XORNADAS ELECTRICIDADE E ELECTRÓNICA 2016

IES Universidade Laboral – Culleredo (A Coruña)
8-9 Setembro 2016

CONTROL COMPATIBLE CON ARDUINO PARA ROBÓTICA E PRÁCTICAS DIXITAIS

Javier Diz Bugarín
IES Escolas Proval (Nigrán)

Por qué facer hardware compatible con Arduino?

1) A propia filosofía de Arduino fomenta o desenvolvemento propio (hardware e software libre).

2) Por flexibilidade, para adaptarse mellor ás nosas necesidades docentes (e incluso comerciais). Pode reaproveitarse outro material existente.

3) Pode saír máis barato que mercar hardware existente (inda que non sempre, depende moito de dónde mercamos, gastos de envío, etc).

4) Para non depender do mercado "secundario" (placas de aplicación): cambios frecuentes de deseños, variabilidade de prezos, moitas fontes de produción e pequenas cantidades.

5) En troques o mercado "primario" (microcontroladores) é máis estable, os modelos teñen unha vida comercial longa e prodúcense en maiores cantidades con pouca variación de prezos.

6) Porque está entre as nosas competencias, os profesores (e alumnos) de electrónica podemos facelo, logo... por qué non?

UNHA OLLADA Ó INTERIOR DE ARDUINO

Orixes e características de Arduino:

CARACTERÍSTICAS:

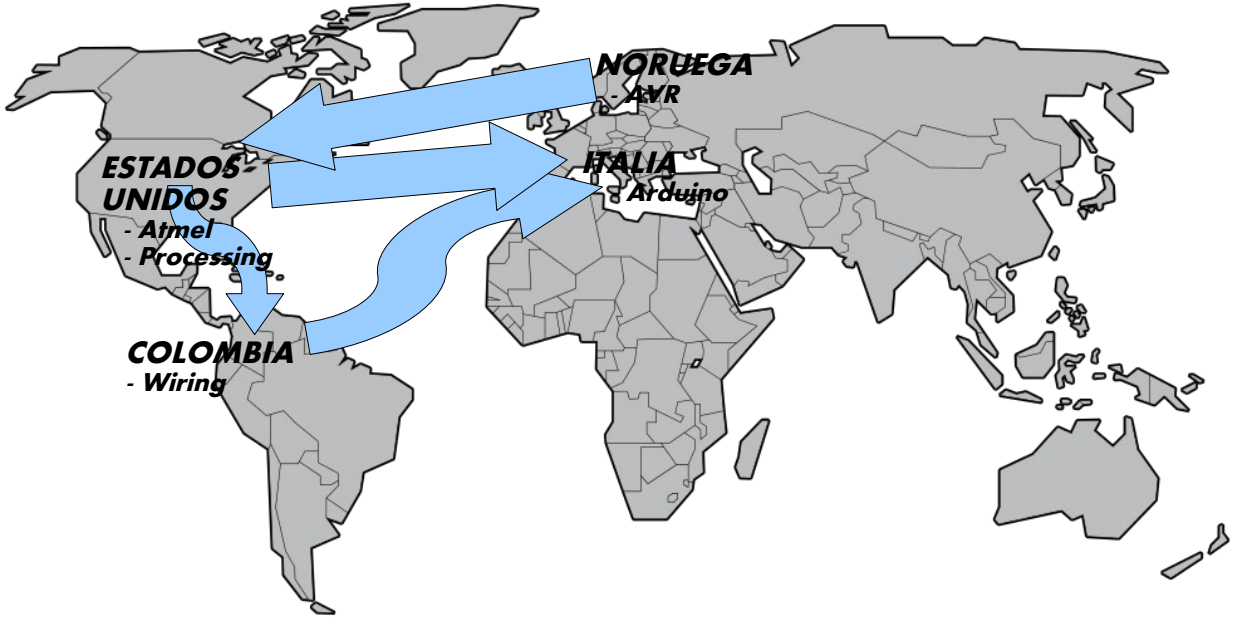
1) CONTORNO DE PROGRAMACIÓN

- Contorno de programación sinxelo
- Exemplos de programas e recursos (bibliotecas comunicacións, motores...)
- Comunidade desarrolladores, foros, moita información
- Conexión co microcontrolador cun cable usb ou adaptador, non necesita un programador (caro).
- Transferencia de código e transmisión de datos por usb
- Programas de uso libre

2) HARDWARE (MICROCONTROLADOR)

- En hardware Arduino non aporta unha novidade importante
- Usa as características dos microcontroladores Atmel con arquitectura AVR
- Placas e esquemas de uso libre, hai moitos sistemas compatibles

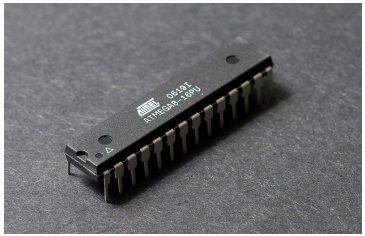
ORIXES DE ARDUINO: ISTO SÍ QUE É GLOBALIZACIÓN



MICROCONTROLADORES ATMEL-AVR

Arquitectura AVR (https://en.wikipedia.org/wiki/Atmel_AVR): microcontroladores RISC (instruccións simples e alta velocidade) deseñados por dous estudantes (Alf-Egil Bogen e Vegard Wollan) no Norwegian Institute of Technology (Noruega) en 1996.

O primeiro circuito fíxose na empresa Nordic VLSI, que vende o deseño a Atmel (EEUU), aparecendo en 1997 a familia de microcontroladores AVR, dos cales forma parte a familia ATMEGA que usa Arduino. Os dous inventores seguiron desenvolvendo a arquitectura desde Noruega para esta nova empresa.



A nova arquitectura tivo un grande éxito, En 2003 levaban vendidos máis de 500 millóns de unidades.

Inicialmente eran compatibles en patillaxe con outra familia, a 8051 de Intel (AT90S8515). Na actualidade Atmel segue a fabricar ambas familias.

Estaban pensados para a programación en linguaxe C.

Aportaban novidades importantes, como usar memoria de programa tipo Flash (en lugar de ROM/EEPROM) ou execución de instrucións nun só ciclo de oscilador o que os facía máis rápidos que outros modelos da época (que empregaban 12 ciclos ou máis).

Levan buses de comunicación serie, memoria de datos integrada e outros recursos.

A memoria de programa ten prevista unha sección de código protexido que pode empregarse para gardar un programa de arrinque, como se fora un pequeno sistema operativo.

Atmel crea tamén un contorno de aplicacións (IDE) chamado AVR Studio (actualmente Atmel Studio).

CARACTERÍSTICAS ARQUITECTURA AVR

- Multifunction, bi-directional general-purpose I/O ports with configurable, built-in pull-up resistors
- Multiple internal oscillators, including RC oscillator without external parts
- Internal, self-programmable instruction flash memory up to 256 kB (384 kB on XMega)
    - In-system programmable using serial/parallel low-voltage proprietary interfaces or JTAG
    - Optional boot code section with independent lock bits for protection
- On-chip debugging (OCD) support through JTAG or debugWIRE on most devices.
- Internal data EEPROM up to 4 kB
- Internal SRAM up to 16 kB (32 kB on XMega)
- External 64 kB little endian data space on certain models, including the Mega8515 and Mega162.
- 8-bit and 16-bit timers
    - PWM output (some devices have an enhanced PWM peripheral which includes a dead-time generator)
    - Input capture that record a time stamp triggered by a signal edge
- Analog comparator
- 10 or 12-bit A/D converters, with multiplex of up to 16 channels
- 12-bit D/A converters
- A variety of serial interfaces, including I²C RS-232, RS-485, Serial Peripheral Interface Bus
- Brownout detection
- Watchdog timer (WDT)
- Multiple power-saving sleep modes
- Lighting and motor control (PWM-specific) controller models
- CAN controller support
- USB controller support
- Ethernet controller support
- LCD controller support
- Low-voltage devices operating down to 1.8 V (to 0.7 V for parts with built-in DC–DC upconverter)
- DMA controllers and "event system" peripheral communication.

CARACTERÍSTICAS ARQUITECTURA AVR

De todo o anterior, o máis importante...

ISP (In system programming) → o microcontrolador pode programarse na mesma placa

Bootcode → o microcontrolador leva un pequeno sistema operativo

O hardware Arduino está baseado nestas dúas características, engadindo un interfaz USB e un programa cargador propio ("bootloader").

Outras características dos micros AVR-mega que usa Arduino:

Portos I/O = entradas/saidas dixitais
Conversores A/D = entradas analóxicas (sensores)
PWM = control motores e potencia, saídas analóxicas
Portos serie (SPI, I2C) = comunicación por usb, bluetooth, etc

CONTORNO DE PROGRAMACIÓN: PROCESSING

Processing: contorno programación sinxelo baseado na linguaxe C.

Creado en 2001 por Benjamin Fry e Casey Reas cando estaban no Aesthetics and Computation Group at the MIT Media Lab.

Feito en Java e multiplataforma (Linux, Windows, Mac).

Estaba pensado para facer esquemas, diagramas, visualización de datos, arte electrónico (non para microcontroladores).

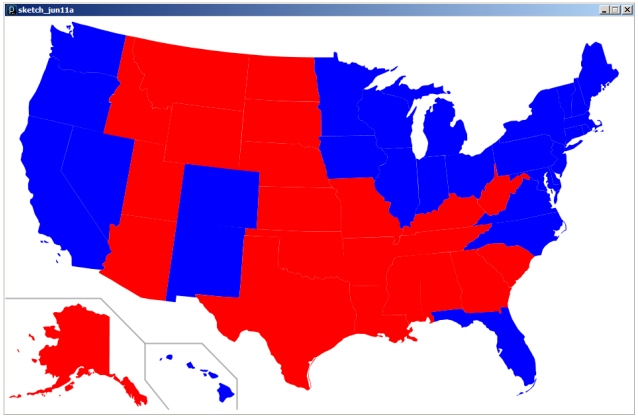
O desenvolvemento de Processing segue, en 2012 crearon a “Processing Foundation” con Daniel Shiffman, que se converte no terceiro coautor do proxecto.

https://en.wikipedia.org/wiki/Processing_%28programming_language%29
https://processing.org/

CONTORNO DE PROGRAMACIÓN: PROCESSING

Exemplo de programa en Processing que debuxa un mapa:

```
PShape usa;
PShape state;
String [] Obama  = { "HI", "RI", "CT", "MA", "ME", "NH", "VT", "NY", "NJ",
         "FL", "NC", "OH", "IN", "IA", "CO", "NV", "PA", "DE", "MD", "MI",
         "WA", "CA", "OR", "IL", "MN", "WI", "DC", "NM", "VA" };
String [] McCain = { "AK", "GA", "AL", "TN", "WV", "KY", "SC", "WY", "MT",
         "ID", "TX", "AZ", "UT", "ND", "SD", "NE", "MS", "MO", "AR", "OK",
         "KS", "LA" };
void setup() {
  size(950, 600);
  // The file Blank_US_Map.svg can be found at Wikimedia Commons
  usa = loadShape("http://upload.wikimedia.org/wikipedia/commons/3/32/Blank_US_Map.svg");
  smooth(); // Improves the drawing quality of the SVG
  noLoop();
}
void draw() {
  background(255);
  // Draw the full map
  shape(usa, 0, 0);
  // Blue denotes states won by Obama
  statesColoring(Obama , color(0, 0, 255));
  // Red  denotes states won by McCain
  statesColoring(McCain, color(255, 0, 0));
  // Save the map as image
  saveFrame("map output.png");
}
void statesColoring(String[] states, int c){
  for (int i = 0; i < states.length; ++i) {
    PShape state = usa.getChild(states[i]);
    // Disable the colors found in the SVG file
    state.disableStyle();
    // Set our own coloring
    fill(c);
    noStroke();
    // Draw a single state
    shape(state, 0, 0);
  }
}
```



WIRING: CONTORNO DE PROGRAMACIÓN + PLACA MICROCONTROLADOR

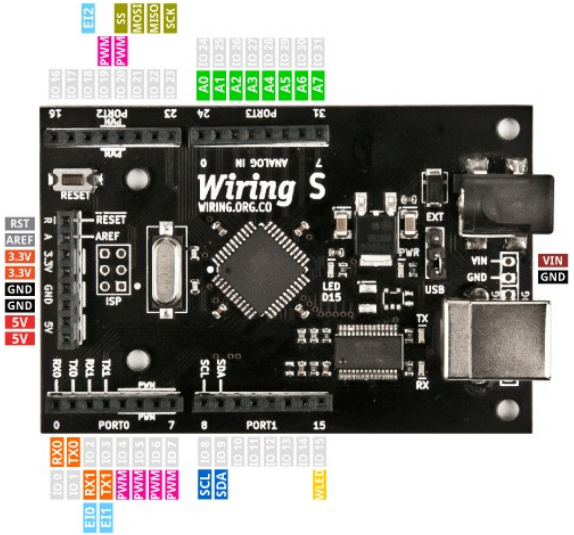
"Wiring" é unha plataforma de desenvolvemento de aplicacións electrónicas formada por unha linguaxe de programación, un contorno de desarrollo e una placa con microcontrolador.

Esta plataforma foi ideada polo colombiano Hernando Barragán en 2003 como tese doutoral no Interaction Design Institute en Ivrea, Italia. Actualmente o proxecto continúa na Escola de Arquitectura e Deseño na Universidad de Los Andes en Bogotá, Colombia.

Está baseado en Processing como contorno de programación, ó que se lle engade un conxunto de bibliotecas de funcións que facilitan o traballo con microcontroladores como *digitalRead()* ou *analogWrite()*.

A biblioteca de funcións "Wiring" segue a usarse en Arduino.

É un sistema aberto, tanto en hardware como en software, con Licenza Creative Commons.



Exemplo de placa de desenvolvemento actual de Wiring

https://en.wikipedia.org/wiki/Wiring_%28development_platform%29
http://wiring.org.co/

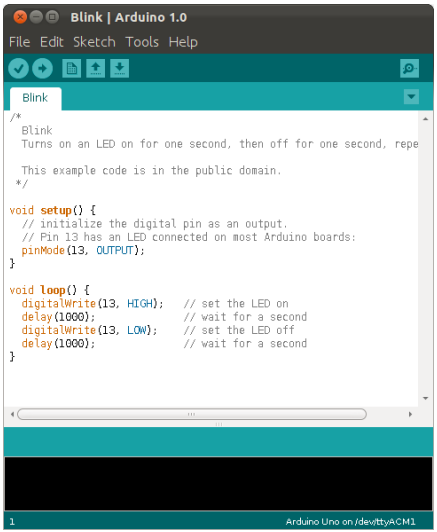
A PLATAFORMA ARDUINO

Arduino é un sistema de desenvolvemento de aplicacións electrónicas creado a partir de 2004 no Interaction Design Institute (Ivrea, Italia) a partir de Processing-Wiring, usando microcontroladores AVR-Atmega de Atmel.

O equipo resultante estaba pensado para os estudantes, era moi fácil de usar e tiña un custo moi reducido, polo que acadóu rápidamente un grande éxito,

Os autores iniciais foron Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Mellis.

Arduino ten o seu propio contorno de desenvolvemento (IDE) feito en Java, polo que tamén é multiplataforma.



A linguaxe de programación está baseada en Processing e usa a biblioteca de funcións do proxecto Wiring, e está pensado para introducir na programación a artistas ou persoas alleas á electrónica e informática.

Ten un editor de código moi sinxelo de usar e os programas se compilan e transfiren ó microcontrolador cun par de “clicks”.

Referencias:
https://en.wikipedia.org/wiki/Arduino
https://www.arduino.cc/

Contribucións ó proxecto:
https://www.arduino.cc/en/Main/Credits

A PLATAFORMA ARDUINO

Un programa feito en Arduino chámase “sketch” e os arquivos teñen a extensión “.ino”.

O programa típico Arduino ten dúas funcións:

- “setup()”, esta función contén código que se executa unha única vez ó principio do programa e se emprega para realizar operacións de inicio, establecer valores de datos,etc.

- “loop()”, esta función se repite indefinidamente ata que se desconecta a placa. Serve para revisar o estado de entradas e sensores e facer as operacións necesarias segundo o resultado (por exemplo, activar un relé, motor, led,...)

Exemplo de sketch Arduino (blink.ino)

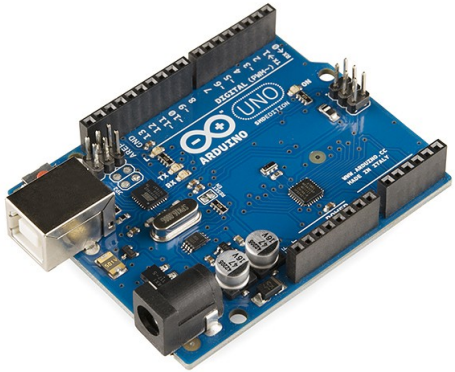
```
#define LED_PIN 13

void setup() {
    pinMode(LED_PIN, OUTPUT);       // Enable pin 13 for digital output
}

void loop() {
    digitalWrite(LED_PIN, HIGH);    // Turn on the LED
    delay(1000);                    // Wait one second (1000 milliseconds)
    digitalWrite(LED_PIN, LOW);     // Turn off the LED
    delay(1000);                    // Wait one second
}
```

A PLATAFORMA ARDUINO

Os sketches de Arduino se traducen mediante programas GNU (como o compilador AVR-GCC) que xa están incluídos no contorno de desenvolvemento. Tamén se emprega un programa de comunicacións de uso libre chamado avrdude (http://savannah.nongnu.org/projects/avrdude) para enviar o código resultante ó microcontrolador.

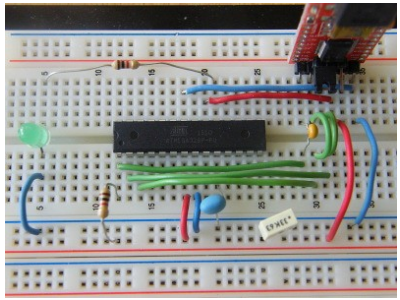
A placa de control ten un pequeno programa chamado Bootloader que tamén é parte de Arduino e se comunica co ordenador para organizar a transferencia de código. Gracias a isto non se precisan elementos externos como programadores e a tarefa de programación e depuración é rápida e sinxela.

Programador AVRISP mkII



Placa de control Arduino UnoR3

PODE FACERSE UN CONTROL COMPATIBLE CON ARDUINO?



Sí, unha placa compatible con Arduino pode facerse cun microcontrolador e uns poucos elementos máis (cuarzo, condensadores, resistencia).

Na documentación de Arduino hai un tutorial que explica cómo: "Building an Arduino on a breadboard", https://www.arduino.cc/en/Main/Standalone

A NOSA PLACA: CRITERIOS DE DESEÑO

1) O elemento principal de Arduino é o microcontrolador. Usaremos o mesmo que levan moitas das placas Arduino (duemilanove, diecimila), o **Atmega328P**. Para que sexa fácil de montar e se poidan quitar escollemos a versión Atmega328P-PU (encapsulado paralelo pdip, 28 patas, separación 0,1"=2,54mm). Ollo, a patillaxe non é igual que na versión smd!!.

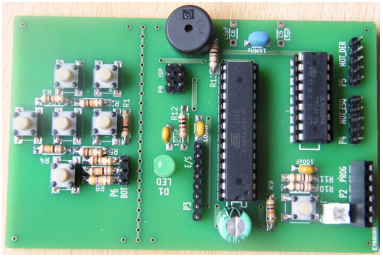
2) As placas Arduino levan o programa de inicio ("***Bootloader***"). Os micros comprados non o traen, polo que hai que gravalo. Veremos cómo facelo con outro tutorial: "From Arduino to a Microcontroller on a Breadboard", https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard
A gravación pode facerse de varias formas: cun programador tipo AVRISP, co módulo hardware ArduinoISP, ou con outra placa Arduino cargando o sketch "Arduino as ISP".

3) Para o deseño da placa usaremos como base a **Arduino Pro-Mini** (Sparkfun). Esta placa leva un conector de 6 contactos no que se enchufa un adaptador serie-usb. Este tipo de circuito non se integra na placa porque é difícil de soldar (smd) e non é imprescindible telo sempre, só cando se reprograma o micro (lembremos que estamos facendo un robot, que é un equipo autónomo e non ten que estar sempre enchufado ó ordenador).

CONTROL COMPATIBLE: CRITERIOS DE DESEÑO

4) Usaremos o contorno Arduino coas súas bibliotecas de funcións e sketches. Calquera programa feito para unha placa tipo duemilanove ou Pro-Mini servirá para ésta. Como o noso robot é unha versión do Escornabot, o mesmo sketch debería servir para os dous.

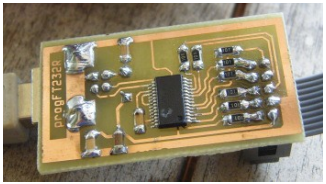
5) Faremos un circuito propio no que se integrará todo o control do robot. Non será necesario facer cableados da placa de control ó teclado, ós motores, etc. Iso evita complicacións e posibles fallos (recordemos de novo que isto é un vehículo autónomo e os cables teñen o costume de afrouxarse...).



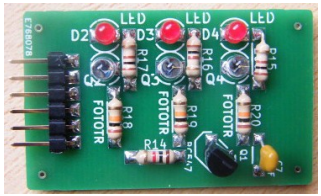
6) Haberá dous modelos de placa, un feito en placa de tiras (que haberá que cablear mediante pontes) e outro con placa de circuíto impreso. A placa de circuíto impreso poderá encargarse (para iso temos os arquivos gerber, noutro apartado veremos cómo e dónde) ou facerse se dispoñemos dun taller electrónico. O deseño das pistas facilitará a construción e montaxe, por tanto as pistas serán anchas, ben separadas, con taladros grandes (1mm ou máis).

7) Para a motorización usaremos o mesmo modelo que o Escornabot, o 28BYJ-48. É un motor paso a paso unipolar con reductora que se controla mediante o circuíto integrado ULN2003, un para cada motor. Na nosa placa tomamos a decisión de substituílos por un único circuíto ULN2803 (ou ULN2804), xa que este último ten 8 saídas de potencia e pode controlar ó mesmo tempo os dous motores. Isto reduce o tamaño da placa, elimina moitos cables e supón un pequeno aforro económico. O uso dun motor paso a paso ten vantaxes e inconvintes, é máis difícil de controlar pero permite facer moitas aplicacións nas que se necesite un control preciso de posición ademáis de movemento.

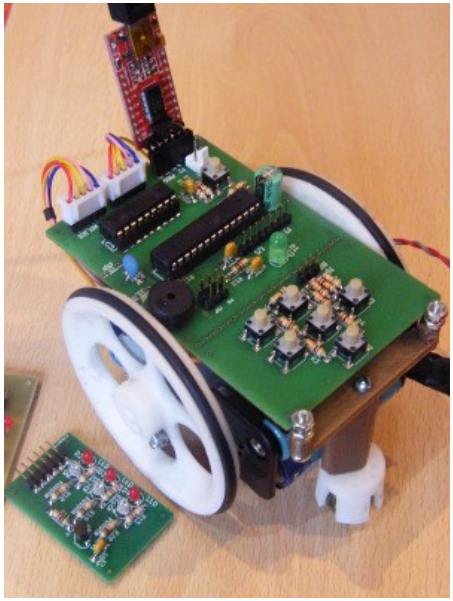
CONTROL COMPATIBLE: CRITERIOS DE DESEÑO



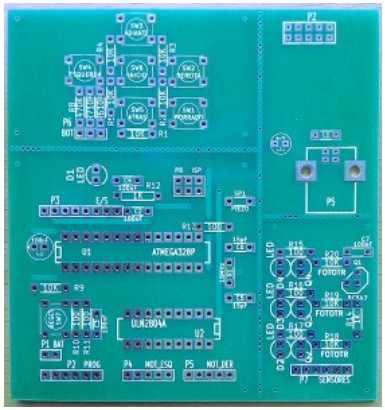
8) O módulo de comunicacións serie-usb pode mercarse ou facerse, temos un deseño de placa (feito para otro sistema de microcontroladores, o AT89 de Atmel) que se pode montar pero que é complicado de soldar.



9) Tamén faremos o deseño dun módulo de sensores infravermellos, que non estaba previsto non Escornabot pero é interesante para facer programas de seguemento de liñas.



10) Todas as placas se inseriron nun cadrado de 10x10cm para que saíra máis barato encargar a fabricación (moitos proveedores traballan con ese tamaño). Nese cadrado está o control, o teclado, a placa de sensores e a de comunicacións serie-usb.



CONTROL COMPATIBLE: CRITERIOS DE DESEÑO

11) Unha cuestión importante: a fonte de alimentación.

- Para simplificar a nosa placa non leva regulador de tensión, polo que a tensión será a mesma da pila ou do usb, dependendo se estamos en modo de programación ou autónomo.

- A tensión do usb son 5V e a das pilas pode variar entre 4 e 6V segundo o tipo de pilas, se son recargables ou non, etc. Incluso podería usarse un pack de baterías de litio (3,7-4,2V).

- Cando se conecte o adaptador usb HAI QUE DESCONECTAR A PILA, non é recomendable ter as dúas fontes ó mesmo tempo.

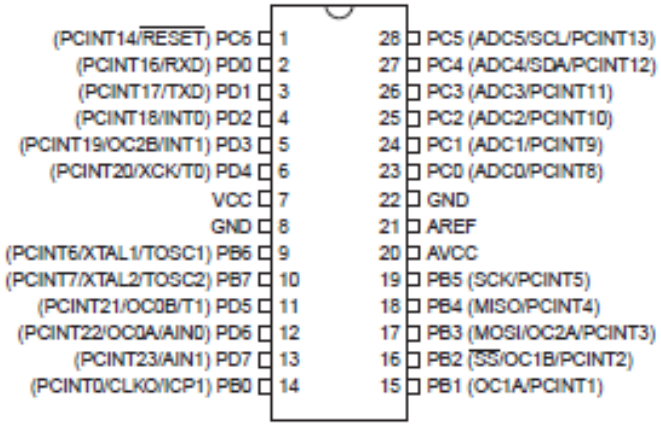
- Existe a posibilidade de colocar díodos nas dúas entradas de alimentación e así poderían estar conectadas ambas, pero tamén producen unha pequena caída de tensión. A última versión da placa xa os leva, nas anteriores poden engadirse con pequenas modificacións (na placa de tiras é inmediato e na de circuíto impreso hai que cortar as pistas nun para de puntos e soldar os díodos). Os díodos máis axeitados son de tipo Schottky (1N5817 ou similar).

12) Por último, na nova versión da placa tamén hai un conector de 4 contactos para o bus de comunicacións SPI. Este bus serve para conectar moitos periféricos e tamén para facer a programación doutros micros (bootloader).

TÁBOA DE CONEXIÓNS DO CONTROL ROBÓTICO

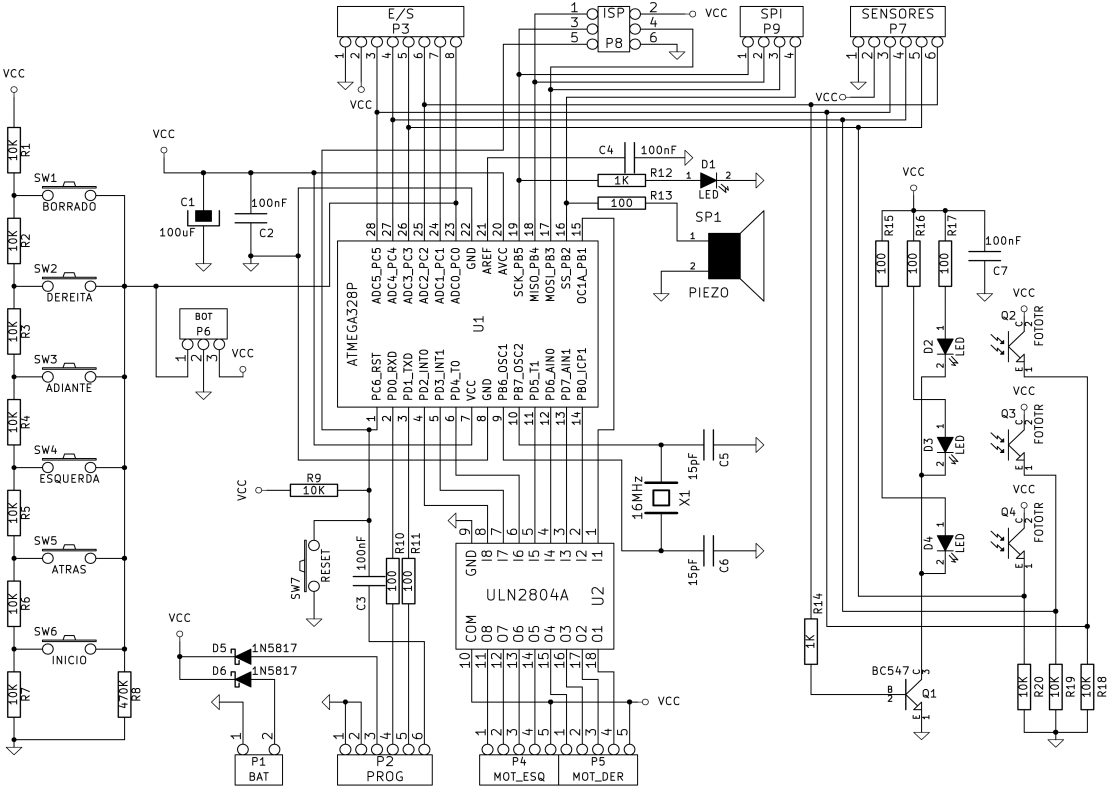
FUNCIÓN	ARDUINO	ATMEGA328P-PDIP	
	nome	nº pin	nome
motor esquerdo IN1	D2	4	PD2 (INT0)
motor esquerdo IN2	D3	5	PD3 (INT1)
motor esquerdo IN3	D4	6	PD4 (T0)
motor esquerdo IN4	D5	11	PD5 (T1)
motor dereito IN1	D6	12	PD6 (AIN0)
motor dereito IN2	D7	13	PD7 (AIN1)
motor dereito IN3	D8	14	PB0
motor dereito IN4	D9	15	PB1
altavoz (piezo)	D10	16	PB2 (SS)
led indicador	D13	19	PB5 (SCK)
botoneira analóxica	A0	23	PC0(ADC0)
led sensores	A2	25	PC2(ADC2)
fotodetector esquerda	A3	26	PC3(ADC3)
fotodetector centro	A4	27	PC4(ADC4)
fotodetector dereita	A5	28	PC5(ADC5)

CONEXIÓNS DO MICROCONTROLADOR ATMEGA328P-PU

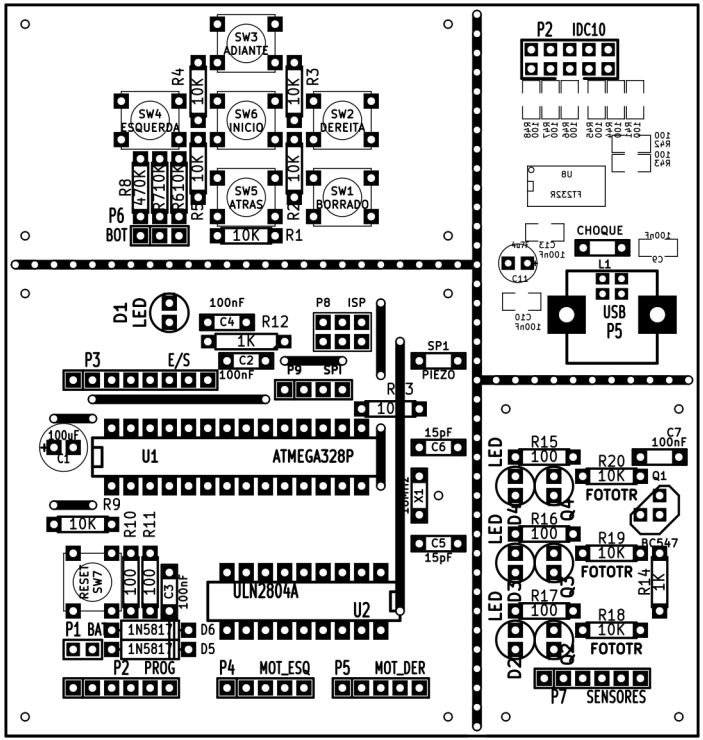


ATMEGA328P

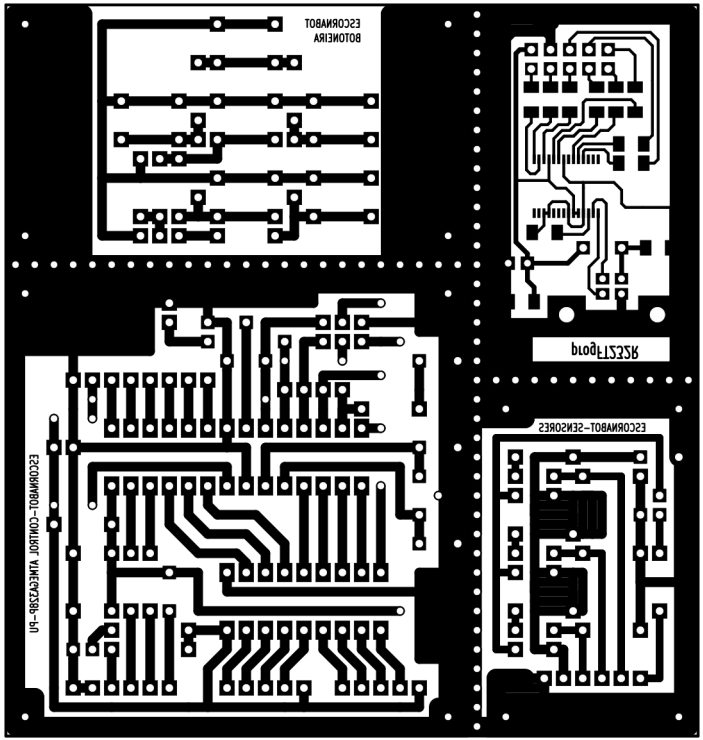
PLACA DE CIRCUITO IMPRESO: ESQUEMA ELECTRÓNICO



PLACA DE CIRCUITO IMPRESO: PLANO DE MONTAXE

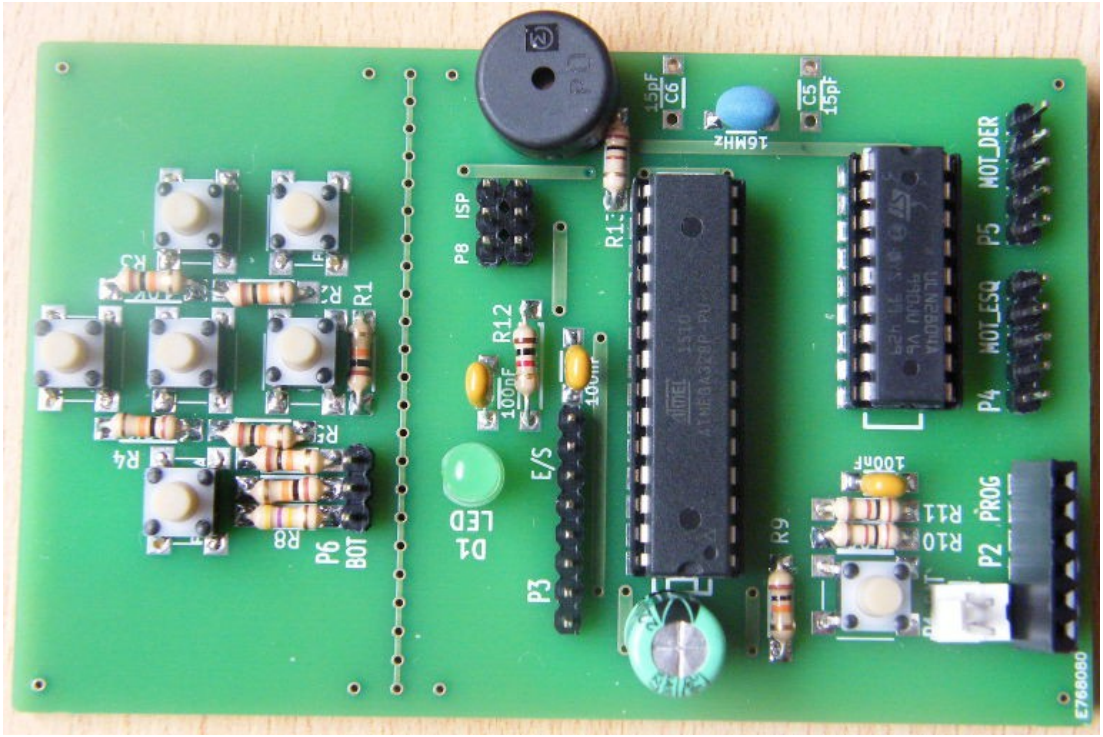


PLACA DE CIRCUITO IMPRESO: TRAZADO DE PISTAS

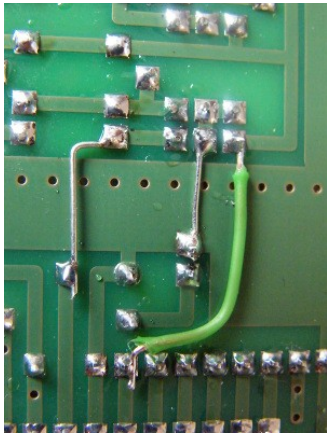


PLACA DE CIRCUITO IMPRESO: PROCESO DE MONTAXE

Imaxe final da placa de circuíto impreso montada indicando a posición e orientación de tódolos compoñentes:



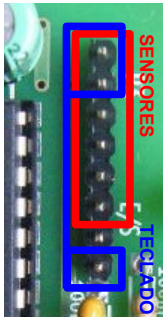
PLACA DE CIRCUITO IMPRESO: PROCESO DE MONTAXE



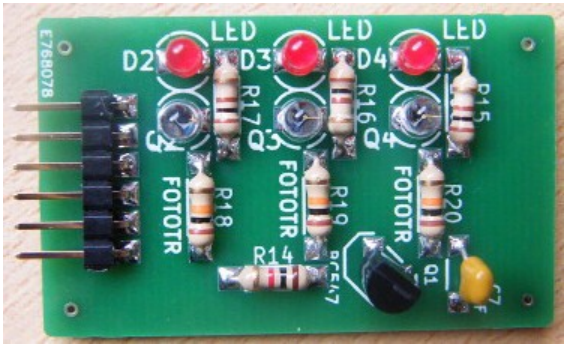
- A placa da botoneira ten unha tira de 3 contactos para conectar á placa principal. Se se manteñen unidas as dúas placas pode facerse a conexión con cables na parte traseira da forma que se indica na figura axunta. Hai que facer as conexións de +5V, GND e saída (A0).

- Na placa de sensores hai que prestar especial atención á orientación dos fototransistores e díodos. Van todos no mesmo sentido cos cátodos ou colectores orientados hacia o conector (lado esquerdo da foto). Tamén hai que fixarse na polaridade dos conectores.

- O conector de E/S da placa de control é de 8 patas porque está pensado para conectar a placa de sensores, o teclado ou como entrada/saída polivalente, para conectar só a placa de sensores hai que aliñar á esquerda deixando dous contactos libres á dereita, segundo o seguinte diagrama.



Control E/S	Fotosensores	
1 GND	GND	1
2 VCC	VCC	2
3 ADC5	FD der.	3
4 ADC4	FD cen.	4
5 ADC3	FD esq.	5
6 ADC2	LED	6
7 ADC1	(sen uso)	
8 ADC0	TECLADO	



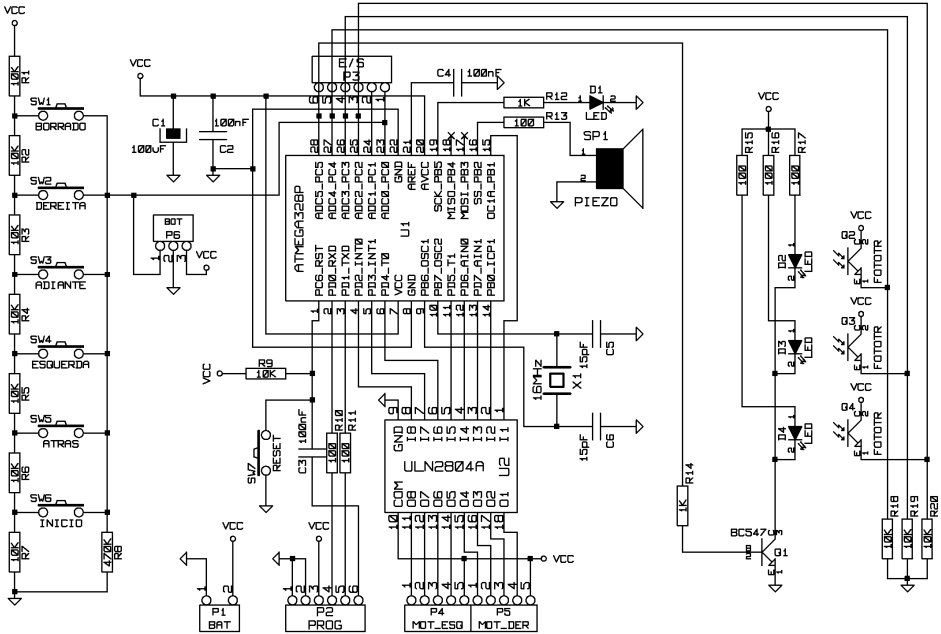
PLACA DE CIRCUITO IMPRESO: LISTA DE MATERIAL

Lista de material electrónico para a placa de control e sensores. Os prezos son orientativos, nalgúns casos con desconto por cantidade. Os códigos de Farnell inclúense como referencia e para as características detalladas.

MATERIAL	CANT	PREZO	TOTAL	
placa de circuito impreso (control+sensores) ou placa de tiras	1	4	4	
microcontrolador atmega328p-pu (Farnell 1715487)	1	1,96	1,96	
circuito integrado ULN2804 (Farnell 1094429)	1	0,5	0,5	
resonador 16mhz 3p (Farnell 2470366)	1	0,33	0,33	substitúe a un cristal de cuarzo de 16MHz e 2 condensadores de 15pF
portapila 4xAA (Farnell 1650685)	1	1,33	1,33	
zumbador piezoeléctrico (Farnell 1192513)	1	0,45	0,45	
microswitch spst-no TECONNECTIVITY (Farnell 1555985)	7	0,09	0,63	
resistencia 100 1/4W	6	0,02	0,12	
resistencia 1K 1/4W	2	0,02	0,04	
resistencia 10K 1/4W	11	0,02	0,22	
resistencia 470K 1/4W	1	0,02	0,02	
condensador cerámico 100nF	4	0,05	0,2	
condensador electrolítico 100uF 25V	1	0,1	0,1	
zócalo dip 28 patas	1	0,5	0,5	
zócalo dip 18 patas	1	0,5	0,5	
diodo led 5mm	1	0,1	0,1	
tira pin macho 2,54mm 40 contactos (Farnell 1654557)	1	1,2	1,2	2x5 (motores), 1x8 (entradas), 2x3 (isp), 1x3 (teclado), 1x6 (sensores), 1x4 (spi)
conector femia 2,54mm 6 contactos (Farnell 1593462)	1	0,6	0,6	pode substituirse por unha tira de zócalo de circuito impreso
conector molex 2,54mm 2 contactos (Farnell 9731148)	1	0,15	0,15	pode substituírse por unha tira de pin de 2 contactos (sen polaridade)
fototr. visible SFH-309-5/6 (RS 665-5381)	3	0,2	0,6	
led visible Vishay TLHA44R1S2 (RS 773-4038)	3	0,2	0,6	
transistor BC547	1	0,2	0,2	
Díodo schottky 1N5817	2	0,1	0,2	
SUMA			14,55	

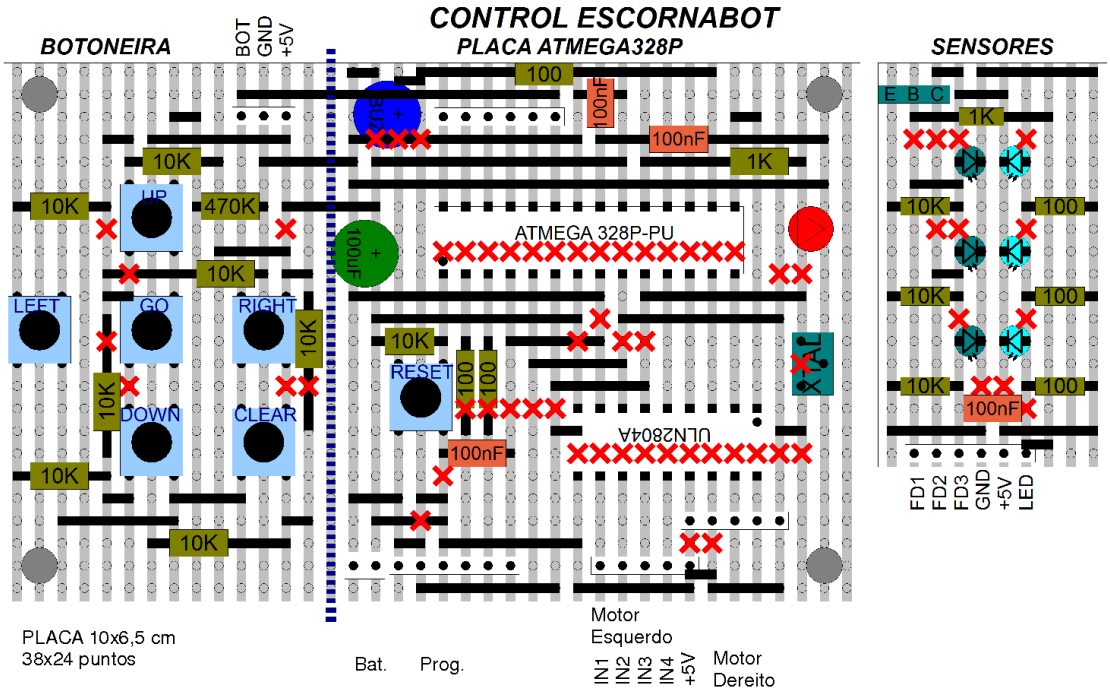
PLACA DE TIRAS: ESQUEMA ELECTRÓNICO

Este esquema é prácticamente igual que o de circuíto impreso, excepto que non leva o conector de programación (bloque de 6 terminais). Tampouco hai un conector de sensores (van cableados). Además, no esquema está previsto que o control dos leds dos sensores se faga coa saida PC5 do micro no canto de PC2 no esquema de circuito impreso (inda que isto pode cambiarse no cableado final).



PLACA DE TIRAS: PLANO DE PONTES E CORTES

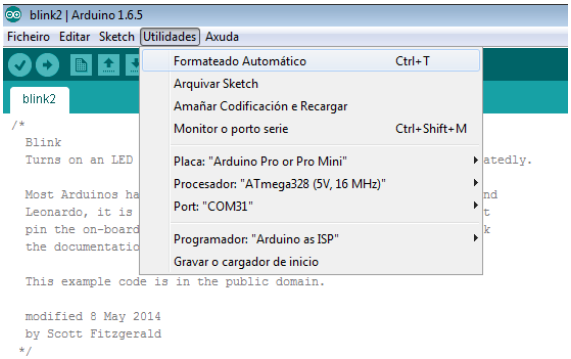
As liñas negras indican as pontes de cable e as cruces vermellas os cortes nas tiras de cobre. Para facelas recoméndase usar unha broca de 3-4mm ou elemento similar e rotar a man facendo centro no burato.
As tres placas poden separarse cortando pola liña de puntos.



GRAVACIÓN DO BOOTLOADER NUN MICRO NOVO USANDO ARDUINO COMO PROGRAMADOR

O bootloader é un programa que está sempre no microcontrolador e se comunica co contorno arduino para transferir o código ("sketch"). Os micros mercados de fábrica non o traen e hai que instalalo.
Este proceso só hai que facelo cando merquemos micros novos. Nese caso gravamos o bootloader en todos eles (leva só uns segundos) e xa quedan preparados para usar nas placas Arduino.
Se non dispoñemos dun gravador específico como o AVRISP pódese usar outra placa Arduino (Nano, Uno, Duemila,...) como programador ISP.
Este procedemento se explica no tutorial: https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard

PASOS A SEGUIR



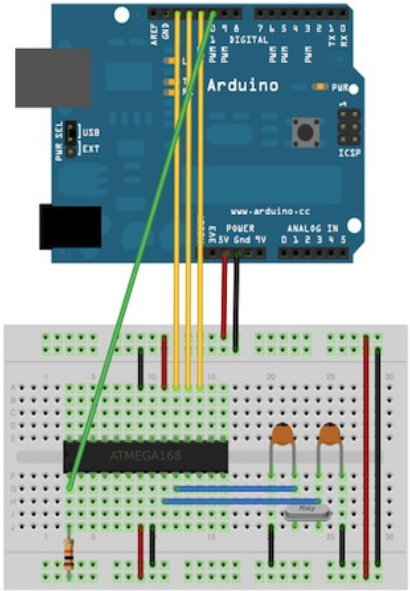
1) Configurar a placa Arduino como gravador ISP
   - Seleccionar en Utilidades" a placa da que dispoñamos.
   - Abrir o sketch "Arduino as ISP", compilar e cargar na placa.

2) Configurar o tipo de Bootloader e modo de transferencia
   - Seleccionar tipo de programador "Arduino as ISP".
   - Seleccionar o tipo de placa **que se usa como gravador** (por exemplo, Arduino Nano, Pro-Mini).
   - Seleccionar o micro que queremos gravar (Atmega328p, 5V, 16MHz).
     MOI IMPORTANTE: Seleccionar o micro correcto, o bootloader é distinto para cada tipo de micro!!!
   - Seleccionar o porto serie no que estea instalado o adaptador (COMxx). Isto só se fai nas placas que non teñan conexión usb directa (como Pro-Mini).

GRAVACIÓN DO BOOTLOADER NUN MICRO NOVO USANDO ARDUINO COMO PROGRAMADOR

3) Facer a seguinte montaxe nunha placa de probas ou similar. Se necesita un cristal de cuarzo con condensadores, unha resistencia e fío ríxido. Pode montarse tamén o led de control na pata 19 do micro (D13) para verificar a carga, xa que está conectado a unha das liñas de comunicacións.

Conexións:	ARDUINO	ATMEGA328P-PU
SS	D10	1 (RESET)
MOSI	D11	17 (MOSI)
MISO	D12	18 (MISO)
SCK	D13	19 (SCK)
cristal 16MHz	-----	9,10
+5V	-----	7, 20
GND	-----	8, 22

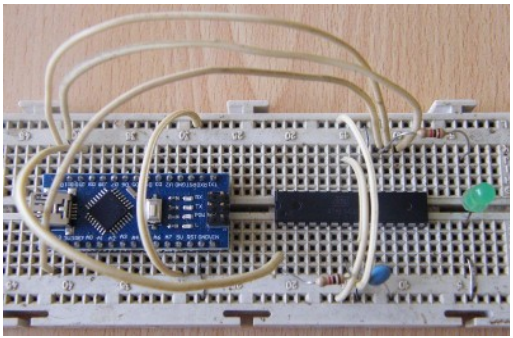


4) Unha vez conectado todo, no menú “Utilidades” seleccionar "Gravar cargador de inicio" e esperar a que remate (son uns poucos segundos).

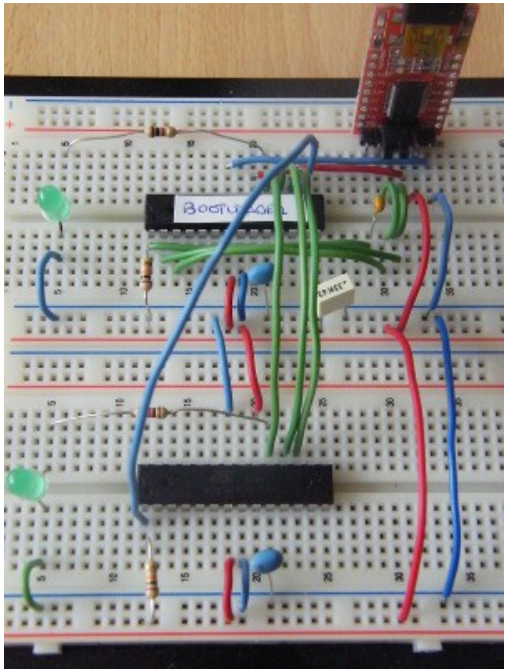
5) Se temos que gravar máis micros, se quita da placa o actual e se repite o proceso tantas veces como sexa necesario (non hai que volver a configurar nada).

GRAVACIÓN DO BOOTLOADER NUN MICRO NOVO USANDO ARDUINO COMO PROGRAMADOR

6) Se temos un micro con bootloader pódese facer unha montaxe en placa de prototipos cos dous micros, un deles actúa como programador e está conectado ó interfaz serie-usb. Cada micro leva o seu propio oscilador ou resonador cerámico.



Gravación de bootloader con Arduino Nano



Gravación con dous micros en breadboard

GRAVACIÓN DUN PROGRAMA ("SKETCH") NUN MICRO CON BOOTLOADER E ADAPTADOR USB

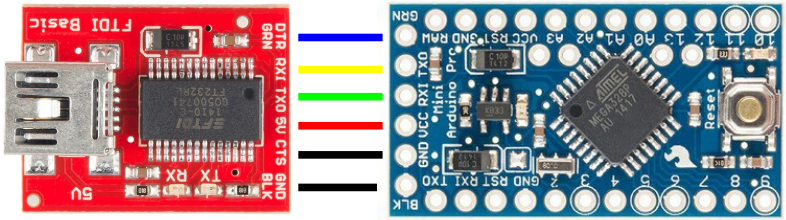
Se a placa xa está conectada ó ordenador cun adaptador serie-usb e o micro ten o *bootloader* de Arduino o proceso é o mesmo que con calquera placa de Arduino.
Para conectar a placa e o adaptador usamos a mesma configuración que no Arduino Pro-Mini. Esta placa leva un conector de comunicacións de 6 patas que se conectan ó adaptador usb.

PROCESO DE GRAVACIÓN
Seleccionar placa Arduino Pro-Mini (sen conexión usb directa).
Seleccionar o tipo de micro (Atmega328P, 5V, 16MHz)
Seleccionar o porto serie no que estea instalado o adaptador (COMxx).

Unha vez axustados estes parámetros, xa pode compilarse e transferir calquera sketch usando a opción normal (Menú "Sketch", opción "Cargar").

DIAGRAMA DE CONEXIÓNS (ARDUINO PRO-MINI)

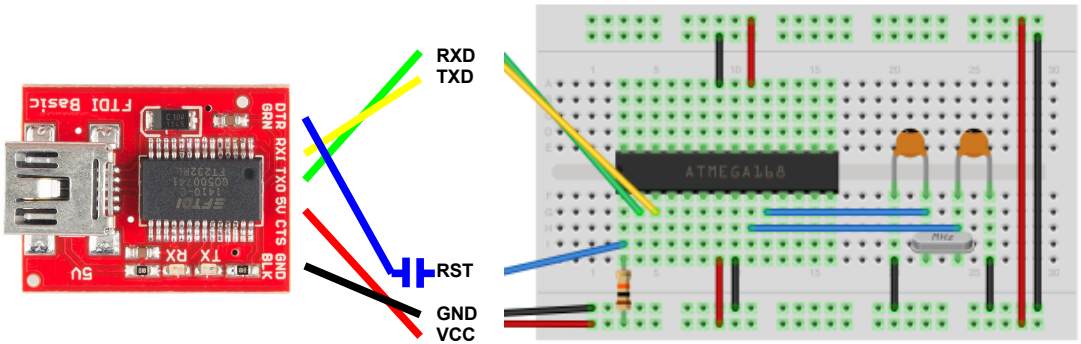
- Se queremos gravar unha placa Arduino Pro-Mini (ou compatible) a forma de conexión sería a seguinte. Son 5 liñas: GND, VCC, RXD, TXD, RST. Nos conectores hai 6 cables porque a masa está duplicada:



GRAVACIÓN DUN PROGRAMA ("SKETCH") NUN MICRO CON BOOTLOADER E ADAPTADOR USB

DIAGRAMA DE CONEXIÓNS (PLACA PROTOTIPOS)

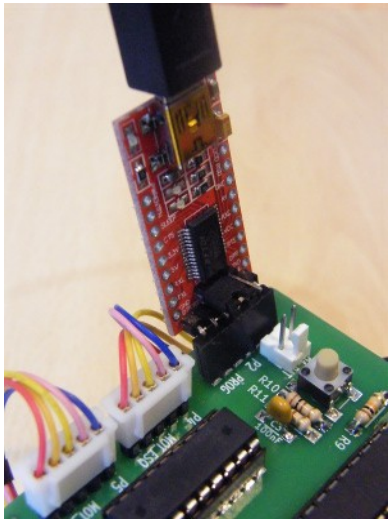
- Se o micro está montado nunha placa de prototipos a conexión sería a seguinte. Tamén son 5 liñas (GND, VCC, RXD, TXD, RST):
- Na entrada Reset hai que poñer un condensador (100nF ou máis), a placa Pro-Mini xa o leva incorporado. Se non se fai así o micro queda en reset permanente e o bootloader non se inicia.



GRAVACIÓN DUN PROGRAMA ("SKETCH") NUN MICRO CON BOOTLOADER E ADAPTADOR USB

DIAGRAMA DE CONEXIÓNS (PLACA COMPATIBLE ESCORNABOT)

- Por último, na nosa placa de control se conecta o adaptador directamente ou cun cable, dependendo do tipo de conector que puxeramos na placa (macho ou femia). O esquema de conexións é o seguinte (incluíndo tamén a opción de adaptador con conector IDC-10).



CONEXIÓNS

Arduino PRO-MINI (ou compatible)		FTDI-IDC10		ATMEGA328P-PU	
pin	nome	pin	nome	pin	nome
1	GND	4	GND	8,22	GND
2	GND	----	----	----	----
3	VCC	2	VCC	7,20	VCC
4	RXI	3	TXD	2	RXI
5	TXO	10	RXD	3	TXO
6	DTR	1	DTR	1	RESET

NOTA: As liñas RXI e TXO se consideran desde o microcontrolador, por ese motivo RXI se conecta a TX na placa FTDI (o adaptador usb envía datos e chegan á liña de recepción do microcontrolador).
A placa de control xa leva o condensador na entrada Reset (igual que a Pro-Mini).

OUTRAS OPCIÓNS DE GRAVACIÓN

OUTROS ADAPTADORES QUE SE PODEN USAR

- Pode usarse un cable de comunicacións serie-usb que teña saidas TTL, sempre que estean as conexions DTR, RX,TX, VCC, GND. Hai moitos no mercado.

- Pode usarse unha placa Arduino normal como adaptador serie-usb, nese caso hai que quitar o micro. Non valen todas, só as que teñan zócalo e se poida quitar o circuito integrado.

- Arduino vende unha placa especial chamada ArduinoISP. Esta placa fai a mesma función que o adaptador usb: https://www.arduino.cc/en/Main.ArduinoISP.
  NOTA: non confundir co sketch “ArduinoasISP”.

GRAVACIÓN DE SKETCHES SEN BOOTLOADER USANDO ARDUINO COMO GRAVADOR ISP

- Tamén se pode usar a placa Arduino como gravador para cargar directamente sketches sen usar o bootloader.
- Neste caso en vez de "cargar" hai que escoller "cargar usando o programador".
- Hai que acordarse de seleccionar o tipo de placa destino (non ten que ser a mesma do programador) antes de compilar o sketch. Tamén hai que especificar correctamente o tipo de micro e velocidade.
- A forma de conexión é igual que para gravar o bootloader.
- Esta opción é interesante se queremos usar todo o espacio de memoria dos micros, xa que pode eliminarse o bootloader.

SENSORES OPTOELECTRÓNICOS

Para completar o robot tamén se fixo o deseño dun circuito de sensores reflectivos con diodos led e fototransistores. Isto permite facer programas de seguemento de liñas marcadas no chan ou detección de obstáculos.
No mercado hai moitos sensores deste tipo para robótica. Como exemplo, está o TCRT5000, que se usa en robots como os Bitbloq (http://www.elecfreaks.com/estore/octopus-hunt-sensor.html).

- Prezo 0,6 euro aprox. Máis barato en versión TCRT5000L (patas longas)
- Distancia detección 2-15mm (20%)
- Intensidade led<60mA (tip. 20mA), tensión led 1,25V
- Saída con fototransistor
- Códigos TCRT5000 (Farnell 1470066, RS 708-5017), TCRT5000L (Farnell 1703519, RS 818-7524)

Outros sensores son o CNY70 (distancia 5mm, funciona ata 1cm, código Farnell 1470063) ou o QRD1114 (distancia 1,5mm, código Farnell 1467858).

Moitos destes sensores teñen saída dixital polo que só poden diferenciar se a luz detectada é superior ou inferior a un valor prefixado (umbral).
Como o microcontrolador Atmega ten 6 canles de entrada analóxicos é posible utilizalos para facer unha lectura máis precisa da luz recibida, con valores numéricos entre 0 (negro) e un máximo (por exemplo 1023=branco). Con esta información o algoritmo de desprazamento do robot pode mellorarse e suavizar os movementos.
Tamén optamos por usar un diodo led e fototransistor separados (en lugar dun módulo como o TCRT5000), isto da máis posibilidades á hora de escoller os compoñentes e tamén se pode modificar a súa posición na placa, orientación (por exemplo para incrementar a distancia de detección).

SENSORES OPTOELECTRÓNICOS: VISIBLES OU INFRAVERMELLOS?

- Hai materiais como moitos plásticos que son transparentes ou reflectantes ó infravermello e teñen o comportamento contrario en visible. Por exemplo os plásticos transparentes de invernadoiro reflicten o infravermello, e as radiografías que parecen oscuras son casi transparentes ó infravermello (por iso non se deben usar para ver o sol!!!). Por tanto é doado equivocarse e pensar que un material é "negro" en visible cando para infravermello non o é. Habería que usar unha cámara infravermella para asegurarse, pero iso non é fácil e menos nunha carreira de robots...
- O motivo de usar sensores infravermellos é que non lles afecte a luz ambiente, especialmente en interior con iluminación fluorescente.
- Iso pódese acadar tamén facendo unha calibración previa: hai que facer dúas lecturas consecutivas, cos leds apagados e acendidos, e restar os valores. Así se elimina o efecto ambiental e se podería usar luz visible (sempre que non varíe moito).
- Outro motivo para usar visible é que facilita o axuste de iluminación e orientación dos leds (por exemplo, para comprobar que iluminan diante dos sensores e non están desviados).
- O diodo led neste caso pode ser calquera (mellor vermello ou laranxa para aproveitar a sensibilidade e porque as luces interiores tenden a ser azul-verdes).

Para ter as dúas opcións:

- Fototransistor SFH310-FA-2/3 (infravermello, 3mm, +/-25º, fotocorrente máx. 2-3mA, RS 654-8744).
- Led: Kingbright L-34F3BT (infravermello, 3mm, +/-25º, 1.5V, RS 860-9560).
- Fototransistor visible: SFH-309-5/6 (3mm, +/-12º, RS 665-5381); SFH 310-2/3 (3mm, RS 654-8542), SFH 300-3/4 (5mm). NOTA: os modelos SFH... FA levan filtro infravermello, os que non levan FA son visibles.
- Led visible alta luminosidade: Vishay TLHA44R1S2 (625nm, +/-30º, 2,6V, RS 773-4038)

Todos estes fotodiodos e fototransistores custan 0,15-0,20 euros cada un.

CIRCUITOS IMPRESOS: CÓMO E DÓNDE ENCARGALOS?

Para facer o deseño dun circuíto impreso hai moitos programas, como o Kicad (libre) ou o Eagle (licencia gratuita limitada).
A partir destes programas pode facerse a placa no taller ou encargala a un proveedor externo. Para iso se necesita un xogo de arquivos de datos que conteñen toda a información das diferentes capas do circuíto (pistas, serigrafía, taladrado, máscaras de soldadura). O formato máis utilizado chámase Gerber.

ARQUIVOS GERBER EN KICAD

Tutorial: http://blog.iteadstudio.com/how-to-generate-gerber-files-from-kicad/
outro: https://code.google.com/p/opendous/wiki/KiCADTutorialCreatingGerberFiles
NOTA: KICAD 2013 crea correctamente os arquivos gerber para os proveedores, os anteriores non.

na opción Plot (Trazar) se exportan os arquivos:

capa	extensión
bottom layer (pistas)	.GBL
top layer (comp)	.GTL
bottom silk	.GBO
top silk	.GTO
bottom solder mask	.GBS
top solder mask	.GTS
edges	.GBR

además hai que crear o arquivo de taladrado .DRL (e logo cambiarlle o nome a .TXT para moitos proveedores). Hai que seleccionar as opcións unidades: pulgadas, formato: suppress leading zeros, precision: 2/3, orixe: absoluto, desmarcar mirror y axis. Hai que axustar diámetro de taladros a 0,040" (1mm) ou similar, tamén se pode facer cando se crea o arquivo de taladrado.

CIRCUITOS IMPRESOS: CÓMO E DÓNDE ENCARGALOS?

Hai fabricantes de circuitos impresos europeos, norteamericanos e en China, India,...
En xeral hai que ter os arquivos gerber (empaquetados nun .zip), cargalos na páxina web e en moitos casos pódense verificar antes de facer o pedido (tamén calcular o prezo). Hai que poñer as extensións correctas nos arquivos, en moitos casos o arquivo de taladrado ten que ser .TXT e o de bordes (edges) .GKO. O resto quedan igual (GBL, GTL, GBO, GTO, GBS, GTS).

En Europa:
EUROCIRCUITS: http://www.eurocircuits.com/ Caro, pero emite factura con IVE.
OLIMEX: https://www.olimex.com/ Máis económico.

EEUU: OSHPARK http://oshpark.com Para múltiplos de 3 placas ou tamaño total de 150” (múltiplos de 10).

CHINA: Calidade aceptable, son moito máis baratos, tardan bastante en entregar as placas (1-2 meses).

SEEEDSTUDIO http://www.seeedstudio.com/service/index.php?r=pcb Económico, ten unha utilidade de verificación que permite cargar e visualizar os gerber antes de facer o pedido.

ITEADSTUDIO http://www.itead.cc/open-pcb/pcb-prototyping.html Económico, non ten utilidade de verificación previa, só se pode ver despóis de facer o pedido.

OUTROS PROVEEDORES

Adaptador serie usb FT232R:

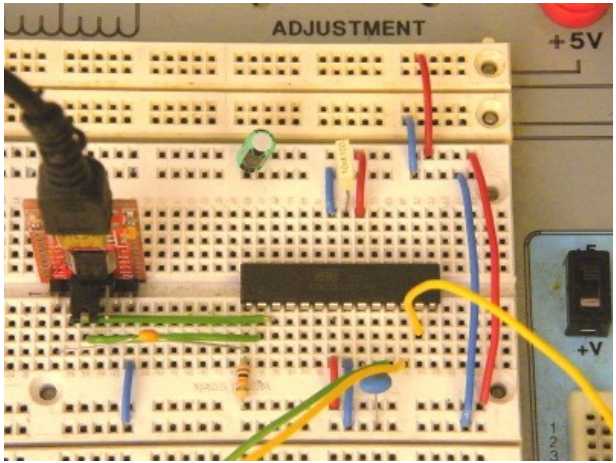
- Amazon: http://www.amazon.com/FT232RL-Serial-Adapter-Module-Arduino/dp/B00HSX3CXE
- Sparkfun: https://www.sparkfun.com/products/12731
- outro de sparkfun: https://www.sparkfun.com/products/9718
- Lightinthebox: http://www.lightinthebox.com/es/ft232rl-usb-a-serie-232-modulo-adaptador-ttl-para-funduino-blue-3-3-5v_p1141439.html
- Aliexpress: http://es.aliexpress.com/item/FT232RL-FTDI-USB-to-TTL-Serial-Adapter-Module-for-Arduino-Mini-Port-3-3V-5V/2043815349.html
- Ebay: http://www.ebay.es/sch/i.html?_from=R40&_sacat=0&_nkw=ft232rl+usb+to+serial+adapter&_sop=15

Cables USB-A mini usb-B: http://es.aliexpress.com/item/2015-Seal-Mini-USB-5-Pin-sync-cable-USB-DATA-and-charger-cable-v3-USB-2/32570474768.html

Micros Atmega328P-PU baratos: http://es.aliexpress.com/store/product/Free-shippin-5pcs-lot-ATMEGA328P-PU-ATMEGA328P-DIP-28-original-authentic/1918854_32524712043.html

APLICACIÓNS: MONTAXES EN ADESTRADOR DIXITAL

- A montaxe pode facerse aproveitando un adestrador do tipo empregado en prácticas de electrónica dixital.



- Poden aproveitarse os pulsadores existentes como entradas e os indicadores led como saídas.

- A alimentación do microcontrolador pode tomarse do adestrador, nese caso a placa usb pode desconectarse despóis de programar (incluso “en quente”) e o circuíto queda independente como en calquera outra práctica.

- Hai que ter coidado de non interconectar as alimentacións e non aplicar tensións elevadas ou negativas ó microcontrolador.

- Pode simularse o comportamento de moitos circuitos dixitais mediante programas propios feitos polo profesor ou aproveitar para que os alumnos comencen a programar.

- Nos apartados seguintes propoñemos algúns exemplos de simulación de circuítos como portas lóxicas, monoestables ou biestables.

APLICACIÓNS: SIMULACIÓN DE PORTAS LÓXICAS

```
////////////////////////////////////////////////////////
//
//  PROGRAMA SIMULADOR DE PORTAS LÓXICAS CON ARDUINO
//
////////////////////////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-PU
// con encapsulado PDIP28
// e un módulo de interfaz serie ttl-usb con ft232r

//  Taboa de verdade
//           A B   S1
// -----------------
// FILA 0:   0 0   0
// FILA 1:   0 1   0
// FILA 2:   1 0   0
// FILA 3:   1 1   1

//definicións previas
#define PIN_A 5
#define PIN_B 6
#define PIN_S1 9
#define PIN_S2 10
#define FILA0_S1 LOW  //modificar para cambiar a táboa!!
#define FILA1_S1 LOW
#define FILA2_S1 LOW
#define FILA3_S1 HIGH

// conexións do microcontrolador
/*         ---------------
           |ATMEGA328P-PU|
     RESET | PC6     PC5 | A5
       RXD | PD0     PC4 | A4
       TXD | PD1     PC3 | A3
        D2 | PD2     PC2 | A2
        D3 | PD3     PC1 | A1
        D4 | PD4     PC0 | A0
       VCC | VCC     GND | GND
       GND | GND    AREF | 100nF
     XTAL1 | PB6     VCC | VCC
     XTAL2 | PB7     PB5 | D13
        D5 | PD5     PB4 | D12
        D6 | PD6     PB3 | D11
        D7 | PD7     PB2 | D10
        D8 | PB0     PB1 | D9
           ---------------  */
```

```
////////////////////////////////////////////////////////////////

void setup() {
  // put your setup code here, to run once:
  // definición de entradas e saidas
  pinMode(PIN_A,INPUT);
  pinMode(PIN_B,INPUT);
  pinMode(PIN_S1, OUTPUT);
  pinMode(PIN_S2, OUTPUT);

  digitalWrite(PIN_S1,LOW);
  digitalWrite(PIN_S2,LOW);
}

////////////////////////////////////////////////////////////////

void loop() {
  // put your main code here, to run repeatedly:

  boolean entrada_a, entrada_b, saida_s1, saida_s2;

  // primeiro facemos a lectura das entradas:
  entrada_a=digitalRead(PIN_A);
  entrada_b=digitalRead(PIN_B);

  // táboa de verdade
  // fila 0
  if( (entrada_a==0) && (entrada_b==0) ) saida_s1=FILA0_S1;
  // fila 1
  if( (entrada_a==0) && (entrada_b==1) ) saida_s1=FILA1_S1;
  // fila 2
  if( (entrada_a==1) && (entrada_b==0) ) saida_s1=FILA2_S1;
  // fila 3
  if( (entrada_a==1) && (entrada_b==1) ) saida_s1=FILA3_S1;

  digitalWrite(PIN_S1,saida_s1);
//  digitalWrite(PIN_S2,saida_s2);
}

////////////////////////////////////////////////////////////////
```

APLICACIÓNS: MONOESTABLE

```
////////////////////////////////////////////////////////////
//
//  PROGRAMA MONOESTABLE CON ARDUINO
//
////////////////////////////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-PU con
encapsulado PDIP28
// e un módulo de interfaz serie ttl-usb con ft232r

//definicións previas
#define PIN_ENTRADA 5
#define PIN_SAIDA   9
#define TEMPO_MS 2000

// conexións do microcontrolador
/*
           ---------------
           |ATMEGA328P-PU|
     RESET | PC6     PC5 | A5
       RXD | PD0     PC4 | A4
       TXD | PD1     PC3 | A3
        D2 | PD2     PC2 | A2
        D3 | PD3     PC1 | A1
        D4 | PD4     PC0 | A0
       VCC | VCC     GND | GND
       GND | GND    AREF | 100nF
     XTAL1 | PB6     VCC | VCC
     XTAL2 | PB7     PB5 | D13
        D5 | PD5     PB4 | D12
        D6 | PD6     PB3 | D11
        D7 | PD7     PB2 | D10
        D8 | PB0     PB1 | D9
           ---------------
 */
```

```
////////////////////////////////////////////////////////////

boolean entrada_anterior;

void setup() {
  // put your setup code here, to run once:
  // definición de entradas e saídas
  pinMode(PIN_ENTRADA,INPUT);
  pinMode(PIN_SAIDA, OUTPUT);

  digitalWrite(PIN_SAIDA,LOW); //saída desactivada
  entrada_anterior=LOW; //estado inicial da entrada
}

void loop() {
  // put your main code here, to run repeatedly:

  boolean entrada_actual;

  // primeiro facemos a lectura das entradas:
  entrada_actual=digitalRead(PIN_ENTRADA);

  // compara o estado actual co anterior
  if( (entrada_actual==HIGH) && (entrada_anterior==LOW) )
  {
    digitalWrite(PIN_SAIDA,HIGH);
    delay(TEMPO_MS);
    digitalWrite(PIN_SAIDA,LOW);
  }

  //copia estado actual para o seguinte paso
  entrada_anterior=entrada_actual;

}

////////////////////////////////////////////////////////////
```

APLICACIÓNS: MULTIVIBRADOR AESTABLE

```
////////////////////////////////////////////////////////////
//
//  PROGRAMA MULTIVIBRADOR AESTABLE CON ARDUINO
//
////////////////////////////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-PU con
encapsulado PDIP28
// e un módulo de interfaz serie ttl-usb con ft232r

//definicións previas
#define PIN_ENTRADA 5
#define PIN_SAIDA   9
#define TEMPO1_MS 100
#define TEMPO2_MS 100

// conexións do microcontrolador
/*
           ---------------
           |ATMEGA328P-PU|
     RESET | PC6     PC5 | A5
       RXD | PD0     PC4 | A4
       TXD | PD1     PC3 | A3
        D2 | PD2     PC2 | A2
        D3 | PD3     PC1 | A1
        D4 | PD4     PC0 | A0
       VCC | VCC     GND | GND
       GND | GND    AREF | 100nF
     XTAL1 | PB6     VCC | VCC
     XTAL2 | PB7     PB5 | D13
        D5 | PD5     PB4 | D12
        D6 | PD6     PB3 | D11
        D7 | PD7     PB2 | D10
        D8 | PB0     PB1 | D9
           ---------------
 */
```

```
////////////////////////////////////////////////////////////

void setup() {
  // put your setup code here, to run once:
  // definición de entradas e saídas
  pinMode(PIN_ENTRADA,INPUT);
  pinMode(PIN_SAIDA, OUTPUT);

  digitalWrite(PIN_SAIDA,LOW); //saída desactivada
}

////////////////////////////////////////////////////////////

void loop() {
  // put your main code here, to run repeatedly:

  boolean entrada;

  // primeiro facemos a lectura das entradas:
  entrada=digitalRead(PIN_ENTRADA);
  // se a entrada está activa fai un ciclo de oscilación
  if(entrada==HIGH)
  {
    digitalWrite(PIN_SAIDA,HIGH);
    delay(TEMPO1_MS);
    digitalWrite(PIN_SAIDA,LOW);
    delay(TEMPO2_MS);
  }

}

////////////////////////////////////////////////////////////
```

APLICACIÓNS: BIESTABLE R-S

```
////////////////////////////////////////////////////////////
//
//  PROGRAMA BIESTABLE R-S CON ARDUINO
//
////////////////////////////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-PU con
encapsulado PDIP28
// e un módulo de interfaz serie ttl-usb con ft232r

//definicións previas
#define PIN_SET   5
#define PIN_RESET 6
#define PIN_Q     9
#define PIN_QN   10

// conexións do microcontrolador
/*
           ---------------
           |ATMEGA328P-PU|
     RESET | PC6     PC5 | A5
       RXD | PD0     PC4 | A4
       TXD | PD1     PC3 | A3
        D2 | PD2     PC2 | A2
        D3 | PD3     PC1 | A1
        D4 | PD4     PC0 | A0
       VCC | VCC     GND | GND
       GND | GND    AREF | 100nF
     XTAL1 | PB6     VCC | VCC
     XTAL2 | PB7     PB5 | D13
        D5 | PD5     PB4 | D12
        D6 | PD6     PB3 | D11
        D7 | PD7     PB2 | D10
        D8 | PB0     PB1 | D9
           ---------------
 */
```

```
////////////////////////////////////////////////////////////

boolean entrada_set, entrada_reset, saida_q, saida_qn;

void setup() {
  // put your setup code here, to run once:
  // definición de entradas e saídas
  pinMode(PIN_SET,   INPUT);
  pinMode(PIN_RESET, INPUT);
  pinMode(PIN_Q,    OUTPUT);
  pinMode(PIN_QN,   OUTPUT);

  saida_q =LOW;
  saida_qn=HIGH;
  digitalWrite(PIN_Q ,saida_q ); //saída desactivada
  digitalWrite(PIN_QN,saida_qn); //saída inversa activa
}

////////////////////////////////////////////////////////////

void loop() {
  // put your main code here, to run repeatedly:

  // primeiro facemos a lectura das entradas:
  entrada_set   = digitalRead(PIN_SET  );
  entrada_reset = digitalRead(PIN_RESET);

  // se a entrada set está alta activa Q e desactiva QN
  if(entrada_set==HIGH) { saida_q=HIGH; saida_qn=LOW; }

  // se a entrada reset está alta activa Q e desactiva QN
  if(entrada_reset==HIGH) { saida_q=LOW; saida_qn=HIGH; }

  // transfire os valores das variables ós pins de saída
  digitalWrite(PIN_Q ,saida_q );
  digitalWrite(PIN_QN,saida_qn);
}

////////////////////////////////////////////////////////////
```

APLICACIÓNS: CONTADOR BINARIO/DECIMAL 4 BITS

```
////////////////////////////////////////////////////////////////
//
//  PROGRAMA CONTADOR BINARIO/DECIMAL 4 BITS CON ARDUINO
//
////////////////////////////////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-PU (PDIP28)
// e un módulo de interfaz serie ttl-usb con ft232r

//definicións previas
#define PIN_CLK   5
#define PIN_RESET 6
#define PIN_Q0    9
#define PIN_Q1   10
#define PIN_Q2   11
#define PIN_Q3   12

#define MAX_CONTADOR 9 //valor máximo de conta: 9 para decimal, 15 para
binario


/* conexións do díxito (SA56) e decodificador (7447)

        g  f  AC a  b           VCC f  g  a  b  c  d  e
        10 9  8  7  6           16 15 14 13 12 11 10  9
        |  |  |  |  |            |  |  |  |  |  |  |  |
        -------------            ----------------------
        |  XXXXXXX  |            |                      |
        | X       X |            |                      |
        | X       X |            |                      |
        |  XXXXXXX  |            |        7447          |
        | X       X |            |                      |
        | X       X |            |                      |
        |  XXXXXXX  |            |                      |
        -------------            ----------------------
        |  |  |  |  |            |  |  |  |  |  |  |  |
        1  2  3  4  5            1  2  3  4  5  6  7  8
        e  d  AC c DP            A1 A2 LT BI RBI A3 A0 GND
 */
////////////////////////////////////////////////////////////////
// define variables globais de entradas e saidas
boolean entrada_clk_anterior, entrada_clk_actual, entrada_reset;
boolean saida_q0, saida_q1, saida_q2, saida_q3;
unsigned char contador;

void setup() {
  // put your setup code here, to run once:
  // definición de entradas e saídas
  pinMode(PIN_CLK,   INPUT);
  pinMode(PIN_RESET, INPUT);
  pinMode(PIN_Q0,   OUTPUT);
  pinMode(PIN_Q1,   OUTPUT);
  pinMode(PIN_Q2,   OUTPUT);
  pinMode(PIN_Q3,   OUTPUT);

  entrada_clk_anterior  = digitalRead(PIN_CLK);

  contador=0;
  saida_q0=bitRead(contador,0);
  saida_q1=bitRead(contador,1);
  saida_q2=bitRead(contador,2);
  saida_q3=bitRead(contador,3);
  digitalWrite(PIN_Q0,saida_q0);
  digitalWrite(PIN_Q1,saida_q1);
  digitalWrite(PIN_Q2,saida_q2);
  digitalWrite(PIN_Q3,saida_q3);
}
////////////////////////////////////////////////////////////////

void loop() {
  // put your main code here, to run repeatedly:

  // primeiro facemos a lectura das entradas:
  entrada_clk_actual  = digitalRead(PIN_CLK);
  entrada_reset = digitalRead(PIN_RESET);

  // compara o estado actual co anterior, se hai cambio avanza o contador
  if( (entrada_clk_actual==HIGH) && (entrada_clk_anterior==LOW) )
  { contador++;
    if (contador==MAX_CONTADOR+1) contador=0;
  }

  // se a entrada reset está alta reinicia a conta
  if(entrada_reset==HIGH) { contador=0; }

  // transfire os valores das variables ós pins de saída
  saida_q0=bitRead(contador,0);
  saida_q1=bitRead(contador,1);
  saida_q2=bitRead(contador,2);
  saida_q3=bitRead(contador,3);
  digitalWrite(PIN_Q0,saida_q0);
  digitalWrite(PIN_Q1,saida_q1);
  digitalWrite(PIN_Q2,saida_q2);
  digitalWrite(PIN_Q3,saida_q3);

  //copia estado actual para o seguinte paso
  entrada_clk_anterior=entrada_clk_actual;
}
////////////////////////////////////////////////////////////////
```

APLICACIÓNS: CONTROL DE MOTOR PASO A PASO UNIPOLAR

```
////////////////////////////////////////////////////////////////
//
//  PROGRAMA CONTROL DE MOTOR PASO A PASO UNIPOLAR CON ARDUINO
//
////////////////////////////////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-PU (PDIP28)
// e un módulo de interfaz serie ttl-usb con ft232r

//definicións previas
#define PIN_AV    5
#define PIN_RET   6
#define PIN_Q0    9
#define PIN_Q1   10
#define PIN_Q2   11
#define PIN_Q3   12

#define MAX_CONTADOR 7
#define RETARDO_MS 10

////////////////////////////////////////////////////////////////
// define variables globais de entradas e saidas
boolean entrada_avance, entrada_retroceso;
boolean saida_q0, saida_q1, saida_q2, saida_q3;
unsigned char contador;

void setup() {
  // put your setup code here, to run once:
  // definición de entradas e saidas
  pinMode(PIN_AV,    INPUT);
  pinMode(PIN_RET,   INPUT);
  pinMode(PIN_Q0,   OUTPUT);
  pinMode(PIN_Q1,   OUTPUT);
  pinMode(PIN_Q2,   OUTPUT);
  pinMode(PIN_Q3,   OUTPUT);

  contador=0;
  // despraza o motor á posición indicada
  motor_paso(contador);

}

////////////////////////////////////////////////////////////////
void motor_paso(unsigned char valor)
{
  unsigned char posicion;

  posicion=0;

  // selecciona patrón de bits de saida en función da variable
  switch(valor)
  { case 0:  posicion=B00000001; break;
    case 1:  posicion=B00000011; break;
    case 2:  posicion=B00000010; break;
    case 3:  posicion=B00000110; break;
    case 4:  posicion=B00000100; break;
    case 5:  posicion=B00001100; break;
    case 6:  posicion=B00001000; break;
    case 7:  posicion=B00001001; break;
    default: posicion=B00000000; break; }

  saida_q0=bitRead(posicion,0);
  saida_q1=bitRead(posicion,1);
  saida_q2=bitRead(posicion,2);
  saida_q3=bitRead(posicion,3);

  digitalWrite(PIN_Q0,saida_q0);
  digitalWrite(PIN_Q1,saida_q1);
  digitalWrite(PIN_Q2,saida_q2);
  digitalWrite(PIN_Q3,saida_q3);
}
////////////////////////////////////////////////////////////////

void loop() {
  // put your main code here, to run repeatedly:

  // primeiro facemos a lectura das entradas:
  entrada_avance    = digitalRead(PIN_AV);
  entrada_retroceso = digitalRead(PIN_RET);

  // se está activa algunha entrada actualiza estado
  if( (entrada_avance==HIGH) || (entrada_retroceso==HIGH) )
  {
    // se a entrada avance está activa incrementa cíclicamente o contador
    if(entrada_avance==HIGH)
    { contador++;
      if(contador>MAX_CONTADOR) contador=0;
      delay(RETARDO_MS); }

    // se a entrada retroceso está activa decrementa cíclicamente o contador
    if(entrada_retroceso==HIGH)
    { contador--;
      if(contador>MAX_CONTADOR) contador=MAX_CONTADOR;
      delay(RETARDO_MS); }

    // despraza o motor á posición indicada
    motor_paso(contador);
  }
  else
    motor_paso(MAX_CONTADOR+1); //saida 0
}
////////////////////////////////////////////////////////////////
```

OUTRAS PROPOSTAS DE PRÁCTICAS

- Outros exemplos: decodificadores binarios ou 7 segmentos, multiplexores, …
- Contadores binarios ou decimais, ascendentes ou descendentes. Pode facerse un programa combinado cun decodificador de 7 segmentos e controlar directamente un visualizador.
- Lectura de sensores analóxicos (o microcontrolador ten incorporado un convertedor AD de 6 canles): fotodiodos, reflectivo TCRT5000, resistencias NTC, ...
- Control de visualizadores LCD.
- Control de motores paso a paso ou lineais.
- Pequenos autómatas para prácticas de domótica.
- E moitas máis, usando toda a información existente sobre Arduino


FIN DA PRESENTACIÓN
E
GRAZAS POLA ATENCIÓN !!
javier.diz@edu.xunta.es