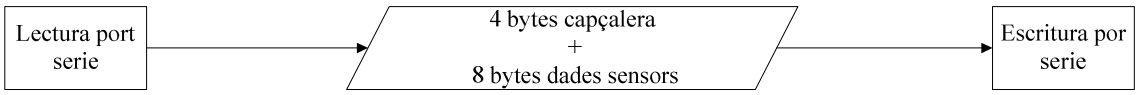


Fig. 7 Diagrama de flux del bloc receptor.

En aquest bloc, igual que en el bloc emissor caldrà tenir en compte la longitud de l'antena.

$$\lambda = \frac{3.10^8}{434.10^6} = 0{,}969 \rightarrow \quad l = \frac{0.69}{2} = 0{,}34 \ metres$$

Al treballar en la mateixa freqüència, la longitud de l'antena és exactament igual que per l'antena del bloc emissor.

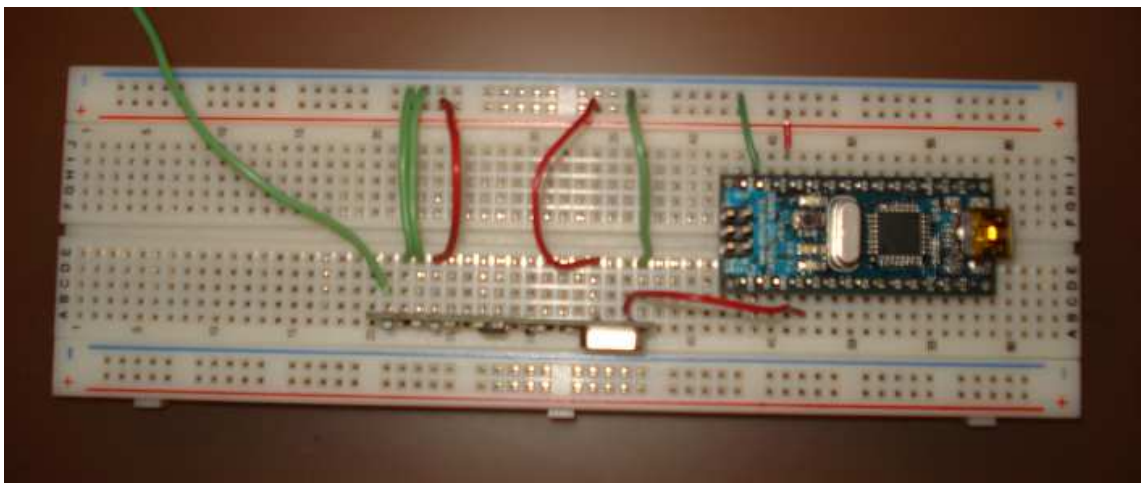


Fig. 8 Foto del bloc receptor.

2.2.3 Unitat de *feedback*

La unitat de *feedback* és l'encarregada d'interpretar les dades que arriben pel port sèrie, procedents del bloc receptor.

És en la unitat de *feedback* on s'hauran d'implementar els futurs continguts que pugui tenir el sistema.

En la següent figura es pot observar un exemple de programa, que es pot implementar fàcilment amb Processing que mostra el funcionament bàsic del sistema.

Es tracta d'un programa que processa les dades que li arriben pel port sèrie, quan detecta la capçalera (definida prèviament per l'usuari/programador) comença assignar les dades de cadascun dels 8 sensors a una variable diferent.

Per pantalla s'imprimeixen 8 requadres dividits en files de 4 requadres cadascuna.

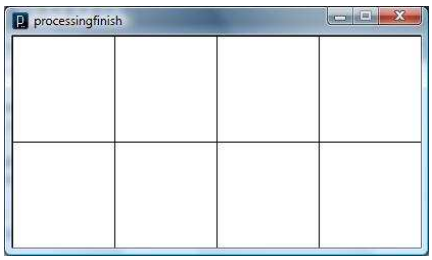


Fig. 9 Captura impressió per pantalla,sistema en estat inicial.

Les cadascuna de les 8 variables corresponents al sensors, llegides pel port sèrie estan lligades a una posició, de manera que cada sensor tindrà assignada una posició en el requadre que surt imprès per pantalla.

Inicialment el color d'aquest requadres es troba de color blanc, però a mesura que s'apliqui una força en un sensor, els colors del requadre aniran canviant depenent de la força que s'apliqui, imprimint el color verd la menor força, vermell la major i taronja una força intermitja.

Fig. 10 Captura impressió per pantalla,sistema en funcionament.

A continuació es pot veure el diagrama de flux del programa:

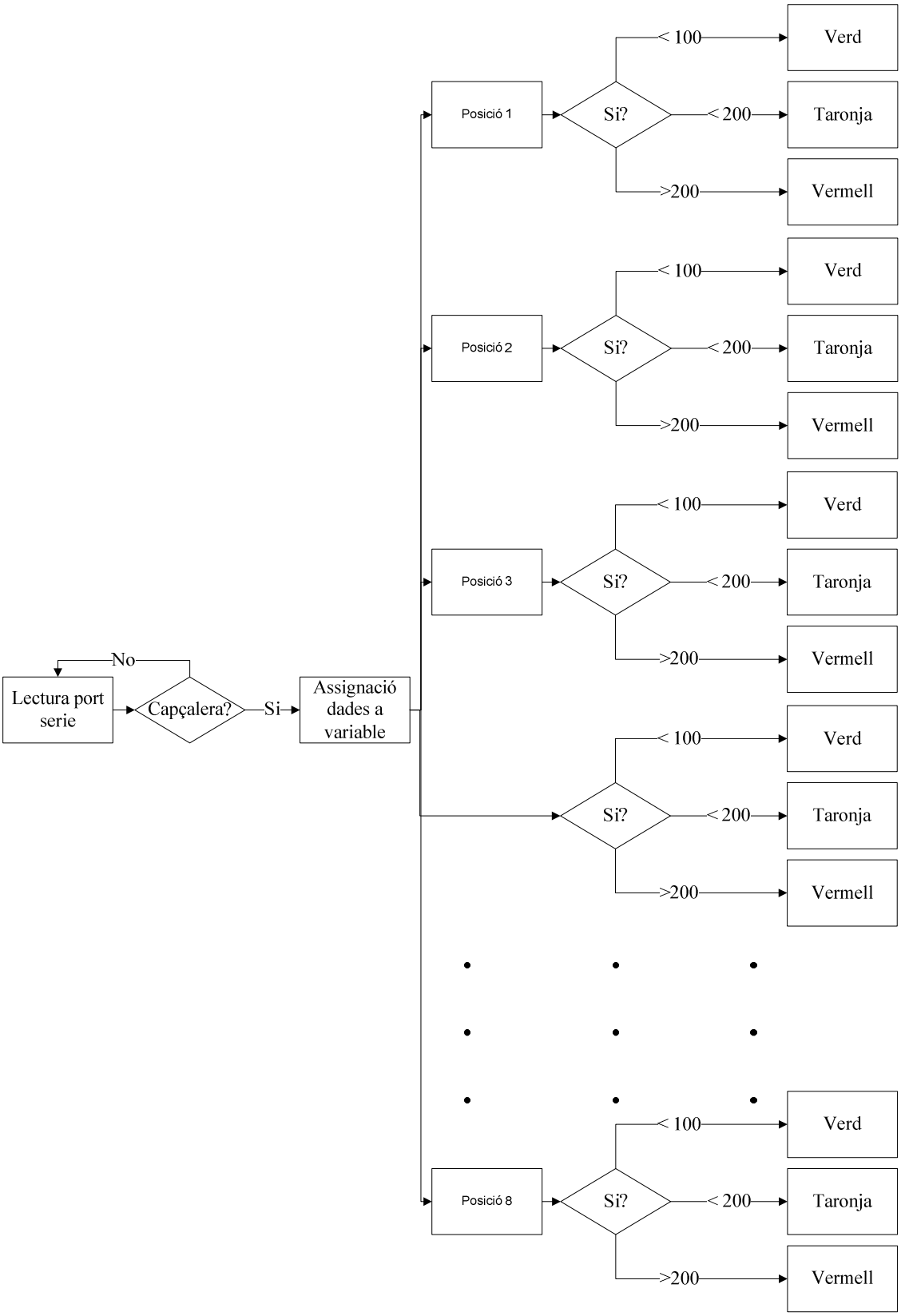


Fig. 11 Diagrama de flux del program

3. Anàlisi funcional i de prestacions

3.1 Anàlisi funcional:

3.1.1 Bloc emissor:

Comprovació del correcte funcionament dels sensors un a un. Per dur a terme aquesta comprovació s'ha portat a terme una petita modificació al codi font.
La modificació consisteix en canviar el *write* de la sortida sèrie per un *println*, això provocarà que en comptes d'imprimir els valors dels sensor per el port sèrie els imprimeixi per pantalla, permetent comprovar si els valors obtinguts dels sensors són els correctes

Serial.write(sensorX); → *Serial.println*(sensorX); on X és el número de sensor.

Sensor 1

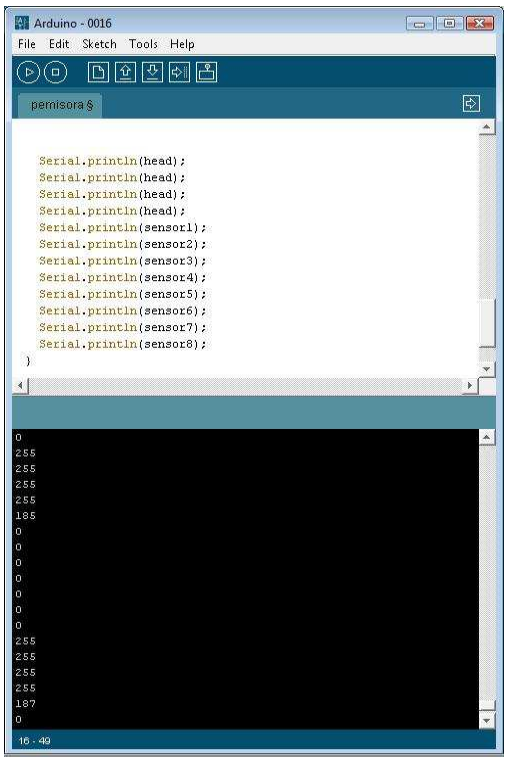


Fig. 12 Fig del Sensor 1

Aquesta prova s'ha realitzat sensor a sensor comprovant el correcte funcionament de cada un d'ells.

3.1.2 Bloc receptor

En el cas del bloc receptor per dur a terme la comprovació del seu correcte funcionament, s'ha observat els valors que escrivia pel port sèrie.

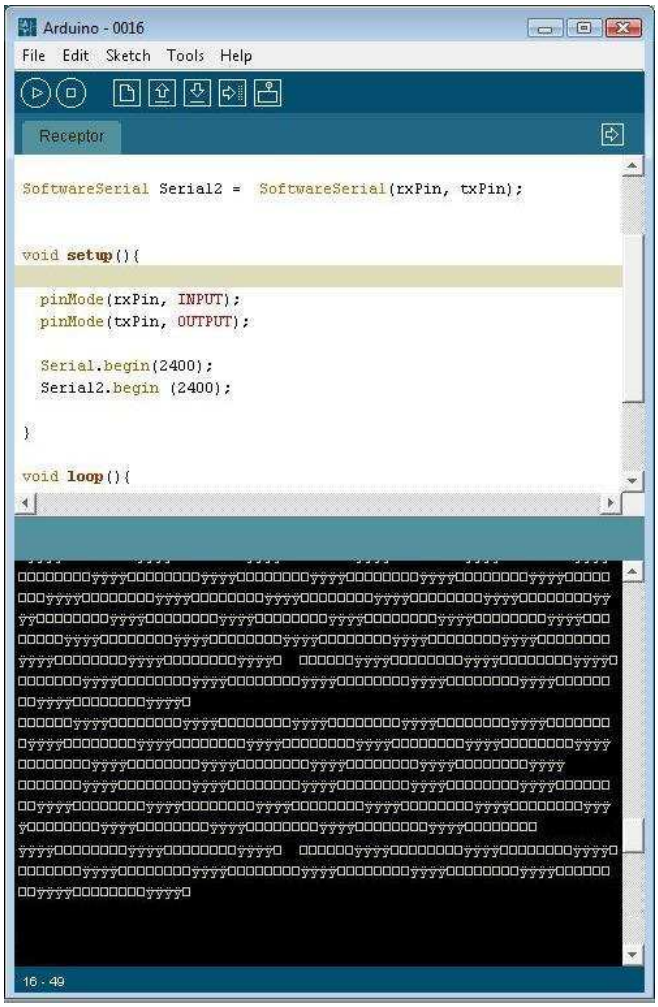


Fig. 13 Comprovació transmissió sèrie bloc receptor

En la figura 18 es pot identificar una seqüència que es repeteix periòdicament de 4 Bytes iguals, seguit de 8 Bytes més. Aquests Bytes són les dades enviades pel bloc emissor, però codificades en ASCII, que és el llenguatge que utilitza Arduino per enviar dades pel port sèrie.

En aquesta seqüència el bloc de 4 Bytes iguals es correspon amb la capçalera (255) i els 8 Bytes que segueixen es corresponen amb els 8 Bytes dels sensors (un per cada sensor).

3.1.3 Unitat de *feedback*

Per finalitzar la comprovació del correcte funcionament del sistema, s'ha de comprovar que les dades rebudes són les correctes, és a dir, que les dades que es llegeixen pel port sèrie són les mateixes que envia el bloc emissor, per això s'ha implementat un petit programa de prova que llista per pantalla les dades rebudes del bloc receptor.

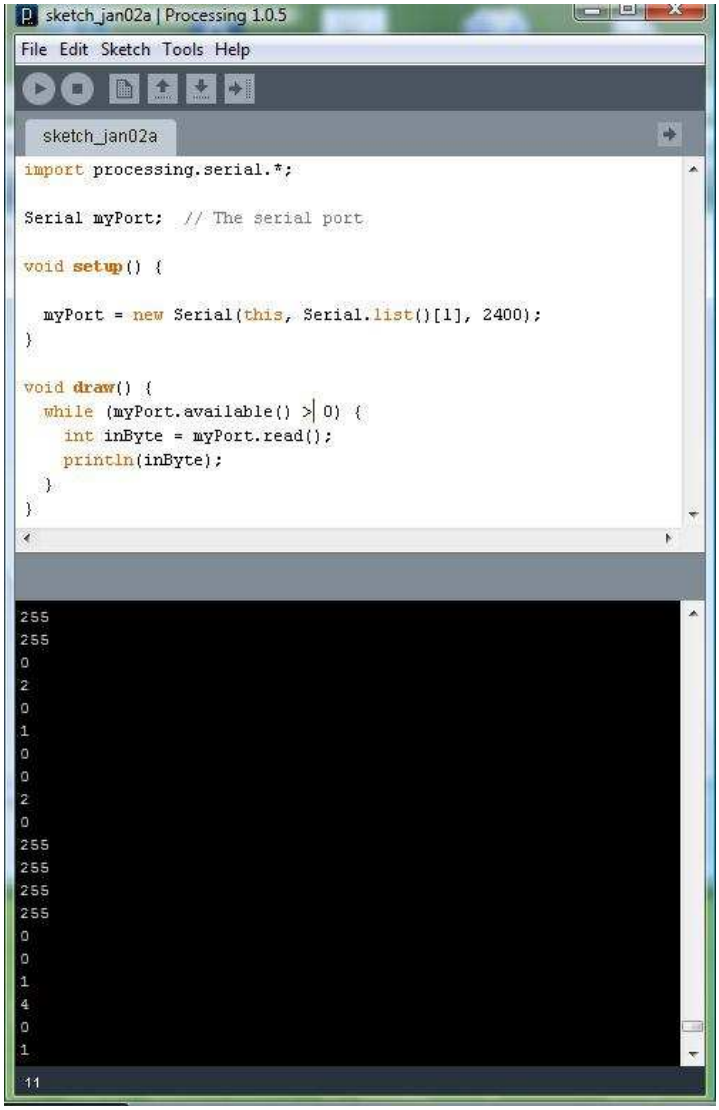


Fig. 14 Dades rebudes a la unitat de feedback.

En la Figura 19 s'observa clarament com la seqüència rebuda és la mateixa que la seqüència que envia el bloc emissor, es poden observar els 4 bytes de capçalera i els 8 zeros corresponents als 8 sensors.

Al fer servir una variable de tipus enter, es perd resolució ja que el valor màxim que es podrà obtenir en la lectura del port sèrie serà 255 en comptes de 550 i fa que les dades rebudes es tornin a veure en format digital.

3.2 Anàlisi de prestacions:

3.2.1 Consum i duració de les bateries

En aquest punt es tracta el consum del bloc emissor. El bloc receptor no es considera ja que va alimentat directament al PC via USB.
El bloc emissor és autònom i s'alimenta amb piles AA , per tant és important definir i saber el seu consum. A continuació se'n fa un anàlisi, primer teòric i després pràctic.

Anàlisi teòric del consum del bloc emissor:

Consum teòric bloc emissor		
Arduino	40	mA
MAX660	0,12	mA
MAX619	0,15	mA
TLC2264N	5	mA
TLC2264N	5	mA
Radio	2,3	mA
Total	52,57	mA

Anàlisi pràctic del consum del bloc emissor:

Consum pràctic bloc emissor		
Arduino	42,1	mA
MAX660	0,15	mA
MAX619	0,16	mA
TLC2264N	4,85	mA
TLC2264N	5,09	mA
Radio	2,5	mA
Total	54,85	mA

Tenint en compte que les piles proporcionen un corrent de 1500mAh, es pot saber aproximadament la duració del sistema en funcionament.

$$duració\ en\ funcionament = \frac{1500mAh}{54,85mA} = 27,34 \approx 27\ hores$$

3.2.2 Cobertura

En telecomunicacions, el terme cobertura es refereix a l'àrea geogràfica que cobreix una estació (emissora o receptora) específica.
La cobertura sol dividir-se en exterior o interior i de veu o de dades, en aquest cas es tracta de mesurar l'àrea cobertura en un entorn interior, enviant un flux constant de dades.
S'ha escollit una distància màxima de 4 metres entre l'emissor i el receptor, que és aproximadament el doble de la distància mínima calculada pel correcte funcionament del sistema. D'altra banda 4 metres ja és una distància considerable a l'hora d'utilitzar el sistema.

En la següent figura es pot observar un planell d'una habitació, amb les àrees de cobertura diferenciades per colors:

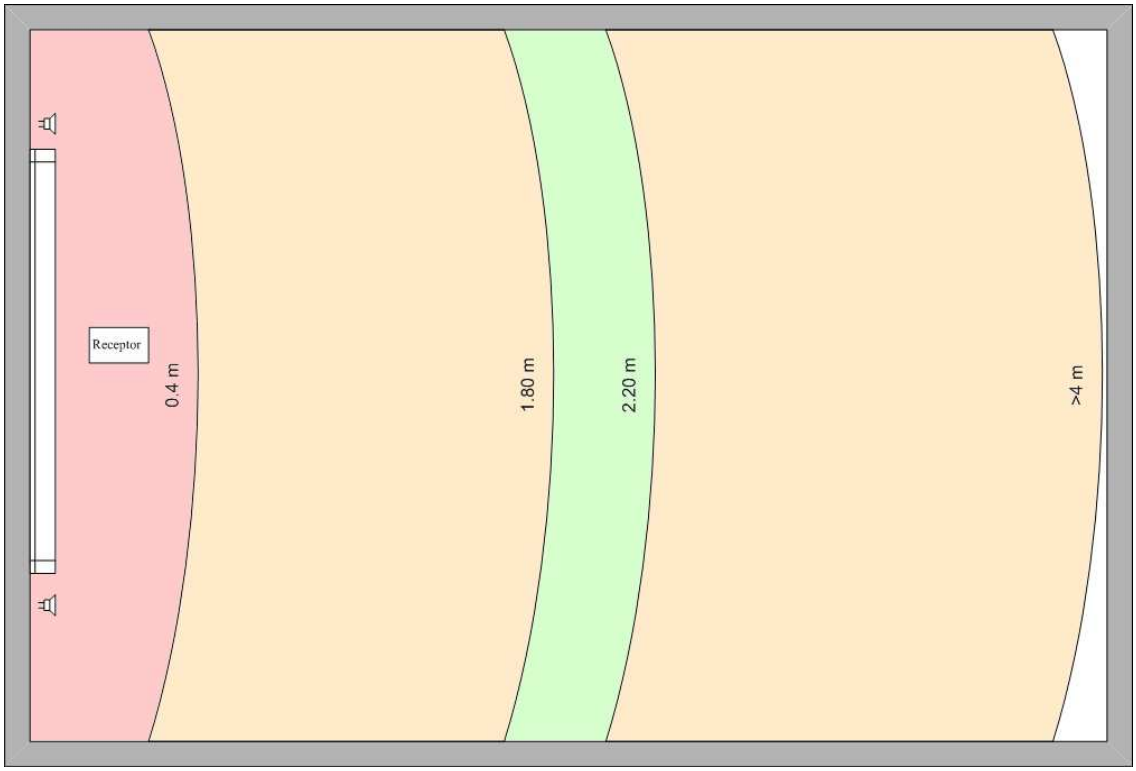


Fig. 15 Àrees de cobertura.

Àrea vermella : Les dades que arriben al receptor són errònies
Àrea taronja: Les dades arriben sense cap error al receptor

Àrea verda:	Distància mínima, calculada, per que el sistema funcioni adequadament

4. Anàlisi d'impacte mediambiental

Complint amb la regulació de la Generalitat de Catalunya sobre l'impacte al medi ambient, redactat al DECRET 114/1988, del 7 d'abril s'elabora el següent informe.

1. Anàlisi detallada de l'indret on es preveu l'obra, l'activitat o la instal·lació, i del seu entorn.

   Aquest projecte està dissenyat principalment per ser utilitzat en un entorn local, com podria ser una habitació amb el suficient espai per poder utilitzar el sistema tenint com a mínim 1,8 metres de separació entre el bloc emissor i el bloc receptor.

2. Descripció general de projecte i exigències previsibles en relació amb la utilització del sòl i d'altres recursos naturals durant les fases de construcció i de funcionament. Estimació dels tipus i quantitat de residus i emissions de matèria o energia resultants.

   El projecte al estar dissenyat per a ser utilitzat en un entorn local, l'impacte que provoca sobre el sòl i altres recursos naturals és mínim, en quant a l'estimació del tipus i quantitat de residus i emissions de matèria o energia resultants, s'ha procurat utilitzar piles amb un 0% de Mercuri i Cadmi per tal de que es puguin reciclar.
3. Avaluació dels efectes previsibles directes i indirectes del projecte sobre la població, la gea, el sòl, la flora i la vegetació, la fauna, l'aire, l'aigua, els factors climàtics, el paisatge i els béns materials, inclòs el patrimoni historicoartístic i arqueològic.

   Com el projecte està dissenyat per ser utilitzat en un entorn local, els efectes ja siguin directes o indirectes són sobre el col·lectiu de la població que pateix paràlisi cerebral. En quant a la gea, el sòl, la flora i la vegetació, la fauna, l'aire,

l'aigua, els factors climàtics, el paisatge i els béns materials incloent el patrimoni historicoartístic i arqueològic els efectes són nuls.

4. Relació detallada i valoració econòmica de les mesures previstes per eliminar, reduir o compensar els efectes ambientals negatius significatius.

   Al ser la major part del material, reciclable o reutilitzable és difícil fer una valoració econòmica, ja que en un principi la principal mesura serà utilitzar les papereres reciclables, per a reciclar les piles.

5. Resum de l'estudi i conclusions, formulats en termes comprensibles fàcilment, i informe, si s'escau, de les dificultats informatives i tècniques trobades en la seva elaboració.

   El sistema està dissenyat principalment per ser utilitzat en un entorn local per la qual cosa gairebé no causa cap impacte a l'exterior, tenint en compte que la majoria de material és reutilitzable o reciclable disminueix dràsticament l'impacte ambiental així com els costos de les mesures previstes per l'eliminació de residus.

6. Programa de vigilància ambiental on es concretin de manera detallada els paràmetres de seguiment de la qualitat dels vectors ambientals afectats, així com els sistemes de mesura i control d'aquests paràmetres.

   L'única vigilància que s'ha de tenir en compte, per el manteniment de l'impacte mediambiental del sistema, és l'ús de piles reciclables amb un percentatge reduït de Candi i Mercuri (ha poder ser un 0% com les utilitzades)

5. Conclusions i treball de futur:

5.1 Conclusions:

L'objectiu d'aquest treball ha estat dissenyar i construir un sistema per ajudar a la rehabilitació de persones amb paràlisi cerebral, compost per un bloc emissor autònom, un bloc receptor i una unitat de *feedback*. El projecte suposa un punt de partida de cara a implementar nous continguts per la unitat de *feedback* (ja sigui en Flash, Processing o Java) .

Les especificacions inicials requerien que el sistema havia de ser capaç d'interactuar amb un mínim de 8 sensors deixant a lliure elecció la resolució dels paràmetres a monitoritzar, el bloc emissor havia de ser totalment autònom donant la llibertat d'utilitzar les piles o bateries tenint en compte que el sistema havia de ser capaç d'estar un mínim de 24 – 48 hores en funcionament.

Els sensors utilitzats han estat piezoresistius, ja que eren els que s'adequaven més a la finalitat del sistema. El mètode de mesura s'ha basat en una interfície directa sensor-microcontrolador i la comunicació sense fils s'ha implementat utilitzant mòduls de Ràdio utilitzant una velocitat d'enllaç de 2400bps. En quant al microcontrolador escollit ha estat l' Atmega d'Amtel ja que és el que utilitza la placa del projecte Arduino. Per l'alimentació s'ha optat per fer servir dues piles AA reciclables d'una marca blanca.

El tractament de les dades s'ha realitzat amb un PC, utilitzant el llenguatge de programació Processing com a base.

S'han realitzat diverses proves per calibrar[1] comprovar el correcte funcionament del sistema. S'ha analitzat que les dades enviades i les dades rebudes es corresponguessin tot i que en alguns dels casos s'ha observat algun petit error a l'hora de rebre les dades, quan el sensor està inactiu, aquest petit problema s'ha solucionat gràcies al calibratge dels sensors.

[1] *Veure Annex V*

5.2 Treball futur:

En el treball, s'ha tractat principalment el disseny i construcció del hardware, en quant a continguts només s'ha realitzat un exemple per comprovar el correcte funcionament del sistema. En el futur caldrà crear nous continguts ja sigui en Flash, Processing o Java pel sistema.

6. Bibliografia

Referències bibliogràfiques:

[1] Dan O'sullivan i Tom Igoe. *Phisical Computing.* COURSE TECHNOLOGY.

[2] Pallas Areny, Ramon. *Sensores i acondicionadores de señal.* Alfa Omega

[3] Faúndez Zanuy, Marcos. *Sistemas de comunicaciones.* Marcombo

Referències web:

[4] http://www.fundmaresme.com/, *Fundació maresme*

[5] http://www.alldatasheet.com/, *Datasheets dels components*

[6] http://www.sparkfun.com/commerce/categories.php, *Botiga i blog on-line*

[7] http://arduino.cc/en/Reference/HomePage, *Referències del llenguatge Arduino*

[8] http://processing.org/reference/ *Referències del llenguatge Processing*

[9] http://www.tucamon.es/contenido/processing-y-arduino *Web d'interaccio d'Arduino amb Processing*

[10] http://en.wikipedia.org/wiki/Battery_(electricity), *Article sobre les bateries*

7. Glossari:

[1] Piezoresistiu: L'efecte piezo resistiu descriu canvis en la resistència elèctrica d'un material, després d'aplicar estres mecanic.

[2] Piezoelectric: és un fenomen presentat per determinats vidres que en ser sotmesos a tensions mecàniques adquireixen una polarització elèctrica en la seva massa, apareixent una diferència de potencial i càrregues elèctriques en la seva superfície.

[3] GFSK: modulació per desplaçament de freqüència, és un tipus de modulació digital similar a la FM analògica on es modifica la reqüència del senyal portador entre un número de valors discrets.

[4] BPSK: La modulació per desplaçament de fase o PSK (és una forma de modulació angular que consisteix en fer variar la fase de la senyal portadora entre un nombre de valors discrets.

[5] QPSK: (codificació per canvi de fase en quadratura), és una particularització de la modulació digital per desplaçament de fase o PSK

[6] QAM : La modulació QAM és un esquema de modulació multi-nivell, on s'envia informació canviant (modulant) l'amplitud de dos senyals portadors.

[7] ASK: La modulació per desplaçament d'amplitud, és una forma de modulació en la qual es representen les dades digitals com variacions d'amplitud de l'ona portadora.

[8] ASSCI: és un joc de caràcters que assigna valors numèrics a les lletres, xifres i signes de puntuació.

Annex I

Projecte Arduino:

1. Que és Arduino?

Arduino és una plataforma de hardware lliure basada en una senzilla placa d'entrades i sortides i un entorn de desenvolupament que implementa el llenguatge de programació Processing/Writing.
El seu cor és un xip Atmega, un xip senzill i de baix cost que permet el desenvolupament de milers de dissenys.
Al ser de hardware lliure tant el seu disseny com la seva distribució són lliures. És a dir, pot fer-se servir lliurement per desenvolupar qualsevol tipus de projecte, sense tenir d'adquirir cap mena de llicència.

2. Per a què puc utilitzar Arduino?

Arduino pot utilitzar-se en el desenvolupament d'objectes interactius autònoms o pot connectar-se al PC a través del port sèrie o d'un connector USB, utilitzant llenguatges como Flash, Processing, MaxMSP, etc ... Les possibilitats de realitzar i desenvolupar projectes basats en Arduino tenen com a límit la imaginació.
Així mateix, la seva senzillesa i el seu baix cost, recomanen el seu ús com a element d'aprenentatge i iniciaciació al món de l'electrònica digital.

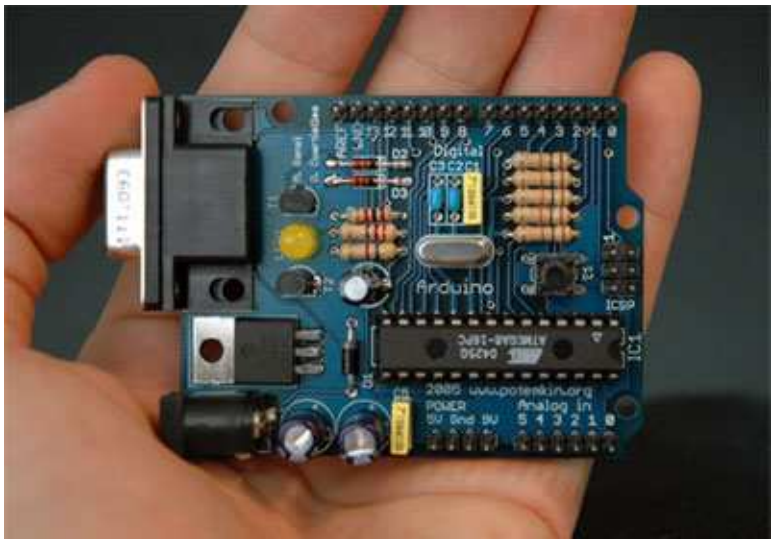
3. Quines versions d'Arduino existeixen?

Hi ha diverses versions de Arduino. En funció al nostre interès per la placa haurem de triar la que més s'adapti al nostre projecte:

a. Placa sèrie

És la placa bàsica, i utilitza una interfície RS232. Aquesta interfície pot ser utilitzada, a més de per a la programació i la placa, per comunicar-se amb

altres elements externs que utilitzin el port sèrie com per exemple un PC. Aquesta placa és molt senzilla d'acoblar. El seu muntatge pot ser fins i tot un exercici pràctic que ens ensenyi a donar els primers passos en el món de la electrònica digital.



b. Placa sèrie de una capa

Aquesta placa és una versió de majors dimensions de la placa sèrie. Aquesta dirigida a aquelles persones que tenen dificultat per accedir a comprar una placa sèrie amb el circuit imprès. Amb l'esquema i els fitxers CAD que es proporciona a la web del projecte es pot muntar Arduino amb una placa PCB i els components corresponents.

c. Placa USB

És una evolució de la placa sèrie que incorpora un port USB per comunicar-se amb el PC. És més difícil d'acoblar perquè inclou un xip SMD

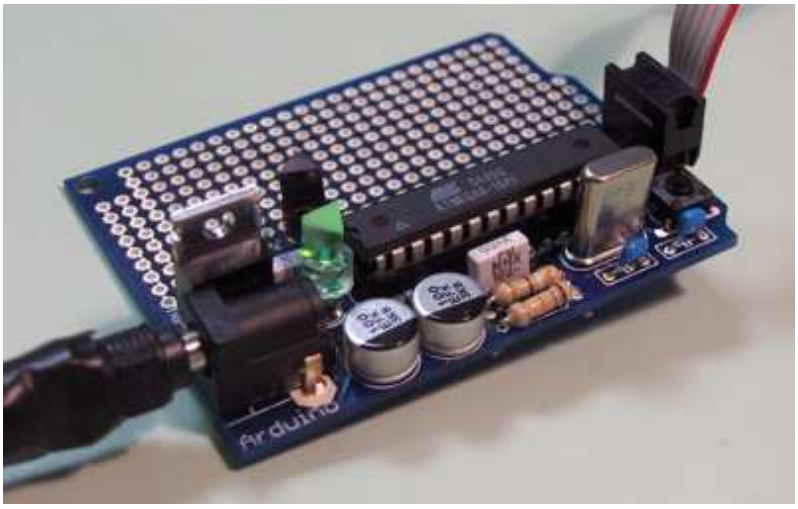
Dintre de les plaques USB podem trobar diferents dissenys i models:

- Arduino Duemilanove
- Arduino Diecimila
- Arduino NG Rev. C
- Arduino NG (Nova generació)
- Arduino Extreme v2
- Arduino Extreme
- Arduino USB v2.0
- Arduino USB

d. Placa de prototips

Aquesta placa està dissenyada per poder incorporar maquinari addicional al disseny de l' Arduino. Incorpora una matriu de forats en la qual poder acoblar el nostre maquinari addicional.

No disposa de port sèrie ni USB, per la qual cosa cal disposar d'una altra placa per programar el xip. En el seu defecte es pot utilitzar un programador paral·lel o AVR-ISP.

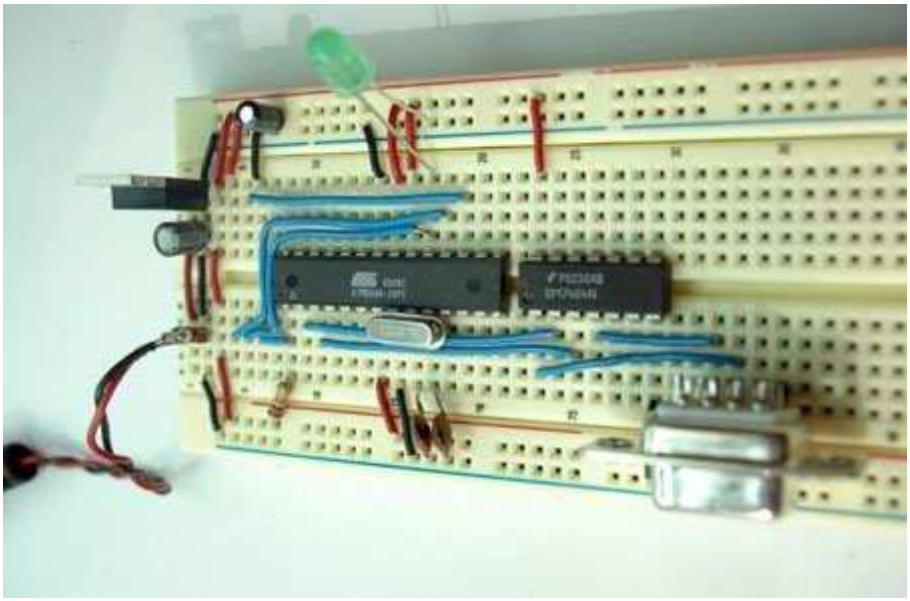


e. Bluetooh

És en la versió que s'està treballant. Elimina la necessitat de cables per comunicar-se amb un PC o qualssevol altre dispositiu bluetooth, com per exemple un telèfon mòbil.

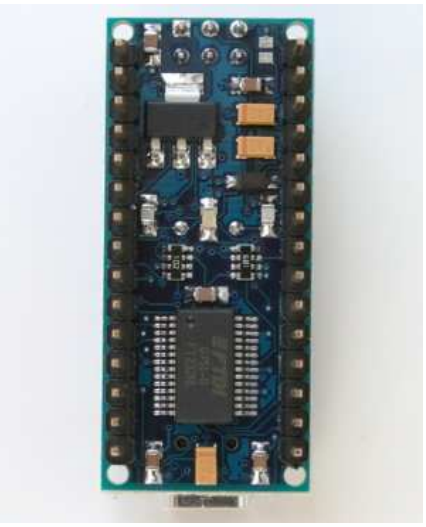
f. Stand-alone

Si el que vols és utilitzar directament el xip Atmega8 sobre qualsevol placa PCB o sobre una placa protoboard, sense utilitzar les parts de l'Arduino que no es necessiten.



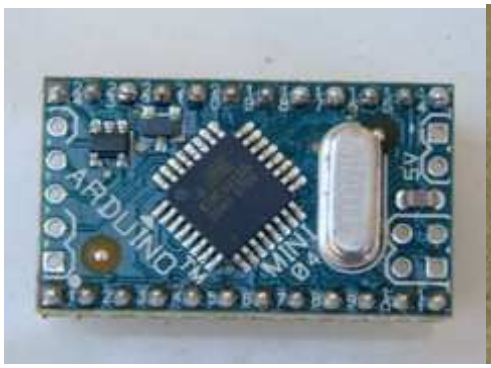
g. Arduino Nano

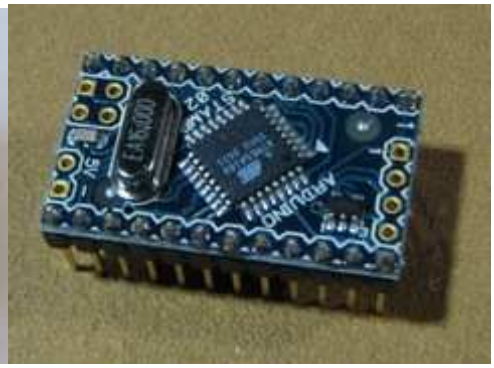
El Nano Arduino és una taula petita, completa, i placa-d'amistat basades en el Atmega328 (Arduino Nano 3,0) o ATmega168 (Arduino Nano 2.x). Té més o menys la mateixa funcionalitat de la Duemilanove Arduino, però en un paquet diferent. Li falta només un connector d'alimentació DC, i treballa amb un mini-B cable USB en lloc d'una normal. El Nano va ser dissenyat i està sent produït per Gravitech.



h. Arduino Mini

L'Arduino mini és la més petita i compacta de les plaques Arduino.



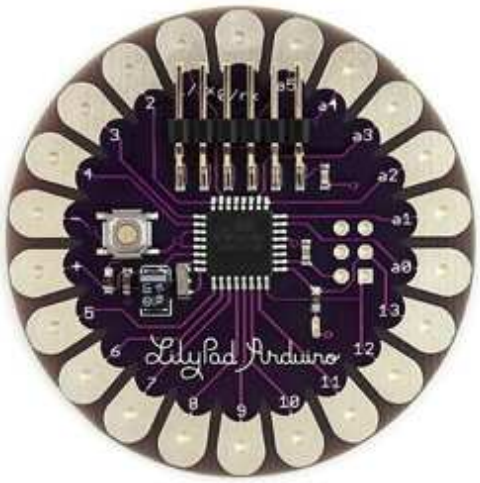


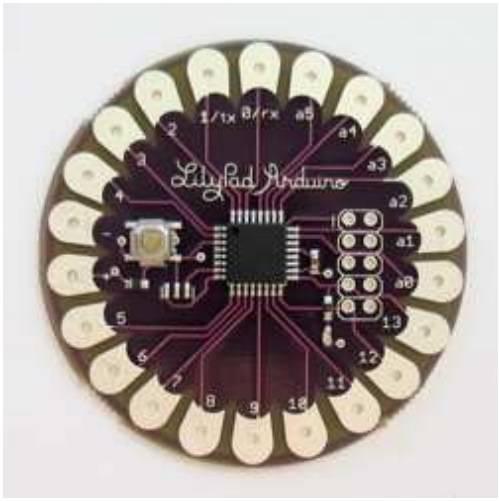
D'aquesta placa també es pot torbar diferents versions.

i. LilyPad Arduino

L'Arduino LilyPad és una placa de microcontrolador dissenyat per portar a sobre i al tèxtil. Pot ser cosit a la tela i de la mateixa manera muntar fonts

d'alimentació, sensors i actuadors amb fil conductor. El cor de la placa es basa en la ATMEGA168V.

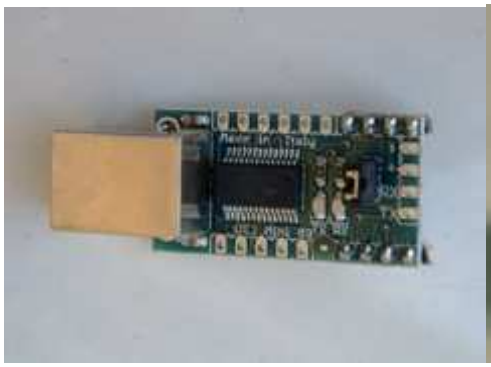


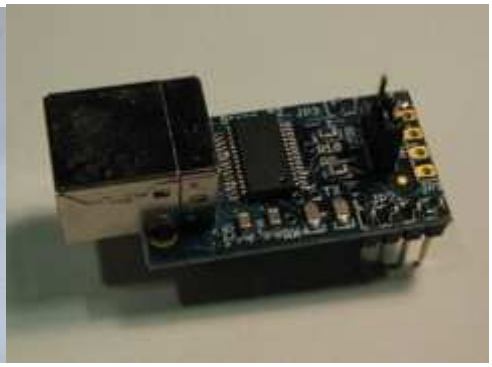


D’aquesta placa també es pot torbar diferents versions.

j. Mini USB Adapter

Tècnicament, això no és una "placa Arduino", ja que no disposa d'un microcontrolador. En canvi, és un adaptador que proporciona el Mini Arduino amb una connexió USB

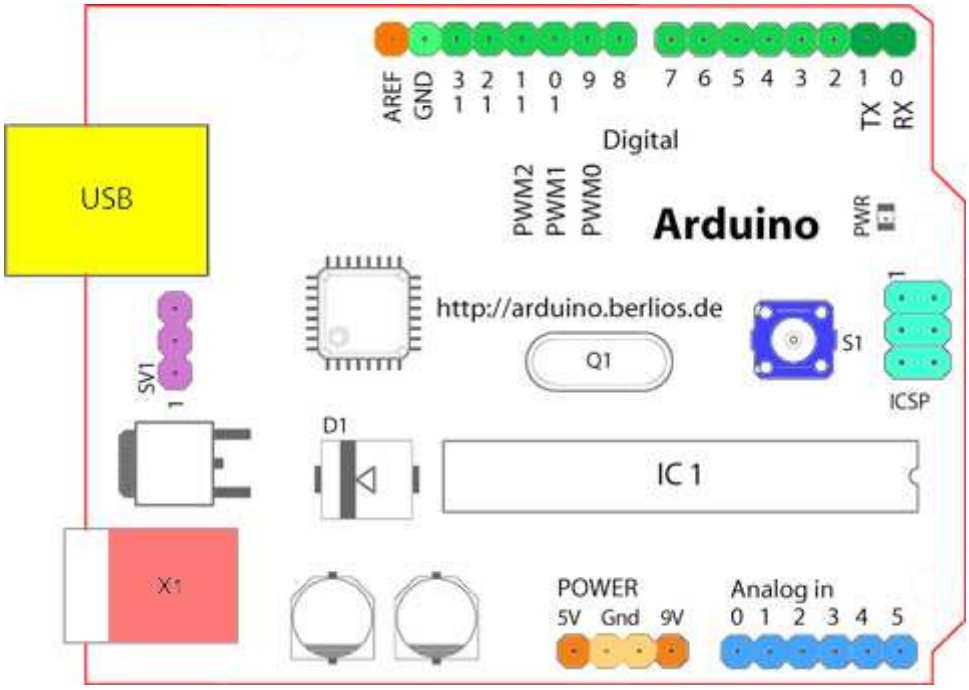




4. Amb quins elements podem interactuar?

La placa Arduino està basada en el chip Atmega8. Al voltant d’ell es monta tota la circuiteria per poder treure-li el màxim rendiment possible .

Prenent com a diferència la placa USB:



Començant en el sentit de les agulles del rellotge des del centre de la part superior:

Pin de referència analògica (taronja)

Sèrie de terra digital (verd clar)

Pins digitals 3-13 (verd)

Pins digitals 1-2 / entrada i sortida del port sèrie: TX/RX (verd fosc)

Botó de reset (blau fosc)

Entrada del circuit del programador sèrie (blau turquesa)

Pins d'entrada analògica 0-5 (blau clar)

Pin d'alimentació (alimentació: taronja)

Pin de terra (terra: taronja clar)

Entrada de la font d'alimentació externa (9-12V DC) X1 (rosa)

Commuta entre la font d'alimentació externa o alimentació a través del port USB SV1 (violeta)

Puerto USB (groc)

Projecte Processing

Processing és un llenguatge de programació, entorn de desenvolupament, i la comunitat en línia que des de 2001 ha promogut l'alfabetització de programari dins de les arts visuals. Inicialment pensat per servir com un quadern d'esbossos de programari i per ensenyar els fonaments de la programació dins d'un context visual, producció ràpidament es va convertir en una eina per a la creació d'acabat el treball professional també.
Processing és lliure i no està lligat a cap llicència, és una alternativa de codi obert a les eines de programari amb llicències molt cares, per tant Proccesing pretén ser l'alternativa accessible a les escoles i els estudiants individuals.
La seva condició de codi obert fomenta la participació comunitària i la col·laboració que és vital per al creixement del projecte.

Col·laboradors i programadors, contribueixen amb codi, responen preguntes en el fòrum de debat i construeixen biblioteques per ampliar les possibilitats del programari. La comunitat Processing ha escrit més de setanta biblioteques per facilitar la visió per ordinador, visualització de dades, música, creació de xarxes, i l'electrònica.

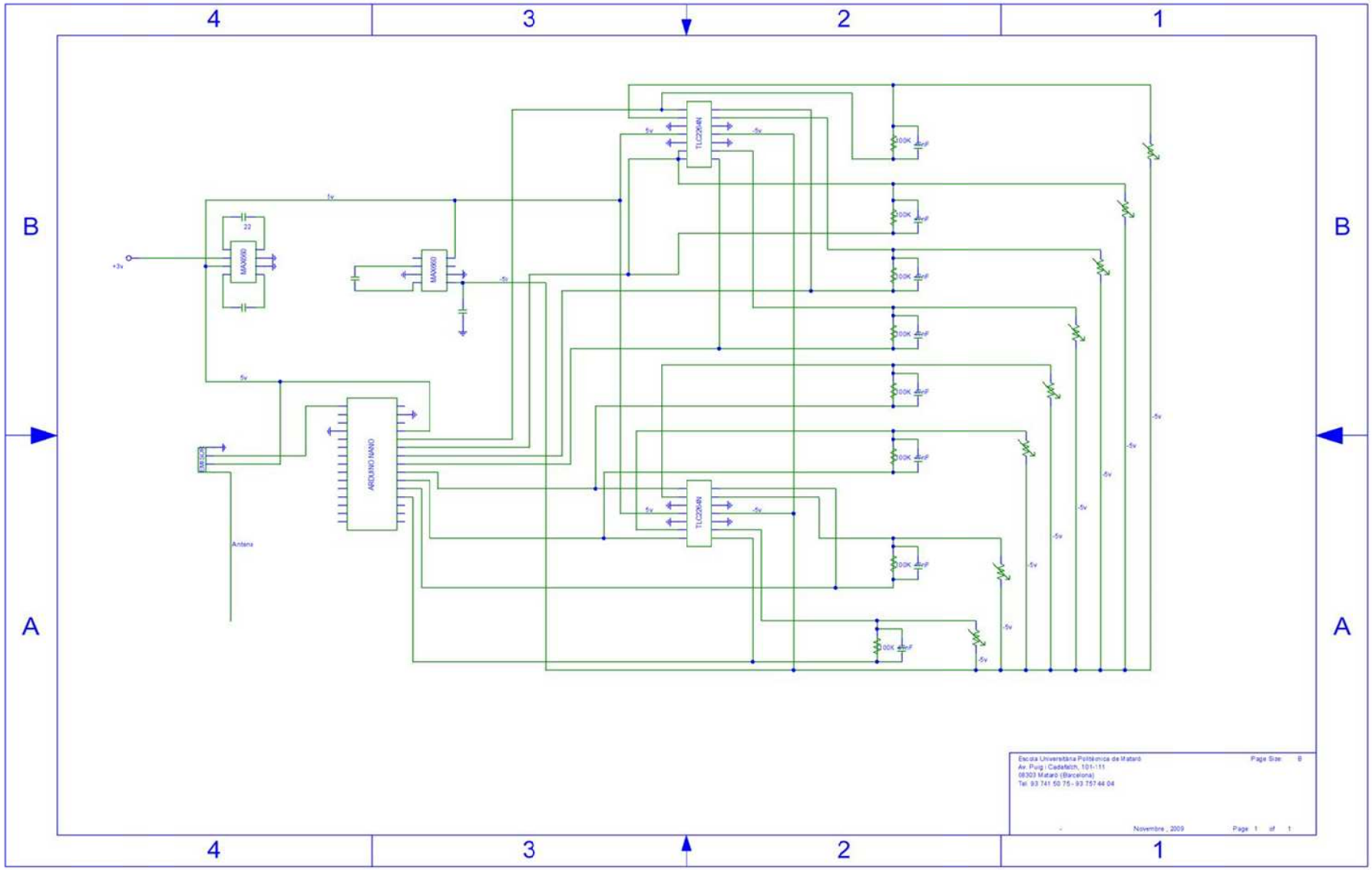
Processing és un context per explorar l'espai conceptual emergent que ens lliuren els mitjans electrònics. És un entorn per aprendre els fonaments de la programació informàtica dins el context de les arts electròniques i és un bloc de notes electròniques per desenvolupar idees.

L'entorn de Processing és el més fàcil compilador de Java / entorn de programació multimèdia i gràfic conegut per l'home. El sistema pot ser utilitzat per produir peces que arrenquen localment, com també Applets de Java incrustats a la web. Deliberadament, el programa està dissenyat per fer un pont entre la programació gràfica educacional, i el Java "real". Processing pot ser utilitzat com roda d'entrenament, però no ha de ser això.

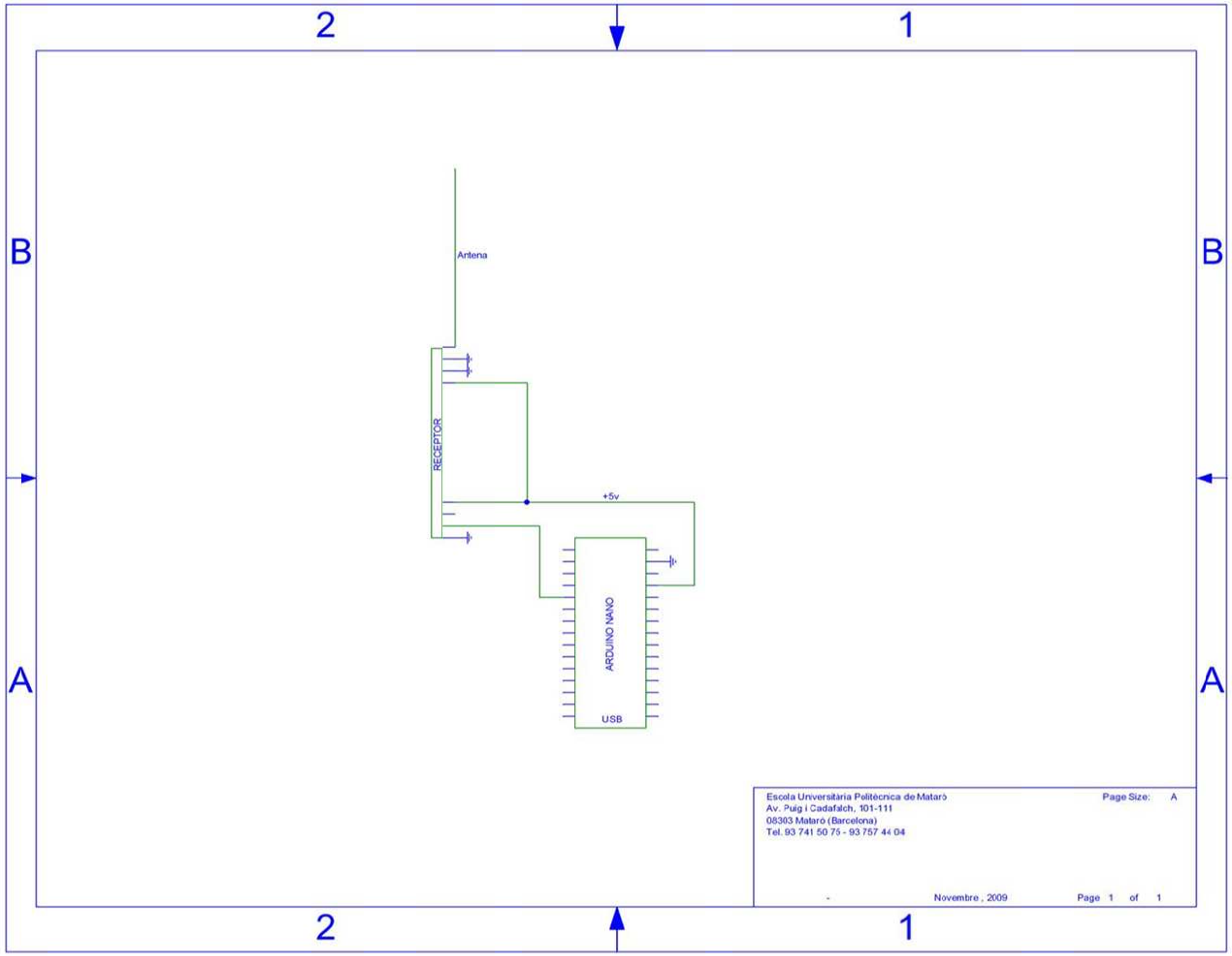
Annex II

Esquemàtics

Bloc emissor


4
3
2
1
B
A
TLC2543N
MAX660
ARDUINO NANO
Antena
Escola Universitària Politècnica de Mataró
Av. Puig i Cadafalch, 101-111
08303 Mataró (Barcelona)
Tel. 93 741 50 75 - 93 757 44 04
Page Size: B
Novembre , 2009
Page 1 of 1

Bloc receptor:


2
1
B
A
Antena
RECEPTOR
+5v
ARDUINO NANO
USB
Escola Universitària Politècnica de Mataró
Av. Puig i Cadafalch, 101-111
08303 Mataró (Barcelona)
Tel. 93 741 50 75 - 93 757 44 04
Page Size: A
Novembre , 2009
Page 1 of 1

Annex III

Components i pressupost

1. Cost del material per al muntatge d'un prototip:

Components	Preu unitari	Quantitat	Preu total
Microcontrolador			
Controlador Arduino Nano.	39,20 €	2	78,40 €
Condensadors			
Condensadors de 22µF	0,15 €	10	1,50 €
Condensadores de 10 µF	0,17 €	2	0,34 €
Resistències			
Resistències de 100KΩ	0,03 €	8	0,24 €
Sensors			
Sensors FlexiForce	13,54 €	8	108,31 €
Circuits integrats			
Inversor MAX660	6,94 €	1	6,94 €
TLC2264CN (Amplificadors operacionals)	0,51 €	2	1,02 €
Regulador de voltatge MAX619	3,36 €	1	3,36 €
Comunicació			
Placa emissora de radio	3,79 €	1	3,79 €
Placa receptora de radio	2,68 €	1	2,68 €
Varis			
Protoboards	35,00 €	2	70,00 €
Porta piles	1,50 €	1	1,50 €
Cables diferents colors (vermell, negre, groc …)	2,90 €	5	14,50 €
Piles	0,75 €	2	1,50 €

TOTAL	292,59 €
TOTAL + IVA	**339,40 €**

2. Costos d'enginyeria:

Concepte	Hores	Preu/hora	Total
Estudi i documentació	50	50 €	2.500 €
Disseny	30	50 €	1.500 €
Muntatge	60	50 €	3.000 €
Implementació	40	50 €	2.000 €
Redacció memòria	50	50 €	2.500 €
TOTAL			11.500 €

3. Amortització Instrumental:

- Instrumentació informàtica:

Equip	Hores	Preu/hora	Total
Ordinador	200	0,5 €	100,0 €
Software Arduino	50	0,0 €	0,0 €
Software Processing	50	0,0 €	0,0 €
Software Orcad	5	2,0 €	10,0 €
Microsoft Office	75	1,0 €	75,0 €

TOTAL	185 €

- Instrumentació electrònica:

Equip	Hores	Preu/hora	Total
Multímetre	10	0,5 €	5,0 €

TOTAL	5 €

4. Cost de fabricació del prototip:

Costos	Preu
Costos de material	339,40 €
Costos d'enginyeria	11.500 €
Costos d'amortització	190 €

TOTAL	12.029,40 €

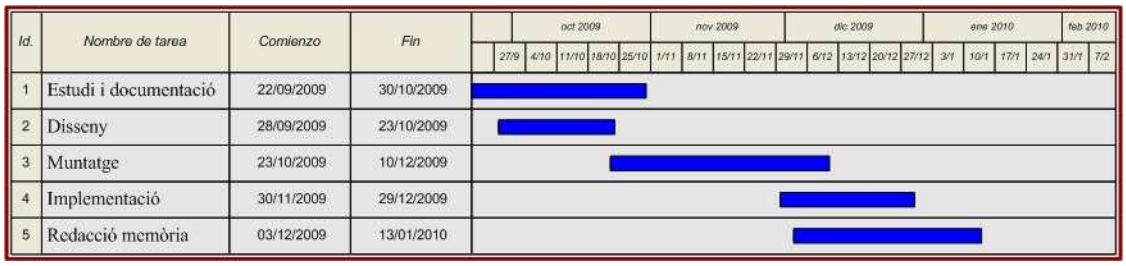
Annex IV

Diagrama de Gantt de la realització del projecte

Temps estimat de la durada de les tasques :

Tasca	Hores
Estudi i documentació	50
Disseny	30
Muntatge	60
Implementació	40
Redacció memòria	50

Diagrama de Gantt:



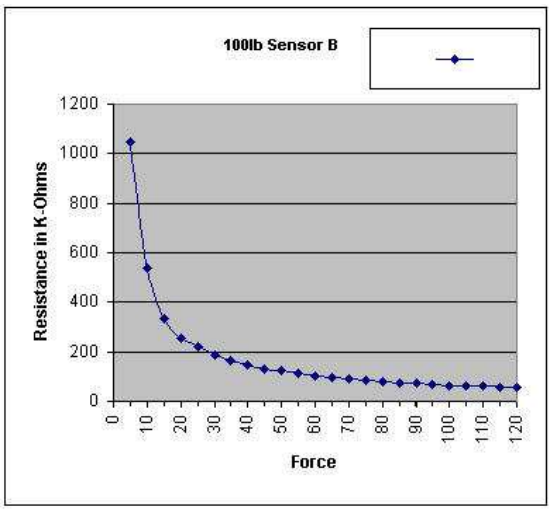
Annex V

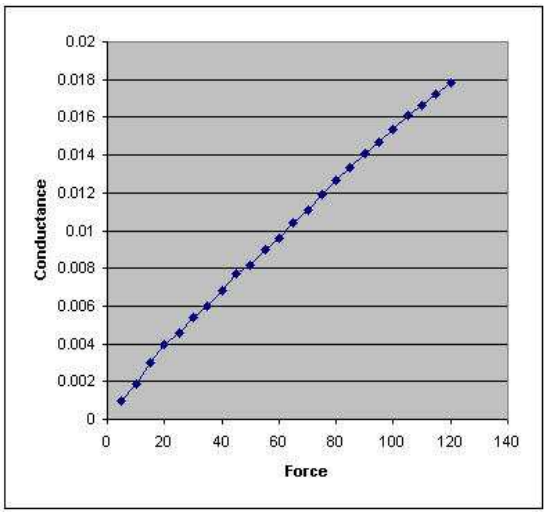
Calibratge

El calibratge és el mètode pel qual la producció elèctrica del sensor està relacionada amb una veritable unitat d'enginyeria, com ara lliures, kilograms o newtons.

Per calibrar, cal aplicar una força coneguda al sensor, i equiparar la sortida del sensor de la resistència a aquesta força.

S'ha d'anar repetint el pas anterior amb una sèrie de pesos coneguts, per poder obtenir un gràfic de relació força-Conductància(1 / R). Es pot dur a terme una interpolació lineal entre la càrrega zero i el calibratge conegut de les diferents càrregues, per determinar que l'abast de la força real coincideix amb el rang de sortida del sensor.





Directius de calibratge

Les següents directrius s'haurien de tenir en compte quan es calibra un sensor:

- Aplicar una càrrega de calibratge que s'aproximi a la càrrega que s'aplicarà durant l'ús del sistema, amb pesos morts o un dispositiu de prova (com un MTS o Instron).

- Evitar la carregar el sensor fins la saturació quan es calibra. Si el sensor es satura en una càrrega menor de la desitjada, ajustar la "sensibilitat".

- Distribuir la càrrega aplicada de manera uniforme en tota la zona de detecció per assegurar que les lectures són el més exactes possibles. Les lectures poden variar lleugerament si es canvia la posició dintre l'àrea de detecció del sensor.

- La temperatura també pot influir a l'hora de calibrar els sensors, per la qual cosa és aconsellable, calibrar els sensors a la mateixa temperatura.

Característiques de rendiment del sensor

Hi ha una sèrie de característiques dels sensors, que poden afectar els seus resultats. Aquesta secció conté una descripció de cada una d'aquestes condicions, i recomanacions sobre com reduir els seus efectes.

- **Repetibilitat**
  La repetibilitat és la capacitat del sensor per a respondre de la mateixa forma a una força aplicada en repetides ocasions

- **Linealitat**

La linealitat es refereix a la resposta del sensor (sortida digital) a la càrrega aplicada, en el rang del sensor. Aquesta resposta ideal ha de ser lineal, i qualsevol no-linealitat del sensor és la quantitat que la seva producció es desviï d'aquesta línia. El calibratge es fa per "alinear" aquesta producció tant com sigui possible.

- **Histèresi**

  La histèresi és la diferència en la resposta de sortida del sensor durant la càrrega i la descàrrega, per la mateixa força. Per a les forces estàtiques i les aplicacions en que la força només s'incrementa, i no disminueix, els efectes d'histèresi són mínims. Si la sol·licitud inclou una disminució de la càrrega, així com l'augment, hi pot haver error introduït per la histèresi que no s'explica per la salivació.

- **Deriva**

  La deriva és el canvi en la sortida del sensor quan una força constant s'aplica durant un període de temps. Si el sensor es manté sota una càrrega constant, la resistència del sensor disminueix contínuament, i la sortida anirà augmentant gradualment. És important tenir en compte quan es deriva de la calibració del sensor, de manera que els seus efectes es poden reduir al mínim. La forma més senzilla d'aconseguir això és realitzar el calibratge del sensor en un termini de temps similar a la que s'utilitzarà en l'aplicació.

- **Sensibilitat a la temperatura**

  En general, els resultats poden variar si es combinen les altes càrregues en el sensor amb altes temperatures. Per assegurar la precisió, cal calibrar el sensor a la temperatura en la que s'utilitzarà en l'aplicació.

- **Vida del sensor**

  La vida del sensor depèn de l'aplicació en la que s'utilitza. Els sensors són reutilitzables, llevat que s'utilitzi en aplicacions en què són sotmesos a

condicions severes. El maneig brusc d'un sensor també escurçar la seva vida útil. Per millorar la vida útil del sensor és important mantenir l'àrea de detecció del sensor neta.

Annex VI

Codi font i llibreries

Codi Font

- Bloc emissor:

```
// Declaració de les variables per les entrades analògiques

int analogInput1 = 0;
int analogInput2 = 1;
int analogInput3 = 2;
int analogInput4 = 3;
int analogInput5 = 4;
int analogInput6 = 5;
int analogInput7 = 6;
int analogInput8 = 7;

// Declaració de les variables correspondents als 8 sensors

int sensor1 = 0;
int sensor2 = 0;
int sensor3 = 0;
int sensor4 = 0;
int sensor5 = 0;
int sensor6 = 0;
int sensor7 = 0;
int sensor8 = 0;

// Declaració de la capçalera, senyala l'inici de la trama
```

```
int head = 255;



void setup(){

//Declaració de les entrades analoguiques i assignació a les variables

  pinMode(analogInput1, INPUT);
  pinMode(analogInput2, INPUT);
  pinMode(analogInput3, INPUT);
  pinMode(analogInput4, INPUT);
  pinMode(analogInput5, INPUT);
  pinMode(analogInput6, INPUT);
  pinMode(analogInput7, INPUT);
  pinMode(analogInput8, INPUT);

// Declaració de la velocitat de l'enllaç

  Serial.begin(2400);

}

// Inici del bucle

void loop(){

 // Asignació dels valors obtinguts dels sensors a les variables.

 sensor1 = analogRead(analogInput1);
 sensor2 = analogRead(analogInput2);
 sensor3 = analogRead(analogInput3);
 sensor4 = analogRead(analogInput4);
 sensor5 = analogRead(analogInput5);
```

```
 sensor6 = analogRead(analogInput6);
 sensor7 = analogRead(analogInput7);
 sensor8 = analogRead(analogInput8);

// Impresió de la capçalera per el port serie.

  Serial.write(head);
  Serial.write(head);
  Serial.write(head);
  Serial.write(head);

// Impresió dels valors dels sensors per el port serie.

  Serial.write(sensor1);
  Serial.write(sensor2);
  Serial.write(sensor3);
  Serial.write(sensor4);
  Serial.write(sensor5);
  Serial.write(sensor6);
  Serial.write(sensor7);
  Serial.write(sensor8);
 }
```

- Bloc receptor:

```
// Crida a la llibreria per convertir un pin digital en un port serie

#include <SoftwareSerial.h>

// Definició dels pins digitals que serviran de RX I TX
```

```
#define rxPin 2
#define txPin 3

//Creació de la variable corresponent al port serie

SoftwareSerial Serial2 =  SoftwareSerial(rxPin, txPin);



void setup(){

//Declaració dels Pin modes

  pinMode(rxPin, INPUT);
  pinMode(txPin, OUTPUT);

//Declaració de la velocitat de l'enllaç

  Serial.begin(2400);

//Declaració de la velocitat de l'enllaç

  Serial2.begin (2400);

}

void loop(){

//Impresió de les dades que ens arriven per el port serie.

  Serial.write(Serial2.read());

}
```

- Unitat de *feedback*

```
import processing.serial.*;
// Variables globals, accessibles des de qualsevol funció
Serial myPort;
int [] sensors = new int [8];
int count = 0;
int inByte = 0;
boolean dades = false;

void setup() {
  myPort = new Serial(this, Serial.list()[0], 2400);
  myPort.buffer(1); // Es genera un event per cada byte rebut
  size(400,200); //tamany de la finestra
}

void draw() {

color c1 = color(#00FF00);
color c2 = color(#FF9900);
color c3 = color (#FF0000);
color c4 = color (#FFFFFF);



if (sensors[0] >= 200){
  fill(c3);
}
else if (sensors[0] >= 100){
  fill(c2);
}
else if (sensors[0] >= 6){
  fill(c1);
```

```
}
else{
  fill (c4);
}

rect(0, 0, 100, 100);

if (sensors[1] >= 200){
  fill(c3);
}
else if (sensors[1] >= 100){
  fill(c2);
}
else if (sensors[1] >= 6){
  fill(c1);
}
else{
  fill (c4);
}
rect(100, 100, 100, 100);
if (sensors[2] >= 200){
  fill(c3);
}
else if (sensors[2] >= 100){
  fill(c2);
}
else if (sensors[2] >= 6){
  fill(c1);
}
else{
  fill (c4);
}
rect(300, 100, 100, 100);
if (sensors[3] >= 200){
```

```
  fill(c3);
}
else if (sensors[3] >= 100){
  fill(c2);
}
else if (sensors[3] >= 6){
  fill(c1);
}
else{
  fill (c4);
}
rect(200, 0, 100, 100);
if (sensors[4] >= 200){
  fill(c3);
}
else if (sensors[4] >= 100){
  fill(c2);
}
else if (sensors[4] >= 6){
  fill(c1);
}
else{
  fill (c4);
}
rect(300, 0, 100, 100);
if (sensors[5] >= 200){
  fill(c3);
}
else if (sensors[5] >= 100){
  fill(c2);
}
else if (sensors[5] >= 6){
  fill(c1);
}
```

```
else{
  fill (c4);
}
rect(200, 100, 100, 100);
if (sensors[6] >= 200){
  fill(c3);
}
else if (sensors[6] >= 100){
  fill(c2);
}
else if (sensors[6] >= 6){
  fill(c1);
}
else{
  fill (c4);
}
rect(0, 100, 100, 100);
if (sensors[7] >= 200){
  fill(c3);
}
else if (sensors[7] >= 100){
  fill(c2);
}
else if (sensors[7] >= 6){
  fill(c1);
}
else{
  fill (c4);
}
rect(100, 0, 100, 100);




}
```

```
void serialEvent(Serial myPort) {
  inByte = myPort.read();
  if (!dades) { // Encara no hem rebut 255/255/255/255
    if (count == 4) { // Si hem rebut 255/255/255/255
      dades = true; // EL seguent byte seran dades
      count = 0; // Reiniciem el comptador
    }
    else if (inByte == 255) { // Si no hem rebut 255/255/255/255 comptem...
      count++;
    }
    else {
      count = 0; // Si els F no son consequtius comencem de nou...
    }
  }
  else if (dades) { // Hem rebut 255/255/255/255, el seguent byte són dades
    if (count < 8) { // No podem rebre més de 8 dades seguides
      sensors[count] = inByte; // Coloquem el Byte
      count++; // Incrementem la posicio
    }
    else { // Un cop hem rebut les 8 dades
      count = 0;
      dades = false;
    }
  }
}
```

Llibreries:

- Llibreries Arduino:

```
/*
  SoftwareSerial.h - Software serial library
  Copyright (c) 2006 David A. Mellis.  All right reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
*/

#ifndef SoftwareSerial_h
#define SoftwareSerial_h

#include <inttypes.h>

class SoftwareSerial
{
  private:
    uint8_t _receivePin;
```

```
  uint8_t _transmitPin;
  long _baudRate;
  int _bitPeriod;
  void printNumber(unsigned long, uint8_t);
 public:
  SoftwareSerial(uint8_t, uint8_t);
  void begin(long);
  int read();
  void print(char);
  void print(const char[]);
  void print(uint8_t);
  void print(int);
  void print(unsigned int);
  void print(long);
  void print(unsigned long);
  void print(long, int);
  void println(void);
  void println(char);
  void println(const char[]);
  void println(uint8_t);
  void println(int);
  void println(long);
  void println(unsigned long);
  void println(long, int);
};

#endif
```

- Llibreries Porcessing:

```
/*
 PSerial - class for serial port goodness
 Part of the Processing project - http://processing.org
```

Copyright (c) 2004-05 Ben Fry & Casey Reas

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General
Public License along with this library; if not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330,
Boston, MA 02111-1307 USA
*/

```
package processing.serial;
import processing.core.*;

import gnu.io.*;

import java.io.*;
import java.util.*;
import java.lang.reflect.*;



public class Serial implements SerialPortEventListener {

  PApplet parent;
  Method serialEventMethod;
```

```
// properties can be passed in for default values
// otherwise defaults to 9600 N81

// these could be made static, which might be a solution
// for the classloading problem.. because if code ran again,
// the static class would have an object that could be closed

public SerialPort port;

public int rate;
public int parity;
public int databits;
public int stopbits;



// read buffer and streams

public InputStream input;
public OutputStream output;

byte buffer[] = new byte[32768];
int bufferIndex;
int bufferLast;

//boolean bufferUntil = false;
int bufferSize = 1;  // how big before reset or event firing
boolean bufferUntil;
int bufferUntilByte;



// defaults

static String dname = "COM1";
```

```
static int drate = 9600;
static char dparity = 'N';
static int ddatabits = 8;
static float dstopbits = 1;


public void setProperties(Properties props) {
  dname =
    props.getProperty("serial.port", dname);
  drate =
    Integer.parseInt(props.getProperty("serial.rate", "9600"));
  dparity =
    props.getProperty("serial.parity", "N").charAt(0);
  ddatabits =
    Integer.parseInt(props.getProperty("serial.databits", "8"));
  dstopbits =
    new Float(props.getProperty("serial.stopbits", "1")).floatValue();
}


public Serial(PApplet parent) {
  this(parent, dname, drate, dparity, ddatabits, dstopbits);
}

public Serial(PApplet parent, int irate) {
  this(parent, dname, irate, dparity, ddatabits, dstopbits);
}

public Serial(PApplet parent, String iname, int irate) {
  this(parent, iname, irate, dparity, ddatabits, dstopbits);
}

public Serial(PApplet parent, String iname) {
  this(parent, iname, drate, dparity, ddatabits, dstopbits);
```

```
}

public Serial(PApplet parent, String iname, int irate,
              char iparity, int idatabits, float istopbits) {
  //if (port != null) port.close();
  this.parent = parent;
  //parent.attach(this);

  this.rate = irate;

  parity = SerialPort.PARITY_NONE;
  if (iparity == 'E') parity = SerialPort.PARITY_EVEN;
  if (iparity == 'O') parity = SerialPort.PARITY_ODD;

  this.databits = idatabits;

  stopbits = SerialPort.STOPBITS_1;
  if (istopbits == 1.5f) stopbits = SerialPort.STOPBITS_1_5;
  if (istopbits == 2) stopbits = SerialPort.STOPBITS_2;

  try {
    Enumeration<?> portList = CommPortIdentifier.getPortIdentifiers();
    while (portList.hasMoreElements()) {
      CommPortIdentifier portId =
        (CommPortIdentifier) portList.nextElement();

      if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
        //System.out.println("found " + portId.getName());
        if (portId.getName().equals(iname)) {
          port = (SerialPort)portId.open("serial madness", 2000);
          input = port.getInputStream();
          output = port.getOutputStream();
          port.setSerialPortParams(rate, databits, stopbits, parity);
          port.addEventListener(this);
```

```
          port.notifyOnDataAvailable(true);
          //System.out.println("opening, ready to roll");
        }
      }
    }

  } catch (Exception e) {
    errorMessage("<init>", e);
    //exception = e;
    //e.printStackTrace();
    port = null;
    input = null;
    output = null;
  }

  parent.registerDispose(this);

  // reflection to check whether host applet has a call for
  // public void serialEvent(processing.serial.Serial)
  // which would be called each time an event comes in
  try {
    serialEventMethod =
      parent.getClass().getMethod("serialEvent",
                                  new Class[] { Serial.class });
  } catch (Exception e) {
    // no such method, or an error.. which is fine, just ignore
  }
}



/**
 * Stop talking to serial and shut things down.
 * <P>
 * Basically just a user-accessible version of dispose().
```

```
 * For now, it just calls dispose(), but dispose shouldn't
 * be called from applets, because in some libraries,
 * dispose() blows shit up if it's called by a user who
 * doesn't know what they're doing.
 */
public void stop() {
  dispose();
}



/**
 * Used by PApplet to shut things down.
 */
public void dispose() {
  try {
    // do io streams need to be closed first?
    if (input != null) input.close();
    if (output != null) output.close();

  } catch (Exception e) {
    e.printStackTrace();
  }
  input = null;
  output = null;

  try {
    if (port != null) port.close();  // close the port

  } catch (Exception e) {
    e.printStackTrace();
  }
  port = null;
}
```

```
/**
* Set the DTR line. Addition from Tom Hulbert.
*/
public void setDTR(boolean state) {
     port.setDTR(state);
}



synchronized public void serialEvent(SerialPortEvent serialEvent) {
  if (serialEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
    try {
      while (input.available() > 0) {
        synchronized (buffer) {
          if (bufferLast == buffer.length) {
            byte temp[] = new byte[bufferLast << 1];
            System.arraycopy(buffer, 0, temp, 0, bufferLast);
            buffer = temp;
          }
          buffer[bufferLast++] = (byte) input.read();
          if (serialEventMethod != null) {
            if ((bufferUntil &&
                 (buffer[bufferLast-1] == bufferUntilByte)) ||
                (!bufferUntil &&
                 ((bufferLast - bufferIndex) >= bufferSize))) {
              try {
                serialEventMethod.invoke(parent, new Object[] { this });
              } catch (Exception e) {
                String msg = "error, disabling serialEvent() for " + port;
                System.err.println(msg);
                e.printStackTrace();
                serialEventMethod = null;
              }
            }
```

```
        }
       }
      }

    } catch (IOException e) {
      errorMessage("serialEvent", e);
    }
  }
}



/**
 * Set number of bytes to buffer before calling serialEvent()
 * in the host applet.
 */
public void buffer(int count) {
  bufferUntil = false;
  bufferSize = count;
}



/**
 * Set a specific byte to buffer until before calling
 * serialEvent() in the host applet.
 */
public void bufferUntil(int what) {
  bufferUntil = true;
  bufferUntilByte = what;
}



/**
 * Returns the number of bytes that have been read from serial
 * and are waiting to be dealt with by the user.
```

```
 */
public int available() {
  return (bufferLast - bufferIndex);
}



/**
 * Ignore all the bytes read so far and empty the buffer.
 */
public void clear() {
  bufferLast = 0;
  bufferIndex = 0;
}



/**
 * Returns a number between 0 and 255 for the next byte that's
 * waiting in the buffer.
 * Returns -1 if there was no byte (although the user should
 * first check available() to see if things are ready to avoid this)
 */
public int read() {
  if (bufferIndex == bufferLast) return -1;

  synchronized (buffer) {
    int outgoing = buffer[bufferIndex++] & 0xff;
    if (bufferIndex == bufferLast) {  // rewind
      bufferIndex = 0;
      bufferLast = 0;
    }
    return outgoing;
  }
}
```

```
/**
 * Same as read() but returns the very last value received
 * and clears the buffer. Useful when you just want the most
 * recent value sent over the port.
 */
public int last() {
  if (bufferIndex == bufferLast) return -1;
  synchronized (buffer) {
    int outgoing = buffer[bufferLast-1];
    bufferIndex = 0;
    bufferLast = 0;
    return outgoing;
  }
}



/**
 * Returns the next byte in the buffer as a char.
 * Returns -1, or 0xffff, if nothing is there.
 */
public char readChar() {
  if (bufferIndex == bufferLast) return (char)(-1);
  return (char) read();
}



/**
 * Just like last() and readChar().
 */
public char lastChar() {
  if (bufferIndex == bufferLast) return (char)(-1);
  return (char) last();
}
```

```
/**
 * Return a byte array of anything that's in the serial buffer.
 * Not particularly memory/speed efficient, because it creates
 * a byte array on each read, but it's easier to use than
 * readBytes(byte b[]) (see below).
 */
public byte[] readBytes() {
  if (bufferIndex == bufferLast) return null;

  synchronized (buffer) {
    int length = bufferLast - bufferIndex;
    byte outgoing[] = new byte[length];
    System.arraycopy(buffer, bufferIndex, outgoing, 0, length);

    bufferIndex = 0;  // rewind
    bufferLast = 0;
    return outgoing;
  }
}


/**
 * Grab whatever is in the serial buffer, and stuff it into a
 * byte buffer passed in by the user. This is more memory/time
 * efficient than readBytes() returning a byte[] array.
 *
 * Returns an int for how many bytes were read. If more bytes
 * are available than can fit into the byte array, only those
 * that will fit are read.
 */
public int readBytes(byte outgoing[]) {
  if (bufferIndex == bufferLast) return 0;
```

```
  synchronized (buffer) {
    int length = bufferLast - bufferIndex;
    if (length > outgoing.length) length = outgoing.length;
    System.arraycopy(buffer, bufferIndex, outgoing, 0, length);

    bufferIndex += length;
    if (bufferIndex == bufferLast) {
      bufferIndex = 0;  // rewind
      bufferLast = 0;
    }
    return length;
  }
}



/**
 * Reads from the serial port into a buffer of bytes up to and
 * including a particular character. If the character isn't in
 * the serial buffer, then 'null' is returned.
 */
public byte[] readBytesUntil(int interesting) {
  if (bufferIndex == bufferLast) return null;
  byte what = (byte)interesting;

  synchronized (buffer) {
    int found = -1;
    for (int k = bufferIndex; k < bufferLast; k++) {
      if (buffer[k] == what) {
        found = k;
        break;
      }
    }
    if (found == -1) return null;
```

```
    int length = found - bufferIndex + 1;
    byte outgoing[] = new byte[length];
    System.arraycopy(buffer, bufferIndex, outgoing, 0, length);

    bufferIndex += length;
    if (bufferIndex == bufferLast) {
      bufferIndex = 0;  // rewind
      bufferLast = 0;
    }
    return outgoing;
  }
}



/**
 * Reads from the serial port into a buffer of bytes until a
 * particular character. If the character isn't in the serial
 * buffer, then 'null' is returned.
 *
 * If outgoing[] is not big enough, then -1 is returned,
 *   and an error message is printed on the console.
 * If nothing is in the buffer, zero is returned.
 * If 'interesting' byte is not in the buffer, then 0 is returned.
 */
public int readBytesUntil(int interesting, byte outgoing[]) {
  if (bufferIndex == bufferLast) return 0;
  byte what = (byte)interesting;

  synchronized (buffer) {
    int found = -1;
    for (int k = bufferIndex; k < bufferLast; k++) {
      if (buffer[k] == what) {
        found = k;
```

```
        break;
      }
    }
    if (found == -1) return 0;

    int length = found - bufferIndex + 1;
    if (length > outgoing.length) {
      System.err.println("readBytesUntil() byte buffer is" +
                         " too small for the " + length +
                         " bytes up to and including char " + interesting);
      return -1;
    }
    //byte outgoing[] = new byte[length];
    System.arraycopy(buffer, bufferIndex, outgoing, 0, length);

    bufferIndex += length;
    if (bufferIndex == bufferLast) {
      bufferIndex = 0;  // rewind
      bufferLast = 0;
    }
    return length;
  }
}



/**
 * Return whatever has been read from the serial port so far
 * as a String. It assumes that the incoming characters are ASCII.
 *
 * If you want to move Unicode data, you can first convert the
 * String to a byte stream in the representation of your choice
 * (i.e. UTF8 or two-byte Unicode data), and send it as a byte array.
 */
public String readString() {
```

```
  if (bufferIndex == bufferLast) return null;
  return new String(readBytes());
}


/**
 * Combination of readBytesUntil and readString. See caveats in
 * each function. Returns null if it still hasn't found what
 * you're looking for.
 *
 * If you want to move Unicode data, you can first convert the
 * String to a byte stream in the representation of your choice
 * (i.e. UTF8 or two-byte Unicode data), and send it as a byte array.
 */
public String readStringUntil(int interesting) {
  byte b[] = readBytesUntil(interesting);
  if (b == null) return null;
  return new String(b);
}


/**
 * This will handle both ints, bytes and chars transparently.
 */
public void write(int what) {  // will also cover char
  try {
    output.write(what & 0xff);  // for good measure do the &
    output.flush();   // hmm, not sure if a good idea

  } catch (Exception e) { // null pointer or serial port dead
    errorMessage("write", e);
  }
}
```

```
public void write(byte bytes[]) {
  try {
    output.write(bytes);
    output.flush();   // hmm, not sure if a good idea

  } catch (Exception e) { // null pointer or serial port dead
    //errorMessage("write", e);
    e.printStackTrace();
  }
}



/**
 * Write a String to the output. Note that this doesn't account
 * for Unicode (two bytes per char), nor will it send UTF8
 * characters.. It assumes that you mean to send a byte buffer
 * (most often the case for networking and serial i/o) and
 * will only use the bottom 8 bits of each char in the string.
 * (Meaning that internally it uses String.getBytes)
 *
 * If you want to move Unicode data, you can first convert the
 * String to a byte stream in the representation of your choice
 * (i.e. UTF8 or two-byte Unicode data), and send it as a byte array.
 */
public void write(String what) {
  write(what.getBytes());
}



/**
 * If this just hangs and never completes on Windows,
 * it may be because the DLL doesn't have its exec bit set.
 * Why the hell that'd be the case, who knows.
```

```
 */
static public String[] list() {
  Vector<String> list = new Vector<String>();
  try {
    //System.err.println("trying");
    Enumeration<?> portList = CommPortIdentifier.getPortIdentifiers();
    //System.err.println("got port list");
    while (portList.hasMoreElements()) {
      CommPortIdentifier portId =
        (CommPortIdentifier) portList.nextElement();
      //System.out.println(portId);

      if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
        String name = portId.getName();
        list.addElement(name);
      }
    }

  } catch (UnsatisfiedLinkError e) {
    //System.err.println("1");
    errorMessage("ports", e);

  } catch (Exception e) {
    //System.err.println("2");
    errorMessage("ports", e);
  }
  //System.err.println("move out");
  String outgoing[] = new String[list.size()];
  list.copyInto(outgoing);
  return outgoing;
}



/**
```

```
  * General error reporting, all corraled here just in case
  * I think of something slightly more intelligent to do.
  */
 static public void errorMessage(String where, Throwable e) {
   e.printStackTrace();
   throw new RuntimeException("Error inside Serial." + where + "()");
 }
}



 /*
 class SerialMenuListener implements ItemListener {
   //public SerialMenuListener() { }

   public void itemStateChanged(ItemEvent e) {
     int count = serialMenu.getItemCount();
     for (int i = 0; i < count; i++) {
       ((CheckboxMenuItem)serialMenu.getItem(i)).setState(false);
     }
     CheckboxMenuItem item = (CheckboxMenuItem)e.getSource();
     item.setState(true);
     String name = item.getLabel();
     //System.out.println(item.getLabel());
     PdeBase.properties.put("serial.port", name);
     //System.out.println("set to " + get("serial.port"));
   }
 }
 */



 /*
 protected Vector buildPortList() {
   // get list of names for serial ports
   // have the default port checked (if present)
```

```
Vector list = new Vector();

//SerialMenuListener listener = new SerialMenuListener();
boolean problem = false;

// if this is failing, it may be because
// lib/javax.comm.properties is missing.
// java is weird about how it searches for java.comm.properties
// so it tends to be very fragile. i.e. quotes in the CLASSPATH
// environment variable will hose things.
try {
 //System.out.println("building port list");
 Enumeration portList = CommPortIdentifier.getPortIdentifiers();
 while (portList.hasMoreElements()) {
  CommPortIdentifier portId =
   (CommPortIdentifier) portList.nextElement();
  //System.out.println(portId);

  if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
   //if (portId.getName().equals(port)) {
   String name = portId.getName();
   //CheckboxMenuItem mi =
   //new CheckboxMenuItem(name, name.equals(defaultName));

   //mi.addItemListener(listener);
   //serialMenu.add(mi);
   list.addElement(name);
  }
 }
} catch (UnsatisfiedLinkError e) {
 e.printStackTrace();
 problem = true;

} catch (Exception e) {
```

```
    System.out.println("exception building serial menu");
    e.printStackTrace();
  }

  //if (serialMenu.getItemCount() == 0) {
    //System.out.println("dimming serial menu");
  //serialMenu.setEnabled(false);
  //}

  // only warn them if this is the first time
  if (problem && PdeBase.firstTime) {
    JOptionPane.showMessageDialog(this, //frame,
                                  "Serial port support not installed.\n" +
                                  "Check the readme for instructions\n" +
                                  "if you need to use the serial port.    ",
                                  "Serial Port Warning",
                                  JOptionPane.WARNING_MESSAGE);
  }
  return list;
}
*/
```

Annex VII

Datasheets :

Amplificador Operacional:

TLC226x, TLC226xA
Advanced LinCMOS™ RAIL-TO-RAIL OPERATIONAL AMPLIFIERS

SLOS177D – FEBRUARY 1997 – REVISED MARCH 2001

- Output Swing includes Both Supply Rails
- Low Noise . . . 12 nV/√Hz Typ at f = 1 kHz
- Low Input Bias Current . . . 1 pA Typ
- Fully Specified for Both Single-Supply and Split-Supply Operation
- Low Power . . . 500 µA Max
- Common-Mode Input Voltage Range Includes Negative Rail
- Low Input Offset Voltage
  950 µV Max at T_A = 25°C (TLC2262A)
- Macromodel Included
- Performance Upgrade for the TS27M2/M4 and TLC27M2/M4
- Available in Q-Temp Automotive
  HighRel Automotive Applications
  Configuration Control/Print Support
  Qualification to Automotive Standards

description

The TLC2262 and TLC2264 are dual and quadruple operational amplifiers from Texas Instruments. Both devices exhibit rail-to-rail output performance for increased dynamic range in single- or split-supply applications. The TLC226x family offers a compromise between the micropower TLC225x and the ac performance of the TLC227x. It has low supply current for battery-powered applications, while still having adequate ac performance for applications that demand it. The noise performance has been dramatically improved over previous generations of CMOS amplifiers. Figure 1 depicts the low level of noise voltage for this CMOS amplifier, which has only 200 µA (typ) of supply current per amplifier.

EQUIVALENT INPUT NOISE VOLTAGE
vs
FREQUENCY

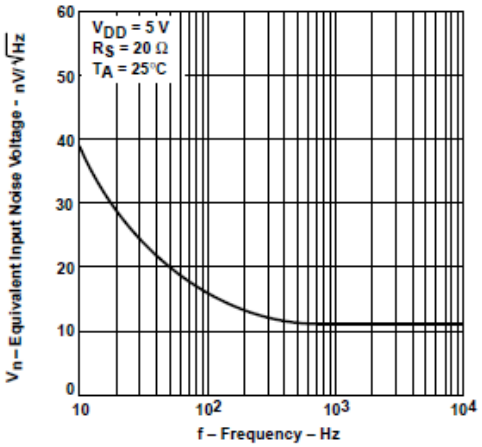


Figure 1

The TLC226x, exhibiting high input impedance and low noise, are excellent for small-signal conditioning for high-impedance sources, such as piezoelectric transducers. Because of the micro-power dissipation levels, these devices work well in hand-held monitoring and remote-sensing applications. In addition, the rail-to-rail output feature with single or split supplies makes this family a great choice when interfacing with analog-to-digital converters (ADCs). For precision applications, the TLC226xA family is available and has a maximum input offset voltage of 950 µV. This family is fully characterized at 5 V and ±5 V.

The TLC2262/4 also makes great upgrades to the TLC27M2/L4 or TS27M2/L4 in standard designs. They offer increased output dynamic range, lower noise voltage and lower input offset voltage. This enhanced feature set allows them to be used in a wider range of applications. For applications that require higher output drive and wider input voltage range, see the TLV2432 and TLV2442. If your design requires single amplifiers, please see the TLV2211/21/31 family. These devices are single rail-to-rail operational amplifiers in the SOT-23 package. Their small size and low power consumption, make them ideal for high density, battery-powered equipment.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Advanced LinCMOS is a trademark of Texas Instruments.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2001, Texas Instruments Incorporated
On products compliant to MIL-PRF-38535, all parameters are tested unless otherwise noted. On all other products, production processing does not necessarily include testing of all parameters.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Sensor:

FlexiForce®
A201 Standard Force & Load Sensors

Actual size of sensor

Sensing area

Physical Properties

Thickness	0.008" (0.208 mm)
Length	7.75" (197 mm),
	optional trimmed lengths: 6" (152 mm), 4" (102 mm), or 2" (51mm)
Width	0.55" (14 mm)
Sensing Area	0.375" diameter (9.53 mm)
Connector	3-pin Male Square Pin (center pin is inactive)
Substrate	Polyester (ex: Mylar)

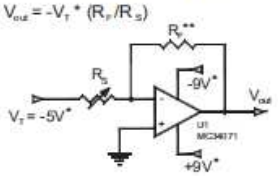
Standard Force Ranges (as tested with circuit shown below)

0 - 1 lb. (4.4 N)
0 - 25 lb. (110 N)
0 - 100 lb. (440 N)*

In order to measure forces above 100 lb (up to 1000 lb), apply a lower drive voltage and reduce the resistance of the feedback resistor (1kΩ min.)

Recommended Circuit

$V_{out} = -V_T * (R_F/R_S)$



- * Supply Voltages should be constant
- ** Reference Resistance R_F is 1kΩ to 100kΩ
- Sensor Resistance R_S at no load is > 5MΩ
- Max recommended current: 2.5 mA

Typical Performance

Linearity (Error)	< ±3%
Repeatability	< ±2.5% of full scale
Hysteresis	< 4.5 % of full scale
Drift	< 5% per logarithmic time scale
Response Time	< 5 µsec
Operating Temperature	15°F - 140°F (-9°C - 60°C)*

Evaluation Conditions

Line drawn from 0 to 50% load
Conditioned sensor, 80% of full force applied
Conditioned sensor, 80% of full force applied
Constant load of 25 lb (111 N)
Impact load, output recorded on oscilloscope
Time required for the sensor to respond to an input force

Regulador de voltatge:

19-0227; Rev 2; 5/96

EVALUATION KIT MANUAL FOLLOWS DATA SHEET

MAXIM

Regulated 5V Charge-Pump DC-DC Converter

MAX619

General Description

The MAX619 step-up charge-pump DC-DC converter delivers a regulated 5V ±4% output at 50mA over temperature. The input voltage range is 2V to 3.6V (two battery cells).

The complete MAX619 circuit fits into less than 0.1in² of board space because it requires only four external capacitors: two 0.22µF flying capacitors, and 10µF capacitors at the input and output.

Low operating supply current (150µA max) and low shutdown supply current (1µA max) make this device ideal for small, portable, and battery-powered applications. When shut down, the load is disconnected from the input.

The MAX619 is available in 8-pin DIP and SO packages.

Features

- Regulated 5V ±4% Charge Pump
- Output Current Guaranteed over Temperature
  20mA (V_{IN} ≥ 2V)
  50mA (V_{IN} ≥ 3V)
- 2V to 3.6V Input Range
- No Inductors; Very Low EMI Noise
- Ultra-Small Application Circuit (0.1in2)
- Uses Small, Inexpensive Capacitors
- 500kHz Internal Oscillator
- Logic-Controlled 1µA Max Shutdown Supply Current
- Shutdown Disconnects Load from Input
- 8-Pin DIP and SO Packages

Applications

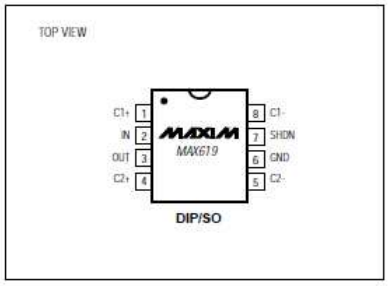
Two Battery Cells to 5V Conversion
Local 3V-to-5V Conversion
Portable Instruments & Handy-Terminals
Battery-Powered Microprocessor-Based Systems
5V Flash Memory Programmer
Minimum Component DC-DC Converters
Remote Data-Acquisition Systems
Compact 5V Op-Amp Supply
Regulated 5V Supply from Lithium Backup Battery
Switching Drive Voltage for MOSFETs in Low-Voltage Systems

Ordering Information

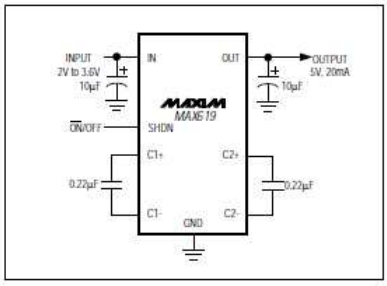
PART	TEMP. RANGE	PIN-PACKAGE
MAX619CPA	0°C to +70°C	8 Plastic DIP
MAX619CSA	0°C to +70°C	8 SO
MAX619C/D	0°C to +70°C	Dice*
MAX619EPA	-40°C to +85°C	8 Plastic DIP
MAX619ESA	-40°C to +85°C	8 SO
MAX619MJA	-55°C to +125°C	8 CERDIP

** Dice are specified at T_A = +25 °C.*

Pin Configuration



Typical Operating Circuit



For free samples & the latest literature: http://www.maxim-ic.com, or phone 1-800-998-8800

Circuit inversor:

19-3293; Rev. 2; 9/96

MAXIM

CMOS Monolithic Voltage Converter

MAX660

General Description

The MAX660 monolithic, charge-pump voltage inverter converts a +1.5V to +5.5V input to a corresponding -1.5V to -5.5V output. Using only two low-cost capacitors, the charge pump's 100mA output replaces switching regulators, eliminating inductors and their associated cost, size, and EMI. Greater than 90% efficiency over most of its load-current range combined with a typical operating current of only 120µA provides ideal performance for both battery-powered and board-level voltage conversion applications. The MAX660 can also double the output voltage of an input power supply or battery, providing +9.35V at 100mA from a +5V input.

A frequency control (FC) pin selects either 10kHz typ or 80kHz typ (40kHz min) operation to optimize capacitor size and quiescent current. The oscillator frequency can also be adjusted with an external capacitor or driven with an external clock. The MAX660 is a pin-compatible, high-current upgrade of the ICL7660.

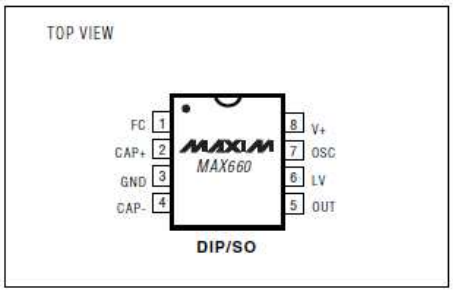
The MAX660 is available in both 8-pin DIP and small-outline packages in commercial, extended, and military temperature ranges.

For 50mA applications, consider the MAX860/MAX861 pin-compatible devices (also available in ultra-small µMAX packages).

Applications

Laptop Computers
Medical Instruments
Interface Power Supplies
Hand-Held Instruments
Operational-Amplifier Power Supplies

Pin Configuration



Features

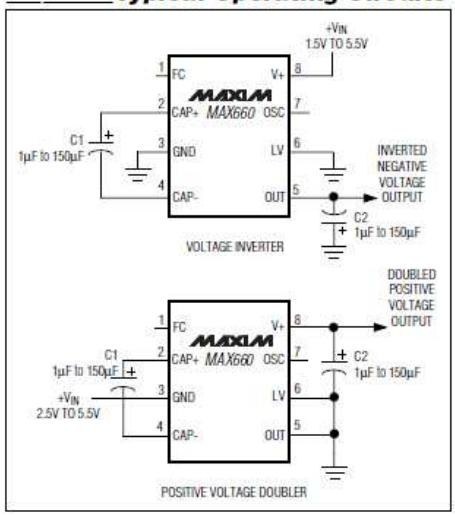
- Small Capacitors
- 0.65V Typ Loss at 100mA Load
- Low 120µA Operating Current
- 6.5Ω Typ Output Impedance
- Guaranteed ROUT < 15Ω for C1 = C2 = 10µF
- Pin-Compatible High-Current ICL7660 Upgrade
- Inverts or Doubles Input Supply Voltage
- Selectable Oscillator Frequency: 10kHz/80kHz
- 88% Typ Conversion Efficiency at 100mA (I_L to GND)

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX660CPA	0°C to +70°C	8 Plastic DIP
MAX660CSA	0°C to +70°C	8 SO
MAX660C/D	0°C to +70°C	Dice*
MAX660EPA	-40°C to +85°C	8 Plastic DIP
MAX660ESA	-40°C to +85°C	8 SO
MAX660MJA	-55°C to +125°C	8 CERDIP

*Contact factory for dice specifications.

Typical Operating Circuits



For free samples & the latest literature: http://www.maxim-ic.com, or phone 1-800-998-8800

Microcontrolador:

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 4/8/16K Bytes of In-System Self-programmable Flash program memory
  - 256/512/512 Bytes EEPROM
  - 512/1K/1K Bytes Internal SRAM
  - Write/Erase cyles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
  - 6-channel 10-bit ADC in PDIP Package
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - DebugWIRE On-Chip Debug System
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48V/88V/168V
  - 2.7 - 5.5V for ATmega48/88/168
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - ATmega48V/88V/168V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
  - ATmega48/88/168: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Low Power Consumption
  - Active Mode:
    - 250 µA at 1 MHz, 1.8V
    - 15 µA at 32 kHz, 1.8V (including Oscillator)
  - Power-down Mode:
    - 0.1µA at 1.8V

Note: 1. See "Data Retention" on page 7 for details.

8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

ATmega48/V
ATmega88/V
ATmega168/V

Note: Not recommended for new designs

Rev. 2545R–AVR–07/09