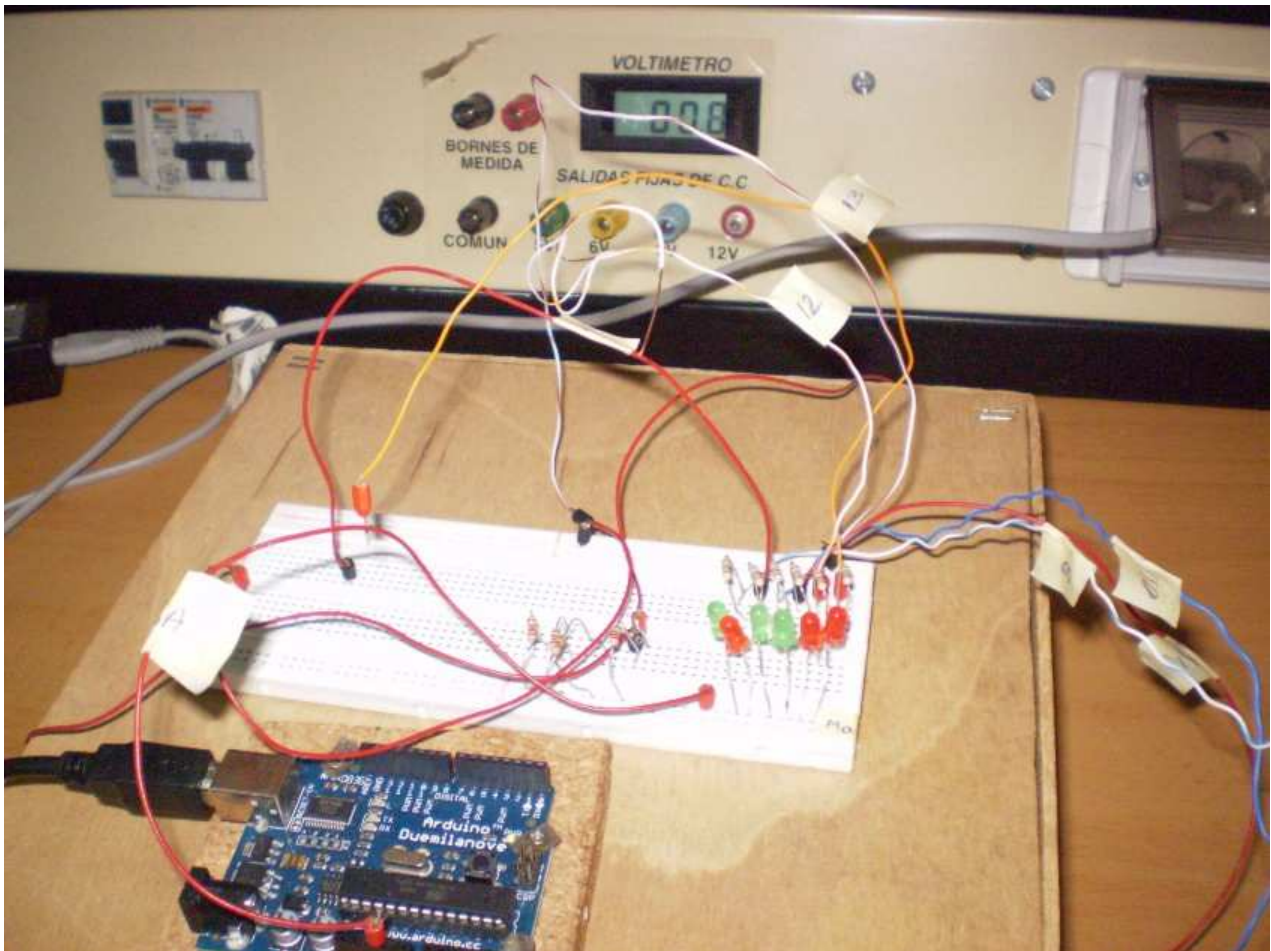


PRÁCTICAS CON ARDUINO

1ª práctica: Preparar e etiquetar o circuío de proba de Arduino.

Usamos como circuío de proba de Arduino unha placa protoboard na cal colocamos 6 Led coas súas correspondentes resistencias en serie (para evitar que estes se queimen).

Etiquetamos os cables de conexión para cada un dos Led que irán conectados a unha saída concreta do Arduino, ademais de indicar os puntos de conexión da tensión de 5 V e da masa. (Ver foto).



Usaremos como saídas dixitais os pines 8, 9, 10, 11, 12 e 13 de Arduino.

2ª práctica: Probar o 1º programa. Parpadeo dun Led.

Utilizamos o programa de exemplo *Blink_Led*, o cargamos no Arduino e comprobamos o seu correcto funcionamento.

O único problema foi que a conexión co Arduino non funcionou ata que lle indicamos que para o noso ordenador o porto de comunicacións usado polo USB era o COM 4. Unha vez solucionado, a conexión e carga do programa foi correcta.

O programa que cargamos en Arduino foi:

```
int ledPin = 13;           // LED connected to digital pin 13
void setup()               // run once, when the sketch starts
{
```

```

    pinMode(ledPin, OUTPUT);    // sets the digital pin as output
}
void loop()                    // run over and over again
{
    digitalWrite(ledPin, HIGH); // sets the LED on
    delay(1000);                // waits for a second
    digitalWrite(ledPin, LOW);  // sets the LED off
    delay(500);                 // waits for a second
}

```

A continuación vamos a realizar varias modificacións no programa, de xeito que cambiaremos a saída que active o Arduino, os tempos de acendido e apagado e tamén o número de Led's que acendamos á vez:

```

int ledPin1 = 10;              // LED connected to digital pin 13
int ledPin2 = 8;
void setup()                   // run once, when the sketch starts
{
    pinMode(ledPin1, OUTPUT);  // sets the digital pin as output
    pinMode(ledPin2, OUTPUT);
}
void loop()                    // run over and over again
{
    digitalWrite(ledPin1, HIGH); // sets the LED on
    digitalWrite(ledPin2, HIGH);
    delay(2000); // waits for a second
    digitalWrite(ledPin1, LOW);  // sets the LED off
    digitalWrite(ledPin2, LOW);
    delay(2000);                // waits for a second
}

```

Engadimos un LED o circuío que se corresponde co programa anterior. Fomos cambiando as saídas e o tempo de funcionamento para comprobar se todo funcionaba axeitadamente.

Engadimos outro LED (3 Led). Estivemos probando a cambiarlle os tempos de funcionamento e en tódalas veces que o fixemos respondeu ben.

3ª práctica: Conexión de saídas dixitais: O bruador (buzzer).

Imos continuar engadíndolle ó circuío un buzzer.

O buzzer, bruador ou bucina ten 2 cables de alimentación (o de cor negra correspóndese coa masa, e o de cor vermella e a alimentación positiva (5 voltios) que se une ao PIN que designemos como saída.

O programa que usamos no Arduino conmuta entre 0 e 1 cos distintos intervalos de tempo (períodos) que lle indicamos, dando lugar a que o bruador emita sons de distintas frecuencias:

```

int speakerOut = 7;
void setup() {
    pinMode(speakerOut, OUTPUT);
}

```

```

}
void loop() {
  for (int i=0;i<100; i++){
    digitalWrite(speakerOut, HIGH);
    delayMicroseconds(2272);
    digitalWrite(speakerOut, LOW);
    delayMicroseconds(2272);
  }
}

```

A práctica da o resultado desexado emitindo o brugador un tono.

Realizamos unha modificación no programa de tal xeito que vaia variando o ton emitido, simulando unha sirena ó cambiar os tempos de retardo no programa.

O ton emitido polo brugador segue a fórmula:

$$F \text{ (frecuencia)} = 1 / T \text{ (período)}.$$

Por exemplo:

$$F = 1 / T = 1 / 1000 \text{ (microseg.)} = 1000 \text{ Hz.}$$

O programa quedou así:

```

int speakerOut = 7;
int tonos[ ] = {4500, 4200, 4000, 3800, 3638, 3450, 3215, 3019, 2875, 2600, 2400, 2200, 2038, 1850, 1615, 1419, 1075, 556};
void setup() {
  pinMode(speakerOut, OUTPUT);
}
void loop() {
  for (int j=0;j<18; j++){
    for (int i=0;i<6; i++){
      digitalWrite(speakerOut, HIGH);
      delayMicroseconds(tonos[j]);
      digitalWrite(speakerOut, LOW);
      delayMicroseconds(tonos[j]);
    }
  }
}

```

4ª práctica: Conexión de entradas dixitais: O Pulsador.

Decidimos utilizar esta práctica para completar o que será un adestrador dixital con 6 entradas (os pulsadores) e 6 saídas (os Leds).

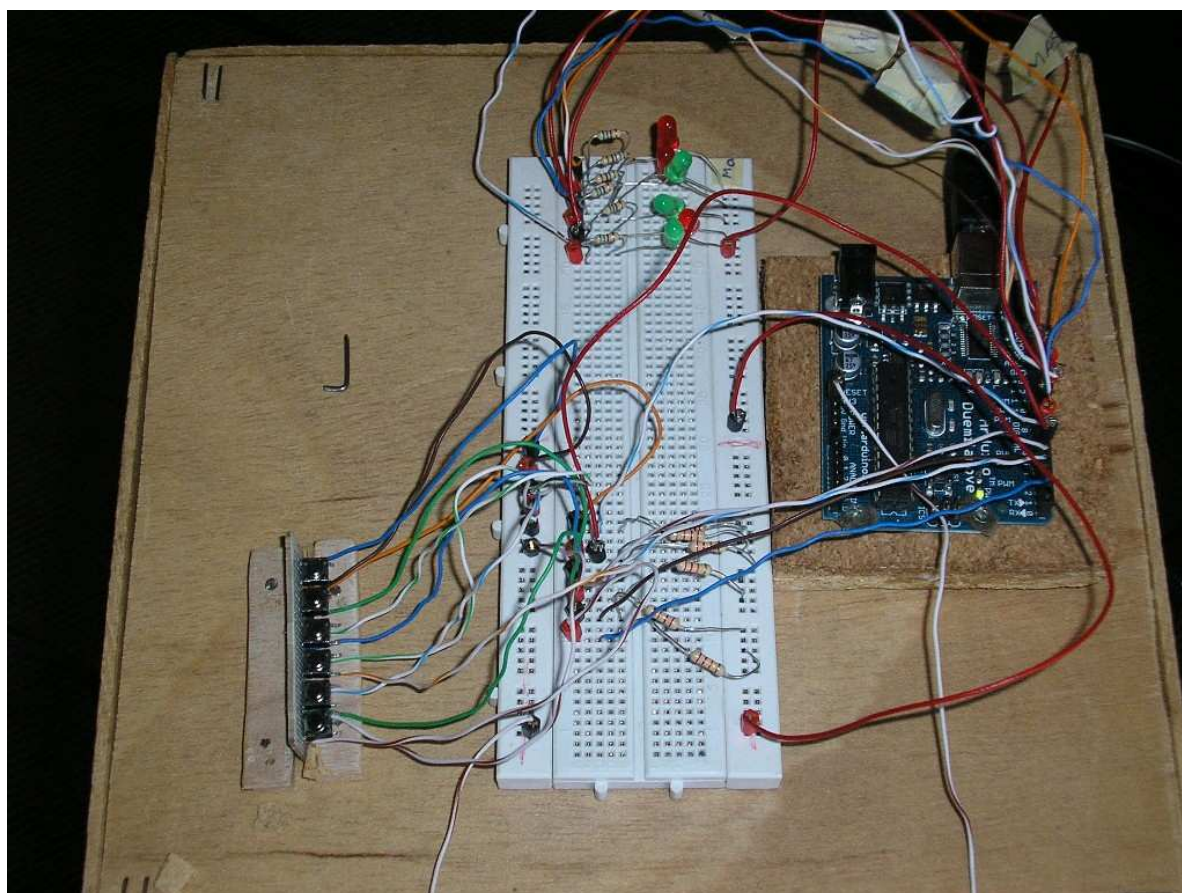
Nesta práctica vamos a utilizar pulsadores como entradas dixitais, utilizamos un anaco de placa impresa para soldar os pulsadores. Utilizamos 6 pulsadores con unha resistencia en serie para cada un, unha vez que os soldamos, fixémoslles sitio para que encaixasen e preparamos cables para conectalos á placa.

Nota importante: Cando estabamos a comprobar o funcionamento das saídas dixitais (Led's), comprobamos que as dúas masas de que dispón o Arduino non funcionan igual. Ó coller a saída da placa

entre as saídas dixitais e a súa masa obtemos a polaridade inversa á que se obtén se collemos a tensión de saída entre as saídas dixitais e a masa da saída de POWER. Hai que ter coidado con isto, para o adestrador dixital estamos a usar a masa das saídas dixitais.

O programa que fixemos para comprobar o adestrador dixital quedou así:

```
int contador = 0;
int pinesentrada[] = {2, 3, 4, 5, 6, 7};
int pinessalida[] = {8, 9, 10, 11, 12, 13};
void setup() {
  for (contador=0;contador<6;contador++) {
    pinMode(pinesentrada[contador], INPUT);
    pinMode(pinessalida[contador], OUTPUT);
  }
}
void loop() {
  for (contador=0;contador<6;contador++) {
    if (digitalRead(pinesentrada[contador]) == HIGH) {
      digitalWrite(pinessalida[contador], HIGH);
    } else {
      digitalWrite(pinessalida[contador], LOW);
    }
  }
}
```



O programa funciona ben, ó activar un pulsador acéndese o Led correspondente, pero hai un problema coa saída 10 (o Led está permanentemente acendido).

A saída 10 se corresponde co pulsador unido á entrada 5, ao comprobar o circuíto dos pulsadores vimos que unha soldadura facía cortocircuíto entre a entrada e a saída do pulsador, retirando o exceso de estaño solucionamos o problema, funcionando correctamente o adestrador dixital.

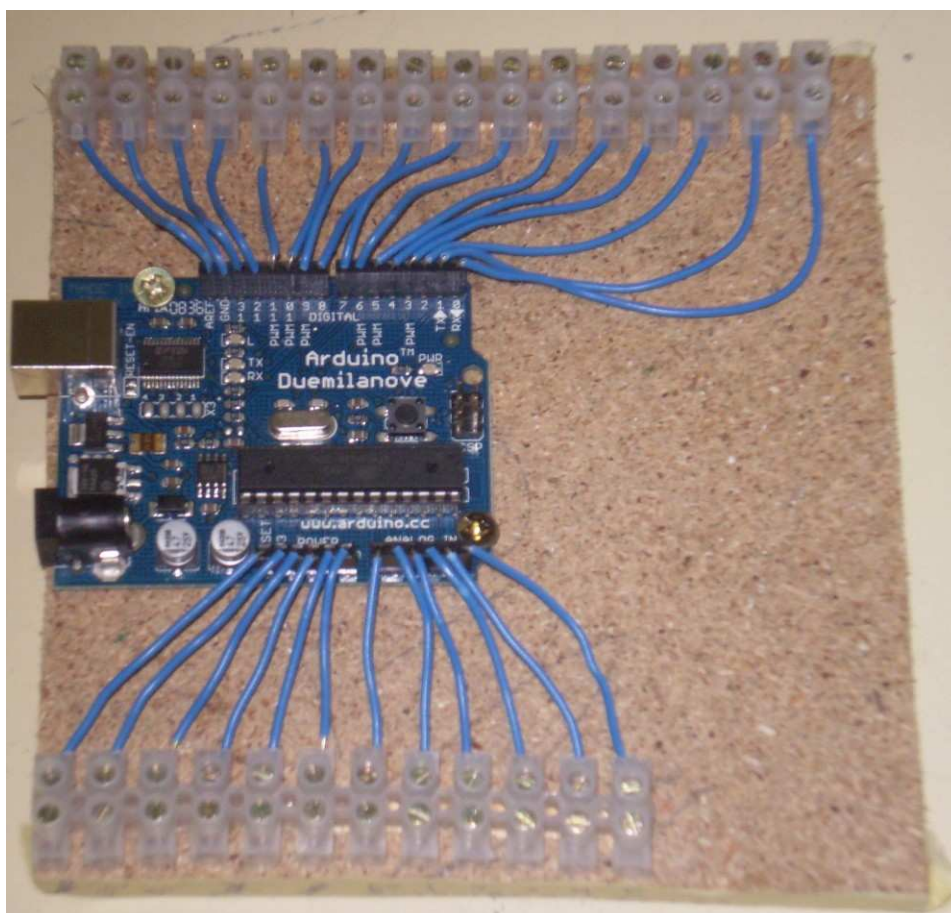
Actualmente o adestrador dixital ten problemas no 2º e 4º pulsadores (16-3-09).

5ª práctica: Activación dun motor paso a paso.

Esta práctica consiste na activación dun motor paso a paso modelo NMB PM42S-048-YTB3 utilizando como interfase a controladora por porto paralelo do PC. É, polo tanto, unha práctica en colaboración co grupo que traballa coa controladora, onde nós faremos o traballo relacionado co Arduino.

Para comezar imos a preparar o 2º Arduino de que dispoñemos (así podemos deixar o outro como adestrador dixital o cal nos permitirá ensaiar os programas nese adestrador antes de utilizalos en cada práctica.

Primeiro colocamos o Arduino nun soporte de madeira e lle poñemos tantas regretas de conexión como conectores ten a placa, etiquetando adecuadamente cada unha.



Así temos preparado este Arduino para conectalo á placa controladora ou a calquera outro dispositivo de entrada e saída a través das regretas de conexión.

O primeiro problema con que nos atopamos ó conectar a placa controladora co Arduino é a equivalencia entre as saídas do conector femia paralelo con cada un dos relés. En principio contamos con 8 conexións

de saída pero só temos 6 relés operativos na nosa controladora, e por isto importante comprobar que saídas do porto paralelo debemos usar.

Localización das saídas:

Para isto, unha vez conectadas a controladora e o Arduino, debemos usar un programa en Arduino que vaia activando as 8 posibles saídas e localizar as 6 que imos a utilizar, marcándoas convenientemente e anotándoas para futuras prácticas.

O programa que usamos foi o seguinte:

```
int contador = 0;
int pinessalida[] = {6, 7, 8, 9, 10, 11, 12, 13};
void setup() {
    for (contador=0;contador<8;contador++) {
        pinMode(pinessalida[contador], OUTPUT);
    }
    Serial.begin(9600);          // activa comunicación serie a 9600 bps
}
void loop() {
    Serial.print("c");           //Envía o carácter c, código ASCII 99 para indicar que comeza
    for (contador=0;contador<8;contador++) {
        Serial.print(pinessalida[contador], BYTE);    // Envía o valor de contador polo porto serie
        digitalWrite(pinessalida[contador], HIGH);
        delay(3000);
        digitalWrite(pinessalida[contador], LOW);
    }
}
```

Coa axuda deste programa localizamos as saídas que imos a usar tanto no Arduino como na controladora:

Pin saída dixital de Arduino	6	7	8	9	10	11	12	13
Nº de saída no cable		3	4	5	6	7	8	
Saída na placa controladora		S2	S3	S4	S5	S6	S7	

Como só necesitamos 6 pines do Arduino, imos a usar só os pines 7 ao 12. O pin 13 non nos gusta usalo como saída pois normalmente se activa nos programas de Arduino como un xeito de confirmación de inicio ou recepción de datos.

Localización das entradas:

Para resolver este problema debemos mirar a información de como está construída a controladora, para coñecer a equivalencia entre os terminais de entrada no cable paralelo e as entradas da controladora e despois buscar o seu equivalente nos pines de entrada dixital de Arduino.

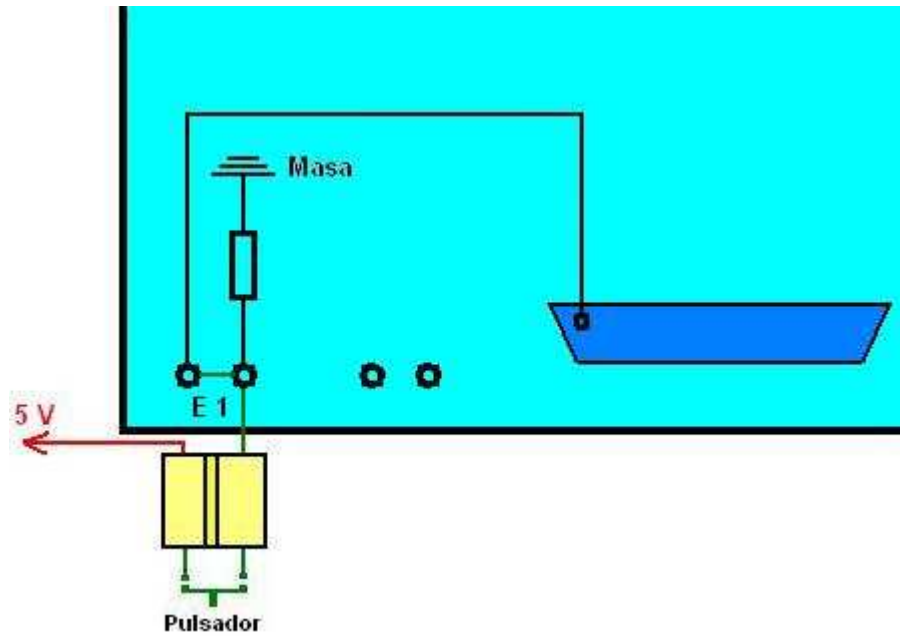
Segundo o esquema da placa controladora as entradas da controladora están unidos cos seguintes terminais do conector por porto paralelo:

Entrada na placa controladora	E1	E2	E3	E4	E5	E6	E7	E8
Nº de entrada no cable	<u>1</u>	<u>14</u>	16	<u>17</u>	13	12	<u>10</u>	11

Pin entrada dixital de Arduino	0	1		2	3	4	5	6
--------------------------------	---	---	--	---	---	---	---	---

(Nota: Os n° de cable da forma X indica que para o porto paralelo do PC esa sinal debía ter a polaridade invertida.)

Outro punto importante no funcionamento das entradas é como deben polarizarse. Mirando o esquema do circuío da controladora, podemos ver que tódalas entradas teñen a seguinte conexión:



A masa se subministra polo cable paralelo desde Arduino, e tal vez sexa necesario subministrar tamén os 5 V desde a placa Arduino.