

The background of the slide is a photograph of a table covered with various electronic components. In the foreground, several white breadboards are visible, some with blue ribbon cables plugged into them. The table is also covered with numerous small electronic parts, including resistors, capacitors, integrated circuits, and other components, some of which are organized into small compartments or trays. The overall scene suggests a workshop or a laboratory setting for electronics education or experimentation.

XORNADAS ELECTRICIDADE E ELECTRÓNICA 2019

**IES Politécnico - Vigo
10 Maio 2019**

***MÓDULOS DE PRÁCTICAS DE ELECTRÓNICA
COMPATIBLES CON ARDUINO***

**Javier Diz Bugarín
IES Escolas Proval (Nigrán)**

A PLATAFORMA DE DESENVOLVEMENTO ARDUINO

Orixes e características de Arduino:

CARACTERÍSTICAS:

1) CONTORNO DE PROGRAMACIÓN

- Contorno de programación sinxelo
- Exemplos de programas e recursos (bibliotecas comunicacións, motores...)
- Comunidade desenvolvedores, 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

A PLATAFORMA DE DESENVOLVEMENTO 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 acadou 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 transfírense ó microcontrolador cun par de “clicks”.

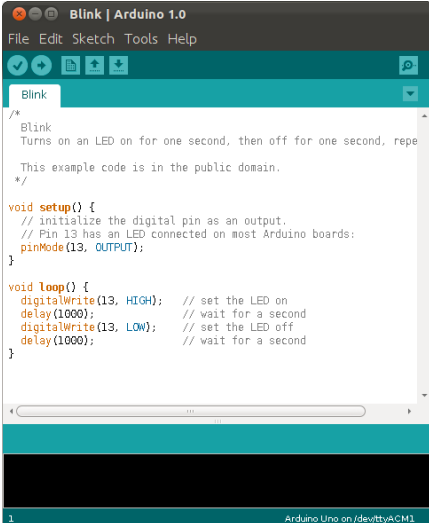
Referencias:

<https://en.wikipedia.org/wiki/Arduino>

<https://www.arduino.cc/>

Contribucións ó proxecto:

<https://www.arduino.cc/en/Main/Credits>



```
Arduino 1.0
File Edit Sketch Tools Help
Blink
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repe
 *
 * This example code is in the public domain.
 */
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}

1 Arduino Uno on /dev/ttyACM1
```

A PLATAFORMA DE DESENVOLVEMENTO 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
}
```

HARDWARE COMPATIBLE CON ARDUINO

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>

CRITERIOS DE DISEÑO

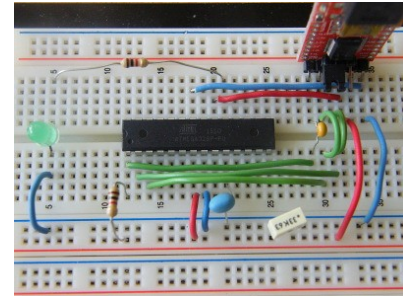
1) O elemento principal de Arduino é o microcontrolador. Usaremos o mesmo que levan moitas das placas Arduino (duemilanove, Uno, Nano), o

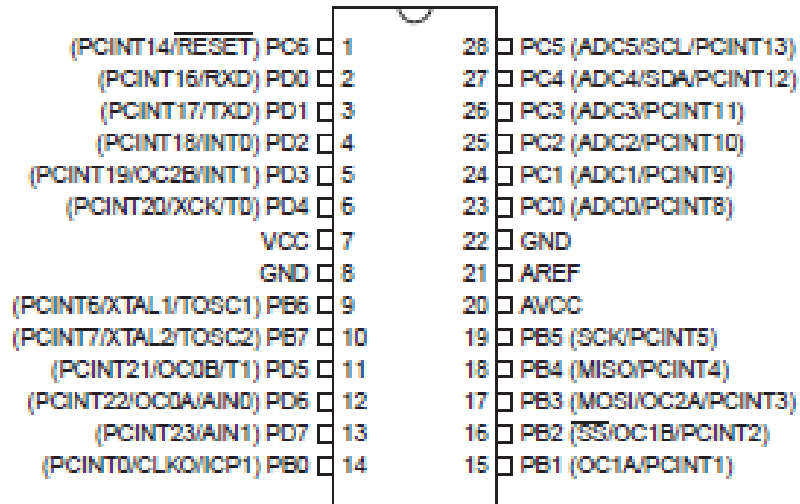
Atmega328P. Para que sexa fácil de montar e se poidan quitar escollemos a versión Atmega328P-PU (encapsulado paralelo pdip, 28 patas, separación 0,1”). Olo, a patillaxe non é igual que na versión smd!!.

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).



CONEXIÓNS DO MICROCONTROLADOR ATMEGA328P-PU

ATMEGA328P

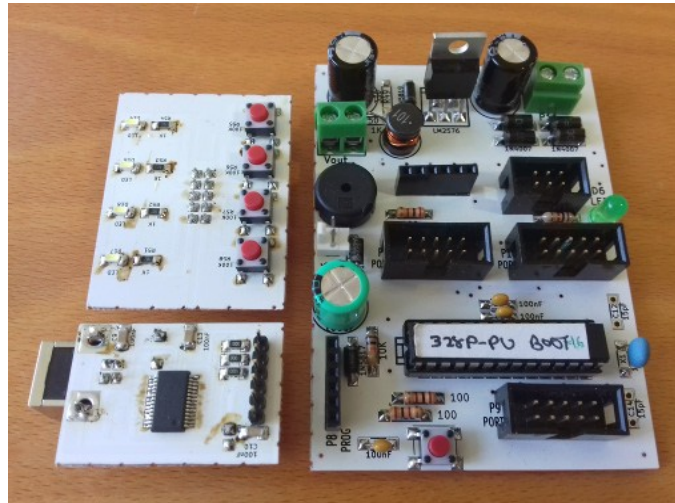
HARDWARE COMPATIBLE CON ARDUINO

Por qué facer hardware compatible con Arduino?

- 1) A filosofía de Arduino fomenta o desenvolvemento propio (hardware e software libre).
- 2) Por flexibilidade, para adaptarse mellor ás nosas necesidades docentes (e incluso comerciais). Podemos facer un deseño a medida para cada aplicación. Tamén 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?

A NOSA PROPOSTA: SISTEMA ARDUINO PROVAL

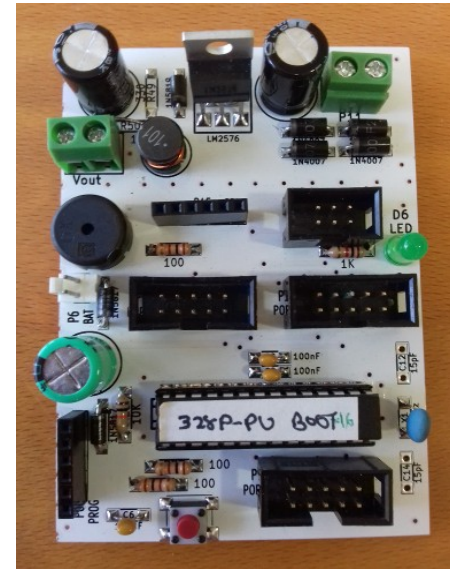
- 1) Diseñar unha placa con microcontrolador compatible co contorno de programación Arduino. O deseño efectuado será libre e estará a disposición dos alumnos ou calquera persoa interesada.
- 2) Usar hardware existente que se poida conectar a esta placa mediante pontes de cable, montaxe e placas de prototipos, etc. igual que con calquera outra placa Arduino.
- 3) Crear un conxunto de módulos propios para diferentes aplicacións: sensores, motores).
- 4) Partindo dos coñecementos adquiridos, facer pequenos sistemas completos específicos con microcontrolador (exemplos: miniautomata, mando a distancia, robot...).



SISTEMA ARDUINO PROVAL: PLACA DE CONTROL

CARACTERÍSTICAS

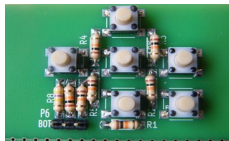
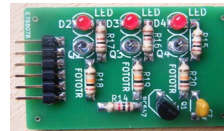
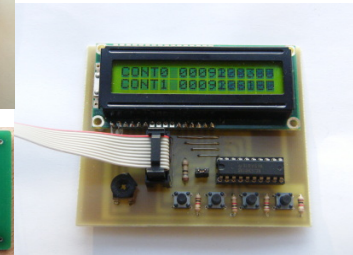
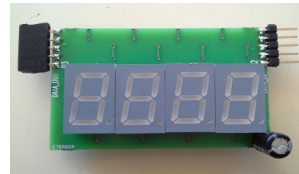
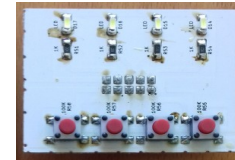
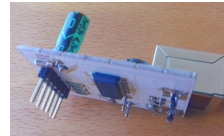
- **SOFTWARE:** Placa compatible con contorno IDE Arduino
- **HARDWARE:** Placa compatible con Arduino Pro-Mini
- Microcontrolador **Atmega328P-PU** (encapsulado PDIP28)
- Fonte conmutada reductora con **LM2576/LM2596** para alimentación externa **AC/DC** ou **batería** (9-24V).
- Usa un adaptador serie-usb externo (pode mercarse ou facerse). Podería ir integrado, pero mellor rachar coa asociación: **Arduino = USB+PC**.
- Conector **SIL-6pin** para módulo USB-serie **FT232R**.
- Conectores para periféricos tipo **IDC10** (cable plano ordenador). Permiten conexión rápida pero tamén facer montaxes permanentes.
- 3 conectores IDC10 portos micro (**PORTB**, **PORTD** 8 bits, **PORTC** 7 bits). NOTA: no porto B hai 2 pins que coinciden co oscilador, no porto C hai un que coincide con reset.
- Conector **MOLEX-2pin** con polaridade para alimentación directa 5V ou pack de baterías (3-6V).
- **LED** (D13) e **altavoz piezoeléctrico** (D10) na placa.
- Conector entrada **ISP IDC6 3x2** (para programación directa do micro mediante un programador externo).
- conector saída **SPI IDC6 3x2** (para conexión periféricos serie, programación).
- **Placa de circuito impreso** con trazado de pistas sinxelo que pode facerse mediante insoladora ou encargala a un fabricante externo (arquivos gerber).



SISTEMA ARDUINO PROVAL

PLACAS DE PERIFÉRICOS

- 1) adaptador serie-usb con circuito integrado FT232R
- 2) placa 4 leds e 4 pulsadores
- 3) visualizador numérico 4 díxitos estático con rexistros de desprazamento 74595
- 4) módulo visualizador con LCD 2x16 e 4 teclas.
- 5) módulo 3 fotosensores para robótica (segueliñas)



- 6) teclado analóxico 6 pulsadores

- 7) módulo control 2 motores paso a paso unipolares 4 fases con ULN2804A

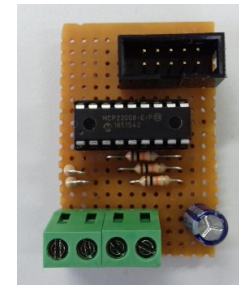
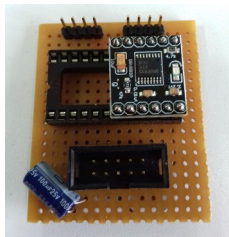
- 8) módulo control 2 motores dc/pap con DRV8833

- 9) módulo control 2 motores pap bipolares con DRV8825

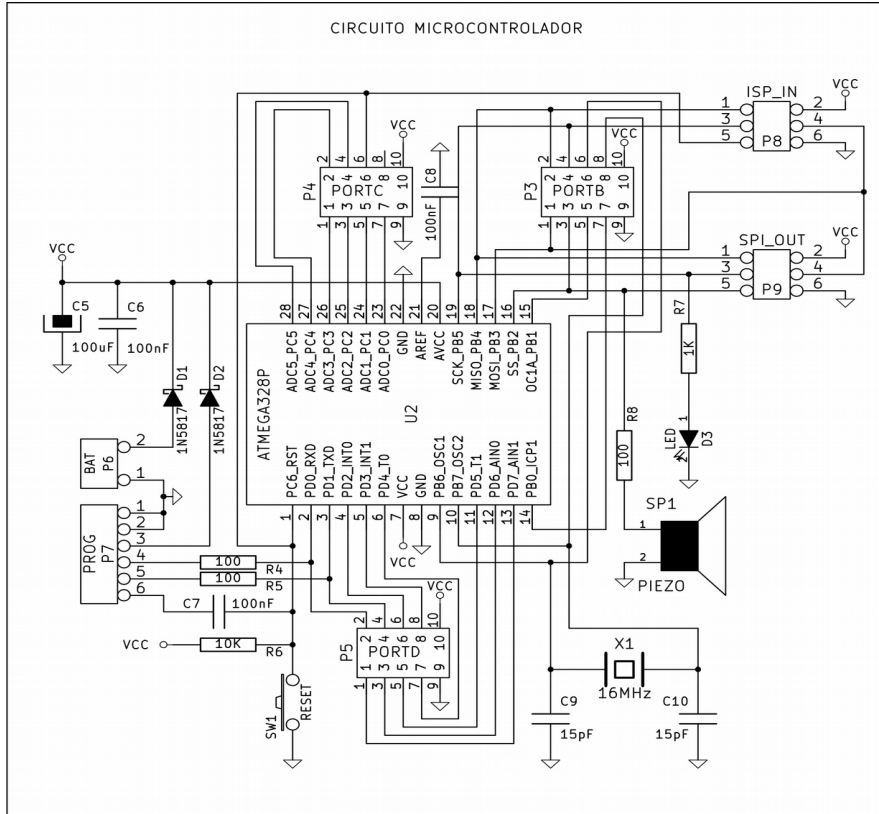
- 10) módulo de expansión E/S dixitais i2c con MCP23008

- 11) módulo transmisor/receptor de radiofrecuencia 433MHz

- 12) módulo receptor de infravermellos para telexendo

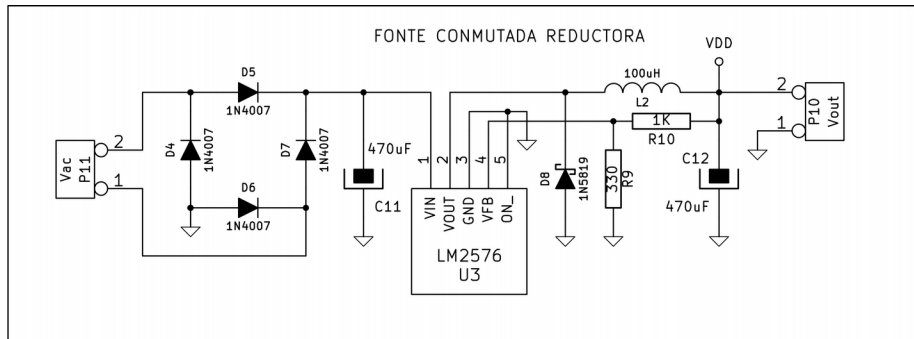


SISTEMA ARDUINO PROVAL: ESQUEMA CIRCUÍTO MICROCONTROLADOR



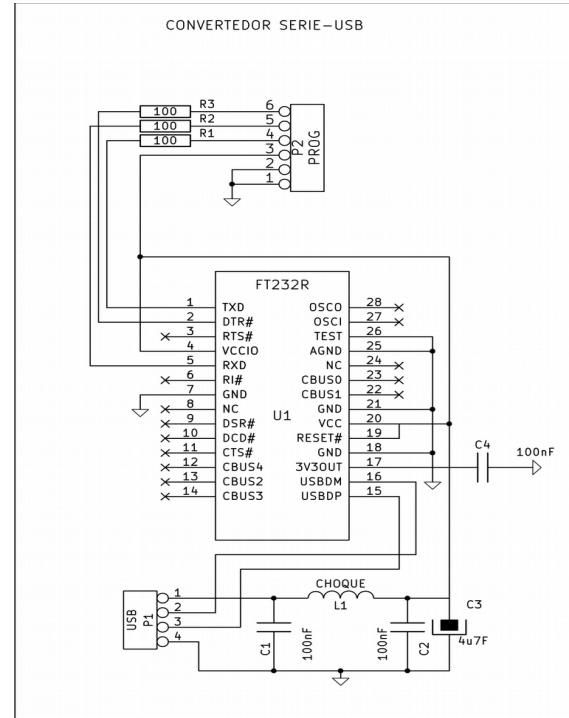
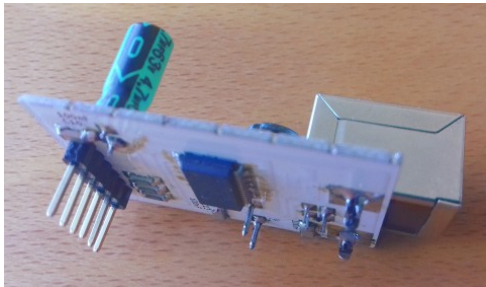
SISTEMA ARDUINO PROVAL: ESQUEMA FONTE CONMUTADA

- A fonte de alimentación está deseñada cun regulador conmutado LM2576/2596 para reducir perdas e queceamento cando o consumo sexa elevado ou se empreguen baterías (ex. 12V) para un sistema autónomo.
- O esquema e placa da fonte son independentes do circuíto do microcontrolador, pode separarse da placa se non é necesaria ou usala para outra finalidade (prácticas).
- A entrada non ten polaridade, admite tensión alterna ou continua. Leva unha ponte de díodos e condensador de filtro.
- O regulador é a versión axustable (LM2576-ADJ), podería usarse a versión de 5V pero isto permite adaptala para outros usos con diferente tensión de saída (3,3V, etc).

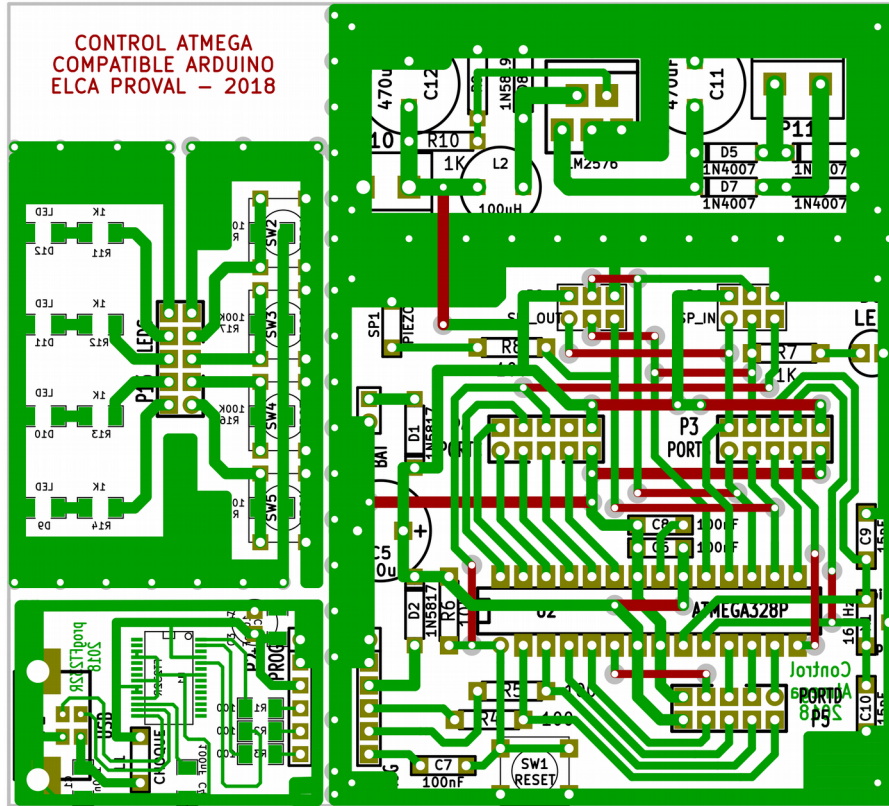


SISTEMA ARDUINO PROVAL: ESQUEMA ADAPTADOR USB

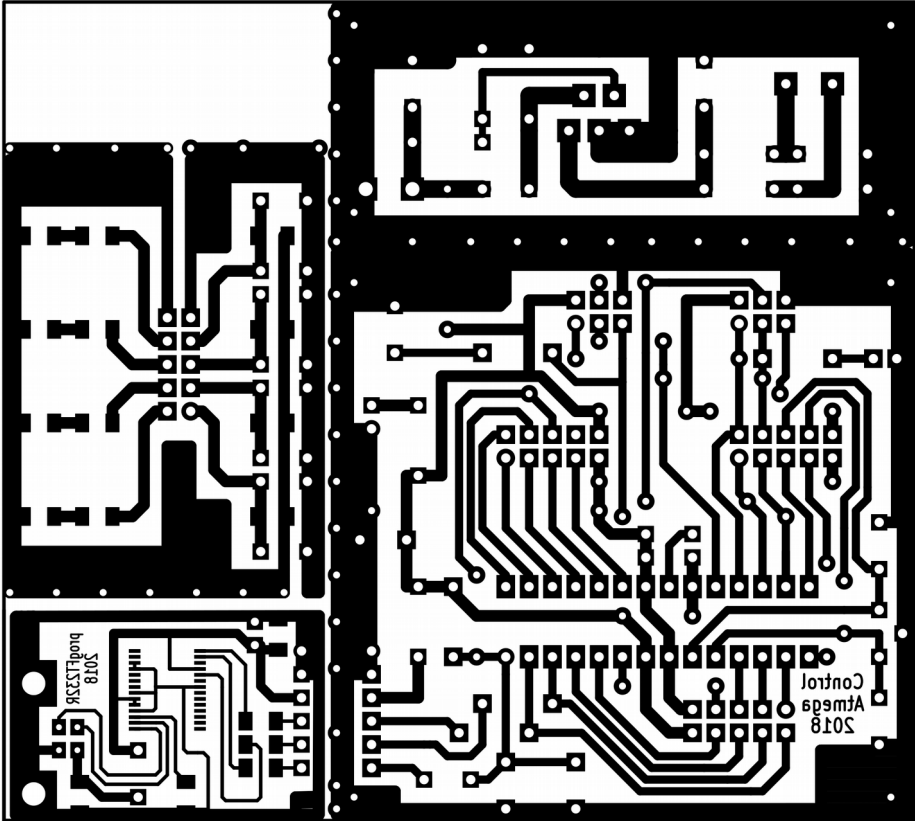
- Baseado no circuíto integrado FT232RL (FTDI). No mercado desde 2005, supón un estándar de facto en conversores serie-usb.
- Drivers de uso libre para todos os sistemas operativos.
- Encapsulado SSOP-28. Isto supón un problema para facer unha placa propia, pero tamén é unha boa práctica de soldadura SMD (non vale queimalo, ten que funcionar!!)
- Alternativas: do mesmo fabricante o FT231X, máis simple e barato.
- Circuitos doutros fabricantes: CP2102 (Silabs), PL2303 (Prolific), CH340 (WCH).
- Se non queremos ou non podemos facer soldadura smd hai moitos módulos comerciais que usan estes circuitos e son moi económicos.



SISTEMA ARDUINO PROVAL: PISTAS E COMPONENTES

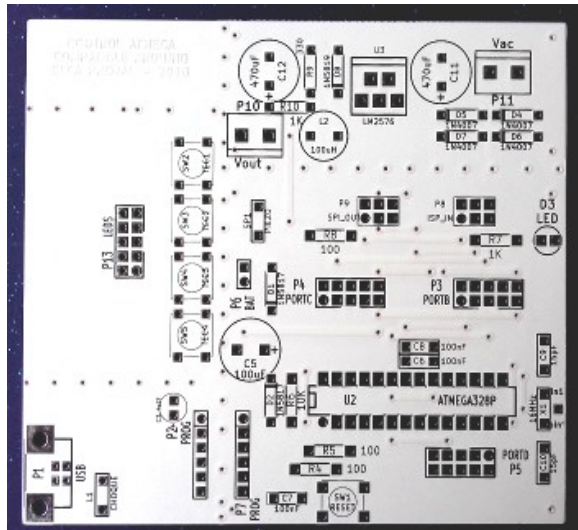


SISTEMA ARDUINO PROVAL: TRAZADO DE PISTAS

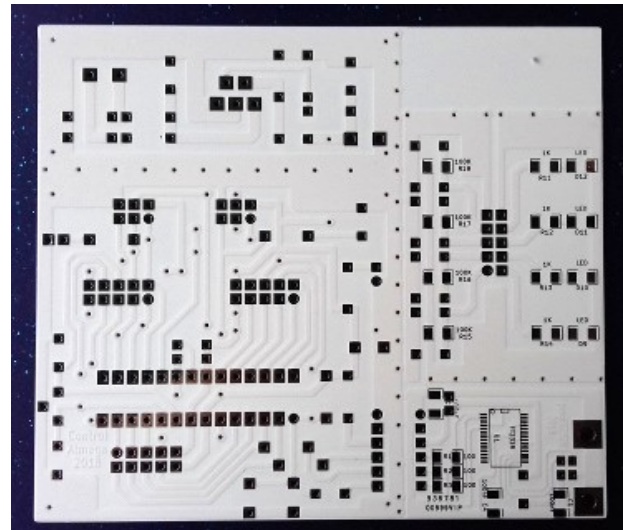


SISTEMA ARDUINO PROVAL: CONXUNTO DE CIRCUITOS IMPRESOS

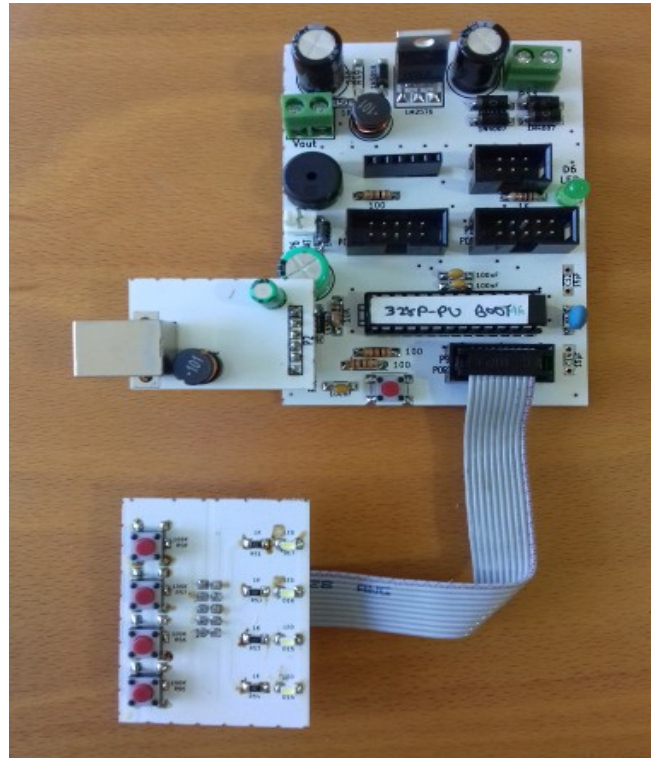
LADO COMPONENTES



LADO PISTAS

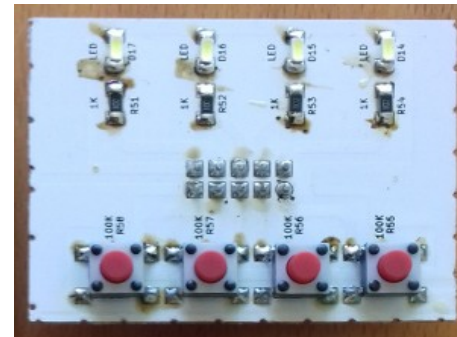
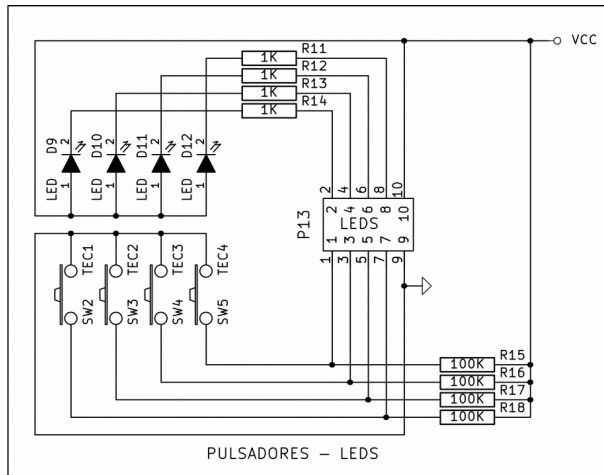


PLACA ARDUINO PROVAL: MÓDULOS FINALIZADOS



SISTEMA ARDUINO PROVAL: PLACA DE INTERFAZ CON DÍODOS LED E PULSADORES

- Esta placa é un periférico moi sinxelo e polivalente.
- Incorpora 4 díodos led smd (OSRAM DURIS E3 PLCC2) con resistencia serie de 1K tamén smd.
- Tamén leva 4 pulsadores miniatura de inserción con resistencia de 100K.
- As placas Arduino só levan un led “de serie” e ningún pulsador.
- A montaxe desta placa serve como práctica de soldadura smd (encapsulados SMD3216 = 3,2x1,6mm, PLCC2 = 3,0x1,4mm). NOTA: primeira resistencias, logo leds!!
- Este módulo serve como botoneira e visualizador para automatismos simples.
- Tamén serve para facer moitas prácticas de electrónica dixital básica (portas lóxicas, biestables, monoestables, decodificadores, etc).
- Isto permite substituír as montaxes con circuitos integrados LSI (debate: prácticas anticuadas?) e introducir a programación en Arduino mediante exemplos simples.



APLICACIONES: CONTROL DE ENTRADAS/SAIDAS DIXITAIS

```

//////////////////////////////////////
// Blink-4bits
// - programa que activa o led D13 e os bits D0-3 do PORTD
//
// modified 8 May 2014 by Scott Fitzgerald
// modificado out 2018 IES Proval
//////////////////////////////////////

// función de inicialización
void setup()
{
  // inicializa saídas dixitais
  pinMode(13, OUTPUT);
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
}

// función de repetición
void loop()
{
  digitalWrite(13, HIGH);
  digitalWrite(0, HIGH);
  digitalWrite(1, LOW);
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  delay(1000); //espera 1 segundo
  digitalWrite(13, LOW);
  digitalWrite(0, LOW);
  digitalWrite(1, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  delay(1000); //espera 1 segundo
}
//////////////////////////////////////

//////////////////////////////////////
// BLINK 4BITS PORTOS
// - Programa de manexo de bits dos portos en bloque para probar
// códigos decimal, binario, hexadecimal
// - instrucións DDRx (dirección bits do porto x, 1=saida)
// - PORTx (escritura 8 bits no porto x)
// - PINx (lectura 8 bits do porto x) */
//////////////////////////////////////

// función de inicialización
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);

  //establece pins 0-3 do porto D como saídas (0=entrada, 1=saida)
  DDRD = 0b00001111;

  // PORTD = 0B00001010; //binario
  // PORTD = 10; //decimal
  PORTD = 0X0A; //hexadecimal
}

// función de repetición
void loop()
{
  //PORTD = 0X0A; //hexadecimal
  //PORTD = 10; //decimal
  PORTD = 0B00001010; //binario
  delay(1000);
  //PORTD = 0X05; //hexadecimal
  //PORTD = 5; //decimal
  PORTD = 0B00000101; //binario
  delay(1000);
}
//////////////////////////////////////

```

APLICACIONES: CONTROL DE ENTRADAS/SAIDAS DIXITAIS

```

////////////////////////////////////// // inicializa pins do porto virtual i2c
// BLINK4BITS REMOTO A TRAVES DE I2C mcp.pinMode(4, OUTPUT);
// - programa que activa o led D13 e os bits D0-3 dun porto mcp.pinMode(5, OUTPUT);
// mcp.pinMode(6, OUTPUT);
// modified 8 May 2014 by Scott Fitzgerald mcp.pinMode(7, OUTPUT);
// modificado out 2018 IES Proval */ }
//
////////////////////////////////////// // the loop function runs over and over again forever
#include <Wire.h> void loop()
#include "Adafruit_MCP23008.h" {
// digitalWrite(13, mcp.digitalRead(0));

// Basic pin reading and pullup test for the MCP23008 I/O expander
// public domain! digitalWrite(13, LOW); // turn the LED on (HIGH is the voltage level)
// Connect pin #1 of the expander to Analog 5 (i2c clock) //alumea os leds do porto D
// Connect pin #2 of the expander to Analog 4 (i2c data) digitalWrite(0, HIGH);
// Connect pins #3, 4 and 5 of the expander to ground digitalWrite(1, LOW );
(address selection) digitalWrite(2, HIGH);
// Connect pin #6 and 18 of the expander to 5V (power and digitalWrite(3, LOW );
reset disable) //alumea os leds do porto virtual i2c
// Connect pin #9 of the expander to ground (common ground) mcp.digitalWrite(4, HIGH);
// Input #0 is on pin 10 so connect a button or switch from mcp.digitalWrite(5, LOW );
there to ground mcp.digitalWrite(6, HIGH);
mcp.digitalWrite(7, LOW );

Adafruit_MCP23008 mcp; delay(1000); // wait for a second
digitalWrite(13, HIGH ); // turn the LED off by making the
// the setup function runs once when you press reset or power voltage LOW
the board //alumea os leds do porto D
void setup() digitalWrite(0, LOW );
{ digitalWrite(1, HIGH);
mcp.begin(); // use default address 0 digitalWrite(2, LOW );
digitalWrite(3, HIGH);
// mcp.pinMode(0, INPUT); //alumea os leds do porto virtual i2c
// mcp.pullUp(0, HIGH); // turn on a 100K pullup internally mcp.digitalWrite(4, LOW );
mcp.digitalWrite(5, HIGH);
mcp.digitalWrite(6, LOW );
mcp.digitalWrite(7, HIGH);
delay(1000); // wait for a second
}
// initialize digital pin 13 as an output.
pinMode(13, OUTPUT);
// inicializa pins do porto D
pinMode(0, OUTPUT);
pinMode(1, OUTPUT);
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
//////////////////////////////////////

```

APLICACIÓNS: SIMULACIÓN DE PORTAS LÓXICAS

```

//////////////////////////////////// void setup()
//
// 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 4
#define PIN_B 5
#define PIN_C 6
#define PIN_D 7
#define PIN_S1 0
#define PIN_S2 1
#define PIN_S3 2
#define PIN_S4 3
// definicións de niveis lóxicos para pulsadores e leds
// LOW=activo, HIGH=inactivo
#define NIVEL_OFF HIGH
#define NIVEL_ON  LOW
#define S1_FILA0 NIVEL_OFF //modificar para cambiar a táboa!!
#define S1_FILA1 NIVEL_OFF
#define S1_FILA2 NIVEL_OFF
#define S1_FILA3 NIVEL_ON
////////////////////////////////////

void setup()
{
  // definición de entradas e saídas
  pinMode(PIN_A, INPUT);
  pinMode(PIN_B, INPUT);
  pinMode(PIN_C, INPUT);
  pinMode(PIN_D, INPUT);
  pinMode(PIN_S1, OUTPUT);
  pinMode(PIN_S2, OUTPUT);
  pinMode(PIN_S3, OUTPUT);
  pinMode(PIN_S4, OUTPUT);
  digitalWrite(PIN_S1, NIVEL_OFF);
  digitalWrite(PIN_S2, NIVEL_OFF);
  digitalWrite(PIN_S3, NIVEL_OFF);
  digitalWrite(PIN_S4, NIVEL_OFF);
}

void loop()
{
  boolean entrada_a, entrada_b, entrada_c, entrada_d, saida_s1,
saida_s2;

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

  // táboa de verdade
  // fila 0
  if( (entrada_d==NIVEL_OFF) && (entrada_c==NIVEL_OFF) )
saida_s1=S1_FILA0;
  // fila 1
  if( (entrada_d==NIVEL_OFF) && (entrada_c==NIVEL_ON) )
saida_s1=S1_FILA1;
  // fila 2
  if( (entrada_d==NIVEL_ON) && (entrada_c==NIVEL_OFF) )
saida_s1=S1_FILA2;
  // fila 3
  if( (entrada_d==NIVEL_ON) && (entrada_c==NIVEL_ON) )
saida_s1=S1_FILA3;

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

```

APLICACIONES: 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: DIAPASÓN 4 NOTAS PARA VIOLÍN

```

////////////////////////////////////// // bucle do programa principal
// DIAPASÓN 4 NOTAS PARA VIOLÍN // variables globais
// // int tecla, tecla1, tecla2, tecla3, tecla4;
// - frecuencias violín: sol3=196.0Hz, re4=293.7Hz, la4=440Hz,
mi5=659.3Hz
//////////////////////////////////////

// definicións de constantes do programa
#define TECLA1 4 //pin da tecla 1 (esquerda)
#define TECLA2 5 //pin da tecla 2
#define TECLA3 6 //pin da tecla 3
#define TECLA4 7 //pin da tecla 4 (dereita)
#define LED1 3 //pin do led 1 (esquerda)
#define LED2 2 //pin do led 2
#define LED3 1 //pin do led 3
#define LED4 0 //pin do led 4 (dereita)
#define FRECUENCIA1 196 //frecuencia SOL3 196.0Hz
#define FRECUENCIA2 294 //frecuencia RE4 293.7Hz
#define FRECUENCIA3 440 //frecuencia LA4 440.0Hz
#define FRECUENCIA4 659 //frecuencia MI5 659.3Hz
#define SPEAKER 10 //pin do altavoz
#define LED_PLACA 13 //pin do led integrado na placa

// código de inicialización de parámetros
void setup()
{
  //define pins de teclas como entradas
  pinMode(TECLA1, INPUT);
  pinMode(TECLA2, INPUT);
  pinMode(TECLA3, INPUT);
  pinMode(TECLA4, INPUT);
  //define pins de leds como saídas
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  //desactiva leds
  digitalWrite(LED1,HIGH);
  digitalWrite(LED2,HIGH);
  digitalWrite(LED3,HIGH);
  digitalWrite(LED4,HIGH);
}

}

void loop()
{
  delay(200); //retardo entre lecturas
  tecla1=digitalRead(TECLA1);
  tecla2=digitalRead(TECLA2);
  tecla3=digitalRead(TECLA3);
  tecla4=digitalRead(TECLA4);

  // tecla1 pulsada: activa frecuencia 1
  tecla=0; //valor de tecla pulsada (0=non, 1-2-3-4 número de
  tecla)
  if(tecla1==LOW) tecla=1;
  if(tecla2==LOW) tecla=2;
  if(tecla3==LOW) tecla=3;
  if(tecla4==LOW) tecla=4;

  switch(tecla)
  {
    case 0: noTone(SPEAKER); //desactiva tono
            digitalWrite(LED1,HIGH); //desactiva leds
            digitalWrite(LED2,HIGH);
            digitalWrite(LED3,HIGH);
            digitalWrite(LED4,HIGH);
            break;
    case 1: digitalWrite(LED1,LOW); //activa led
            tone(SPEAKER,FRECUENCIA1); //activa tono
            break;
    case 2: digitalWrite(LED2,LOW); //activa led
            tone(SPEAKER,FRECUENCIA2); //activa tono
            break;
    case 3: digitalWrite(LED3,LOW); //activa led
            tone(SPEAKER,FRECUENCIA3); //activa tono
            break;
    case 4: digitalWrite(LED4,LOW); //activa led
            tone(SPEAKER,FRECUENCIA4); //activa tono
            break;
  } // fin do switch
} //fin do programa

```

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 ); //saida desactivada
  digitalWrite(PIN_QN,saida_qn); //saida 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; }

  // transfere 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 dixito (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          Al A2 LT BI RBI A3 A0 GND

*/
////////////////////////////////////
// define variables globais de entradas e saídas
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; }

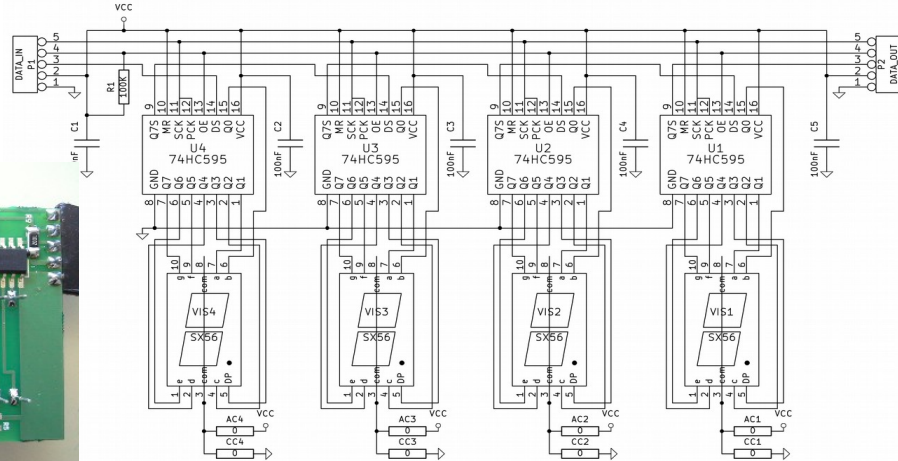
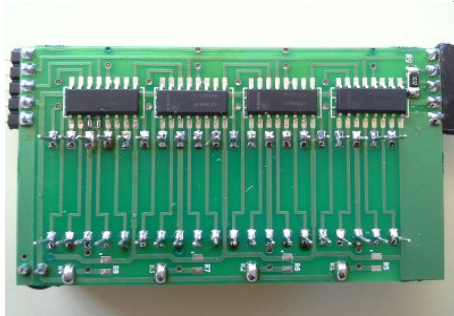
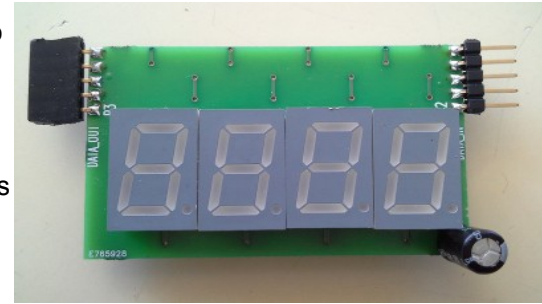
  // transfere os valores das variables ós pins de saída
  saida_q0=bitRead(contador,0);
  saida_q1=bitRead(contador,1);
  saida_q2=bitRead(contador,2);
  saida_q3=bitRead(contador,3);
  digitalWrite(PIN_Q0,saida_q0);
  digitalWrite(PIN_Q1,saida_q1);
  digitalWrite(PIN_Q2,saida_q2);
  digitalWrite(PIN_Q3,saida_q3);

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

```

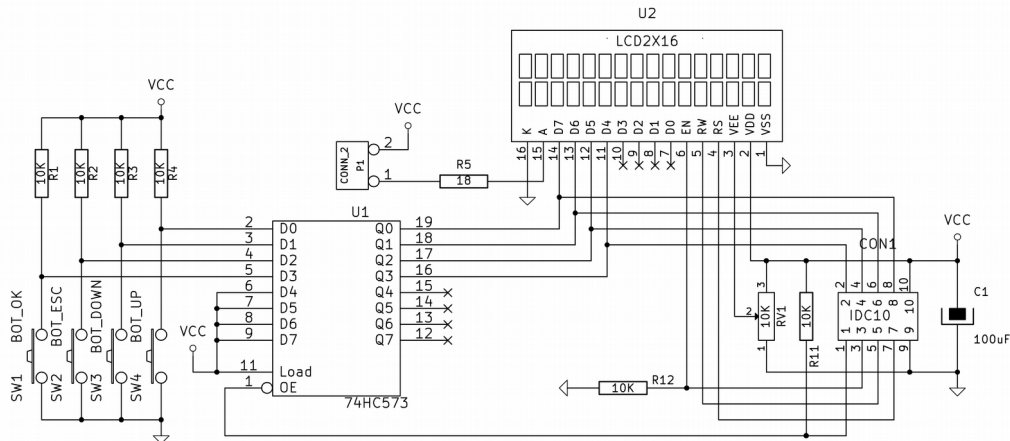
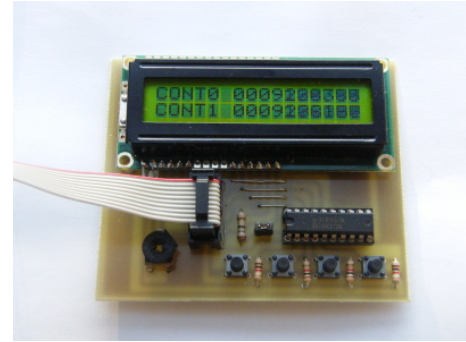
SISTEMA ARDUINO PROVAL: VISUALIZADOR 7 SEGMENTOS CON REXISTROS 595

- Este visualizador está formado por dígitos led de 7 segmentos controlados por rexistros de desprazamento serie-paralelo 74595. Cada rexistro controla un único dígito de forma estática.
- Os datos procedentes do microcontrolador chegan en formato serie mediante dúas liñas (clock/data) e recorren todos os rexistros. É posible interconectar máis de un módulo para facer visualizadores ampliados.
- Pode controlarse a corrente de saída mediante unha liña OE que se aplica a tódolos rexistros.
- O consumo pode ser elevado, hai que prestar atención ó cableado e alimentación.



SISTEMA ARDUINO PROVAL: VISUALIZADOR LCD 2X16 CON PULSADORES

- Esta placa ten un visualizador lcd de 2 liñas e 16 caracteres con control HD44780 e 4 pulsadores conectados ás liñas de datos mediante un rexistro 74573.
- Está pensada como interfaz de usuario para visualización de datos e control de menús ou parámetros en sistemas máis complexos.
- O control da placa só precisa 8 liñas de datos=un único porto.
- Compatible coa biblioteca de Arduino "*LiquidCrystal*". Tamén pode usarse un adaptador i2c para conexión serie.



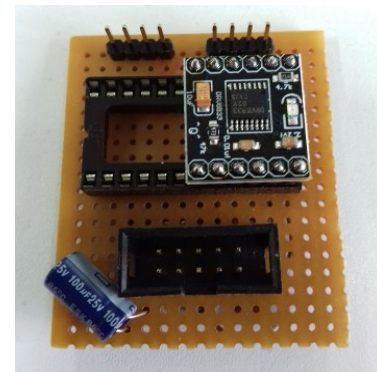
APLICACIONES: VISUALIZADOR LCD 2x16

- Este exemplo emprega a biblioteca "LiquidCrystal" para controlar un visualizador lcd de 2 filas x 16 caracteres con controlador compatible HD44780

```
////////////////////////////////////  
// LiquidCrystal Library - Hello World  
//  
// Demonstrates the use a 16x2 LCD display. The LiquidCrystal  
// library works with all LCD displays that are compatible with the  
// Hitachi HD44780 driver. There are many of them out there, and you  
// can usually tell them by the 16-pin interface.  
//  
// This sketch prints "Hello World!" to the LCD  
// and shows the time.  
//  
// Library originally added 18 Apr 2008 by David A. Mellis  
// library modified 5 Jul 2009 by Limor Fried (http://www.ladyada.net)  
// example added 9 Jul 2009 by Tom Igoe  
// modified 22 Nov 2010 by Tom Igoe  
  
// This example code is in the public domain.  
  
// http://www.arduino.cc/en/Tutorial/LiquidCrystal  
  
// include the library code:  
#include <LiquidCrystal.h>  
  
// initialize the library with the numbers of the interface pins  
// LiquidCrystal(rs, rw, en, d4, d5, d6, d7)  
LiquidCrystal lcd(4, 5, 6, 0, 1, 2, 3);  
  
void setup() {  
  {  
    pinMode(7,OUTPUT);  
    digitalWrite(7, HIGH);  
    // set up the LCD's number of columns and rows:  
    lcd.begin(16, 2);  
    // Print a message to the LCD.  
    lcd.print("hello, world!");  
  }  
  
  void loop() {  
    // set the cursor to column 0, line 1  
    // (note: line 1 is the second row, since counting begins with 0):  
    lcd.setCursor(0, 1);  
    // print the number of seconds since reset:  
    lcd.print(millis() / 1000);  
  }  
}////////////////////////////////////
```

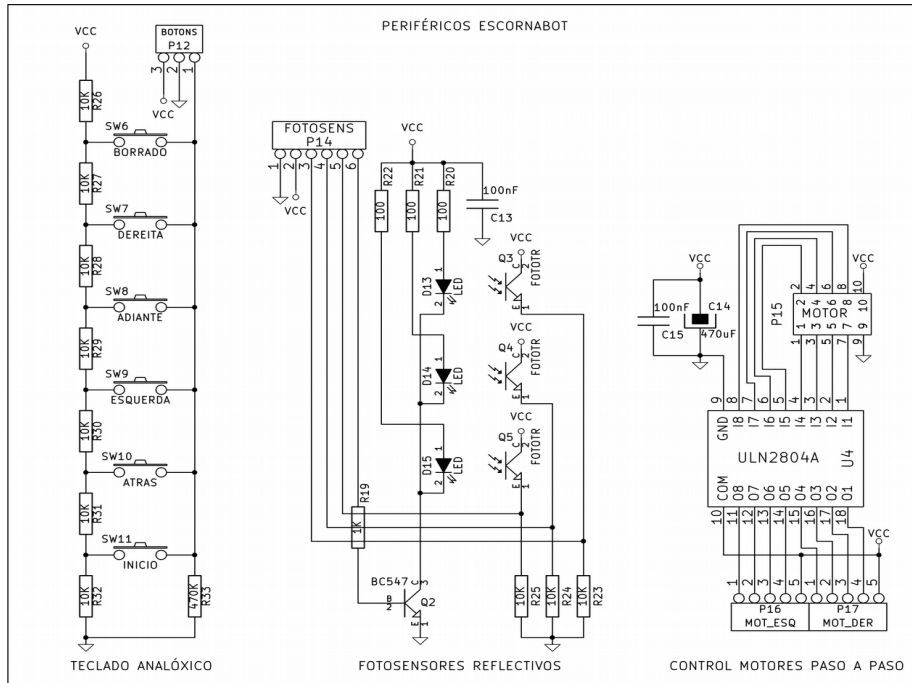
SISTEMA ARDUINO PROVAL: MÓDULO CONTROL DE MOTORES CON DRV8833

- O circuito DRV8833 é unha dobre ponte H con transistores mosfet con capacidade para controlar un motor paso a paso bipolar ou dous motores de corrente contínua.
- A tensión de alimentación dos motores pode variar entre 2,7 e 10,8V polo que pode funcionar con diferentes tipos de fontes e baterías.
- Un único módulo conectado a un microcontrolador abonda para facer un pequeno robot con capacidade de desprazamento bidireccional e xiro.
- O control pode facerse mediante as funcións PWM de Arduino (*analogWrite*) ou as bibliotecas de motores paso a paso.



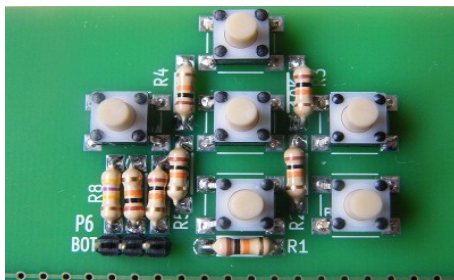
SISTEMA ARDUINO PROVAL: PERIFÉRICOS ROBÓTICA

- 1) Teclado analóxico con 6 pulsadores
- 2) Sensor reflectivo triple con led visibles/infravermellos e fototransistores
- 3) Control doble motor paso a paso unipolar con ULN2804



SISTEMA ARDUINO PROVAL: PERIFÉRICOS ROBÓTICA

- O módulo de 3 fotosensores permite facer programas de seguemento de liñas ou detección de obstáculos.
- Hai moitos sensores que se poden usar, como o TCRT5000 (infravermello), CNY70 ou QRD1114. Moitos destes sensores teñen saída dixital polo que só poden diferenciar se a luz detectada é superior ou inferior a un valor prefixado (umbral).
- Optamos por usar un diodo led e fototransistor separados, 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). A lectura é analóxica para facer poder facer algoritmos máis precisos e calibración de valores de referencia.
- Os leds e fototransistores poden ser visibles (SFH-309-5/6 + Vishay TLHA44R1S2) ou infravermellos (SFH310-FA-2/3 + Kingbright L-34F3BT).



- O módulo de teclado analóxico está formado por un divisor de tensión con pulsadores conectados a unha saída común.
- A saída leva unha resistencia a masa para definir o nivel cando non hai ningún pulsador activado.

APLICACIÓNS: CONTROL DE MOTOR PASO A PASO UNIPOLAR

```

////////////////////////////////////
//
//  PROGRAMA CONTROL DE MOTOR PASO A PASO UNIPOLAR CON ARDUINO
//
////////////////////////////////////
// Este programa emprega o microcontrolador ATMEGA328P-FU (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 saídas
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 saídas
  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 saída 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 ciclicamente o contador
    if(entrada_avance==HIGH)
    { contador++;
      if(contador>MAX_CONTADOR) contador=0;
      delay(RETARDO_MS); }

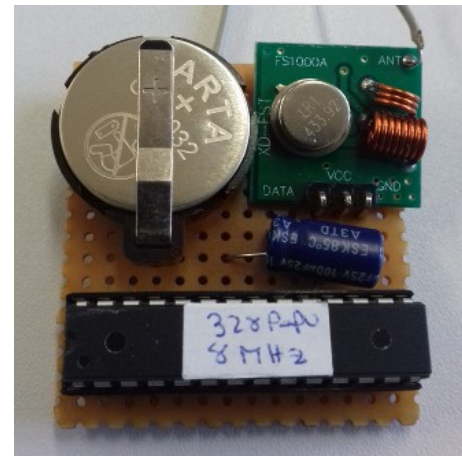
    // se a entrada retroceso está activa decrementa ciclicamente o contador
    if(entrada_retroceso==HIGH)
    { contador--;
      if(contador<0) contador=MAX_CONTADOR;
      delay(RETARDO_MS); }

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

```

EXEMPLO DE APLICACIÓN: MANDO A DISTANCIA RF 433MHZ

- Este exemplo consiste na montaxe dun transmisor de radiofrecuencia compatible con receptores comerciais de código fixo.
- A placa é totalmente minimalista, só leva o microcontrolador, pila, módulo de radiofrecuencia e un pulsador.
- O microcontrolador está programado usando o oscilador interno de 8MHz, polo que non é preciso engadir cristal nin resonador.
- Usa a biblioteca de Arduino "RCSwitch". Esta biblioteca inclúe os métodos de transmisións máis empregados en telexmandos de código fixo, só hai que escoller o dato que se quere transmitir e envíalo.
- Pode facerse tamén a placa receptora completa ou conectar o circuito receptor a unha placa arduino normal.



EXEMPLO DE APLICACIÓN: MANDO A DISTANCIA RF 433MHZ

```
////////////////////////////////////  
// TRANSMISOR RADIOFRECUENCIA CON ARDUINO  
//  
// https://github.com/sui77/rc-switch/  
//  
////////////////////////////////////  
  
#include <RCSwitch.h>  
  
RCSwitch mySwitch = RCSwitch();  
  
void setup() {  
  
    Serial.begin(9600);  
  
    // Definición do transmisor: pin 12, protocolo 1, ancho pulso 188us  
    mySwitch.enableTransmit(12);  
    mySwitch.setProtocol(1);  
    mySwitch.setPulseLength(188);  
}  
  
void loop() {  
  
    // Envía o código 1.000.000 (pode cambiarse)  
    mySwitch.send(1000000, 24);  
    delay(1000); //espera un segundo entre transmisións  
  
}
```

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.

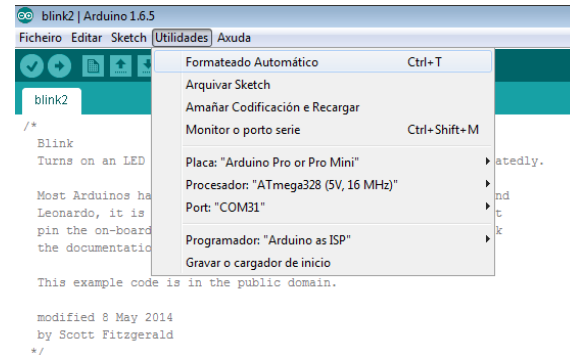
Se non dispoñemos dun gravador específico como o AVRISP pódese usar outra placa Arduino (Nano, Uno, Duemila,...) como programador ISP.

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).

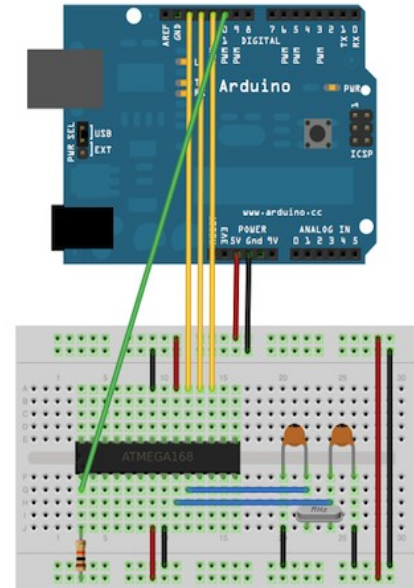
GRAVACIÓN DO BOOTLOADER NUN MICRO NOVO USANDO ARDUINO COMO PROGRAMADOR

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

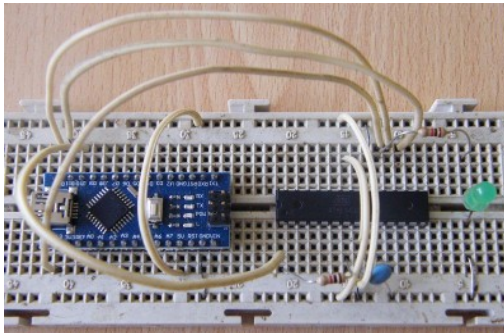
4) Unha vez conectado todo, no menú "Utilidades" seleccionar "Gravar cargador de inicio" e esperar a que remate (son uns poucos segundos).

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).

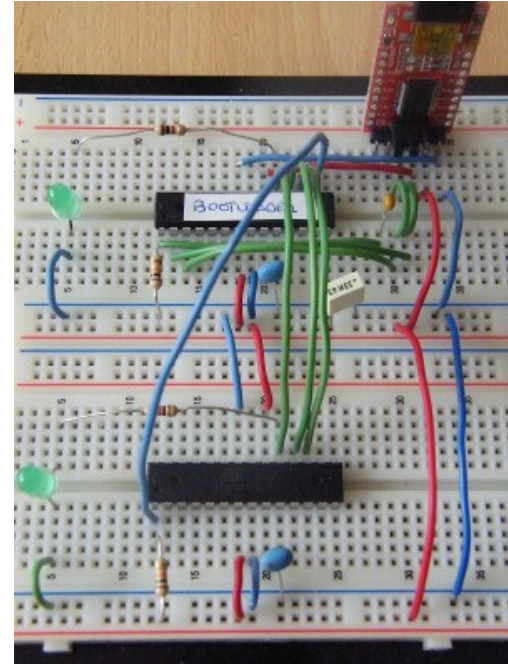


GRAVACIÓN DO BOOTLOADER NUN MICRO NOVO USANDO ARDUINO COMO PROGRAMADOR

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).

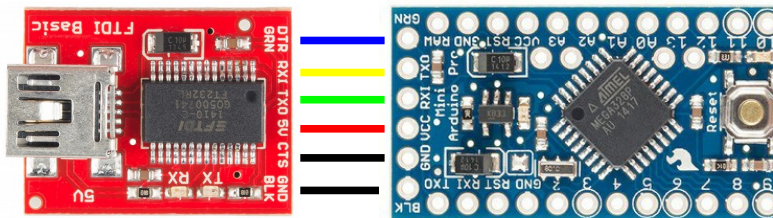
Seleccionar o tipo de micro (Atmega328P, 5V, 16MHz)

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)

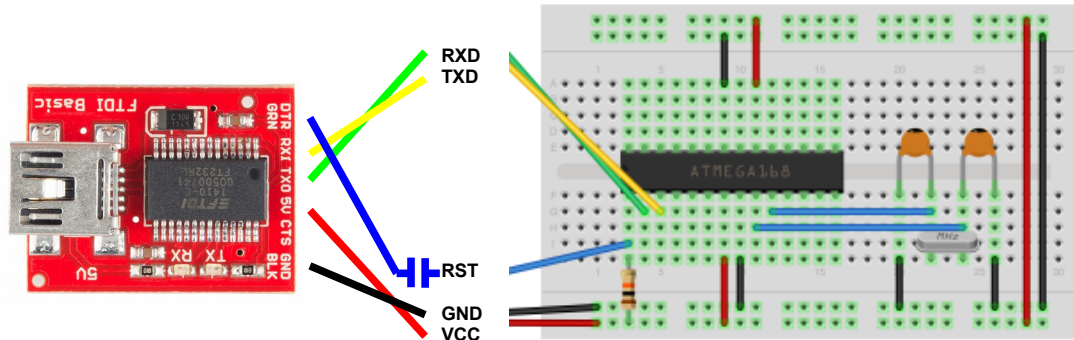
- Se queremos gravar unha placa Arduino Pro-Mini (ou compatible) a forma de conexión sería a seguinte. Son 5 liñas: GND, VCC, RXD, TXD, RST. Nos conectores hai 6 cables porque a masa está duplicada:



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.



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/opensourcous/wiki/KiCADTutorialCreatingGerberFiles>

NOTA: KICAD 2013 crea correctamente os arquivos gerber para os provedores, 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

ademáis hai que crear o arquivo de taladrado .DRL (e logo cambiarlle o nome a .TXT para moitos provedores). Hai que seleccionar as opcións unidades: pulgadas, formato: suppress leading zeros, precisión: 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).

SEEDSTUDIO <http://www.seedstudio.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 despois de facer o pedido.

APLICACIONES: MONTAXES EN ADESTRADOR DIXITAL

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

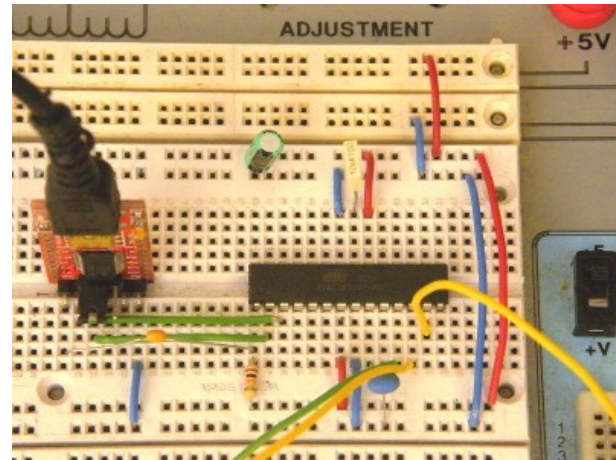
- Poden aproveitarse os pulsadores existentes como entradas e os indicadores led como saídas.

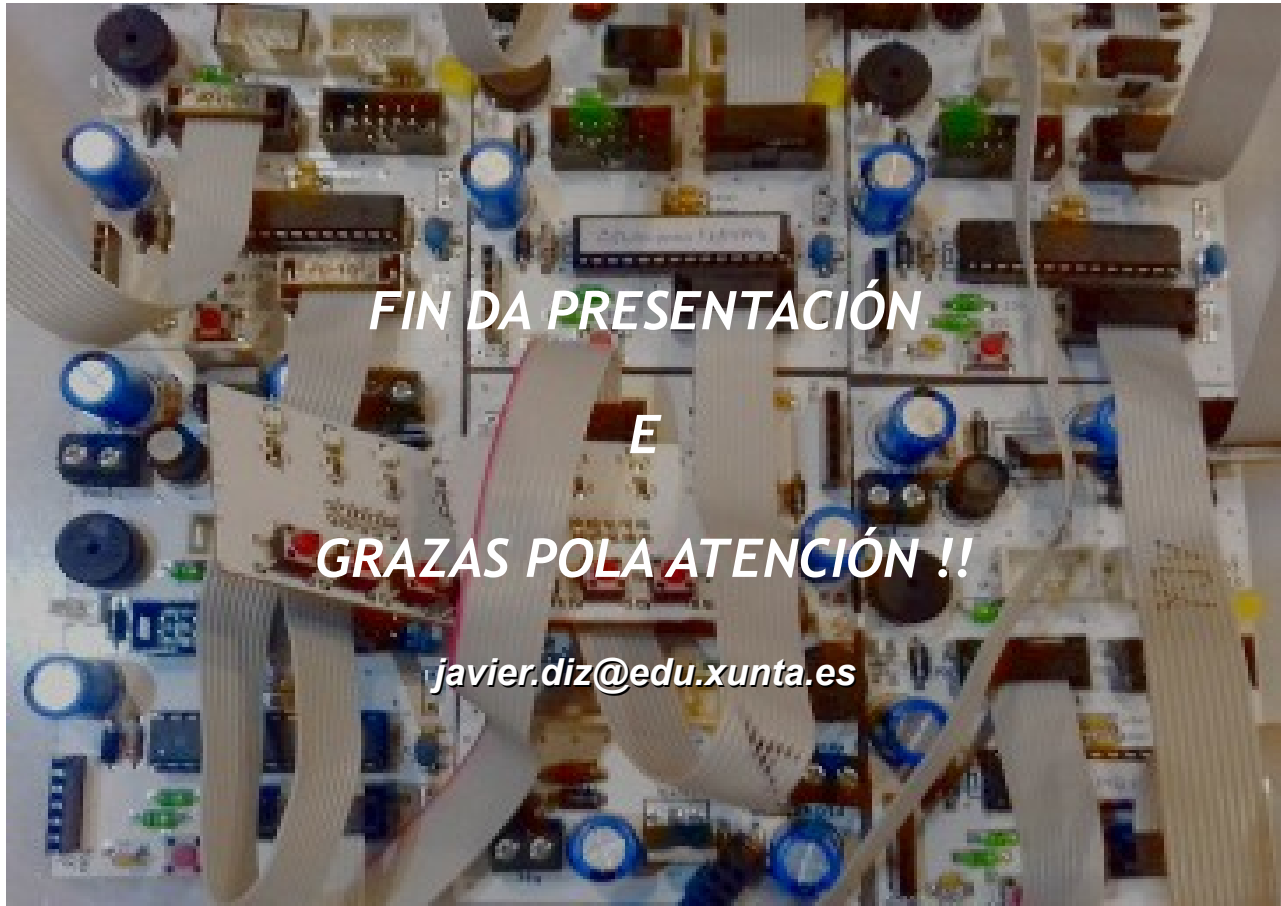
- A alimentación do microcontrolador pode tomarse do adestrador, nese caso a placa usb pode desconectarse despóis de programar (incluso “en quente”) e o circuíto queda independente como en calquera outra práctica.

- Hai que ter coidado de non interconectar as alimentaciónns 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 circuitos como portas lóxicas, monoestables ou biestables.





FIN DA PRESENTACIÓN

E

GRAZAS POLA ATENCIÓN !!

javier.diz@edu.xunta.es