Trabajo Práctico 4: Entrada/Salida

Luz Alba Posse, Josefina Jahde, Dafydd Jenkins, Valentino Cannestraci

Tecnología Digital II - Junio 2023

1 Trabajo Previo

¿Cuál es la diferencia entre un pin digital y un pin analógico?

Los pines digitales en Arduino son utilizados para leer o escribir señales digitales, lo que significa que solo pueden tomar dos valores distintos: HIGH (alto) o LOW (bajo). Estos pines son comúnmente utilizados para conectar y controlar dispositivos que trabajan con señales digitales, como LEDs, botones, etc. Cada pin digital en una placa Arduino puede funcionar como una entrada (para leer señales) o como una salida (para enviar señales).

Después, están los pines analógicos que son utilizados para leer señales analógicas, lo que significa que pueden tomar valores continuos en un rango específico. Estos pines son utilizados para leer sensores analógicos, como potenciómetros, sensores de temperatura, sensores de luz, etc. Los pines analógicos pueden tomar valores desde 0 hasta un valor máximo determinado por la resolución del conversor analógico-digital (ADC) de la placa, generalmente 10 bits en las placas más comunes.

¿Qué hace la función `digitalPinToInterrupt`?

La función `digitalPinToInterrupt()` se utiliza para asignar un número de interrupción a un pin digital específico. Las interrupciones son eventos que pueden interrumpir la ejecución típica del programa cuando ocurre algún cambio en un pin específico. Cuando se produce una interrupción, el programa puede detener temporalmente lo que está haciendo y ejecutar una función especial llamada "rutina de interrupción" que se asoció con ese pin.

La función `digitalPinToInterrupt()` toma como argumento el número del pin digital y devuelve el número de interrupción correspondiente. Es útil cuando se quiere utilizar una interrupción en un pin digital específico. Algunas placas Arduino tienen pines designados para admitir interrupciones, y `digitalPinToInterrupt()` proporciona la correspondencia entre los pines digitales y los números de interrupción asociados.

¿Para qué sirve la función `map`?

Se utiliza para convertir un valor de un rango a otro rango. Permite escalar o mapear un valor de entrada a un rango diferente de salida. La sintaxis es:

```
map(valor, rangoEntradaMin, rangoEntradaMax, rangoSalidaMin, rangoSalidaMax);
```

- `valor`: el valor que se desea mapear al nuevo rango.
- `rangoEntradaMin` y `rangoEntradaMax`: el rango de valores de entrada dentro del cual se encuentra el valor.
- `rangoSalidaMin` y `rangoSalidaMax`: el rango de valores de salida al que se desea mapear el valor.

Entonces `map()` toma el valor de entrada y lo convierte proporcionalmente al nuevo rango especificado. Por ejemplo, si quieres mapear un valor de entrada que está en el rango de 0 a 1023 (como el resultado de la lectura de un sensor analógico) al rango de salida deseado que es, por ejemplo, de 0 a 255, puedes usar la función map de la siguiente manera:

```
int valorEntrada = analogRead(A0);
int valorMapeado = map(valorEntrada, 0, 1023, 0, 255);
```

En este ejemplo, el valor de entrada se mapea del rango de 0 a 1023 al rango de 0 a 255. Si el valor de entrada es 512, la función map() convertirá ese valor a aproximadamente 127 en el nuevo rango.

Es útil cuando se necesita adaptar un valor de entrada a un rango específico requerido por un componente o dispositivo en particular. En general, se usa para mapear valores de sensores analógicos a un rango de salida deseado o para ajustar valores de entrada antes de enviarlos a un actuador.

¿Para que sirve habilitar la conexión serial?

Habilitar la conexión serial permite establecer una comunicación bidireccional entre la placa Arduino y un dispositivo externo, como una computadora o un módulo Bluetooth. La conexión serial se basa en la comunicación serie, donde los datos se transmiten uno después del otro a través de un único canal de comunicación.

2 Uso de Leds

Explicar que realiza el programa.

```
const int ledPin0 = 4; // Acá se configuran los pines de los leds.
```

```
const int ledPin1 = 5;
const int ledPin2 = 6;
void setup() {
pinMode(ledPin0, OUTPUT);//Se define que se van a usar para
pinMode(ledPin1, OUTPUT); // enviar señales (controlar leds)
pinMode(ledPin2, OUTPUT); }
void loop() {
digitalWrite(ledPin0, 1); // Se prenden los tres
digitalWrite(ledPin1, 1); // pines (están en 1)
digitalWrite(ledPin2, 1);
delay(1000);
digitalWrite(ledPin0, 0); // Se apagan los tres pines
digitalWrite(ledPin1, 0); // (están en 0)
digitalWrite(ledPin2, 0);
delay(1000); }
```

Este código enciende los leds conectados a los pines 4, 5 y 6 de la placa Arduino, espera 1 segundo y después los apaga.

Opción A - Leds

```
/* Modificar el programa para que presente la siguiente secuencia
de encendido de leds: 000, 001, 011, 111, 110, 100. La misma se debe
repetir todo el tiempo en intervalos regulares de 1 segundo entre cada valor.*/
const int ledPin0 = 4;
const int ledPin1 = 5;
const int ledPin2 = 6;
void setup() {
pinMode(ledPin0, OUTPUT);
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT); }

void loop() {
// 000
digitalWrite(ledPin0, 0);
digitalWrite(ledPin1, 0);
digitalWrite(ledPin2, 0);
delay(1000);
// 001
digitalWrite(ledPin0, 0);
digitalWrite(ledPin1, 0);
digitalWrite(ledPin2, 1);
delay(1000);
// 011
digitalWrite(ledPin0, 0);
```

```
digitalWrite(ledPin1, 1);
digitalWrite(ledPin2, 1);
delay(1000);
// 111
digitalWrite(ledPin0, 1);
digitalWrite(ledPin1, 1);
digitalWrite(ledPin2, 1);
delay(1000);
// 110
digitalWrite(ledPin0, 1);
digitalWrite(ledPin1, 1);
digitalWrite(ledPin2, 0);
delay(1000);

// 100
digitalWrite(ledPin0, 1);
digitalWrite(ledPin1, 0);
digitalWrite(ledPin2, 0);
delay(1000);
}
```

Opción B - Leds

```
/*Modificar el programa anterior para que por cada iteración de la secuencia,
el tiempo entre cada valor se reduzca en 0,05 segundos. Una vez que llegue a
0 se debe volver a reiniciar a 1 segundo*/

const int ledPin0 = 4;
const int ledPin1 = 5;
const int ledPin2 = 6;

void setup() {
  pinMode(ledPin0, OUTPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
}

void loop() {
  int tiempo = 1000;

  while (tiempo > 0) {
    digitalWrite(ledPin0, HIGH);
    digitalWrite(ledPin1, HIGH);
    digitalWrite(ledPin2, HIGH);
    delay(tiempo);
```

```
    digitalWrite(ledPin0, LOW);
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    delay(tiempo);

    tiempo = tiempo - 50;
      if (tiempo <= 0) {
      tiempo = 1000;
    }
  }
}
```

3 Uso de Joystick

Explicar qué hace el programa

```
const int ledPin0 = 4; // Asigna el pin 4 al led (es digital)
const int VRy = A0; // Asigna el pin A0 que es analógico
const int VRx = A1;
const int SW = 2;
void setup() {
Serial.begin(9600);
pinMode(ledPin0, OUTPUT); // Esto va a enviar señales
pinMode(VRx, INPUT); // Esto va a recibir señales
pinMode(VRy, INPUT);
pinMode(SW, INPUT_PULLUP);
}
void loop() {
int xPosition = analogRead(VRx);
int yPosition = analogRead(VRy);
int SW_state = digitalRead(SW);
int mapX = map(xPosition, 0, 1023, -512, 512); // Convierte la posición
int mapY = map(yPosition, 0, 1023, -512, 512);
Serial.println(mapX); // Lo imprime en monitor serie
Serial.println(mapY);
Serial.println(SW_state);
digitalWrite(ledPin0, SW_state);
delay(10);
}
```

Básicamente, se leen e interpretan las entradas del joystick y el estado de un botón. O sea, lee las posiciones X e Y del joystick, el estado del botón y los imprime en el monitor serie. También controla un LED en función del estado de ese botón. Entonces, muevo el joystick y pasan cosas.

Opción A - Joystick

```
/* Modificar el programa de forma que se enciendan
los leds respetando el siguiente patron:
010 -> arriba
101 -> abajo
110 -> izquierda
011 -> derecha
El umbral de encendido debe ser
del 10% del recorrido del joystick */

const int ledPin0 = 4;
const int ledPin1 = 5;
const int ledPin2 = 6;
const int VRy = A0;
const int VRx = A1;
const int SW = 2;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin0, OUTPUT);

  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  pinMode(SW, INPUT_PULLUP);
}

void loop() {
  int xPosition = analogRead(VRx); // Lees analógicamente
  int yPosition = analogRead(VRy); // Aca tambien jaja
  int SW_state = digitalRead(SW); // Lees digital el botón.
  int mapX = map(xPosition, 0, 1023, -512, 512);
  int mapY = map(yPosition, 0, 1023, -512, 512);

  Serial.println(mapX);
  Serial.println(mapY);
  Serial.println(SW_state);

  // Verificar los patrones de movimiento
  if (mapX < 51 && -51 < mapX && mapY >= 51) {
  // Patrón: arriba (010) // 10% es 461
    digitalWrite(ledPin0, 0);
    digitalWrite(ledPin1, 1);
    digitalWrite(ledPin2, 0);
  } else if (mapX < 51 && -51 < mapX && mapY < -51) {
  // Patrón: abajo (101)
```

```
    digitalWrite(ledPin0, 1);
    digitalWrite(ledPin1, 0);
    digitalWrite(ledPin2, 1);
  } else if (mapX <= -51 && mapY >= -51 && mapY <= 51) {
  // Patrón: izquierda (110)
    digitalWrite(ledPin0, 1);
    digitalWrite(ledPin1, 1);
    digitalWrite(ledPin2, 0);
  } else if (mapX >= 51 && mapY >= -51 && mapY <= 51) {
  // Patrón: derecha (011)
    digitalWrite(ledPin0, 0);
    digitalWrite(ledPin1, 1);
    digitalWrite(ledPin2, 1);
  }
  else {
    digitalWrite(ledPin0, 0);
    digitalWrite(ledPin1, 0);
    digitalWrite(ledPin2, 0);
  }


  delay(10);
}

/*Se utilizan las variables mapX y mapY para almacenar las lecturas mapeadas
de los sensores analógicos. Después, se verifica el estado de los botones y las
lecturas mapeadas para determinar el patrón de movimiento. Si se encuentra una
coincidencia con uno de los patrones especificados, se enciende el LED (ledPin0).*/
```

4 Uso de Servomotor

Explicar qué hace el programa

```
#include <Servo.h>
const int servoPin = 10; // En qué pin está el servomotor
Servo servo1;
void setup() {
servo1.attach(servoPin);
} void loop() {
servo1.write(0); // Ángulo 0
delay(1000);
servo1.write(30); // Ángulo 30
delay(1000);
servo1.write(60); // Ángulo 60
```

```
delay(1000);
servo1.write(90); // Ángulo 90
delay(1000);
servo1.write(180); // Ángulo 180
delay(1000);
}
```

El código mueve el servomotor conectado al pin 10 en diferentes ángulos (0, 30, 60, 90 y 180 grados) en un ciclo. Cada posición se mantiene durante un segundo antes de pasar a la siguiente posición. Esto permite controlar la posición del servomotor y crear movimientos específicos según los ángulos proporcionados.

Opción A - Servomotor

```
/*Modificar el programa para que el Servo reproduzca
el siguiente patrón de comportamiento
una y otra vez, cambiando de posición cada 2 segundos.
1 0 grados
2 90 grados 3 45 grados 4 180 grados */

#include <Servo.h>
const int servoPin = 10;
Servo servo1;
void setup() {
servo1.attach(servoPin);
}
void loop() {
servo1.write(0);
delay(2000);
servo1.write(90);
delay(2000);
servo1.write(45);
delay(2000);
servo1.write(180);
delay(2000);
}
```

Opción B - Servomotor

```
/*Modificar el programa para que el movimiento de izquierda a
derecha del Joystick, reproduzca el mismo comportamiento
que el Servo de 0 a 180 grados.*/

#include <Servo.h>
```

```
const int VRx = A1;
const int servoPin = 10;
Servo servo1;

void setup() {
  servo1.attach(servoPin);
}

void loop() {
  int xPosition = analogRead(VRx);
  int mapX = map(xPosition, 0, 1023, 0, 180);
  servo1.write(mapX);
  delay(10);
}
```

5 Interrupciones

Explicar qué hace esta función

```
#include <Servo.h>
Servo servo1;
const int servoPin = 10; // Define el pin
const byte interruptPin = 2;
volatile byte button = 0;
const int ledPin0 = 4;
const int ledPin1 = 5;
const int VRy = A0;
const int VRx = A1;
const int SW = 2;
void setup() {
servo1.attach(servoPin); // Le digo que el servo está acá
pinMode(ledPin0, OUTPUT); // Esto envia señales
pinMode(ledPin1, OUTPUT);
pinMode(interruptPin, INPUT_PULLUP); // Recibe señales
attachInterrupt(digitalPinToInterrupt
(interruptPin), click, RISING); pinMode(VRx, INPUT);
pinMode(VRy, INPUT);
pinMode(SW, INPUT_PULLUP);
}
void loop() {
int xPosition = analogRead(VRx);
// Se guarda en variables lo que se lee
int yPosition = analogRead(VRy);
int SW_state = digitalRead(SW);
```

```
int angulo1 = map(xPosition, 0, 1023, 0, 180);
// Mapea para convertir a ángulo
servo1.write(angulo1);
int brillo = map(yPosition, 0, 1023, -255, 255);
// Mapea para convertir a brillo o intensidad
analogWrite(ledPin1, abs(brillo));
if( button ) {
digitalWrite(ledPin0, 1);
} else {
digitalWrite(ledPin0, 0);
}
}
void click() {
if( button == 0 ) {
button = 1;
} else {
button = 0;
}
}
```

Utiliza las lecturas del joystick para controlar la posición de un servomotor y el brillo de un LED. Además, utiliza un botón para cambiar el estado de una variable que controla el encendido/apagado de otro LED. La posición del joystick se mapea a un ángulo y se utiliza para mover el servomotor, mientras que el valor del eje Y del joystick se mapea a un brillo y se aplica a un LED.

Opción A - Interrupciones

```
/*Modificar el programa para que cada vez que se presiona el
botón del Joystick se prendan todos los leds y estos queden
prendidos. Cuando se presione nuevamente,
estos se deben apagar.*/

#include <Servo.h>
Servo servo1;
const int servoPin = 10;
const byte interruptPin = 2;
volatile byte button = 0;
const int ledPin0 = 4;
const int ledPin1 = 5;
const int ledPin2 = 6;
const int VRy = A0;
const int VRx = A1;
const int SW = 2;
```

```
int time_stamp = 0;
// Usamos time_stamp para solucionar el problema con las
interrupciones

void setup() {
  servo1.attach(servoPin);
  pinMode(ledPin0, OUTPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin),
  click, RISING);
  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  pinMode(SW, INPUT_PULLUP);

}

void loop() {
  int xPosition = analogRead(VRx);
  int yPosition = analogRead(VRy);
  int SW_state = digitalRead(SW);
  int angulo1 = map(xPosition, 0, 1023, 0, 180);
  servo1.write(angulo1);
  int brillo = map(yPosition, 0, 1023, -255, 255);
  analogWrite(ledPin1, abs(brillo));

  if (button) {
    digitalWrite(ledPin0, HIGH);
    digitalWrite(ledPin1, HIGH);
    digitalWrite(ledPin2, HIGH);
  } else{
    digitalWrite(ledPin0, LOW);
    //digitalWrite(ledPin1, LOW);
    Como se juega con el brillo de este pin,
    lo dejamos relacionado al estado que tenga
    digitalWrite(ledPin2, LOW);
  }
  if (time_stamp > 0){
    time_stamp = time_stamp - 1;
  }
  delay(2);
}

void click() {
  if(time_stamp == 0){
```

```
if (button == 0) {
    button = 1;
    time_stamp = 250;
  } else {
    button = 0;
    time_stamp = 250;
  }
  }
}
```

Opción B - Interrupciones

```
/*Modificar el programa para que cada vez que se presiona
el botón del Joystick se intercambie el sentido de rotación
del Servo.*/

#include <Servo.h>

Servo servo1;
const int servoPin = 10;
const byte interruptPin = 2;
volatile byte button = 0;
const int ledPin0 = 4;
const int ledPin1 = 5;
const int VRy = A0;
const int VRx = A1;
const int SW = 2;
int time_stamp = 0;

void setup() {
  servo1.attach(servoPin);
  pinMode(ledPin0, OUTPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin),
  click, RISING);
  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  pinMode(SW, INPUT_PULLUP);
}

void loop() {
  int xPosition = analogRead(VRx);
  int yPosition = analogRead(VRy);
  int SW_state = digitalRead(SW);
  int angulo1 = (map(xPosition, 0, 1023, 0, 180));
```

```
  if (button == 1) {
    digitalWrite(ledPin0, HIGH);
    angulo1 = (map(xPosition, 0, 1023, 180, 0));
  } else {
    digitalWrite(ledPin0, LOW);
  }

  servo1.write(angulo1);
  int brillo = map(yPosition, 0, 1023, -255, 255);
  analogWrite(ledPin1, abs(brillo));

  if (time_stamp > 0) {
    time_stamp = time_stamp - 1;
  }

  delay(10);

}

void click() {
  if (time_stamp == 0) {
    if (button == 0) {
      button = 1;
      time_stamp = 40;
    } else {
      button = 0;
      time_stamp = 40;
    }
  }
}
```

6 Resumen

```
/*Utilizado lo aprendido en los puntos anteriores
construir un programa que respete el siguiente comportamiento:
Los leds van a mostrar contando del 000 al 111 la posición del
Joystick en sentido vertical.
El movimiento del Joystick en sentido horizontal deberá mover
el Servo en todo su rango.
Si se presiona el botón del Joystick se
intercambiará el comportamiento de los ejes vertical y horizontal. */

#include <Servo.h>
```

```
Servo servo1;
const int servoPin = 10;
const byte interruptPin = 2;
volatile byte button = 0;
const int ledPin0 = 4;
const int ledPin1 = 5;
const int ledPin2 = 6;
const int VRy = A0;
const int VRx = A1;
const int SW = 2;
int time_stamp = 0;

void setup() {
  servo1.attach(servoPin);
  pinMode(ledPin0, OUTPUT);
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), click, RISING);
  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  pinMode(SW, INPUT_PULLUP);

  // Estado inicial: LEDs verticales, servomotor horizontal
  int horizontal = 90;
  // Posición horizontal media para el servomotor
  int vertical = 0;
  // Valor inicial para los LEDs verticales
  servo1.write(horizontal);
  digitalWrite(ledPin0, (vertical & 1));
  digitalWrite(ledPin1, (vertical & 2));
  digitalWrite(ledPin2, (vertical & 4));
}

void loop() {
  int xPosition = analogRead(VRx);
  int yPosition = analogRead(VRy);
  int horizontal, vertical;

  if (button) {
    horizontal = map(yPosition, 0, 1023, 0, 180);
    vertical = map(xPosition, 0, 1023, 0, 7);
  } else {
    horizontal = map(xPosition, 0, 1023, 0, 180);
    vertical = map(yPosition, 0, 1023, 0, 7);
  }
```

```
  servo1.write(horizontal);

  digitalWrite(ledPin0, (vertical & 1));
  digitalWrite(ledPin1, (vertical & 2));
  digitalWrite(ledPin2, (vertical & 4));

  if (time_stamp > 0) {
    time_stamp = time_stamp - 1;
  }

  delay(10);
}

void click() {
  if (time_stamp == 0) {
    if (button == 0) {
      button = 1;
      time_stamp = 40;
    } else {
      button = 0;
      time_stamp = 40;
    }
  }
}
```

Repositorio: https://github.com/luzalbaposse/Entrada-Salida-Arduino/tree/main