# Obligatorio 3 - Monitor Serie, Control de flujo, tiempos y 2 patas.

IIE - Facultad de Ingeniería - Universidad de la República

Tallerine Biónico 2024

El objetivo de este obligatorio es comprender el funcionamiento de la herramienta “Monitor Serie” y formalizar brevemente el concepto de control de flujo y sus diferentes formas de implementarlo.

Además se agrega una forma práctica de obtener el tiempo que demora un servo en recorrer un ángulo dado.

Por último, armar 2 patas.

## 1. Monitor Serie

El Monitor Serie de Arduino IDE es una interfaz para poder testear el funcionamiento de los programas. Esta interfaz permite muy fácilmente enviar y recibir información entre la placa Microbit y la terminal.

Para utilizar el Monitor Serie en la Microbit se dispone de una librería especial, la cual no es necesario declarar ni generar la variable asociada (variable objeto) pues esto ya existe por defecto. La variable objeto para el Monitor Serie se llama `Serial`.

Las funciones a utilizar son:

- `Serial.begin(9600);` Inicializa el Monitor a 9600 baudios.
- `Serial.available();` Devuelve VERDADERO si hay un dato nuevo.
- `Serial.read();` Para leer un dato presente en el Monitor.
- `Serial.print("texto")`; Escribe “texto” en el Monitor.
- `Serial.println("texto");` Escribe “texto” con “retorno de carro” en el Monitor.

El Monitor Serie en Arduino IDE se abre con el icono con forma de lupa que se encuentra en la parte superior derecha de la ventana, con *Herramientas -> Monitor Serie o* mediante las teclas Ctrl+Mayúscula+M. Al activarlo se despliega una nueva ventana (figura 1).

Una vez abierta la nueva ventana, en el campo superior se muestra lo que se le envía desde el monitor a la placa (“a” en la figura 1) y debajo del anterior lo que se escribe en el monitor desde la placa (“Dato correcto……” en la figura 1).

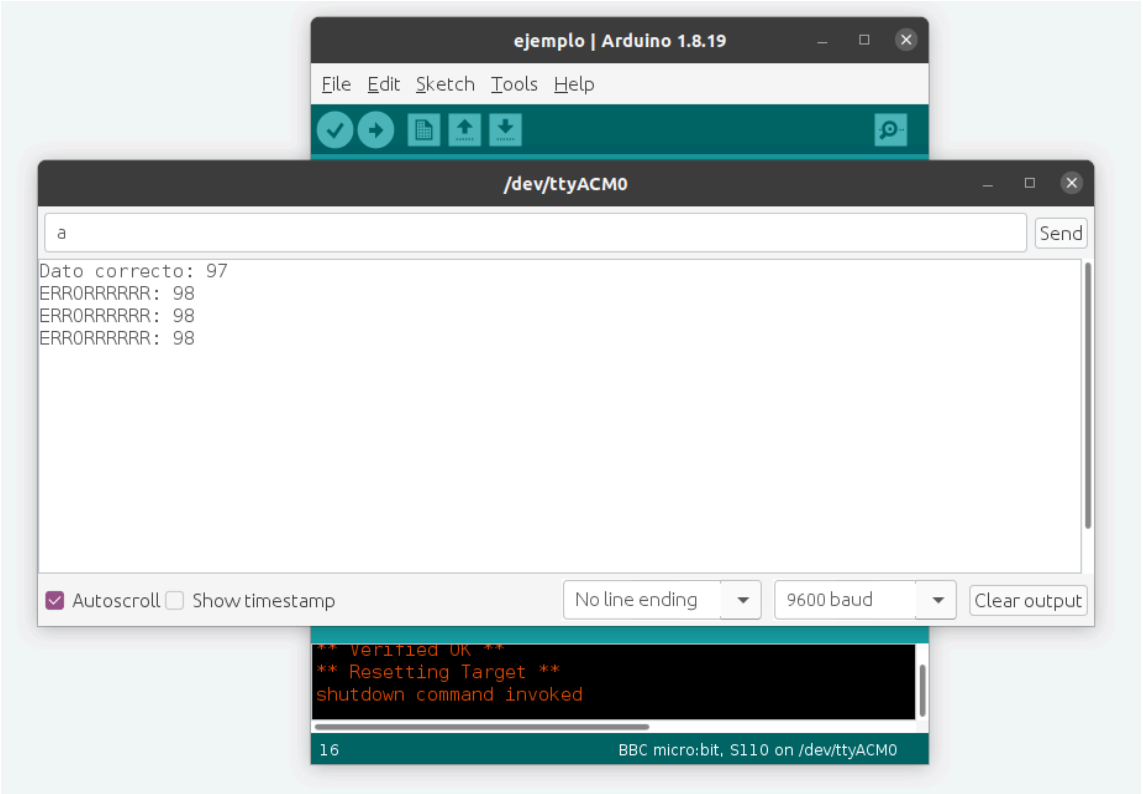


***Figura 1**: Monitor Serie abierto.*

El siguiente ejemplo muestra el sketch **`serial_microbit.ino`** el cual está disponible en EVA.

```
const int COL1 = 3;          // columna 1 Cj
const int LED = 26;          // 'fila 1' Ri
int dato;                    // define la variable "dato" como un número entero.

void setup() {
  Serial.begin(9600);        //Se inicializa el monitor en 9600 Baudios
  pinMode(COL1, OUTPUT);
  digitalWrite(COL1,LOW);    // habilita la Columna 1
  pinMode(LED,OUTPUT);
  digitalWrite(LED, LOW);    // LED inicialmente apagado.
}

void loop() {
   if (Serial.available()>0){   //Se fija si hay un nuevo dato disponible enviado desde el monitor
     dato =  Serial.read();     //Lee el número
     if (dato == 'a')    {      //Si el valor leído es la letra 'a'
        Serial.print("Dato correcto: ");
        Serial.println(dato);
        digitalWrite(LED, HIGH);
     }
     else {                          // Si el valor no es 'a'
        Serial.print("ERRORRRRRR: ");
        Serial.println(dato);
        digitalWrite(LED, LOW);
     }
   }
   delay(50);                  // Retardo de 50ms
}
```

## 2. Control de Flujo

El control de flujo refiere al control del orden en que se ejecutan las instrucciones de un programa. Si no existiera el control de flujo, los programas se ejecutarían en forma secuencial, comenzando por la primera instrucción y finalizando por la última. Las estructuras de control de flujo más importantes para el curso son:

A. if-else
B. switch case
C. while
D. for

### A. if-else

Ya se trabajó con esta estructura y se asume que se conoce bien.

### B. switch - case

Al igual que el **`if-else`**, el **`switch-case`** es una estructura de control de flujo del programa que permite ejecutar una porción de código u otra dependiendo de la condición que se cumple. Mientras el `if-else` se utiliza cuando se cumple una condición o no, el **`switch-case`** se aplica cuando las condiciones son múltiples.

Sintaxis

```
switch (var) {
  case label1:
          // código para valor 1
          break;
  case label2:
          // código para valor 2
          break;
  case label3:
          // código para valor 3
          break;
  default:
          // código para demás casos
          break;
}
```

La sentencia **`switch`** compara el valor de una variable **`(var)`** con los valores especificados en cada una de las etiquetas **`(label)`** de los **`case`**. Cuando se encuentra un **`case`** cuyo valor coincide con el de la variable, se ejecuta el código bajo ese **`case`**.

La palabra clave **`break`** se utiliza al final de cada caso para indicarle al programa que abandone el **`switch`**. Si no se utiliza **`break`**, se continuará ejecutando el siguiente **`case label`**. Si la variable **`var`** no coincide con ninguna de las etiquetas, se ejecuta el código bajo la sentencia **`default`**. Incluir el **`default`** es opcional.

Se dispone del código **`swith_case_microbit.ino`** (disponible en EVA) el cual espera un comando (’a’,’b’ o ’c’) proveniente del Monitor Serie y mueve un servo a un ángulo determinado de acuerdo a:

- 'a' → 0°
- 'b' → 45°
- 'c' → 90°
- 'otro' → 180°

```
#include <udriver_pca9685_fing.h>
using namespace UDriver_PCA9685;

///Servos
const I2CAddress ADDR = 0x6A;          // Lo requiere la librería.
PCA9685ServoController servo_ctrl(ADDR); //Crea la variable "servo_ctrl"
const pin_t PIN_SERVO = SV1;           // Pin donde se conecta el servo (cable naranja)
                                       // pin_t se define en la librería y refiere a SV1 ...SV16
char comando;                          // Indicar el comando ingresado

void setup()
{
     Serial.begin(9600);   // Inicializo a 9600 el monitor serial
     servo_ctrl.begin();         // Inicializa el controlador de servos
     servo_ctrl.move_servo(PIN_SERVO, 135); // Posición inicial del servo en 135°
}
void loop()
{
     if (Serial.available())
     {
          comando = Serial.read(); // lee el comando
          switch (comando){ // ejecuta según el comando leído
               case 'a':    servo_ctrl.move_servo(PIN_SERVO, 0);
                            delay(500);
                            break;
               case 'b':    servo_ctrl.move_servo(PIN_SERVO, 45);
                            delay(500);
                            break;
               case 'c':    servo_ctrl.move_servo(PIN_SERVO, 90);
                            delay(500);
                            break;
               default:     servo_ctrl.move_servo(PIN_SERVO, 180);
                            delay(500);
     }
delay(50);
}
```

Observar que si no hay un dato disponible en el Monitor Serie, el switch-case no se ejecuta. Se puede observar que la estructura es muy intuitiva y rápida de entender. Se sugiere pensar como resolver este mismo problema utilizando únicamente la estructura **`if-else`**.

## C. while

Esta estructura ejecuta un conjunto de instrucciones repetidas veces mientras una condición se mantiene verdadera. La condición debe cambiar de alguna forma para dejar de ejecutar estas instrucciones.

Ejemplo 1:La condición se modifica en el propio código

```
var = 0;
while (var < 200)
{
// instrucciones a ejecutar 200 veces
var++; // Incrementa en 1 el valor de    var
}
```

Ejemplo 2:La condición se modifica en forma externa.

```
while( digitalRead (10) == LOW ) //lee pin 10 y lo compara con LOW
{
// instrucciones a ejecutar un n mero  indeterminado de veces,
// hasta que el pin 10 sea HIGH
}
```

***Figura 2**. Estructura **while**. La condición se modifica: arriba en el propio código y abajo de forma externa.*

En general la estructura **while** no se utiliza cuando se conoce a priori el número de iteraciones. Para esto está la estructura **for**.

## D. for

La estructura ***for*** se utiliza para ejecutar un conjunto de instrucciones un número determinado de veces. Se utiliza un contador el cual se incrementa (o decrementa) en cada pasada. La cantidad de veces está determinada por una condición.

Ejemplo 1:Contador por incremento

```
for (i=0; i< 200;i++)
{
// instrucciones a ejecutar 200 veces
}
```

Ejemplo 2:Contador por decremento.

```
for (i=200; i> 0;i--)
{
// instrucciones a ejecutar 200 veces
}
```

***Figura 3**. Estructura **for**. Arriba (abajo) el contador se incrementa (decrementa).*

**Notas:** i++ es lo mismo que i = i + 1 ; i-- es lo mismo que i = i - 1.

Esta estructura es un poco más general, pero para el curso lo aquí indicado es suficiente.

-

# 3. Ejercicios

1. A partir del código **`serial_microbit.ino`**.
   a. Estudiar el código.
   b. Cargar el código en la placa Microbit, activar el “Monitor Serie” y probar el código.
   c. ¿Puede explicar qué es lo que se muestra en el monitor serie al oprimir una tecla? Comparar con una tabla ASCCI.
   d. ¿Cuál es el código ASCII de la letra ‘Y’? ¿el de la letra ‘y’? Verifique esto usando el monitor
   e. Modifique el código cambiando **int dato** por **char dato** ¿Qué cambios observa en el

monitor? Intente explicar el motivo.

f. A partir de la pata armada y calibrada, conecte el servo del muslo en un Pin a elección. Modifique el código de forma de cumplir lo siguiente:
   - Inicialmente el LED debe estar prendido y el muslo posicionado en 90°
   - Si desde el Monitor Serie se recibe la letra "b" el LED debe encenderse y el muslo posicionarse en 10°
   - Si desde el Monitor Serie se recibe una letra distinta a la letra "b" el LED debe apagarse y el servo posicionarse en 135°.
   - Mientras no haya recepción desde el Monitor Serie, el LED y el servo deben quedar incambiados.

2. En este ejercicio se busca averiguar por ensayo y error cuánto tiempo demora un servo en recorrer un ángulo dado. Se utiliza para esto el sketch **`tiempo_servo.ino`** (disponible en EVA).
   a. Estudiar el código
   b. A partir de la pata armada, conectar el servo de la pierna, cargar el código en la placa Microbit y probar distintos valores de "n".
   c. Buscar por ensayo y error el valor óptimo de "d" (tiempo que requieren los servos en hacer el recorrido).
   d. Modificar el ángulo recorrido y proporcionalmente el valor "d", ¿que observa?
   e. Sustituir en el código la estructura for por la estructura while de forma que el comportamiento no cambie.

3. Estudiar y probar el código **`switch_case_microbit.ino`**.
   a. Estudiar el código.
   b. A partir de la pata armada y calibrada, conectar el servo de la pierna, cargar el código en la placa Microbit, activar el "Monitor Serie" y probar el código. Comparar lo que observa con el propio código.
   c. Si se envían 2 comandos juntos (es decir 2 caracteres juntos ej: "ab"), ¿se ejecutan ambos?, ¿se ejecuta 1 solo?, ¿cuál? Intente explicar el motivo de lo que observa.
   d. Conectar ambos servos, de la pierna y del muslo, y modificar el código de forma de realizar lo que se indica en la tabla. Primero se debe mover la pierna, 500 ms después el muslo y finalmente esperar 300 ms antes de quedar esperando la recepción del siguiente comando.

| Comando | Servo pierna | Servo muslo |
|---|---|---|
| 'a' | 10° | 90° |
| 'b' | 60° | 10° |
| 'c' | 120° | 60° |
| 'm' | 45° | 135° |
| en otro caso | 170° | 170° |

Cuadro 1: Correspondencia comando-servo-grados

4. Se va a comenzar a comprender el movimiento de "caminar", moviendo los servos uno a uno. Más adelante veremos que hay movimientos que deben ser simultáneos.
   Como criterio considerar que se trabaja sobre la pata delantera derecha del bicho.
   La pata debe estar calibrada.

a. Escribir un sketch que inicialice la pata de forma que el muslo quede alineado con la cadera y la pierna a 80° del muslo hacia abajo. ¿En cuantos grados se debe inicializar la pierna para que forme 80° con la horizontal?.

b. Cargar el sketch en la placa y probar.

c. Hacer que la pierna "camine" hacia adelante o imite algo similar a un braceado de nado. Para esto, una vez iniciada la pierna, realizar lo siguiente:

   - Levantar pierna.
   - Esperar 0,5 segundos.
   - Mover muslo hacia adelante.
   - Esperar 0,5 segundos.
   - Bajar la pierna a posición inicial.
   - Esperar 0,5 segundos.
   - Mover muslo hacia atrás, a posición inicial.
   - Esperar 0,5 segundos.
   - Repetir esta secuencia en forma indefinida.

   Los valores de los ángulos a mover los servos son de elección libre.

   Observar intuitivamente estos movimientos en los brazos de Uds. para ir comprendiendo la analogía "bicho" vs ser vivo.

d. Modificar el código de forma que utilice el Monitor Serie para recibir "comandos" (letras), utilice la estructura **`switch case`** y cumpla lo siguiente:

| Comando | Comportamiento de la pata |
|---|---|
| 'a' | "Camine" hacia adelante |
| 'b' | "Camine" hacia atrás |
| 'c' | Permanezca en la posición inicial |
| en otro caso | Quede completamente estirada |

Cuadro 2: Lista de comandos

**Observaciones:**

- Observar que caminar hacia atrás es igual a la secuencia de ir hacia adelante, pero en sentido inverso.
- La estructura **`switch case`** no debe estar dentro del if que pregunta si hay un dato disponible en el Monitor Serie. Con esto el **`switch case`** se ejecuta siempre independientemente de haber o no un dato nuevo recibido del Monitor Serie.

-

5. En este ejercicio se busca armar 2 patas en forma calibrada (referirse al procedimiento dado en el obligatorio 2):

   a. Armar 2 patas y unirlas con la cadera. Recordar que primero se debe llevar cada servo a 90° y luego armar ambas patas en posición completamente estirada, tratando de que los piñones no se muevan.

   b. Determinar el ángulo de error que debe agregarse al ángulo inicial para que la posición real de cada servo sea la correcta. Referirse al obligatorio 2, ejercicio 5. Estos valores obtenidos deberán utilizarse **siempre** para lograr la posición correcta al realizar los movimientos. Cada vez que se desarme y arme una pata, debe realizarse este procedimiento.