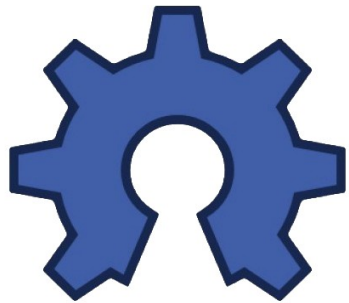




open hardware



open source



OPEN ACCESS

Arduino: Simon dice, paso a paso

Alberto Labarga – Experimental Serendipity S.L.

Laboratorio de Fabricación Digital, Mutilva, 14 de Marzo de 2014

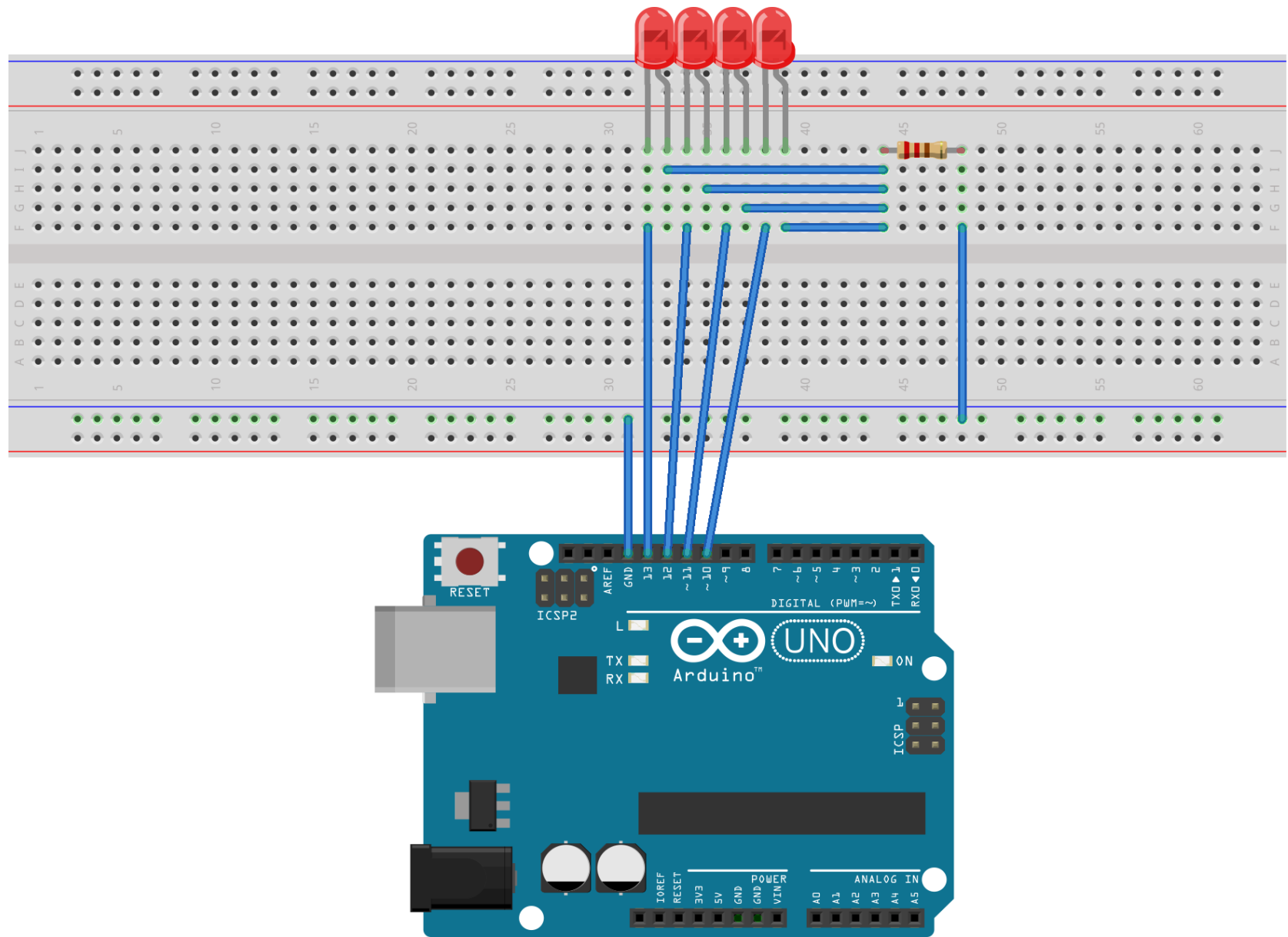


<http://www.apptivismo.org/laboratorio-fabricacion-digital/>



Paso a paso

- Reproducir una secuencia de luces
- Controlar los cambios de estado del pulsador
- Simón dice
- Cargar liberías. Reproducir sonidos
- Simón dice con sonidos



fritzing

Setup

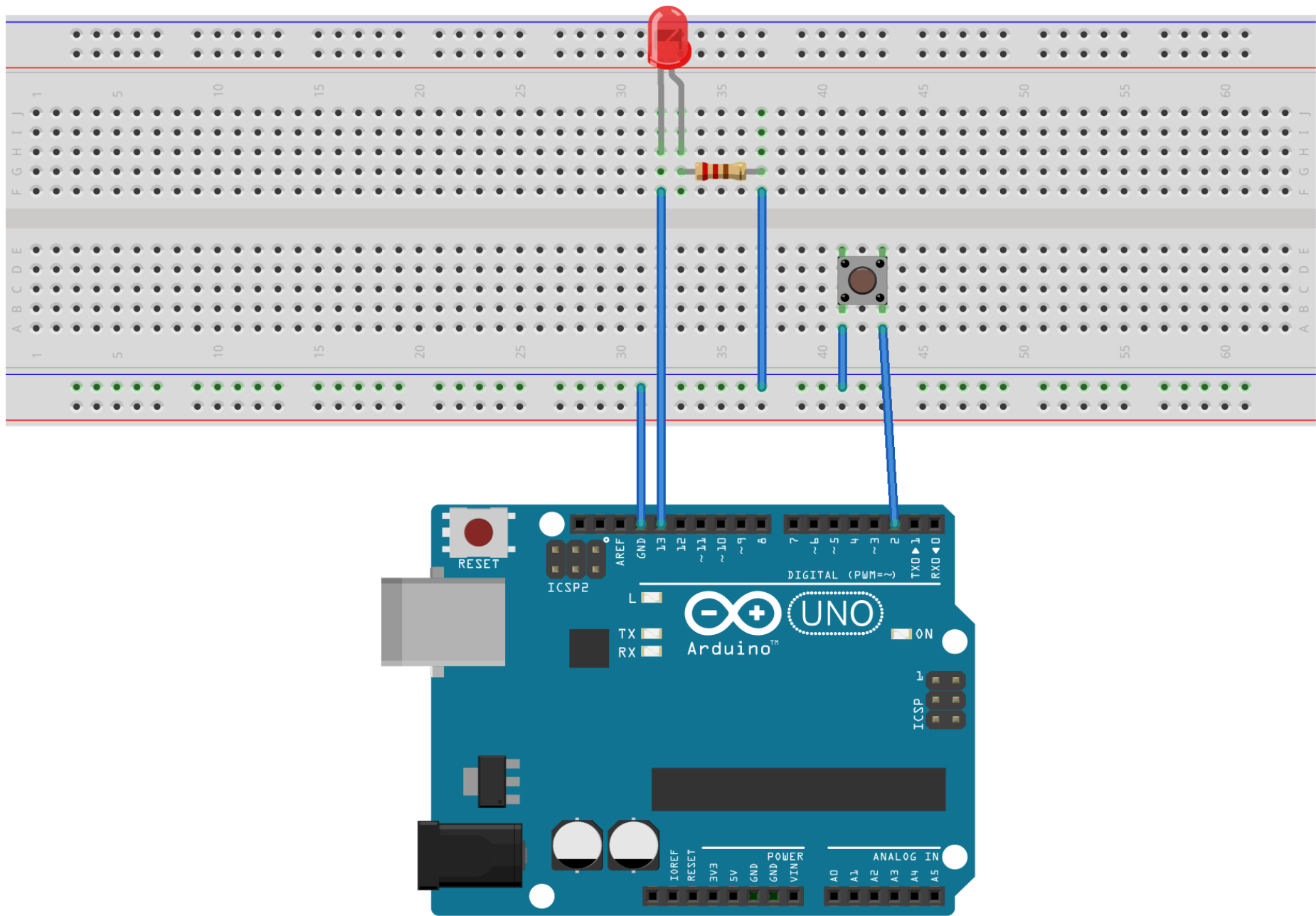
```
int simon[20];  
int leds[] = {10, 11, 12, 13}; // LED pins  
void setup() {  
  // inicializar los pines como salida  
  for (int i=0; i<4; i++){  
    pinMode(leds[i], OUTPUT);  
  }  
  // inicializar la semilla de nuemeros aleatorios  
  randomSeed(analogRead(0));  
}
```

Reproducir secuencia

```
void playSequence(int seq[],int n){  
  
    for (int i=0; i<n; i++){  
  
        int s = seq[i];    // qué led debo encender en el turno i?  
  
        int led = leds[s]; // qué pin corresponde a ese led?  
  
        digitalWrite(led, HIGH); // enciendo el led  
        delay(1000);             // espero un segundo  
        digitalWrite(led, LOW);  // apago el led  
        delay(1000);             // espero un segundo  
    }  
}
```

Loop

```
void loop() {  
    for (int n=0; n<10; n++){  
        simon[n] = random(0, 4);  
    }  
    playSequence(simon, 10);  
}
```



fritzing

<http://www.apptivismo.org/laboratorio-fabricacion-digital/descargas/codigo/CambioEstado/>

Pulsador con cambio de estado

```
// Variables para guardar los estados del pulsador
int estado = 1;          // estado del pulsador (como usamos PULL UP, comienza a HIGH)
int estado_anterior = 1; // estado anterior
bool encendido = false;  // el led comienza apagado

void setup() {

    // inicializamos el pin del pulsador como INPUT (con resistencia PULL UP interna)
    pinMode(buttonPin, INPUT_PULLUP);
    // inicializamos el pin del led como OUTPUT
    pinMode(ledPin, OUTPUT);

}
```

Pulsador con cambio de estado

```
void loop() {  
    // leemos el estado del pulsador  
    estado = digitalRead(buttonPin);  
  
    // comparamos el estado actual con el anterior  
    if (estado != estado_anterior) {  
        // cuando soltamos el pulsador  
        if (buttonState == HIGH) {  
            if (encendido) {          // si estaba encendido lo apago  
                digitalWrite(ledPin, LOW);  
                encendido = false;  
            } else { // si estaba apagado lo enciendo  
                digitalWrite(ledPin, HIGH);  
                encendido = true;  
            }  
        }  
    }  
  
    estado_anterior = estado;  
}
```

Reproducir secuencia

```
int simon[] = {0, 2, 1, 1, 3, 2, 0, 1, 2, 3}; // Secuencia de 10 elementos
```

```
int leds[] = {10, 11, 12, 13};           // LED pins
```

```
void setup() {
```

```
    // inicializar los pines como salida
```

```
    for (int i=0; i<4; i++){
```

```
        pinMode(leds[i], OUTPUT);
```

```
    }
```

```
}
```

Simon dice

```
int simon[20]; // tabla donde guardo la secuencia generada por el juego  
int jugador[20]; // tabla donde guardo la secuencia introducida por el jugador
```

```
int botones[] = {2, 3, 4, 5}; //The four button input pins  
int leds[] = {10, 11, 12, 13}; // LED pins
```

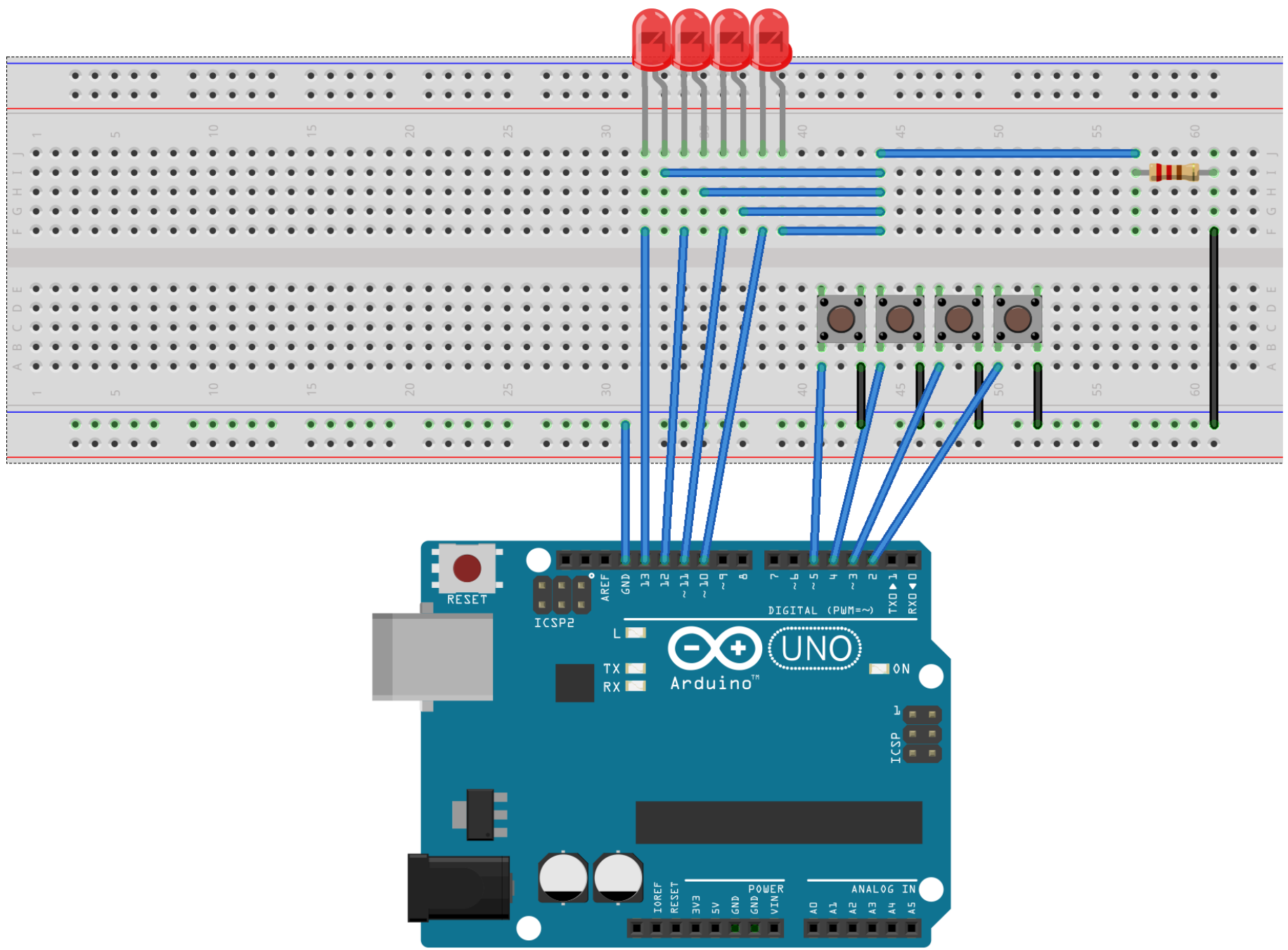
```
int turno = 0; // el turno  
boolean pulsado = false; // hemos pulsado el boton  
boolean game_over = false; // hemos perdido
```

```
// variables auxiliares
```

```
int ledPin;  
int botonPin;  
int estado;
```

Simon dice

```
void setup() {  
    // inicializar los pines como salida  
    for (int i=0; i<4; i++){  
        pinMode(leds[i], OUTPUT);  
    }  
    // inicializar los pines botones como entrada  
    for (int i=0; i<4; i++){  
        pinMode(botones[i], INPUT_PULLUP);  
    }  
    // inicializar la semilla de numeros aleatorios  
    randomSeed(analogRead(0));  
    // abrir la conexión serie con el PC  
    Serial.begin(9600);  
}
```



fritzing

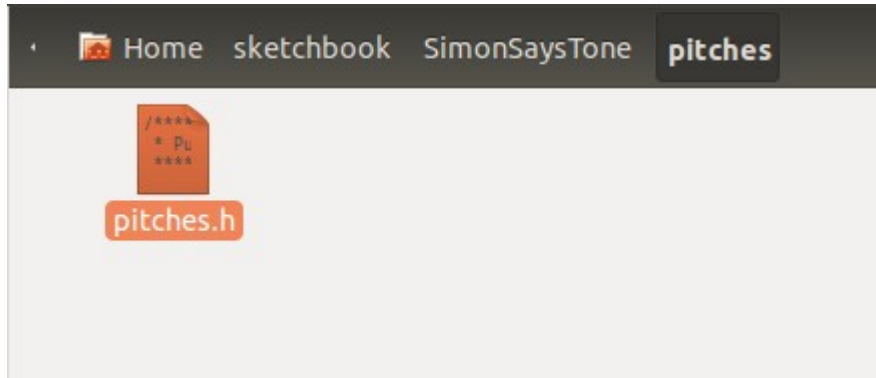
Simon dice

```
// reproducir una secuencia seq de longitud n
void playSequence(int seq[],int n){
  int s;
  // para cada elemento de la secuencia, enciendo y apago el led
  for (int i=0; i<n; i++){
    s = seq[i];
    ledPin = leds[s];
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(500);
    // mostrar info en pantalla
    Serial.print(s);
    Serial.print(" ");
  }
  Serial.println();
  Serial.println("Tu turno");
}
```

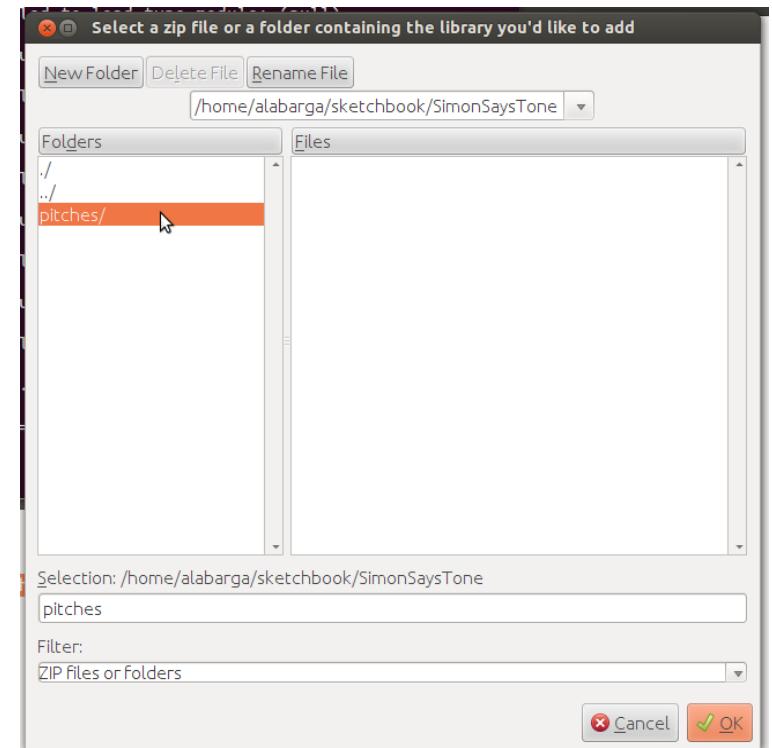
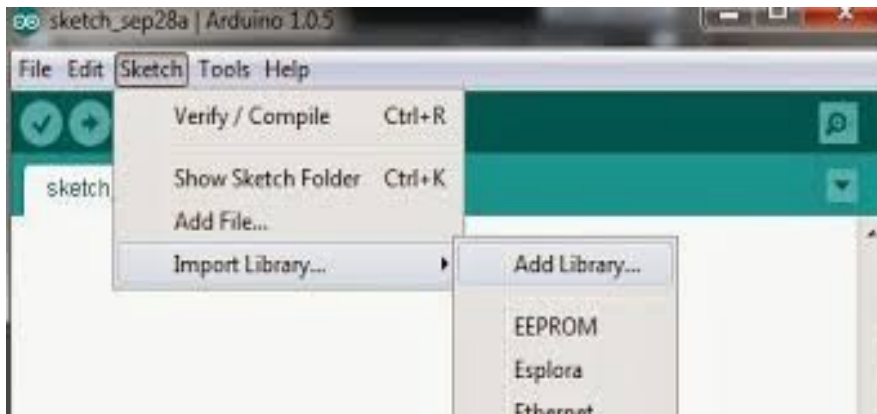
Leer secuencia

```
• // ahora leo la secuencia del jugador
• for (int n=0; n<turno; n++){
• // espero a que pulse un botón
• while (!pulsado){
• // voy leyendo los botones uno a uno
• for (int i=0; i<4; i++){
• // miro que pin toca
• botonPin = botones[i];
• // leo el estado del pulsador en ese pin
• estado = digitalRead(botonPin);
• // si está a LOW quiere decir que está pulsado (usamos PULL UP interno)
• if (estado == LOW){
•
•     pulsado = true; // actualizamos el estado
•     jugador[n] = i; // almacenamos el número de botón en la tabla del jugador
•
•     ledPin = leds[i]; // miro que pin es el del led correspondiente
•     digitalWrite(ledPin, HIGH); // enciendo el pin
•
•     // mostrar info en pantalla
•     Serial.print("has pulsado ");
•     Serial.println(i);
•
•     // si no es el color que tocaba, hemos perdido
•     if (jugador[n] != simon[n]) {
•         game_over = true;
•     }
•
•     break; //salimos del bucle, ya no hace falta mirar los botones que faltan
• }
• }
• }
```


Importar librerías



Ponemos el fichero de cabecera (*.h) en un directorio con el mismo nombre





<http://www.apptivismo.org/laboratorio-fabricacion-digital/descargas/codigo/SimonSaysTone/>