

Aprende a Programar con Arduino

Omar Castillo
Sheyla N. Barreda

Lema del texto

“Es imposible aprender a programar limitándose a leer apuntes o a seguir pasivamente una explicación en clase”.

Componentes Básicos de un Programa

Todo programa sin importar el tamaño o funcionalidad del mismo está compuesto de las siguientes partes.

- **Entrada:** Recibir datos del teclado, o de un archivo o de otro dispositivo.
- **Salida:** Mostrar datos en el monitor o enviar datos a un archivo a otro dispositivo.
- **Matemáticas:** Ejecutar operaciones básicas, como la adición y multiplicación.
- **Operación Condicional:** Probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones determinada.
- **Repetición:** Ejecutar alguna acción repetidas veces, usualmente con alguna variación.

Aunque sea difícil de creer, *todos los programas* existentes *están formados por estas instrucciones*, básicamente se rompe una tarea en varias más pequeñas hasta que puedan ser ejecutadas por una instrucción básica.

Algoritmos:

Un algoritmo, es sencillamente **una secuencia de pasos orientada a la consecución de un objetivo.**

Un algoritmo puede expresarse en cualquier lenguaje de programación. Sin embargo no es adecuado ya que.

- Los programadores no conocen TODOS los lenguajes de programación.
- Los lenguajes de programación presentan características únicas que pueden interferir en la expresión clara de la solución a un problema.

Algoritmos usando lenguaje natural (Promedio de tres números)

1. Dame el primer número.
2. Dame el segundo número.

3. Dame el tercer número.
4. A la suma de los tres números divídelo entre 3.
5. Mostrar el resultado.

*Todo Algoritmo **debe tener un final**, es decir debe **acabar** tras la ejecución de un número finito de pasos.¹*

Características de un algoritmo

Los algoritmos deben tener las siguientes características para estar correctamente definido.

1. *Datos de entrada.*
2. *Datos de salida.*
3. *Definido con exactitud.*
4. *Es finito.*
5. *Es efectivo. (realizable).*
6. *Es eficiente. (óptimo).*

Otra forma de Representar algoritmos

Hay varias formas de representar algoritmos, las principales son.

- Diagramas de Flujo.
- Pseudocódigo.

¿Que es un Arduino?

Es un *circuito electronico de desarrollo* para computación física, también es un *lenguaje de programación*, para más información puedes visitar la pagina web www.arduino.cc

Estructura de un programa en Arduino

Todo programa en arduino tiene la siguiente estructura básica.

```
void setup() {  
  // coloca tu código setup aquí, se ejecutará una sola vez  
}  
  
void loop() {  
  // coloca tu código principal aquí, se ejecutará continuamente
```

¹ no como One Piece que nunca va acabar ^_^

}

Bueno, ahora a escribir código, (como dice el maestro Miyagi a encerar, pulir).

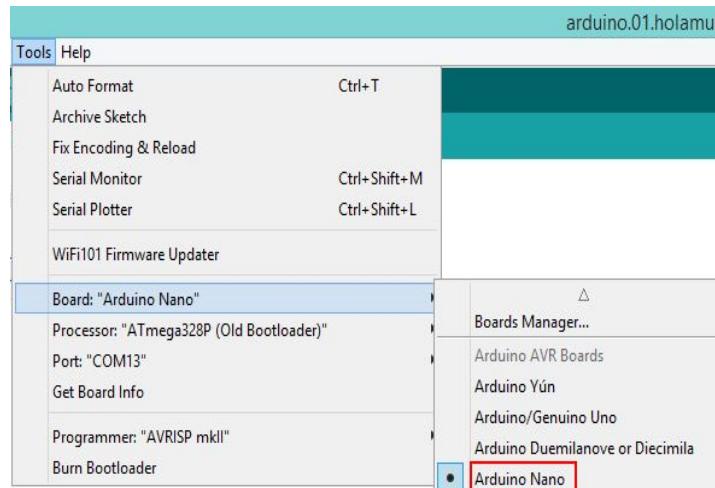
1. Mostrar “Hola Mundo” en la pantalla usando Arduino

El primer programa a escribir en un lenguaje siempre es el conocido “hola mundo”, en el caso de Arduino podemos tener dos hola mundo, un hola mundo mediante texto en forma de comunicación serial con la PC y otro hola mundo en forma de un led que se enciende y se apaga, hagamos los dos.

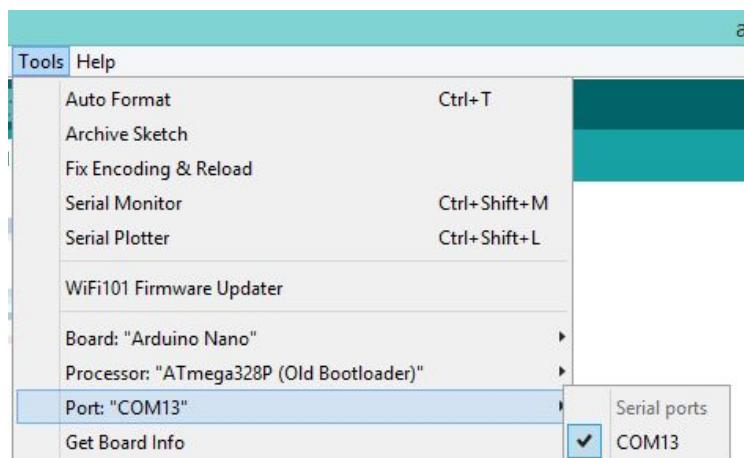
```
void setup() {
Serial.begin(9600);
}

void loop() {
Serial.println("Hola Mundo");
}
```

Para subir el programa debemos configurar el tipo de *Arduino* en nuestro *Arduino IDE* y seleccionar el puerto *COM* respectivo así como el tipo de tarjeta *Arduino*.



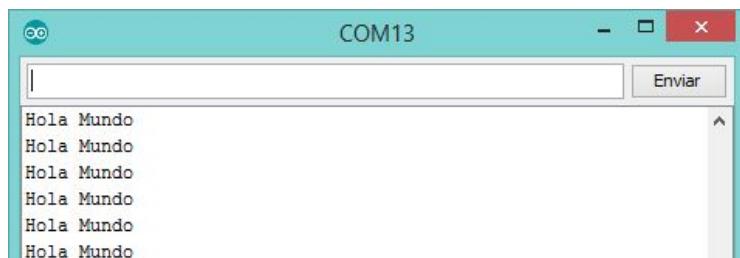
Si tu *Arduino NANO* es algo antiguo, sea original o clon, es probable que el *bootloader* que tenga no esté actualizado, en ese caso debemos escoger un *bootloader antiguo* (old bootloader) al momento de compilar en el *Arduino IDE*, si marca error al momento de compilar tus programas es probable que sea por un *bootloader antiguo*.



Ahora si abrimos el *Monitor serie* (Herramientas -- Monitor Serie), tendremos la salida del programa.



La salida del programa sería la siguiente:



Debemos tener cuidado que la *velocidad en baudios* del programa sea igual a la *velocidad del Monitor serie*, para este texto usaremos siempre la velocidad *9600 bps* (baudios por segundo).

Podemos observar que el mensaje “*Hola Mundo*” se repite una y otra vez ya que se encuentra en la función *loop()* del programa y por lo tanto siempre se repetirá.

La función *setup()* solo se ejecuta una sola vez, para poder entender como la función *setup()* solo se ejecuta una sola vez vamos a modificar nuestro programa actual de hola mundo para que el mensaje *solo se ejecute una sola vez*.

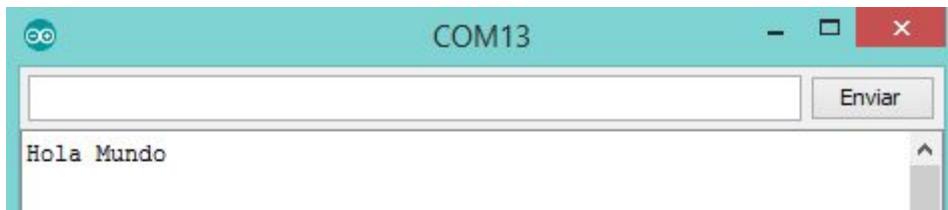
2. Mostrar “Hola Mundo” en la pantalla usando Arduino y la función setup.

Para este caso vamos a modificar nuestro anterior programa y vamos a hacer que el mensaje ya no esté en la función *loop()* sino en la función *setup()*, el código quedaría como sigue.

```
void setup() {
  Serial.begin(9600);
  Serial.println("Hola Mundo");
}

void loop() {
```

Guardar el programa, y *subirlo* al *Arduino* con el **botón** (→) o con ***CTRL+U***, y después abrir el *Monitor serie*, tendremos algo similar a esto.



Punto!!!!, ahora nos damos cuenta que el código para mostrar el mensaje solo se ejecutó una sola vez, siempre tomar nota de esta diferencia ya que nos servirá para hacer cosas interesantes más adelante.

Resumen

```
Las instrucciones en la función setup() solo se ejecutan una vez.  
Las instrucciones en la función loop() se ejecutan (y se repiten)  
siempre
```

El objetivo de los anteriores programas era mostrarles un componente básico de los programas, la *salida*, en este caso mensajes por el puerto serial, ahora vamos a ver otro tipo de *salida*, un led el cual vamos a encender un tiempo y apagar por otro tiempo.

3. Mostrar “Hola Mundo” usando el led incorporado en el Arduino.

Para este programa vamos a usar el led incorporado en todos los Arduinos, y que está

conectado a la salida de el pin digital 13, recordemos que un led es solo un diodo que emite luz, además tomar nota que seguimos hablando de *salida*, pero ahora vamos a introducir un nuevo concepto, el *retardo* o *delay*, el cual permite que el Arduino no haga nada en un determinado tiempo, es como decir *espera un rato no hagas nada*. El código sería el siguiente.

```
void setup() {           //esta función solo se ejecuta una sola vez
  pinMode(13, OUTPUT); //configura PIN 13 como salida
}

void loop() {           //esta función siempre se repite
  digitalWrite(13, HIGH); //salida digital 13 encendido
  delay(100);           //retardo de 100 ms
  digitalWrite(13, LOW); //salida digital 13 apagado
  delay(100);           //retardo de 100 ms
}
```

Vayamos por partes, en este último programa aparecen nuevas instrucciones y funciones, por ejemplo.

pinMode(13,OUTPUT), esta función configura el pin 13 como *salida*, *OUTPUT* significa *salida* en inglés.

digitalWrite(13,HIGH), esta función recibe dos argumentos, el primero es el pin en el que va hacer la salida digital y el otro es el tipo de valor lógico que va tener en este caso *HIGH* (alto en inglés), que quiere decir **enciende**.

digitalWrite(13,LOW), esta función es similar pero en vez de encender, la palabra *LOW* (bajo en inglés), quiere decir **apaga** la salida.

delay(100), quiere decir que no hagas nada por 100 milisegundos, *delay* significa *retraso* en inglés, puedes cambiar este valor y experimentar con distintos valores para ver como cambia la luz del led.

Hasta este momento hemos hablado de *salidas*, de configuración y de retardos, ahora es tiempo de hablar de *entradas* y de *operaciones matemáticas*, que son las otras partes componentes de un programa.

4. Calcular la suma de dos números y mostrarla usando el monitor serial.

Para este ejercicio vamos a hacer un programa que sume los números *a* y *b* con valores 3 y

8, luego mostrará el resultado de la *operación matemática* por el *Monitor serie* del *Arduino IDE*, con esto vamos a introducir el concepto de *variable* y *tipo de dato*, el código es el siguiente.

```

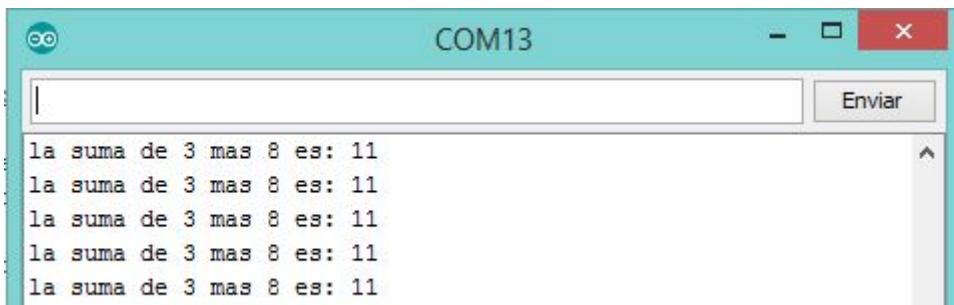
int a = 3;          //se declara la variable a de tipo entero y con valor 3
int b = 8;          //se declara la variable b de tipo entero y con valor 8
int suma = 0;        //se declara la variable suma de tipo entero y con
                     //valor 0, esto solo para reservar espacio de memoria

void setup() {         //esta función solo se ejecuta una sola vez
    Serial.begin(9600); //configura puerto serial a 9600 de velocidad
}

void loop() {           //esta función siempre se repite
    suma = a + b;       //la variable suma se define como a+b
    Serial.print("la suma de "); //se muestra el texto la suma de
    Serial.print(a);      //se imprime la variable a (que vale 3)
    Serial.print(" mas "); //se muestra el texto mas
    Serial.print(b);      //se imprime la variable b (que vale 8)
    Serial.print(" es: "); //se muestra el texto es:
    Serial.println(suma); //se imprime la suma y se crea otra línea
                         //con println
}

```

Después de subir el programa al arduino y abrir el monitor serial se tiene lo siguiente.



Podemos ver que la función *loop()* muestra linea tras linea el resultado de la suma de $3+8$ que eran las variables *a* y *b*.

Recordar siempre la diferencia entre *Serial.print()* y *Serial.println()*, el primero imprime todo seguido y el segundo imprime una línea nueva al final (también conocido como retorno de carro), podemos hacer uso del manual de referencia incluido en el *Arduino IDE*, siempre nos será de ayuda (para abrirlo *Herramientas -- Referencia*).

Hasta el momento ya vimos *salidas* y *operaciones matemáticas*, en el siguiente ejercicio vamos a continuar con la *operaciones matemáticas*, recordando siempre que todos son *componentes de un programa*.

5. Un programa que calcule el área de un cuadrado

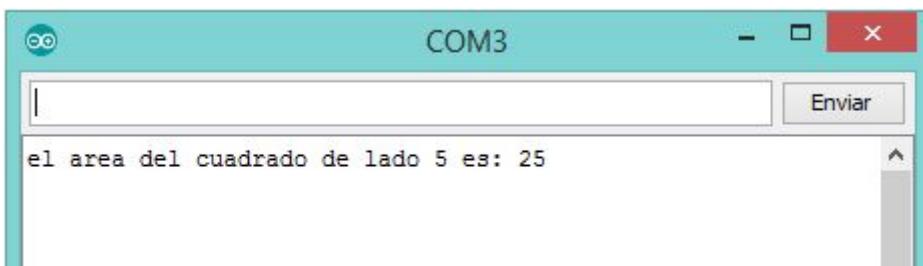
Hasta el momento se hizo programas con *salidas* y *operaciones matemáticas* ahora vamos a hacer un ejemplo de *cálculo matemático*, para este ejemplo tenemos la función *pow()*, que ejecuta la operación de potencia.

```
int lado = 5;           //se declara la variable lado tipo int y valor 5
int area = 0;           //se declara la variable área tipo entero y valor 0

void setup() {          //esta función solo se ejecuta una sola vez
    Serial.begin(9600);   //configura puerto serial a 9600 de velocidad
    area = pow(lado, 2);  //área es lado al cuadrado, con función pow()
    Serial.print("el área del cuadrado de lado ");
    Serial.print(lado);    //se imprime la variable lado
    Serial.print(" es: ");   //
    Serial.println(area);   //se imprime la variable área
}

void loop() {            //esta función siempre se repite, no la usamos
}
```

Ahora si subimos el programa al Arduino y abrimos el Monitor Serial tendremos lo siguiente.



Como vemos se puede hacer *cálculos matemáticos* con el Arduino y mostrarlos con una *salida* por el puerto serie, pero hasta el momento para introducir el lado del cuadrado lo hemos declarado en el mismo *código fuente* del programa, para cambiar el lado debemos compilar el código de nuevo a cada rato, lo mejor sería que podamos introducir como una

entrada el lado del cuadrado por teclado,y eso es lo que veremos a continuación.

6. El área de un cuadrado con entrada de lado por teclado

En este ejercicio vamos a calcular el área un un cuadrado con una *operación matemática* pero la *entrada* del valor del lado por nuestro querido *monitor serial*, el código es el siguiente.

```

int lado = 0;      //se declara la variable lado tipo entero y valor 0
int area = 0;      //se declara la variable área tipo entero y valor 0

void setup() {          //esta función se ejecuta una sola vez

    Serial.begin(9600);  //configura puerto serial a 9600 de velocidad
    Serial.println("Calculando el Area de un Cuadrado");
    Serial.println("*****");
    Serial.println("");
    Serial.println("Dame el valor del lado: ");
}

void loop() {           //esta función siempre se repite

    if (Serial.available() > 0) { //si hay dato en puerto serial entrar a
        //esta condicional Si---Entonces
        lado = Serial.parseInt(); //se aplica parseInt al puerto serial
        //se guarda dato en la variable "lado"
        Serial.print("lado = ");   //mostrar el valor
        Serial.println(lado);     //de la variable "lado"
        area = pow(lado, 2);     // potencia de 2 es función pow(lado,2)
        Serial.print("el area del cuadrado de lado ");
        Serial.print(lado);       //se imprime la variable lado
        Serial.print(" es: ");    //
        Serial.println(area);    //se imprime la variable área
        Serial.println("");      //línea en blanco
        Serial.println("Dame el valor del lado: "); //muestra dame lado
        //repite loop() esperando en puerto serial.
    }
}

```

Como vemos aparece un nuevo concepto que es la *condicional*, y se encuentra en la referencia como *Control de flujo*, evaluando una *condición* puedes controlar el comportamiento de un programa, vamos a explicarlo ahora.

```
if (algunaVariable > 50) {
    // hacer algo aca
}
```

como vemos *algunaVariable > 50*, puede tener dos valores.

- *algunaVariable* es menor que 50, lo que daría como resultado *falso*.
- *algunaVariable* es mayor que 50, lo que daría *verdadero*.

Si se cumple que la condición es *verdadera* se ejecuta el código entre las llaves {}.

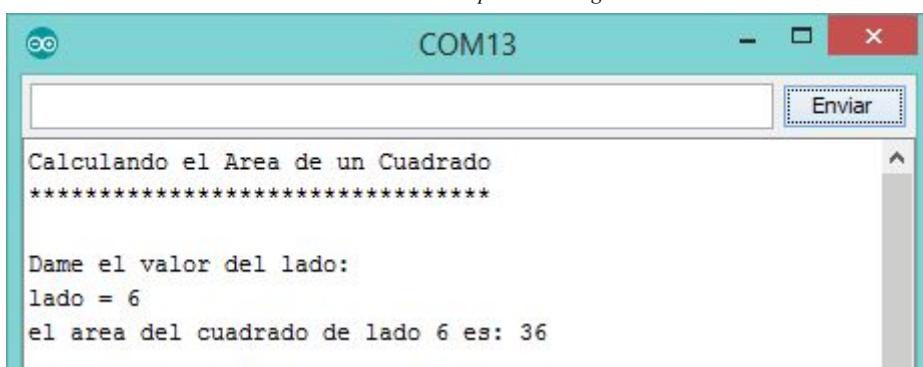
Ahora volvamos a nuestro programa, ¿Dónde está la condicional?

```
if (Serial.available() > 0) { //si hay dato en puerto serial entrar
}
```

Realmente la función *Serial.available()* devuelve el número de bytes presentes en el buffer del puerto serial del *Arduino*. El *monitor serie* siempre está leyendo el puerto, así que siempre está presente el valor *cero* (no hay bytes en el buffer), por eso cuando lees el puerto serial con alguna función de lectura como puede ser *Serial.read()*, se tiene siempre el valor *0* (cero).

Ahora cuando enviamos un dato al puerto serie, obviamente el valor del buffer cambia y ya no es cero, de esta forma se activa la condición y se ejecuta el código encerrado entre las llaves {}.

Si ejecutamos nuestro programa y enviamos el dato del *lado*, vamos a tener algo como esto.



Como el programa de *entrada* y *operaciones matemáticas* está en el bucle *loop()* este se va repetir indefinidamente, siempre pedirá lado y siempre te devolverá el área del cuadrado.

Y ahora nuevamente nuestro mantra (*ohmmm*). Porque repetir es volver a vivir (total siempre se revive en base).

- **Entrada**: recibir datos del teclado, o de un archivo o de otro aparato.
- **Salida**: mostrar datos en el monitor o enviar datos a un archivo a otro aparato.
- **Matemáticas**: ejecutar operaciones básicas, como la adición y multiplicación.
- **Operación Condicional**: probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.
- **Repetición**: ejecutar alguna acción repetidas veces, usualmente con alguna variación.

7. Hola amigo humano.

Hasta el momento hemos visto *entradas*, *salidas* y *operaciones matemáticas*, vamos bien, ahora vamos a hacer un programa que reciba una *cadena de texto* por teclado (nuestro nombre) y nos haga un saludo, el típico *hola mundo* pero con texto, es decir *Hola Omar!*. El código sería el siguiente.

```
String nombre = "alguien"; //variable "nombre", tipo String = "alguien"

void setup() { //esta función se ejecuta una sola vez
```

```

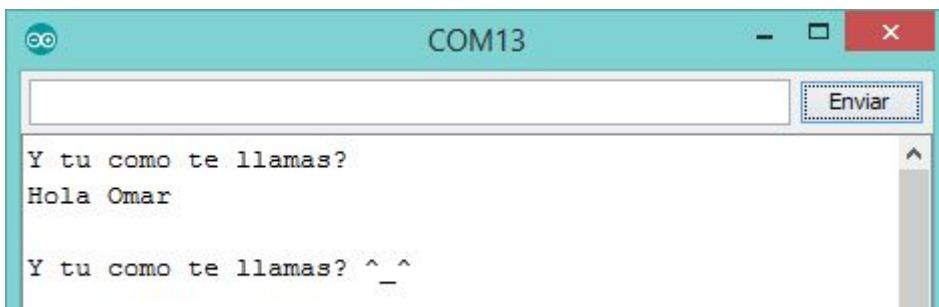
Serial.begin(9600);      //configura puerto serial a 9600 de velocidad
Serial.println("Y tu como te llamas? ");
}

void loop() {           //esta función siempre se repite

    if (Serial.available() > 0) { //si hay dato en el serial se cumple la
                                //condición
        nombre = Serial.readString(); //readString() al puerto serial y
                                //guarda dato en variable "nombre"
        Serial.print("Hola ");       //mostrar el texto hola y seguido
        Serial.println(nombre);     //el texto de la variable "nombre"
        Serial.println("");
        Serial.println("Y tu como te llamas? "); //repite saludo y espera
                                                //algún dato en puerto serial
    }
}

```

Si lo subimos al *Arduino*, tenemos lo siguiente en el *Monitor serie*, le damos nuestro nombre, en este caso el mio, Omar y voila..



Vamos a continuar con *operaciones matemáticas* a continuación veremos cómo calcular el volumen de una esfera, y le vamos a dar el radio por teclado.

8. Cálculo del volumen de una esfera con entrada de datos por teclado.

En este ejercicio vamos a continuar con las *entradas, salidas y operaciones matemáticas*, si es más de lo mismo, pero la práctica afianza lo aprendido, tu dale nomas, la fórmula del volumen de la esfera es ($ve=4/3*pi*radio^3$)

```

int radio = 0;           //variable radio tipo entero valor 0
float volumen = 0;       //variable volumen tipo float (con

```

```

                //decimales) y valor 0
void setup() {           //esta funcion solo se ejecuta una sola vez
Serial.begin(9600);      //configura el puerto serial a 9600 bps
Serial.println("Calculo del Volumen de una esfera");
Serial.println("");
Serial.println("Dame el radio de la esfera");
}

void loop() {             //esta funcion siempre se repite

if (Serial.available() > 0) {           //si hay dato en serial
                                         //condicion cambia
    radio = Serial.parseInt();          //parseInt() al serial y
                                         //se guarda el dato en la variable "radio"
    volumen = 4/3*3.14159264*pow(radio, 3); //se aplica la
                                         //formula (ve=4/3*pi*radio^3) y se asigna a volumen
    Serial.print("el volumen de la esfera de radio ");
    Serial.print(radio);
    Serial.print(" es = ");
    Serial.println(volumen);            //mostrar el volumen
    Serial.println("");
    Serial.println("Dame el radio de la esfera");
}
}
}

```

Similar al ejercicio anterior, si hay algún dato en el puerto serial, guardarlo en la variable *radio*, luego operar la fórmula y mostrarla en el puerto serial, después repetir todo por la función *loop()*.

Hay que notar que aparece un nuevo tipo de dato el tipo *float*, el cual es similar al tipo entero *int* pero tiene la posibilidad de manejar decimales, más adelante explicaremos que *elegir el tipo adecuado de dato influye significamente en el desempeño de tus programas*.

Hasta ahora nos hemos divertido con programas sencillos y con entradas por teclado de un solo dato ya que es relativamente sencillo, *solo es un dato*. Ahora veremos cómo introducir más de un dato por el puerto serie, esto lo vamos a hacer controlando el flujo del programa, vamos a establecer una *condición* inicial, luego el programa ejecutará una acción gracias a esa *condición* y luego al finalizar esta acción cambiará la *condición* para que se ejecute otra sección de código y así sucesivamente, ahora acompaña a ver esta triste historia.

9. Suma de dos números introducidos por teclado

```

int a = 0; //se declara variable a de tipo entero y con valor 0
int b = 0; //se declara variable b de tipo entero y con valor 0
int suma = 0; //se declara variable suma tipo entero con valor 0
int condicion = 0; //condicion controla el flujo del programa

void setup() { //esta funcion solo se ejecuta una sola vez
    Serial.begin(9600); //configura puerto serial 9600 de velocidad
}

void loop() {
    Serial.println("Sumando dos numeros");
    Serial.println("dame el valor de a: ");

    while (condicion == 0) { //si se cumple condicion=0 entrar
        if (Serial.available() > 0) { //si hay dato en serial entrar
            a = Serial.parseInt(); //el valor de a es "parse" entero
            Serial.print("a = "); //mostrar el valor
            Serial.println(a); //de la variable "a"
            Serial.println("dame el valor de b: ");
            condicion = 1; //cambia la condicion por 1 para salir
        }
    }

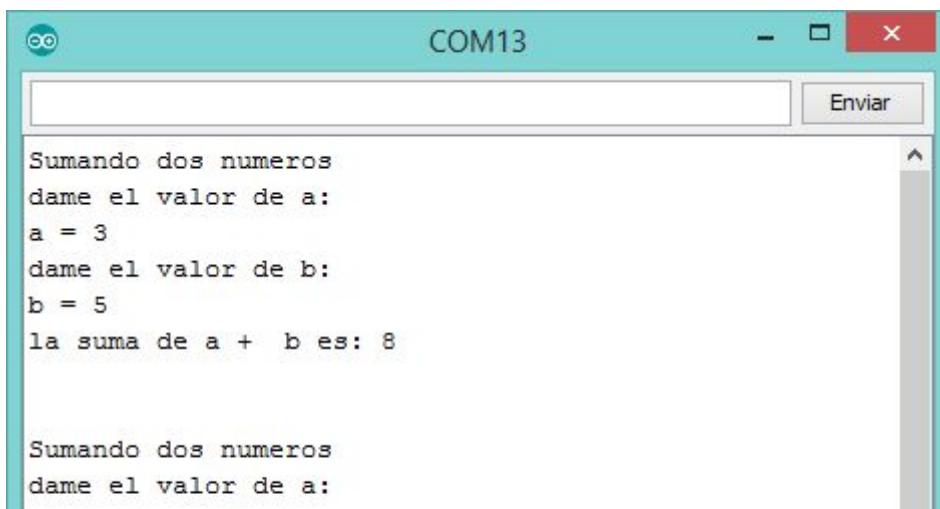
    while (condicion == 1) { //si condicion vale 1 entra a while
        if (Serial.available() > 0) { //si hay dato en serial entra
            b = Serial.parseInt(); //se entra valor de b tipo int
            Serial.print("b = "); //muestra el valor
            Serial.println(b); //de la variable "b"
            condicion = 2; //cambia condicion a 2 para salir
        }
    }

    //salimos de los bucles while ahora vamos a calcular la suma
    suma = a + b; //se suma a+b y se guarda en la variable suma
    Serial.print("la suma de a + b es: ");
    Serial.println(suma); //se muestra la variable suma
    Serial.println();
    Serial.println();
    condicion = 0; //se cambia la condicion a 0 de esa manera se
}

```

```
//regresa al bucle while que pide valor de a
}
```

Después de subir el programa al *Arduino* y de abrir el *Monitor serie*, le enviaremos los valores 3 y 8 y tenemos lo siguiente.



Como vemos, podemos ingresar más de un valor al *Monitor serie*, lo que hacemos es crear una condición, *entrada* de un valor y ahora a otra *condición, entrada* de otro valor y así sucesivamente, este truco nos puede servir para hacer un programa configurable por puerto serial (que grabe la memoria EEPROM por ejemplo), aplicaciones como pueden ser un robot o quizás un controlador de tiempo etc, las posibilidades son varias como veremos más adelante. Ahora continuemos con las *operaciones matemáticas*.

10. Recibe altura y base de un triángulo y muestra el área del mismo.

Para este ejercicio vamos a usar nuestro programa para entrada de datos y vamos a añadir una rutina de cálculo de área del triángulo, la fórmula del área del triángulo es sencilla. Veamos el código fuente.

```
//Calculo del area de un triangulo
int base = 0;           //se declara variable base entero y con valor 0
int altura = 0;          //se declara variable altura entero y valor 0
float area = 0;          //se declara variable suma tipo float valor 0
int condicion = 0;        //condicion que controla flujo del programa

void setup() {            //esta funcion solo se ejecuta una sola vez
```

```

Serial.begin(9600);           //configura serial a 9600 de velocidad
}

void loop() {
    Serial.println("Hallar el area de un triangulo");
    Serial.println("dame la base del triangulo: ");

    while (condicion == 0) {    //si cumple condicion de cero entrar
        if (Serial.available() > 0) { //si hay dato en serial entrar
            base = Serial.parseInt(); //valor de base es "parse" entero
            Serial.print("base = ");           //mostrar el valor
            Serial.println(base);           //de la variable "a"
            Serial.println("dame la altura del triangulo: ");
            condicion = 1;               //cambia la condicion por 1 para salir
        }
    }

    while (condicion == 1) {    //ahora la condicion vale 1 se entra
        if (Serial.available() > 0) { //si hay dato en serial entra
            altura = Serial.parseInt(); //introduce el valor de altura
            Serial.print("altura = ");           //muestra el valor
            Serial.println(altura);           //de la variable "altura"
            condicion = 2;               //cambia condicion por 2 para salir
        }
    }

    //ya salimos de los bucles while ahora a calcular el area
    area = float(base) * float(altura) / 2;      //base*altura/2 en area
    Serial.print("el area del triangulo es: ");
    Serial.println(area);           //se muestra la variable area
    Serial.println();
    Serial.println();
    condicion = 0;           //se cambia la condicion a 0 de esa manera se
                            //regresa al bucle while donde se pide base
}

```

Subimos el programa al arduino y se tiene lo siguiente en el monitor serial.

```
Hallar el area de un triangulo
dame la base del triangulo:
base = 2
dame la altura del triangulo:
altura = 3
el area del triagulo es: 3.00
```

El programa funciona bien, pero hagamos la prueba con una base y una altura con valor 1, ¿que se tiene?, ¿Cual es el error?, trata de corregirlo (la respuesta es el tipo de dato *int* en la base y altura). No te detengas, en el siguiente programa convertiremos grados sexagesimales a radianes.

11. Convertir grados sexagesimales a radianes

La equivalencia de grados está definido como $2*pi \text{ radianes} = 360^\circ$, para convertir grados debemos multiplicar el dato ingresado por una razón de relación para obtener radianes, el código sería el siguiente.

```
//De sexagesimales a radianes
int angulo = 0; //variable angulo tipo entero de valor 0
float radianes = 0; //variable float (decimales) y valor 0
void setup() { //esta funcion se ejecuta una sola vez
    Serial.begin(9600); //configura el serial a 9600 velocidad
    Serial.println("Convertir de Sexagesimales a Radianes");
    Serial.println("");
    Serial.println("Dame el angulo en sexagesimales");
}

void loop() { //esta funcion siempre se repite

    if (Serial.available() > 0) { //si hay dato en serial entrar
        angulo = Serial.parseInt(); //parseInt() al serial se
                                    //guarda variable "angulo"
        radianes = (2 * 3.14159264 / 360) * angulo; //conversion
                                                    //(2*3.14159264/360)
        Serial.print("el angulo ");
        Serial.print(angulo);
}
```

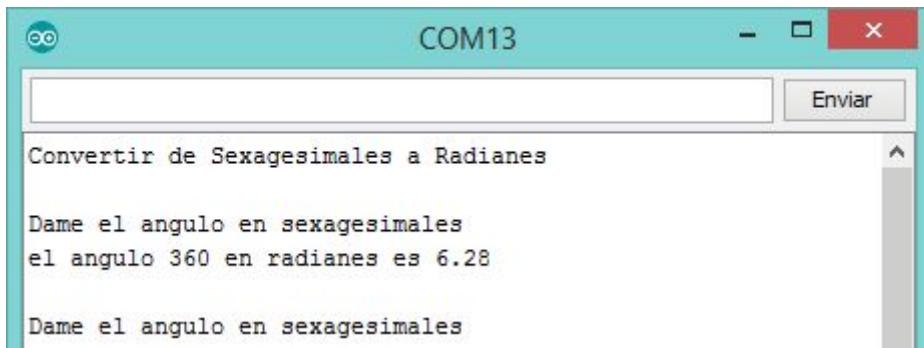
```

Serial.print(" en radianes es ");
Serial.println(radianes);      //mostrar el angulo en radianes
Serial.println("");
Serial.println("Dame el angulo en sexagesimales");
}

}

```

Subimos el programa al *Arduino* y se tiene en el *Monitor serie* lo siguiente.



12. Calcular la corriente si se tiene voltaje y resistencia

Para este ejercicio usaremos la ley de ohm $V=I*R$ así que la *entrada* serán los valores de voltaje y de resistencia luego haremos una *operación matemática* y tendremos a la *salida* el valor de la corriente, el código es el siguiente.

```

//Calculo de la resistencia
float voltaje = 0;          //se declara voltaje tipo float y con valor 0
float corriente = 0;         //se declara corriente tipo float y con valor 0
float resistencia = 0;       //se declara resistencia tipo float con valor 0
int condicion = 0;           //se declara condicion, controla flujo del programa

void setup() {                //esta funcion solo se ejecuta una sola vez
  Serial.begin(9600);        //configura puerto serial a 9600 de velocidad
}

void loop() {
  Serial.println("Calcular la corriente ");
  Serial.println("dame el voltaje ");

```

```

while (condicion == 0) { //si se cumple condicion que es cero entrar
    if (Serial.available() > 0) { //si hay dato en serial entrar
        voltaje = Serial.parseFloat(); //valor de voltaje en float
        Serial.print("voltaje = ");
        Serial.println(voltaje); //mostrar el valor de
        Serial.println("dame la resistencia:");
        condicion = 1; //cambia condicion por 1 para salir del bucle
    }
}

while (condicion == 1) { //ahora la condicion vale 1 se entra a este
    if (Serial.available() > 0) { //verificamos si hay dato en el
        puerto
            //serial, si hay se entra
        resistencia = Serial.parseFloat(); //entra resistencia tipo float
            //gracias a parseFloat()
        Serial.print("resistencia = ");
            //muestra el valor
        Serial.println(resistencia); //de la variable
        "resistencia"
        condicion = 2; //despues de dar el valor de "resistencia"
    cambia
            //condicion por 2 para salir
    }
}
// ya salimos de los bucles while ahora vamos a calcular la
corriente.
corriente = voltaje / resistencia; //se calcula corriente y se guarda
Serial.print("la corriente es: ");
Serial.println(corriente); //se muestra la variable
corriente
Serial.println();
Serial.println();
condicion = 0; //cambia condicion a 0 y regresa al bucle while
inicial
}

```

ESTRUCTURAS DE CONTROL

Hasta ahora nuestros “programas” en nuestro Arduino han sido del tipo:

1. Se ingresan datos
2. Se hacen cálculos
3. Se imprimen resultados.

No se tiene control sobre el flujo del programa, es decir hasta ahora el flujo de ejecución es SECUENCIAL (uno tras de otro).

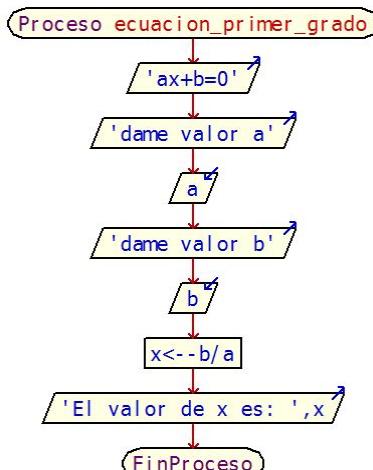
Ahora veremos un nuevo concepto *el control de flujo* mediante **estructuras de control de flujo** o abreviado **estructuras de control**.

SENTENCIAS CONDICIONALES

A escribir código, uno dos.. uno dos...

13. Resolver una ecuación de primer grado.

Una ecuación de primer grado tiene la forma $ax+b=0$ para solucionarla podríamos guiarnos por el siguiente diagrama de flujo.



Hagamos el programa para nuestro *Arduino*, el código es el siguiente.

```

//Ecuacion de primer grado ax+b=0 ... x=-b/a
float a = 10; //se declara variable a tipo float y con valor 5
float b = 2; //se declara variable b tipo float y con valor 6
float x = 0; //se declara variable x tipo float con valor 0
void setup() { //esta funcion solo se ejecuta una sola vez
  
```

```

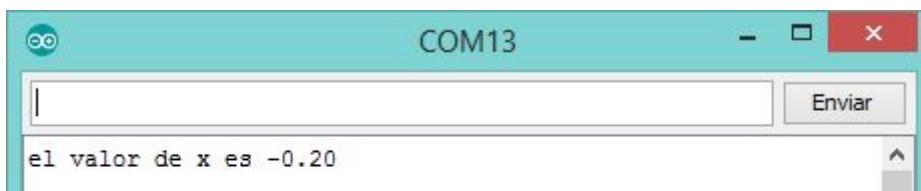
Serial.begin(9600); //configura el puerto serial a 9600 de
velocidad
x = -1 * b / a; //se halla el valor de x
Serial.print("el valor de x es ");
Serial.println(x); //se imprime la variable x
}

void loop() { //esta funcion siempre se repite, ahora no la
usamos

}

```

Como podemos ver no tiene *entrada* de datos, estos están directamente declarados al inicio del código fuente, el valor de *a* es 10 y de *b* es 2, por lo tanto después de subirlo al *Arduino* y de abrir el *Monitor serie* tendremos.



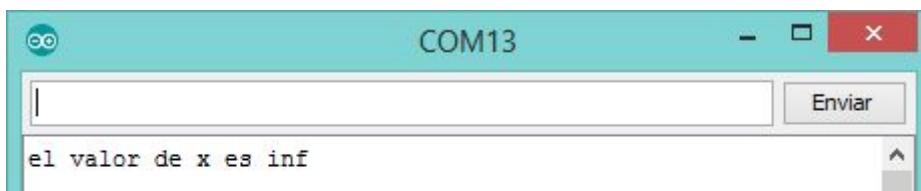
Pero si ahora cambiamos el valor de *a* por 0 y *b* por 5 como a continuación.

```

float a = 0; //se declara variable a tipo float y con valor 0
float b = 5; //se declara variable b tipo float y con valor 5

```

Después de subirlo al *Arduino* y de activar el *monitor serial* se tendría la siguiente *salida*



La *salida* es *inf*, ¿pero qué es eso?, es una *salida* que se obtiene cuando se hace una división entre cero, lo cual es indeterminado.

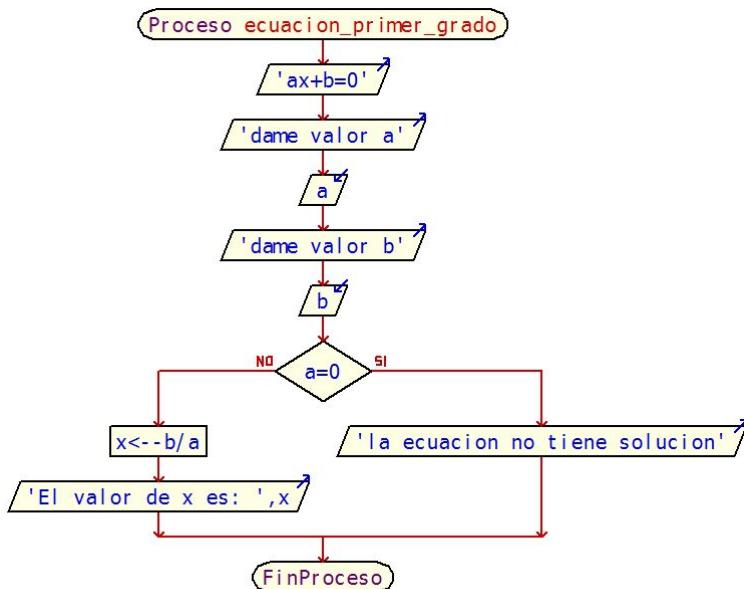
¿Cómo podemos “saltar” la condición indeterminada?

Lo primero que debemos analizar es que el valor de “ a ” sea diferente de “0”, para evitar la división entre cero, solo si es *verdadero* recién evaluar la división para obtener el valor de x .

Si es cierto que a es distinto de cero entonces evaluar la fórmula de solución.

Sentencia Condicional IF

“Si esta condición es **cierta**, entonces ejecuta estas acciones”



En este caso, según el diagrama de flujo anterior si a es 0 la ecuación no tiene solución, veamos.

- **SI $a=0$ ENTONCES** “la ecuación no tiene solución”.
- **SI $a \neq 0$ ENTONCES** $x=-b/a$, y “el valor de x es ”, x

El programa en *Arduino* sería ahora como sigue.

```
//Ecuacion de primer grado verificando que a!=0
float a = 0; //se declara a de tipo float y con valor 0
float b = 0; //se declara b de tipo float y con valor 0
float x = 0; //se declara x de tipo float y con valor 0
```

```
int condicion = 0; //condicion que controla el flujo del programa

void setup() { //esta funcion solo se ejecuta una sola vez
    Serial.begin(9600); //configura el puerto serial a 9600 de velocidad
}

void loop() {
    Serial.println("");
    Serial.println("Ecuacion de primer grado ");
    Serial.println("ax+b=0 ");
    Serial.println("dame a : ");

    while (condicion == 0) { //si se cumple la condicion cero entrar
        if (Serial.available() > 0) { //si hay dato en el serial entrar
            a = Serial.parseFloat(); //el valor de "a" en float.
            Serial.print("a = "); //mostrar el valor de
            Serial.println(a); //de la variable "a"
            Serial.println("dame b : ");
            condicion = 1; //cambia la condicion por 1 para salir
        }
    }

    while (condicion == 1) { //como ahora la condicion vale 1 se entra
        if (Serial.available() > 0) { //si hay dato en el serial, se entra
            b = Serial.parseFloat(); //entra el valor de b tipo float
            parseFloat()
            Serial.print("b = "); //muestra el valor
            Serial.println(b); //de la variable "b"
            condicion = 2; //cambia la condicion por 2 para salir
        }
    }
}
```

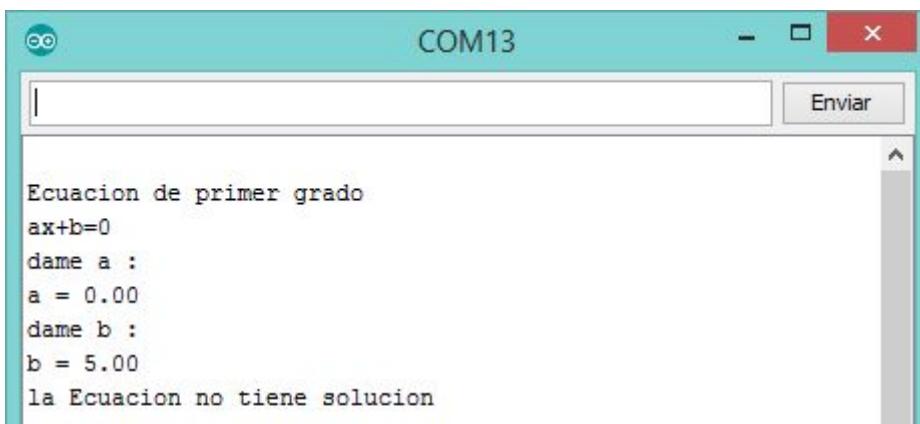
```

    }
}

//salimos de los bucles while, verificar que a!=0
if (a == 0) { //si a==0 entonces no tiene solucion
    Serial.println("la Ecuacion no tiene solucion");
    condicion = 0; //se cambia la condicion y se regresa al
bucle inicial
}
if (a != 0) {
    x = -1 * b / a; //se calcula "x" y se guarda en "x"
    Serial.print("x es: ");
    Serial.println(x); //se muestra la variable x
    Serial.println("");
    condicion = 0; //cambia la condicion a 0 se regresa al
bucle inicial
}
}
}

```

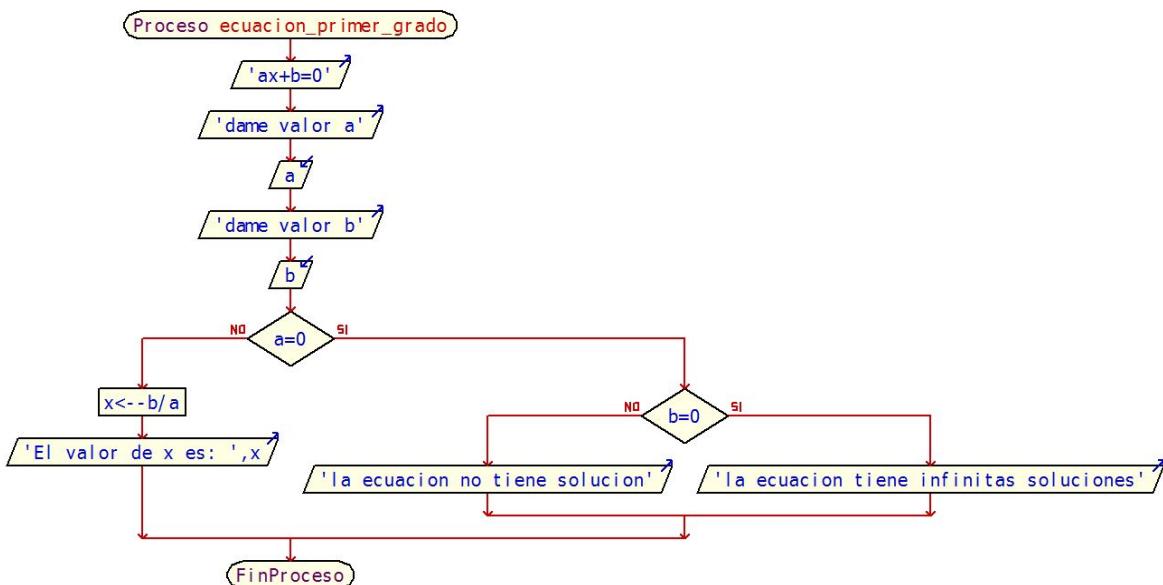
Después de subirlo al *Arduino* y de activar el *Monitor serial* y con valores de $a=0$ y $b=5$ se tiene que podemos detectar cuando la ecuación no tiene solución.



SENTENCIAS CONDICIONALES ANIDADAS

Hey por fin!, bueno como te explico, aun no vimos el caso en que $a=0$ y además $b=0$ donde se darían infinitas soluciones, si alimentamos nuestro programa con $a=0$ y $b=0$ tendríamos que *La Ecuación no tiene solución*, pero no es cierto ya que se tienen *infinitas soluciones*, aun falta arreglar nuestro programa, vamos a usar condiciones *if* anidadas (uno dentro de

otro) podemos guiarnos con el siguiente diagrama de flujo.



Como podemos notar si se cumple la *condición* que $a=0$ y además que $b=0$ entonces se tendrían infinitas soluciones, ahora a picar código.

```

//Ecuacion de primer grado verificando que a!=0
float a = 0; //se declara a de tipo float y con valor 0
float b = 0; //se declara b de tipo float y con valor 0
float x = 0; //se declara x de tipo float y con valor 0
int condicion = 0; //condicion que controla el flujo del programa

void setup() { //esta funcion solo se ejecuta una sola vez
    Serial.begin(9600); //configura el puerto serial a 9600 de velocidad
}

void loop() {
    Serial.println("");
    Serial.println("Ecuacion de primer grado ");
    Serial.println("ax+b=0 ");
    Serial.println("dame a : ");

    while (condicion == 0) { //si se cumple la condicion cero
    }
}
    
```

```

entrar
    if (Serial.available() > 0) { //si hay dato en el serial entrar
        a = Serial.parseFloat();           //el valor de "a" en float.
        Serial.print("a = ");             //mostrar el valor de
        Serial.println(a);               //de la variable "a"
        Serial.println("dame b : ");
        condicion = 1;                  //cambia la condicion por 1 para
salir
    }
}

while (condicion == 1) {      //como ahora la condicion vale 1 se entra
    if (Serial.available() > 0) { //si hay dato en el serial, se
entra
    b = Serial.parseFloat(); //se entra el valor de b tipo float
    Serial.print("b = ");    //muestra el valor
    Serial.println(b);       //de la variable "b"
    condicion = 2;          //cambia la condicion por 2 para salir
}
}

//salimos de los bucles while, verificar que a!=0
if (a == 0) { //si a==0 , no tiene solucion o infinitas soluciones
    if (b == 0) { //si b=0 se tiene infinitas soluciones
        Serial.println("La Ecuacion tiene infinitas soluciones");
        condicion = 0;
    }
    if (b != 0) { //si b tiene algun valor no se tiene solucion.
        Serial.println("La Ecuacion no tiene solucion");
        condicion = 0; //se cambia la condicion y se regresa al inicio
    }
}

if (a != 0) {
    x = -1 * b / a; //se calcula "x" y se guarda en "x"
    Serial.print("x es: ");
    Serial.println(x); //se muestra la variable x
    Serial.println("");
    condicion = 0; //cambia la condicion a 0 se regresa al bucle
inicial
}

```

}

Ahora si ejecutamos nuestro programa y lo alimentamos con valores de ejemplo como $a=2$, $b=3$ luego $a=0$ y $b=0$ y finalmente $a=0$ y $b=3$ tendremos lo siguiente.

```

Ecuacion de primer grado
ax+b=0
dame a :
a = 2.00
dame b :
b = 3.00
x es: -1.50

Ecuacion de primer grado
ax+b=0
dame a :
a = 0.00
dame b :
b = 0.00
La Ecuacion tiene infinitas soluciones

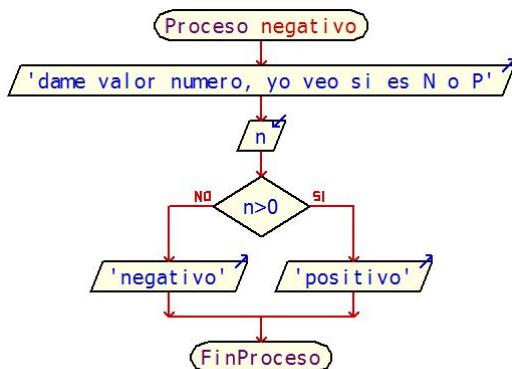
Ecuacion de primer grado
ax+b=0
dame a :
a = 0.00
dame b :
b = 3.00
La Ecuacion no tiene solucion
  
```

The screenshot shows a terminal window titled "COM13" with three distinct runs of a program. Each run starts with "Ecuacion de primer grado" and "ax+b=0". It prompts for values of "a" and "b". In the first run, "a" is set to 2.00 and "b" to 3.00, resulting in the solution "x es: -1.50". In the second run, both "a" and "b" are set to 0.00, leading to the message "La Ecuacion tiene infinitas soluciones" (The equation has infinite solutions). In the third run, "a" is 0.00 and "b" is 3.00, resulting in the message "La Ecuacion no tiene solucion" (The equation has no solution). The terminal also includes standard controls like "Enviar" (Send), "Autoscroll" checked, and baud rate settings at the bottom.

Ahora si nuestro programa de solución de ecuaciones de primer grado está **terminado**, ha sido un camino largo pero se vio claramente el control de flujo mediante *condicionales*, ahora sigamos con más ejercicios, ¿como podemos evaluar si un número es positivo o si es negativo?, no te enojes es por tu bien.

14. Leer un número y ver si es Positivo o Negativo

Para que un número sea positivo o negativo se debe cumplir la condición que ese número sea mayor o que sea menor que cero, podemos guiarnos del siguiente diagrama de flujo.



Ahora a escribir código, el programa es el siguiente.

```

int numero = 0; //se declara la variable numero a verificar

void setup() {
    Serial.begin(9600); //configura el puerto serial a 9600 de velocidad
    Serial.println("Verificar si un numero es positivo o negativo");
    Serial.println("*****");
    Serial.println("");
    Serial.println("Envia el numero a verificar: ");
}

void loop() {

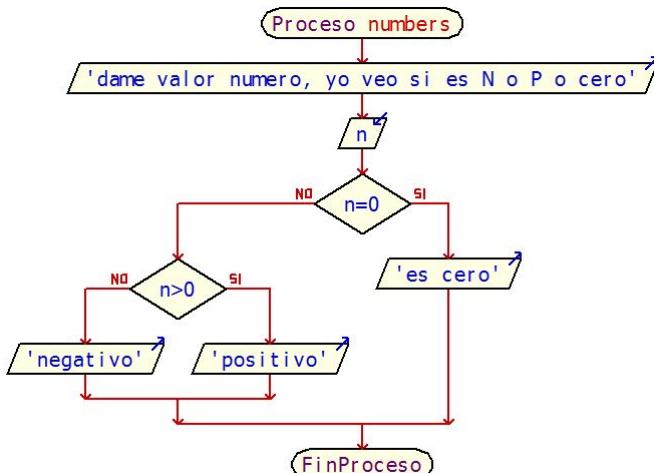
    if (Serial.available() > 0) { //si hay dato en serial entrar a
    condicional if
        numero = Serial.parseInt(); //parseInt al serial se guarda en
    "numero"
        Serial.print("numero = "); //mostrar el valor
        Serial.println(numero); //de la variable "numero"
        if (numero > 0) {
            Serial.println("El numero es positivo");
        }
        if (numero < 0) {
            Serial.println("El numero es negativo");
        }
}
    
```

```
}
```

Si hacemos la prueba con número positivos, negativos funciona bien, pero veamos lo que ocurre cuando le damos de *entrada* un valor igual a cero

```
Verificar si un numero es positivo o negativo
*****
Envia el numero a verificar:
numero = 6
El numero es positivo
numero = -8
El numero es negativo
numero = 0
```

Hey que paso?, bueno *0* no es negativo ni positivo, por lo tanto no cumple ninguna *condición*, vamos a corregir este problema guiandonos de el siguiente diagrama de flujo.



Hay varias formas de solucionar este problema, una podría ser una nueva condición *if* verifique que el número es cero (*numero == 0*), otra forma sería usar un *else* (que no usaremos por ahora). El código es el siguiente.

```

int numero = 0;           //se declara la variable numero a verificar

void setup() {
    Serial.begin(9600); //configura el puerto serial a 9600 de velocidad
    Serial.println("Verificar si un numero es positivo o negativo");
    Serial.println("*****");
    Serial.println("");
    Serial.println("Envia el numero a verificar: ");
}

void loop() {
    if (Serial.available() > 0) { //si hay dato en el serial entrar
        condicional if
        numero = Serial.parseInt(); //parseInt al serial se guarda en
        "numero"
        Serial.print("numero = "); //mostrar el valor
        Serial.println(numero); //de la variable "numero"
        if (numero > 0) { //condicional +
            Serial.println("El numero es positivo");
        }
        if (numero < 0) { //condicional -
            Serial.println("El numero es negativo");
        }
        if (numero == 0) { //si no es + o - debe ser
            Serial.println("El numero es cero");
        }
    }
}

```

Después de subir el programa a nuestro *Arduino* y conectarnos por *Monitor serie* tendremos lo siguiente.

```

COM13
Enviar

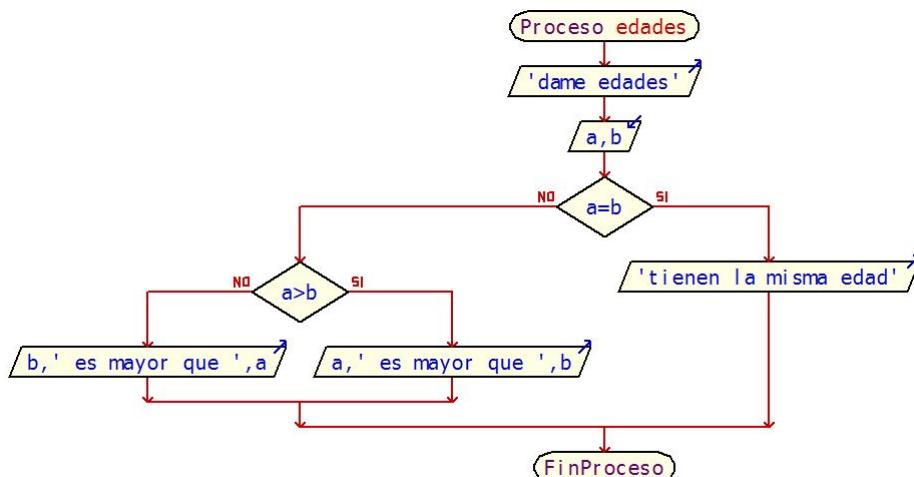
Verificar si un numero es positivo o negativo
*****
Envia el numero a verificar:
numero = 2
El numero es positivo
numero = -5
El numero es negativo
numero = 0
El numero es cero

```

Como podemos ver ahora que si se cumplen todas las *condiciones*, podemos verificar si es positivo, negativo y si es cero. Ahora veremos un programa muy parecido que compare la edad de dos personas y te diga cual es mayor, menor o si tienen la misma edad (*a encerar, pulir*).

15. Compara las edades de dos personas, cual es mayor, menor o si tienen la misma edad.

Para este ejercicio nos ayudaremos del siguiente *diagrama de flujo*.



Es un problema es sencillo, básicamente es una *entrada* de dos valores enteros, evaluar las *condiciones* de mayor, menor o igual y dar una *salida*. El código es el siguiente.

```
//Compara edades
int luis = 0; //se declara a luis de tipo entero y con valor 0
```

```

int miguel = 0; //se declara a miguel de tipo entero y con valor 0
int condicion = 0; //condicion la cual controla el flujo del programa

void setup() { //esta funcion solo se ejecuta una sola vez
    Serial.begin(9600); //configura el puerto serial a 9600 de
}
}

void loop() {
    Serial.println("");
    Serial.println("Compara edades");
    Serial.println("dame la edad de luis: ");

    while (condicion == 0) { //se cumple la condicion cero entrar a este bucle
        if (Serial.available() > 0) { //si dato en serial entrar condicional if
            luis = Serial.parseInt(); //la edad de luis es un entero.
            Serial.print("la edad de luis es = "); //mostrar el valor
            Serial.println(luis); //de la variable "luis"
            Serial.println("dame la edad de miguel: ");
            condicion = 1; //cambia la condicion por 1 para salir
        }
    }

    while (condicion == 1) { //ahora la condicion vale 1 se entra while
        if (Serial.available() > 0) { //si hay dato en serial se entra
            miguel = Serial.parseInt(); //se entra la edad de miguel tipo int
            Serial.print("la edad de miguel es = "); //muestra el valor
            Serial.println(miguel); //de la variable "miguel"
            condicion = 2; //cambia condicion por 2 para salir de while.
        }
    }
}

```

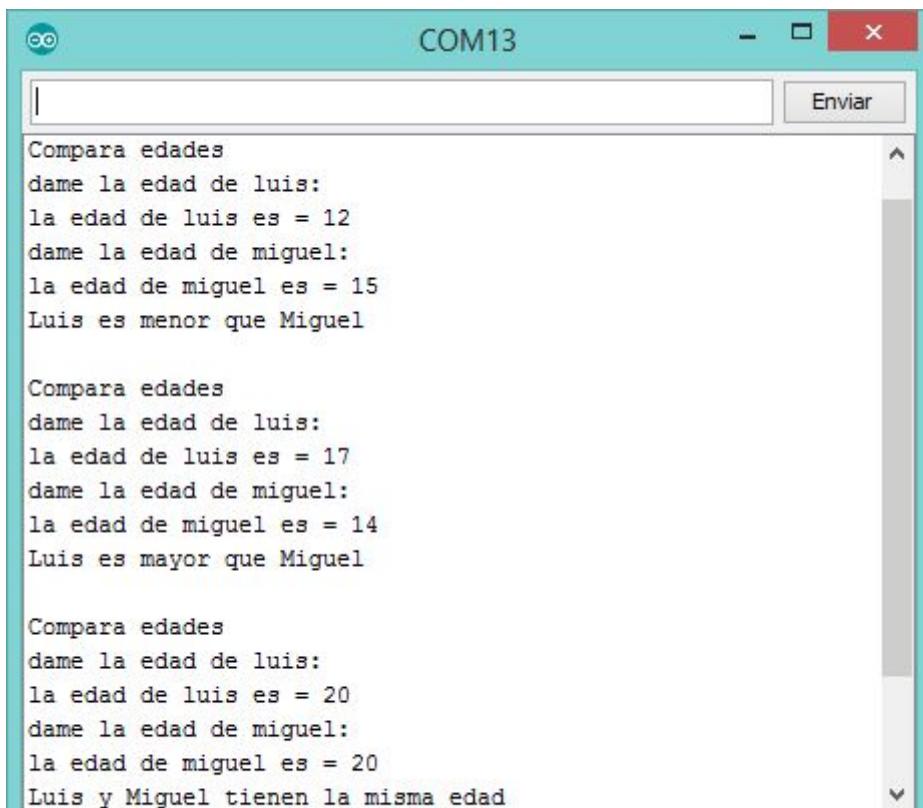
```

if (luis > miguel) { //luis es mayor que miguel.
    Serial.println("Luis es mayor que Miguel");
}
if (luis < miguel) { //luis es menor que miguel.
    Serial.println("Luis es menor que Miguel");
}
if (luis == miguel) { //luis y miguel tienen la misma edad
    Serial.println("Luis y Miguel tienen la misma edad");
}

condicion = 0;      //condicion a 0 se regresa al bucle inicial
}

```

Si subimos nuestro programa a nuestro *Arduino* y abrimos el *monitor serial* tendremos lo siguiente.

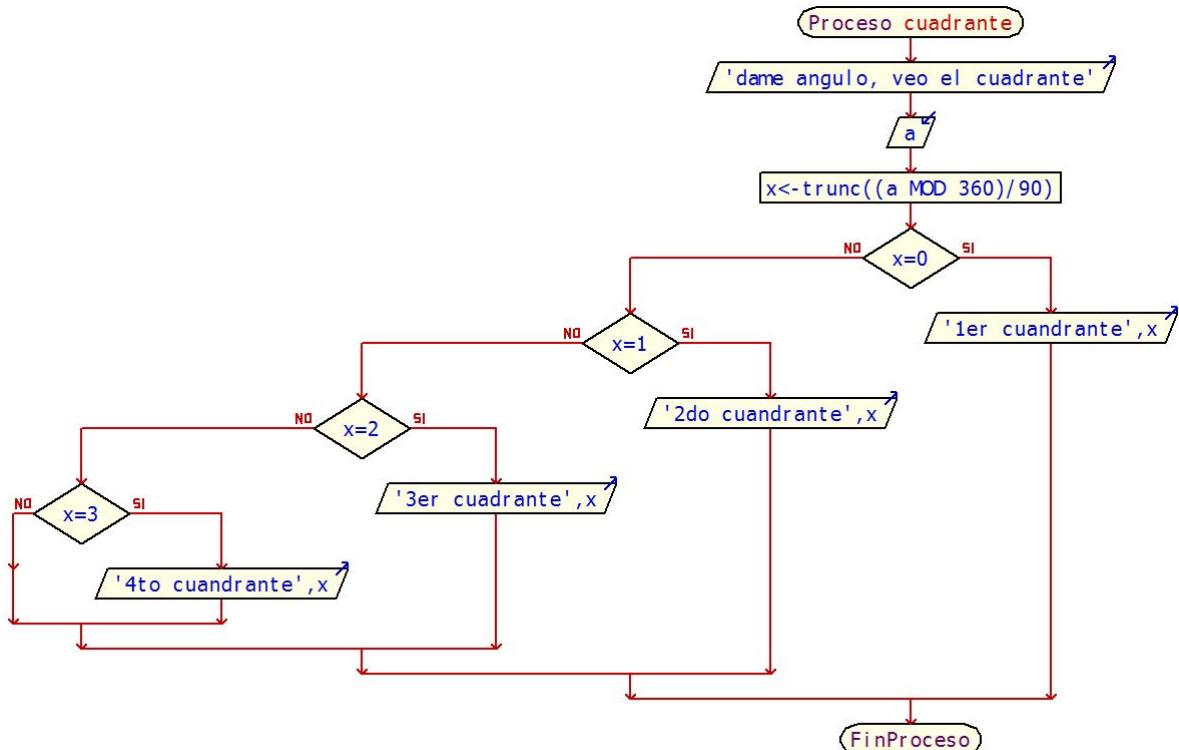


Una solución más elegante hubiese sido el uso de *condicionales* tipo *if ... else*, veremos como usar ese tipo de *condicionales* más adelante. Por el momento nos concentraremos en las *operaciones matemáticas* y escribiremos un programa que reciba un grado sexagesimal y

nos diga en qué cuadrante trigonométrico esta (I, II, III o IV cuadrante).

16. En que cuadrante trigonométrico esta un grado sexagesimal

Para el presente ejercicio haremos uso de la fórmula para establecer la pertenencia a un determinado cuadrante así como del siguiente *diagrama de flujo*.



En el diagrama de flujo se lee *TRUNC*, esa es la función de redondeo o mejor dicho *quédate con la parte entera de un número con decimales*.

La fórmula de pertenencia a un determinado cuadrante es:

$$x = \text{parte_entera}[(\text{grado MODULO } 360/90)]$$

La operación *MODULO* es el **residuo** que queda de una división, por ejemplo.

1 MODULO 2 =1, ya que la cociente es cero (en entero), queda 1 en el *residuo*

4 MODULO 2 =0, ya que el cociente es 2 y no queda nada en el *residuo*

11 MODULO 6 = 5, ya que el cociente es 1 pero queda 5 en el *residuo*.

Volviendo a nuestro programa, el código es el siguiente.

```

int grado = 0; //variable grado entero
int x = 0; //variable x entero

void setup() {
  Serial.begin(9600); //puerto serial a 9600
  Serial.println("Dame el angulo ");
}

void loop() {
  if (Serial.available() > 0) {//si hay dato en el serial entrar
    grado = Serial.parseInt(); //leer entero del serial
    Serial.println(grado); //mostrar el dato ingresado
    x = ((grado % 360) / 90); //evaluar el grado ingresado
    Serial.print("Resultado evaluacion es: ");
    Serial.println(x); //mostrar el resultado
    if (x == 0) { //si es V esta en 1er cuadrante
      Serial.println("Pertenece al primer cuadrante");
    }
    if (x == 1) { //si es V esta en 2do cuadrante
      Serial.println("Pertenece al segundo cuadrante");
    }
    if (x == 2) { //si es V esta en 3er cuadrante
      Serial.println("Pertenece al tercer cuadrante");
    }
    if (x == 3) { //si es V esta en 4to cuadrante
      Serial.println("Pertenece al cuarto cuadrante");
    }
}
}

```

Como podemos ver no implementamos *TRUNC* del diagrama de flujo ya que la variable *x* ya era un *entero* al momento de declararla al inicio del programa (*int*). Si probamos nuestro programa con el *Monitor serie* tendremos.

The screenshot shows the Arduino Serial Monitor window titled "COM13". It contains the following text:

```

Dame el angulo
45
Resultado evaluacion es: 0
Pertenece al primer cuadrante
91
Resultado evaluacion es: 1
Pertenece al segundo cuadrante
190
Resultado evaluacion es: 2
Pertenece al tercer cuadrante
290
Resultado evaluacion es: 3
Pertenece al cuarto cuadrante

```

Continuamos con ejercicios con *condicionales*, ahora veremos el caso cuando un número es par o cuando es impar.

17. Cuando un número es PAR o IMPAR

Para el presente ejercicio, recordamos que un número es PAR cuando el *residuo* de dividirlo entre 2 es igual a 0. Y como vimos antes el *residuo* es el operador *MODULO*, en *Arduino* se usa el símbolo % para definir *MODULO*.

```

int numero = 0; //numero a ingresar
int x = 0; //para el modulo entre 2

void setup() {
    Serial.begin(9600); //velocidad serial a 9600
    Serial.println("Numero Par o impar");
    Serial.println("Dame numero: ");
}

void loop() {
    if (Serial.available() > 0) { //si hay dato en serial entrar
        numero = Serial.parseInt(); //dato serial entero
        Serial.println(numero);
        x = numero % 2; //modulo de numero entre dos
        if (x == 0) { //si no hay residuo es par

```

```

    Serial.println("El numero es par ");
}
else {                                //o si no, es impar
    Serial.println("El numero es impar ");
}
Serial.println("Dame numero: ");
}
}

```



Recordar que un número es *par* cuando el *residuo* de dividirlo entre 2 es *CERO*.

18. Desglosar una cantidad de dinero en billetes y monedas de circulación común.

Para este ejercicio, nos dan una *cantidad de dinero* y el objetivo es desglosar esa cantidad de dinero en billetes de *distinta denominación en forma óptima*, desde billetes de 200 hasta monedas de 1 sol.

Por ejemplo: “Tenemos 366 soles, entonces lo dividimos entre 200 y tenemos 1 billete de 200 y 166 de módulo o dinero actual, luego lo dividimos esos 166 entre 100 y nos da un billete de 100, con saldo de 66 soles, luego dividimos entre 50 y tenemos 1 billete de 50 y 16 soles de saldo, ahora lo dividimos entre 20, no contiene, no tenemos ningún billete de 20 soles, seguimos con 16 soles de saldo, ahora lo dividimos entre 10 y tenemos un billete de 10 y 6 soles de saldo, ahora el saldo lo dividimos entre 5 y tenemos 1 moneda de 5 soles, ahora tenemos de saldo 1 sol que dividimos entre 2 y no se tiene ninguna moneda de 2 porque no contiene, el saldo sigue siendo 1 sol y dividimos entre 1 y tenemos la última moneda de 1 sol, y con saldo de cero soles”.

El código del programa para nuestro *Arduino* es el siguiente.

```
//desglosa dinero en 200,100,50,20,10,5,2,1 soles
```

```

int dinero = 0; //el dinero a desglosar
int d200 = 0; //billetes de 200
int d100 = 0; //billetes de 100
int d50 = 0; //billetes de 50
int d20 = 0; //billetes de 20
int d10 = 0; //billetes de 10
int d5 = 0; //monedas de 5
int d2 = 0; //monedas de 2
int d1 = 0; //monedas de 1

void setup() {
    Serial.begin(9600);
    Serial.println("Desglosar Billetes");
    Serial.println("Dame cantidad:");
}

void loop() {

    if (Serial.available() > 0) { //si hay dato en serial entrar y
        ejecutar
        dinero = Serial.parseInt(); //entrar "dinero" entero por serial
        Serial.println(dinero); //mostrar "dinero"
        d200 = dinero / 200; dinero = dinero % 200; // $200 y se actualiza
        saldo
        d100 = dinero / 100; dinero = dinero % 100; // $100 y se actualiza
        saldo
        d50 = dinero / 50; dinero = dinero % 50; // y asi sucesivamente
        d20 = dinero / 20; dinero = dinero % 20;
        d10 = dinero / 10; dinero = dinero % 10;
        d5 = dinero / 5; dinero = dinero % 5;
        d2 = dinero / 2; dinero = dinero % 2;
        d1 = dinero / 1; dinero = dinero % 1;

        Serial.print(d200); Serial.println(" billetes de 200 "); // $200
        Serial.print(d100); Serial.println(" billetes de 100 "); // $100 y
        asi
        Serial.print(d50); Serial.println(" billetes de 50 ");
        //sucesivamente
        Serial.print(d20); Serial.println(" billetes de 20 ");
        Serial.print(d10); Serial.println(" billetes de 10 ");
}

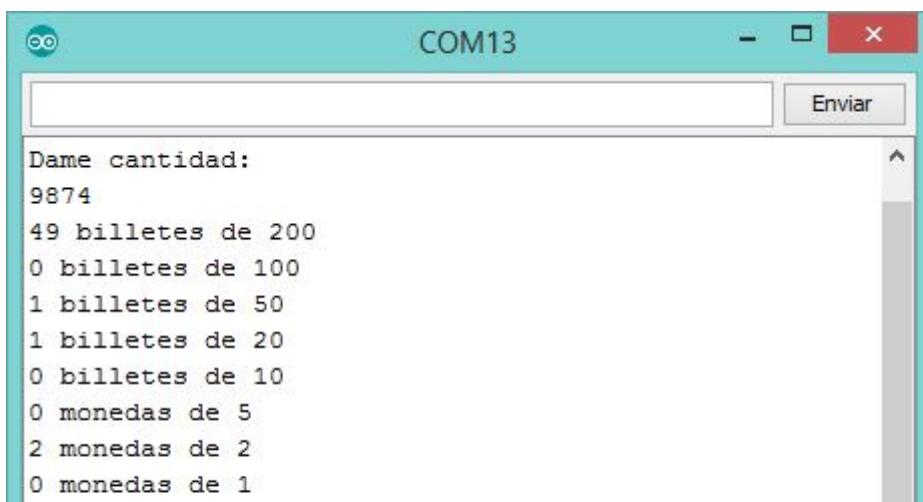
```

```

Serial.print(d5);    Serial.println(" monedas de 5 ");
Serial.print(d2);    Serial.println(" monedas de 2 ");
Serial.print(d1);    Serial.println(" monedas de 1 ");
Serial.println("");
Serial.println("Desglosar Billetes");
Serial.println("Dame cantidad:");
}
}

```

Si subimos nuestro programa y nos conectamos a nuestro querido *Monitor serie* tendremos lo siguiente para un dinero de 9874.



Funciona!, ahora te invito a hacer la prueba con el valor 99999, ¿que sucede? ¿cuál es el límite en que puede desglosar dinero correctamente? ¿Porque?

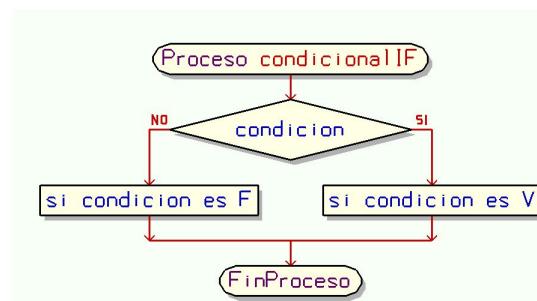
ESTRUCTURAS DE CONTROL

SENTENCIAS DE CONTROL *IF*

Hasta ahora nuestros “programas” han sido del tipo:

1. Se ingresan datos
2. Se hacen cálculos
3. Se imprimen resultados.

No se tiene control sobre el flujo del programa, es decir hasta ahora el flujo de ejecución es SECUENCIAL.



**Si se cumple la condición se ejecuta un lado
Si no se cumple la condición se ejecuta el otro.**

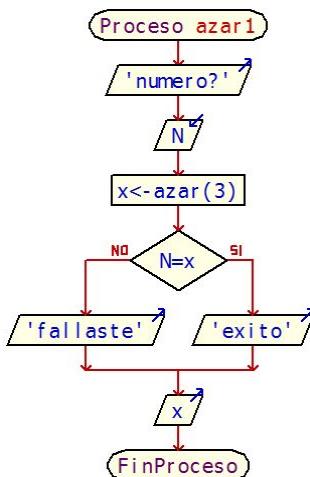
No nos olvidemos del mantra

- **Entrada:** recibir datos del teclado, o de un archivo o de otro aparato.
- **Salida:** mostrar datos en el monitor o enviar datos a un archivo a otro aparato.
- **Matemáticas:** ejecutar operaciones básicas, como la adición y multiplicación.
- **Operación Condicional:** probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.
- **Repetición:** ejecutar alguna acción repetidas veces, usualmente con alguna variación.

19. Generar un número al azar, y adivinar qué número es

Para este ejercicio, vamos a generar valores aleatorios usando la función `random(maximo)` que genera un número al azar desde cero hasta `maximo`². Podemos guiarnos por el siguiente diagrama de flujo.

² min y max son palabras reservadas por el lenguaje de Arduino y no se pueden declarar como variables o como funciones.



El código fuente de nuestro programa sería el siguiente.

```

int numero = 0; //numero a ingresar
int azar = 0;    //numero a adivinar

void setup() {
  Serial.begin(9600);
  azar = random(5);
  Serial.println("escoge numero del 0 al 5");
}

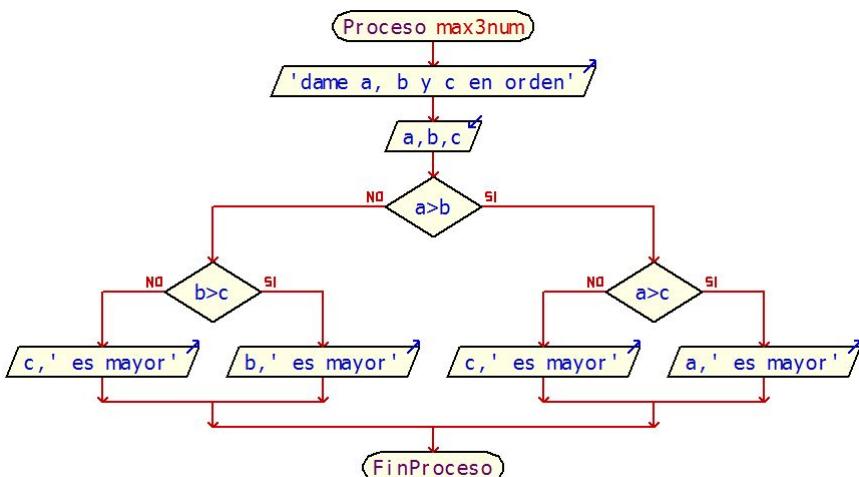
void loop() {
  if (Serial.available() > 0) { //si hay dato en el serial entrar
    numero = Serial.parseInt(); //numero es dato serial en entero
    if (numero == azar) {      //si "numero" y "azar" son iguales
      Serial.println("Acertaste");
      Serial.println("resetea el Arduino para jugar de nuevo");
    }
    else {                    //si no son iguales repite
      Serial.println("Intenta de nuevo");
    }
}
}
  
```

Como vemos se genera el valor al *azar* con la función *random(5)* en la función *setup()*, el valor se introduce en la función *loop()*, y si son iguales sale el mensaje que *ganaste* y si son diferentes sale el mensaje que *intentes de nuevo* hasta ganar, para repetir el juego debes

presionar *RESET* en el *Arduino*.

20. Reciba 3 números y muestra el número máximo introducido

Para este ejercicio vamos a recibir 3 *entradas* por teclado haciendo uso de *condicionales* y *repeticiones* con *while*, luego de introducir los datos vamos a *comparar* las 3 *entradas* y vamos a mostrar en la *salida* el número más grande introducido. El código sería el siguiente. Podemos guiarnos por el siguiente diagrama de flujo.



El código sería el siguiente.

```

//Que numero es mas grande
int a = 0; //variable a
int b = 0; //variable b
int c = 0; //variable c
byte condicion = 0; //condicion de control, byte ocupa menos espacio

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Dame 3 numero y comparo");
  Serial.println("Numero a?");
}

void loop() {
  while (condicion == 0) { //condicion inicial cero se entra
    if (Serial.available() > 0) { //si hay dato se entra
      a = Serial.parseInt(); //el dato en serial se guarda en "a"
    }
    if (a > b) {
      if (a > c) {
        Serial.print(a);
        Serial.print(" es el mayor ");
      } else {
        Serial.print(c);
        Serial.print(" es el mayor ");
      }
    } else {
      if (b > c) {
        Serial.print(b);
        Serial.print(" es el mayor ");
      } else {
        Serial.print(c);
        Serial.print(" es el mayor ");
      }
    }
  }
}
  
```

```
Serial.print("a=");
Serial.println(a);
Serial.println("Numero b?");
condicion = 1; //se cambia condicion para cambiar de bucle
}
}

while (condicion == 1) { //se entra este bucle con condicion 1
    if (Serial.available() > 0) { //si hay dato en serial entrar
        b = Serial.parseInt(); //guardar entero del serial en "b"
        Serial.print("b=");
        Serial.println(b);
        Serial.println("Numero c?");
        condicion = 2; //se cambia para cambiar de bucle
    }
}

while (condicion == 2) { //se entra a este bucle con condicion 2
    if (Serial.available() > 0) { //si hay dato en serial entrar
        c = Serial.parseInt(); //entero en serial se guarda en "c"
        Serial.print("c=");
        Serial.println(c);
        Serial.println("Comparando...");
        condicion = 3; //para salir de bucle
    }
}

if (a > b) { //condicionales if--else que comparan los datos a,b,c
    if (a > c) { //si a> c entrar
        Serial.print(a);
        Serial.println(" es el mayor");
    }
    else { //o si no entrar
        Serial.print(c);
        Serial.println(" es el mayor");
    }
}
else { //o si no
    if (b > c) { //si b>c entrar
        Serial.print(b);
        Serial.println(" es el mayor");
    }
}
```

```

else { //o si no
Serial.print(c);
Serial.println(" es el mayor");
}
}

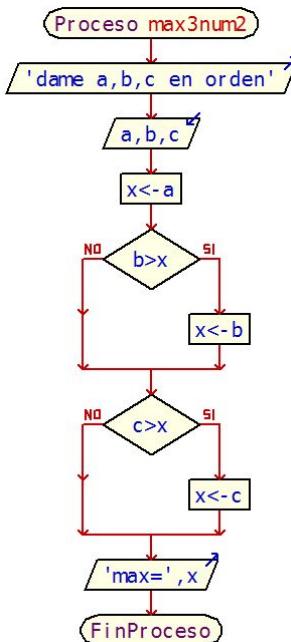
condicion = 0; //se reinicia la condicion para repetir el programa
Serial.println("Numero a?");
}
}

```

Como vemos en el ejercicio lo que tenemos son varias *condiciones* que se van evaluando después de haber introducido los valores de a, b y c como *entrada*, en el siguiente ejercicio vamos a ver una aproximación distinta del problema, vamos a usar una variable que se va *actualizando en sucesivas comparaciones*, es más sencillo.

21. Reciba 3 números y muestra el número máximo introducido (otro)

Vamos a solucionar el problema anterior haciendo uso de una variable auxiliar *x* que se va actualizar en cada comparación, esta variable define actualmente qué número es mayor y mediante sucesivas comparaciones con el resto de variables a, b y c se tiene *al final* el número mayor. Nos vamos a guiar del siguiente diagrama de flujo.



El código fuente el programa sería como sigue.

```

//Que numero es mas grande (otro)
int a = 0; //variable a
int b = 0; //variable b
int c = 0; //variable c
int x = 0; //variable x valor mayor actual
byte condicion = 0; //condicion de control, byte ocupa menos espacio

void setup() {
    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Dame 3 numero y comparo");
    Serial.println("Numero a?");
}

void loop() {
    while (condicion == 0) { //condicion inicial cero se entra
        if (Serial.available() > 0) { //si hay dato se entra
            a = Serial.parseInt(); //el dato en serial se guarda entero en
            "a"
            Serial.print("a=");
            Serial.println(a);
            Serial.println("Numero b?");
            condicion = 1; //se cambia condicion para cambiar de
            bucle
        }
    }

    while (condicion == 1) { //se entra a este bucle despues de entrar
    "a"
        if (Serial.available() > 0) { //si hat dato en serial entrar
            b = Serial.parseInt(); //guardar entero del serial en "b"
            Serial.print("b=");
            Serial.println(b);
            Serial.println("Numero c?");
            condicion = 2; //se cambia para cambiar de bucle
        }
    }

    while (condicion == 2) { //se entra a este bucle con condicion 2
        if (Serial.available() > 0) { //si hay dato en serial entrar
            c = Serial.parseInt(); //entero en serial se guarda en "c"
            Serial.print("c=");
    }
}

```

```

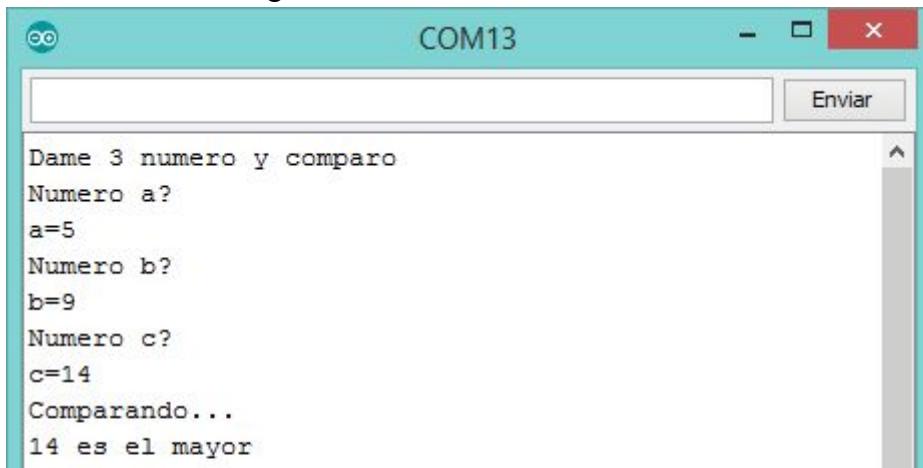
Serial.println(c);
Serial.println("Comparando... ");
condicion = 3;           //para salir de bucle
}
}

x = a; //suponemos que a es el mayor
if (b > x) { //si b > x entrar
    x = b; //ahora x toma el valor de b
}
if (c > x) { //si c > x entrar
    x = c; //ahora x toma el valor de c
}

Serial.print(x);
Serial.println(" es el mayor");
Serial.println("Numero a?");
condicion = 0; //se reinicia la condicion para repetir el programa
}

```

Como vemos esta aproximación es más simple y más efectiva y con el mismo procedimiento podemos ampliarlo a más números para comparar. Si activamos nuestro *Monitor serie* tendríamos lo siguiente.



Más mantra

UN DILEMA CON LAS VARIABLES

¿Qué sucede cuando ejecutamos lo siguiente?

```
x=1  
x=x+1
```

¿Tiene Lógica?

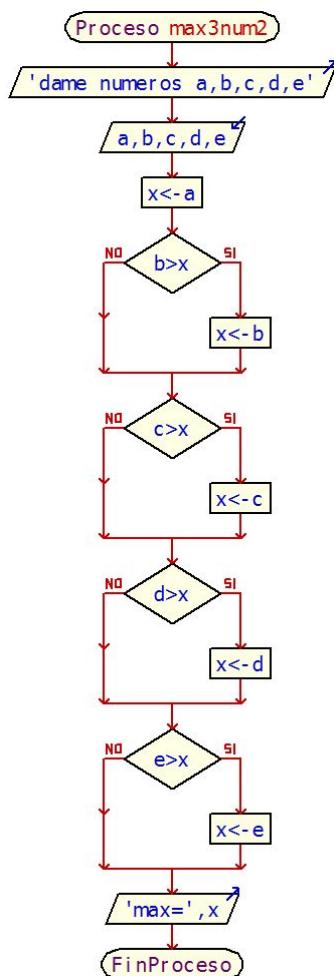
Desde el punto de vista de *matemática no hay Lógica* pero si lo hay en la programación, hay que recordar que una *definición en programación NO ES UNA ECUACIÓN*.

La primera expresión es una **ASIGNACIÓN** $x=1$, es decir x tiene el valor de 1.

La segunda expresión dice que $x=x+1$, significa **INCREMENTO**. En este caso un incremento en 1.

22. Recibe 5 números y muestra el número mayor introducido.

Si resolviste los ejercicios anteriores este va ser simple, solo debemos ampliar el código que vimos para cuando son 3 valores, veamos este diagrama de flujo.



Ahora el código fuente de la comparación de 5 números.

```

//Que numero es mas grande (otro)
int a = 0; //variable a
int b = 0; //variable b
int c = 0; //variable c
int d = 0; //variable d
int e = 0; //variable e
int x = 0; //variable x valor mayor actual
byte condicion = 0; //condicion de control, byte ocupa menos espacio

void setup() {
    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Dame 5 numeros y comparo");
}
    
```

```
Serial.println("Numero a?");
}

void loop() {
    while (condicion == 0) { //condicion inicial cero se entra
        if (Serial.available() > 0) { //si hay dato se entra
            a = Serial.parseInt(); //el dato en serial se guarda entero en
            "a"
            Serial.print("a=");
            Serial.println(a);
            Serial.println("Numero b?");
            condicion = 1;           //se cambia condicion para cambiar de
            bucle
        }
    }
    while (condicion == 1) { //se entra a este bucle despues de entrar
    "a"
        if (Serial.available() > 0) { //si hat dato en serial entrar
            b = Serial.parseInt(); //guardar entero del serial en "b"
            Serial.print("b=");
            Serial.println(b);
            Serial.println("Numero c?");
            condicion = 2;           //se cambia para cambiar de bucle
        }
    }
    while (condicion == 2) { //se entra a este bucle con condicion 2
        if (Serial.available() > 0) { //si hay dato en serial entrar
            c = Serial.parseInt(); //entero en serial se guarda en "c"
            Serial.print("c=");
            Serial.println(c);
            Serial.println("Numero d?");
            condicion = 3;           //para salir de bucle
        }
    }
    while (condicion == 3) { //se entra a este bucle con condicion 3
        if (Serial.available() > 0) { //si hay dato en serial entrar
            d = Serial.parseInt(); //entero en serial se guarda en "d"
            Serial.print("d=");
            Serial.println(d);
            Serial.println("Numero e?");
```

```

condicion = 4;           //para salir de bucle
}
}
while (condicion == 4) { //se entra a este bucle con condicion 4
    if (Serial.available() > 0) { //si hay dato en serial entrar
        e = Serial.parseInt(); //entero en serial se guarda en "e"
        Serial.print("e=");
        Serial.println(e);
        Serial.println("Comparando...");
        condicion = 5;           //para salir de bucle
    }
}

x = a; //suponemos que a es el mayor
if (b > x) { //si b > x entrar
    x = b; //ahora x toma el valor de b
}
if (c > x) { //si c > x entrar
    x = c; //ahora x toma el valor de c
}
if (d > x) { //si d > x entrar
    x = d; //ahora x toma el valor de d
}
if (e > x) { //si e > x entrar
    x = e; //ahora x toma el valor de e
}

Serial.print(x);
Serial.println(" es el mayor");
Serial.println("Numero a?");
condicion = 0; //se reinicia la condicion para repetir el programa
}

```

Si subimos nuestro programa a nuestro *Arduino* y abrimos el *monitor serial* tenemos lo siguiente.

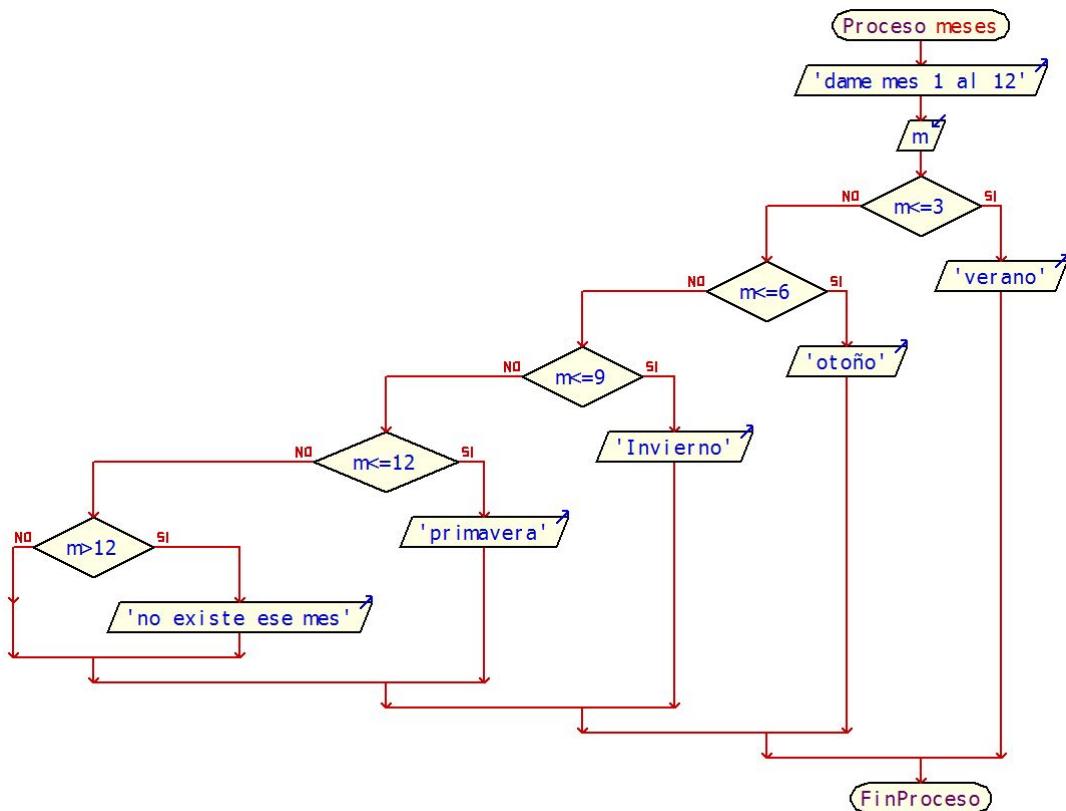
The screenshot shows a terminal window titled "COM13". The window has a red "X" button in the top right corner. Below the title, there is a text input field with a cursor and a "Enviar" (Send) button. The terminal displays the following text:

```
Dame 5 numeros y comparo
Numero a?
a=1
Numero b?
b=2
Numero c?
c=3
Numero d?
d=4
Numero e?
e=5
Comparando...
5 es el mayor
```

Divertido, ahora vamos a ver el caso de un programa que reciba un mes y que nos diga en qué estación estamos.

23. Recibe un mes y dice en qué estación estamos.

Para este ejercicio usaremos varias *condicionales if* que evalúan un mes, el intervalo en que se cumpla una *condición* indica la estación en la que está ese mes. Podemos guiarnos del siguiente diagrama de flujo.



El código sería el siguiente.

```

int mes = 0; //variable de mes entero

void setup() {
    Serial.begin(9600); //velocidad serial a 9600
    Serial.println("Digo en que estacion estamos");
    Serial.println("Dame mes (1-12)");
}

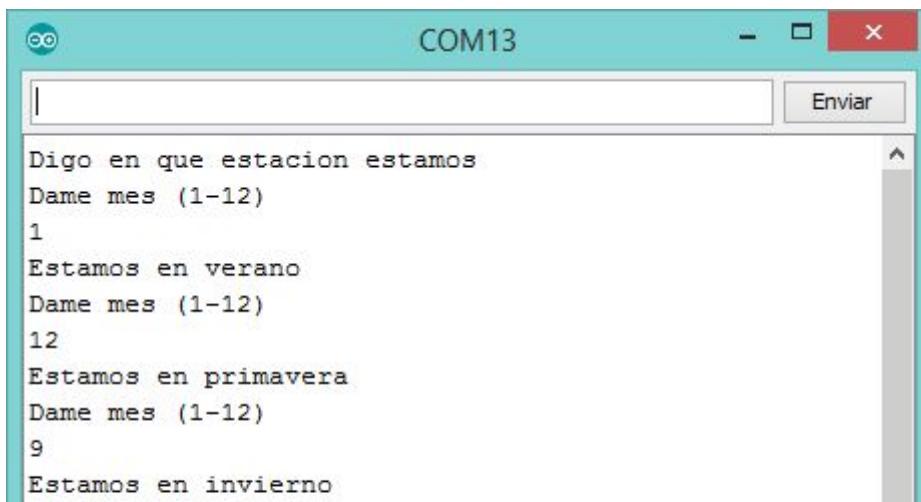
void loop() {
    if (Serial.available() > 0) { //si ah y dato en serial entrar
        mes = Serial.parseInt(); //entero en serial es mes
        Serial.println(mes);
        if (mes < 1 or mes > 12) { //menor a 0 , mayor a 12 no
            existe
            Serial.println("Ese mes no existe");
        }
    }
}
    
```

```

    }
    if (mes > 0 and mes < 4) { //si esta entre 1 y 4 es verano
      Serial.println("Estamos en verano");
    }
    if (mes > 3 and mes < 7) { //si esta entre 4 y 6 es otono
      Serial.println("Estamos en otono");
    }
    if (mes > 6 and mes < 10) { //si esta entre 7 y 9 es
invierno
      Serial.println("Estamos en invierno");
    }
    if (mes > 9 and mes < 13) { //si esta entre 10 y 12 es
primavera
      Serial.println("Estamos en primavera");
    }
    Serial.println("Dame mes (1-12)");
}
}

```

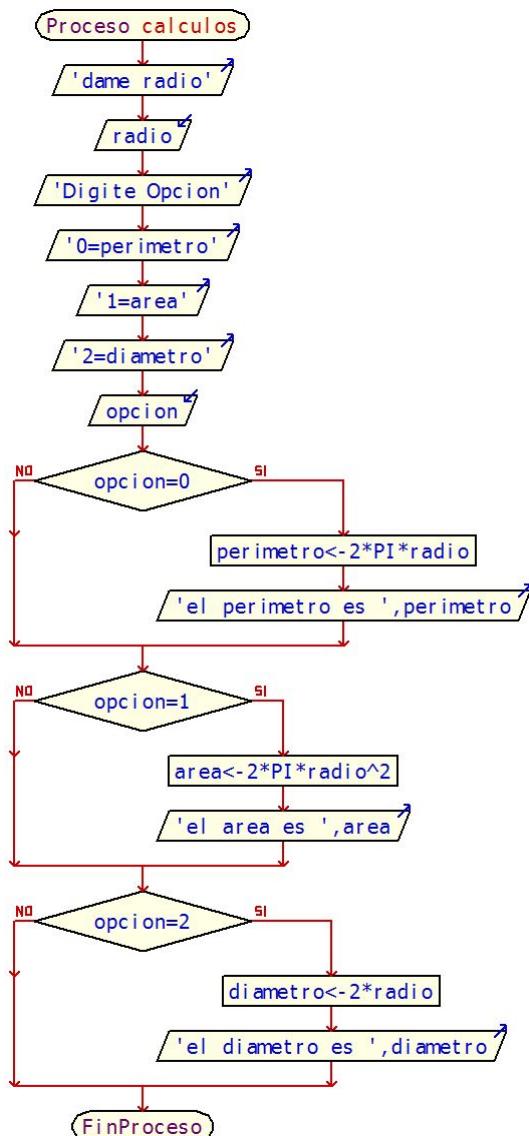
Luego se subirlo a nuestro *Arduino* y abrir nuestro *Monitor serie* tendremos.



24. Calcula el área, perímetro y circunferencia de un círculo, mediante un menú

Para el presente ejercicio mostraremos un menu, luego se hará una *entrada de dato de radio*, luego se escoge una accion del menu y despues se aplicara *operaciones matemáticas*

para mostrar en la *salida* lo elegido en el menú, podemos guiarnos del siguiente diagrama de flujo.



El código del programa sería el siguiente.

```

int radio = 0;
int opcion = 0;
int condicion = 0;

void setup() {

```

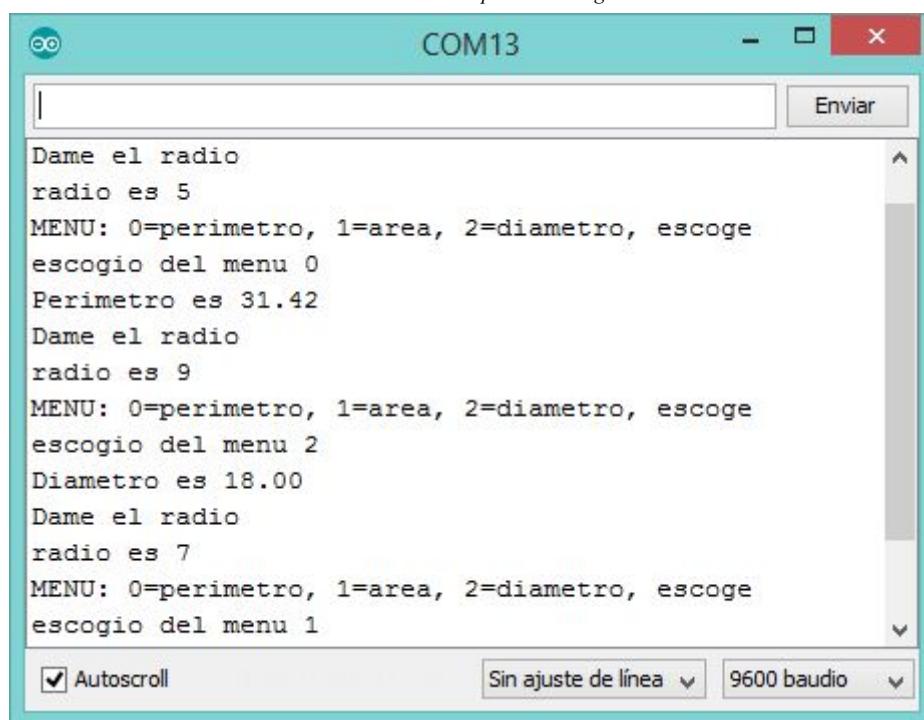
```
Serial.begin(9600);
Serial.println("Dame el radio");
}

void loop() {
    while (condicion == 0) {
        if (Serial.available() > 0) {
            radio = Serial.parseInt();
            Serial.print("radio es ");
            Serial.println(radio);
            Serial.println("MENU: 0=perimetro, 1=area, 2=diametro,
escoge");
            condicion = 1;
        }
    }
    while (condicion == 1) {
        if (Serial.available() > 0) {
            opcion = Serial.parseInt();
            if (opcion > 2 or opcion < 0) {
                Serial.println("Opcion de Menu no valido");
                Serial.println("Dame el radio");
                condicion = 0;
            }
            else {
                Serial.print("escogio del menu ");
                Serial.println(opcion);
                condicion = 2;
            }
        }
    }

    if (opcion == 0) {
        float perimetro = 2 * 3.1416 * radio;
        Serial.print("Perimetro es ");
        Serial.println(perimetro);
        Serial.println("Dame el radio");
    }
}
```

```
condicion = 0;
}
if (opcion == 1) {
    float area = 2 * 3.1416 * radio * radio;
    Serial.print("Area es ");
    Serial.println(area);
    Serial.println("Dame el radio");
    condicion = 0;
}
if (opcion == 2) {
    float diametro = 2 * radio;
    Serial.print("Diametro es ");
    Serial.println(diametro);
    Serial.println("Dame el radio");
    condicion = 0;
}
}
```

Ahora si lo subimos a nuestro *Arduino* y luego abrimos el *Monitor serie* tendríamos lo siguiente.



The screenshot shows the Arduino Serial Monitor window titled "COM13". The text output is as follows:

```

Dame el radio
radio es 5
MENU: 0=perimetro, 1=area, 2=diametro, escoge
escogio del menu 0
Perimetro es 31.42
Dame el radio
radio es 9
MENU: 0=perimetro, 1=area, 2=diametro, escoge
escogio del menu 2
Diametro es 18.00
Dame el radio
radio es 7
MENU: 0=perimetro, 1=area, 2=diametro, escoge
escogio del menu 1
  
```

At the bottom of the window, there are three buttons: "Autoscroll" (checked), "Sin ajuste de línea" (unchecked), and "9600 baudio" (selected).

SENTENCIAS ITERATIVAS (BUCLE ... LOOP)

EL BUCLE WHILE

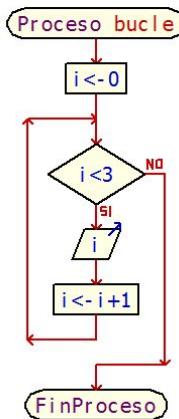
WHILE significa *MIENTRAS*, Se usa así:

```

WHILE condición;
  acción
  acción
  ...
  acción
  
```

MIENTRAS se cumpla esta condición, REPITE estas acciones

Las sentencias de repetición se les denomina *BUCLES*, por ejemplo analizemos el siguiente diagrama de flujo y piquemos código, en este caso meteremos el bucle *while* al *setup()* para solo ejecutarlo una sola vez y analizar su funcionamiento.



El código en *Arduino* sería el siguiente.

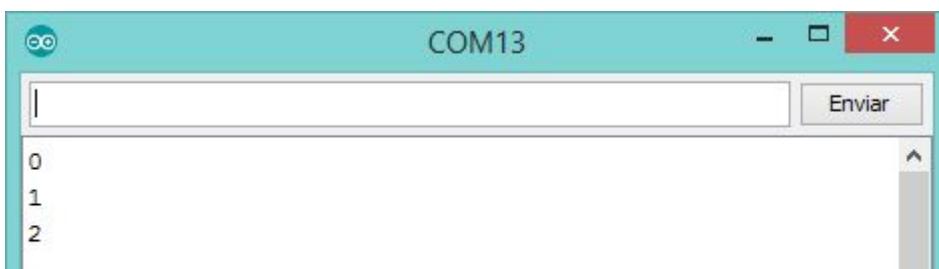
25. El bucle while

```

int i = 0;                      //declara i entero
void setup() {
  Serial.begin(9600);          //velocidad serial
  i = 0;                      //i vale 0
  while (i < 3) {              //mientras "i" sea menor a 0
    Serial.println(i); //mostrar i
    i = i + 1;      //aumentar i en una unidad, INCREMENTO
  }
}

void loop() {
}
  
```

Subimos nuestro programa a nuestro *Arduino* y la salida del *Monitor serie* es.



Explicando, en cada repetición se muestra el valor actual de *i*, este se va *incrementando* de *uno en uno* debido a $x=x+1$, como el límite del bucle *while* es 3, cuando *i* toma el valor de 3 ya no se cumple la *condición* por lo tanto se sale del *bucle* y hasta ahí queda el programa, prueba variando el límite superior de *i* por un número más grande, compila, abre el *Monitor serie* y observa qué sucede.

Nuestros programas anteriores usan bucles *while* para tener de *entrada* datos, gracias a ellos el programa “*espera*” a que ingreses los datos, **un pequeño hack** que nos ayuda como *entrada* de varios datos por *serial* sin necesidad de métodos más complicados.

ESTRUCTURAS DE CONTROL

(ohmmmmmm)

Entrada: recibir datos del teclado, o de un archivo o de otro aparato.

Salida: mostrar datos en el monitor o enviar datos a un archivo a otro aparato.

Matemáticas: ejecutar operaciones básicas, como la adición y multiplicación.

Operación Condicional: probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.

Repetición: ejecutar alguna acción repetidas veces, usualmente con alguna variación.

Recordando Variables (incremento)

¿Qué sucede cuando ejecutamos lo siguiente?

```
x=1  
x=x+1
```

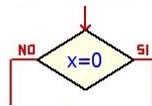
Sentencia Condicional IF

“Si esta condición es cierta, entonces ejecuta estas acciones”

SENTENCIA CONDICIONAL ELSE

Significa

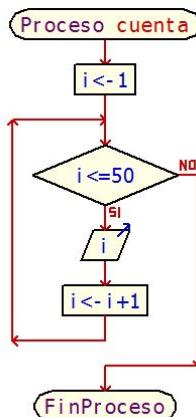
En caso contrario ejecuta esto , tambien significa “o si no”.



SENTENCIAS ITERATIVAS (BUCLE ... LOOP)

26. Muestra los números naturales del 1 al 50.

Este ejercicio es para mostrar en la *salida* los números del 1 al 50 en forma seguida. Nos ayudaremos con el siguiente diagrama de flujo.



El código fuente para nuestro *Arduino*, es el siguiente

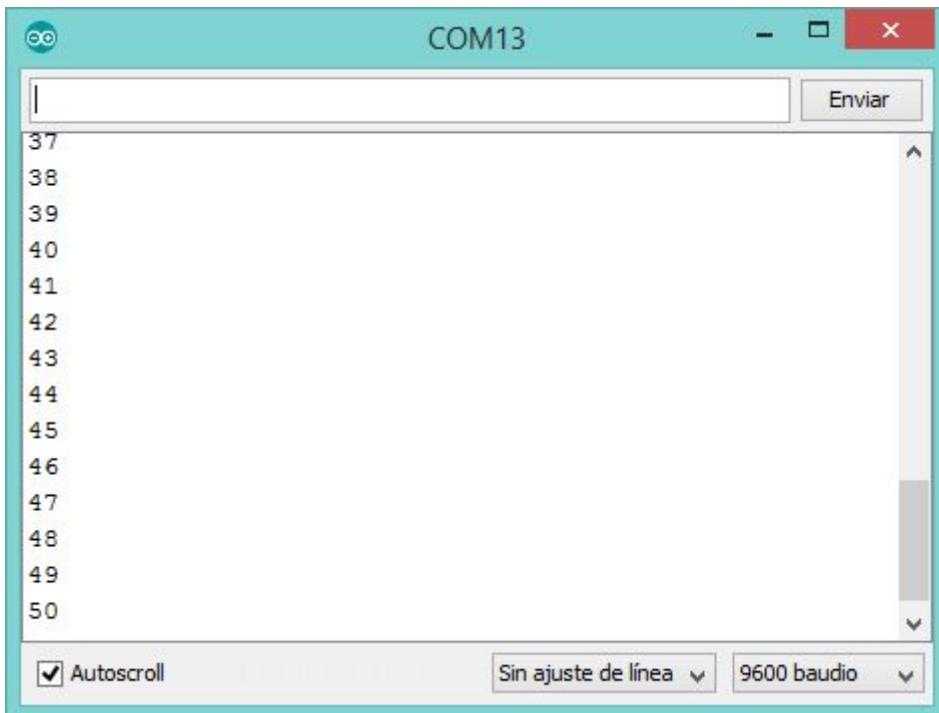
```

int i = 0; //declara i entero
void setup() {
  Serial.begin(9600); //velocidad serial
  i = 1; //i vale 1
  while (i <= 50) { //mientras "i" sea menor o igual a 50
    Serial.println(i); //mostrar i
    i = i + 1; //aumentar i en una unidad INCREMENTO
  }
}

```

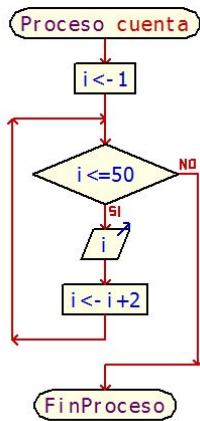
```
void loop() {  
}
```

Después de subir al *Arduino* y abrir el *Monitor serie* tendremos.



27. Muestra los números naturales del 1 al 50 de dos en dos

Para este ejercicio vamos a cambiar el incremento ya no será de uno en uno sino será de dos en dos es decir $x=x+2$, más claro si nos guiamos en el siguiente diagrama de flujo.



El código del programa para nuestro *Arduino* sería el siguiente.

```

int i = 0; //declara i entero
void setup() {
    Serial.begin(9600); //velocidad serial
    i = 1; //i vale 1
    while (i <= 50) { //mientras "i" sea menor o igual a 50
        Serial.println(i); //mostrar i
        i = i + 2; //aumentar i en dos INCREMENTO
    }
}

void loop() {
}
    
```

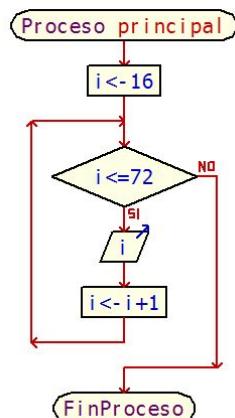
Usando nuestro *monitor serial*, la salida sería la siguiente.

```
25
27
29
31
33
35
37
39
41
43
45
47
49
```

Vemos, que comienza en 1, el siguiente valor será 3 (aumentó en 2) y así sucesivamente hasta que alcance el valor de 49 (menor a 50).

28. Muestra los números naturales del 16 al 72

Para este ejercicio vamos a mostrar en la *salida*, los numero desde un *inicio* 16 hasta un *final* 72 con un *incremento* de 1 en 1. Podemos guiarnos por el siguiente diagrama de flujo.



El programa para nuestro *Arduino* es el siguiente.

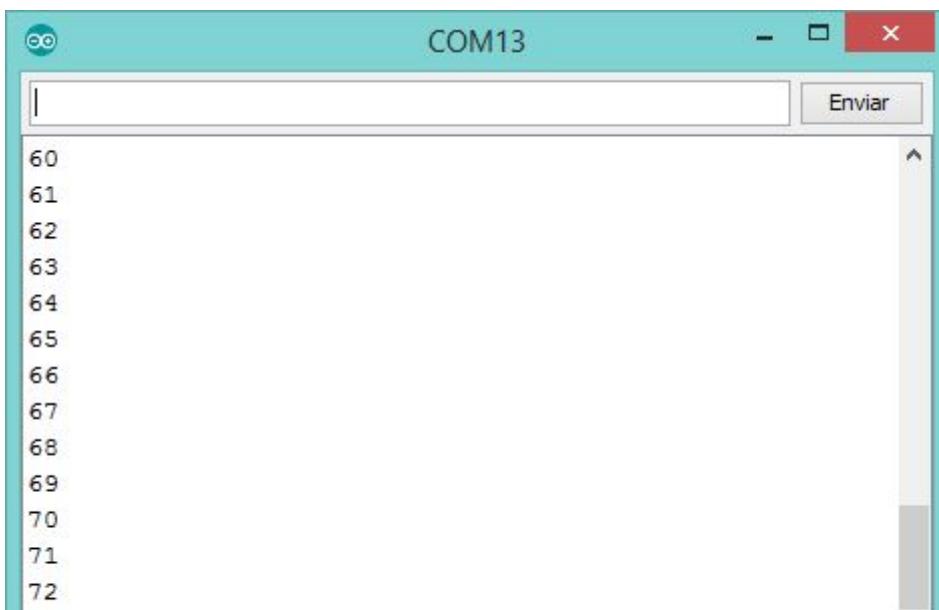
```
int i = 0; //declara i entero
void setup() {
```

```

Serial.begin(9600); //velocidad serial
i = 16; //i vale 16
while (i <= 72) { //mientras "i" sea menor o igual a 72
    Serial.println(i); //mostrar i
    i = i + 1; //aumentar i en 1, INCREMENTO
}
}

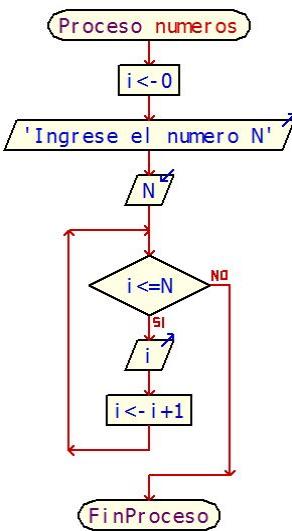
void loop() {
}

```



29. Muestra los números naturales de 0 hasta “n”

En el ejercicio actual vamos a mostrar en *salida* los números naturales desde un *inicio* igual a *0*, hasta un *final* definido por una *entrada* con un valor ingresado por teclado. Como siempre tenemos nuestro diagrama de flujo correspondiente



El código fuente de nuestro programa para nuestro *Arduino* es el siguiente.

```

int i = 0; //declara i contador
int N = 0; //declara N limite superior
void setup() {
  Serial.begin(9600); //velocidad serial
  Serial.println("Imprimo numeros");
  Serial.println("Dame limite superior 'N'");
}

void loop() {
  if (Serial.available() > 0) {
    N = Serial.parseInt();
    while (i <= N) {
      Serial.println(i);
      i = i + 1;
    }
    Serial.println("Dame limite superior 'N'");
  }
}
  
```

Ahora después de subir el programa al *Arduino* y abrir el *Monitor Serie* tendremos.

```

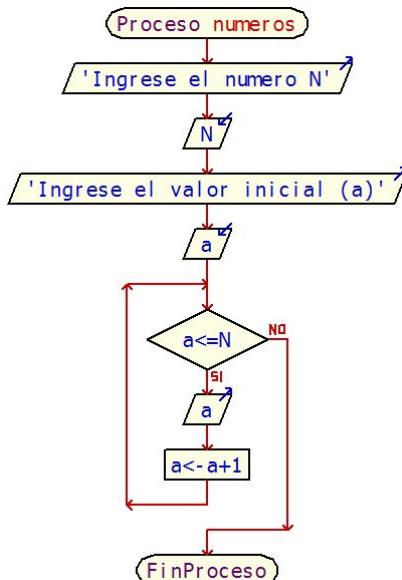
COM13
Enviar

Imprimo numero
Dame limite superior 'N'
0
1
2
3
4
5
6
7
Dame limite superior 'N'

```

30. Muestra números naturales desde un inicio “a”, hasta un final “n”

En este ejercicio vamos a mostrar en *salida* los números naturales desde un *inicio a* hasta un *final n*, pero estos límites van a ser *entradas* por teclado. La guia es el siguiente diagrama de flujo.



El código de nuestro programa en *Arduino* es el siguiente.

```

int a = 0; //declara a limite inferior
int N = 0; //declara N limite superior

```

```

int condicion = 0; //condicion para control de flujo

void setup() {
  Serial.begin(9600); //velocidad serial
  Serial.println("Imprimo numeros");
  Serial.println("Dame inicio 'a'");
}

void loop() {
  while (condicion == 0) { //condicion 0 se entra en este
while y se espera dato
    if (Serial.available() > 0) { //si hay dato en serial
entrar
      a = Serial.parseInt(); //el dato en serial guardar en a
      Serial.print("Inicio es ");
      Serial.println(a); //mostrar a
      Serial.println("Dame final 'N' ");
      condicion = 1; //cambiar condicion para ir al
siguiente bucle while
    }
  }
  while (condicion == 1) { //ahora condicion vale 1 entrar
    if (Serial.available() > 0) { //espera dato en serial
      N = Serial.parseInt(); //el dato en serial es entero
guardar en N
      Serial.print("Final es ");
      Serial.println(N);
      condicion = 2; //cambiar condicion para salir
de bucle
    }
  }
  while (a <= N) { //mientras a sea menor a N
    Serial.print(a); //mostrar a
    Serial.print(";");
    a = a + 1; //contador incremento en 1
  }
}

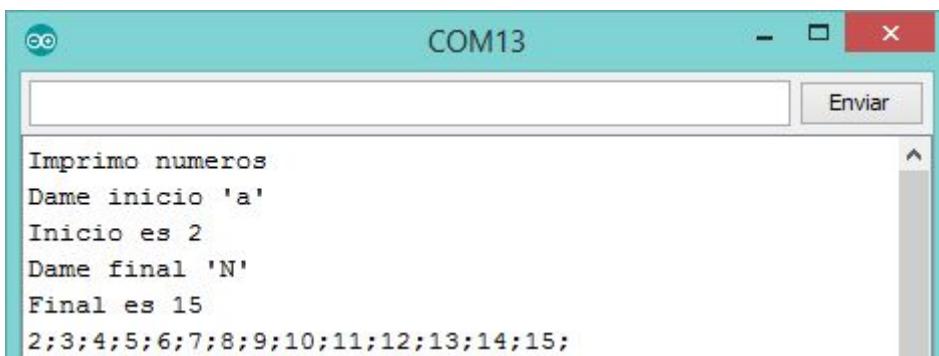
```

```

Serial.println("");
Serial.println("Dame inicio 'a'");
condicion = 0; //cambiar condicion para iniciar
de nuevo en "a"
}

```

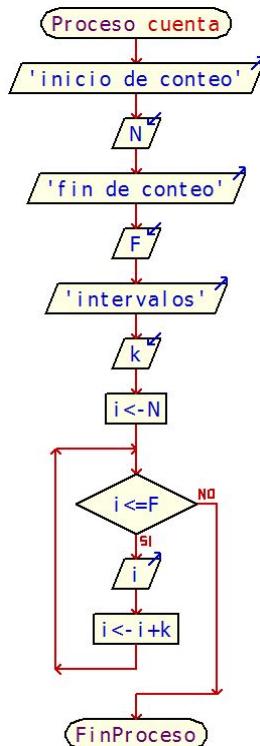
Después de subirlo a nuestro *Arduino* y de abrir nuestro *Monitor Serie* tendríamos.



El ejercicio que acabamos de implementar nos sirve para analizar el comportamiento de *Serial.println()* y *Serial.print()*, son diferentes y dependiendo de lo que necesitemos vamos a usar uno u otro.

31. Hacer un programa que imprima números naturales desde un valor “inicio”, hasta un valor “final” y con intervalos de “intervalo”

Para este ejercicio vamos a mostrar en *salida* los números naturales desde una *entrada* por teclado de *inicio*, *final* e *intervalo*. Vamos a guiarnos por el siguiente *diagrama de flujo*.



El código del presente ejercicio es el siguiente.

```

int inicio = 0; //declara a limite inferior
int fin = 0; //declara N limite superior
int intervalo = 0; //declara intervalo de incremento
int condicion = 0; //condicion para control de flujo

void setup() {
  Serial.begin(9600); //velocidad serial
  Serial.println("Imprimo numeros");
  Serial.println("Dame inicio 'a'");
}

void loop() {
  while (condicion == 0) { //condicion 0 se entra while y
  se espera dato
    if (Serial.available() > 0) { //si hay dato en serial
    entrar
  }
}
  
```

```

    inicio = Serial.parseInt();           //el dato en serial guardar
en a
    Serial.print("Inicio es ");
    Serial.println(inicio);            //mostrar a
    Serial.println("Dame final ");
    condicion = 1;                   //cambiar condicion para ir al
siguiente while
}
}

while (condicion == 1) {           //ahora condicion vale 1 entrar
    if (Serial.available() > 0) { //espera dato en serial
        fin = Serial.parseInt();   //el dato en serial es entero
guardar en N
        Serial.print("Final es ");
        Serial.println(fin);
        Serial.println("Dame Intervalo ");
        condicion = 2;             //cambiar condicion para
siguiente bucle
    }
}

while (condicion == 2) {           //ahora condicion vale 2 entrar
    if (Serial.available() > 0) { //espera dato en serial
        intervalo = Serial.parseInt(); //el dato en serial es
entero guardar en intervalo
        Serial.print("Intervalo es ");
        Serial.println(intervalo);
        condicion = 3;             //cambiar condicion para salir
de bucle
    }
}

while (inicio <= fin) {           //mientras a sea menor a N
    Serial.print(inicio);         //mostrar a
    Serial.print(";");
    inicio = inicio + intervalo; //contador
incremento en 1
}

```

```

Serial.println("");
Serial.println("Dame inicio 'a'");
condicion = 0; //cambiar condicion para iniciar
de nuevo en "a"
}

```

Después de subir nuestro programa en *Arduino*, y de conectarnos con el *Monitor serie*, tendremos lo siguiente.

The screenshot shows the Arduino Serial Monitor window titled "COM13". The window has a teal header bar with the title and standard window controls. The main area is a white text box containing the following text:

```

Imprimo numeros
Dame inicio 'a'
Inicio es 1
Dame final
Final es 55
Dame Intervalo
Intervalo es 2
1;3;5;7;9;11;13;15;17;19;21;23;25;27;29;31;33;35;37;39
Dame inicio 'a'

```

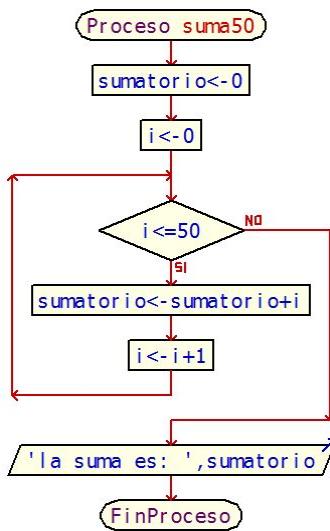
Como podemos ver la única variación con el programa anterior fue el cambio de la *variable* contador por una *variable* intervalo que sea *entrada* por teclado, ahora escribiremos un programa que continúe con *operaciones matemáticas*, el cálculo de sumatorias.

32. Cálculo de sumatorios, Hacer un programa que sume los primeros 50 números naturales.

Para este ejercicio vamos a implementar la *sumatoria* desde un *0* hasta un número *50*, veámoslo más simple.

$$\text{sumatorio}(50) = 0 + 1 + 2 + 3 + 4 + 5 + \dots + 50$$

Vamos a guiarnos del siguiente diagrama de flujo.



El código en *Arduino* sería el siguiente, escribamoslo y a subirlo.

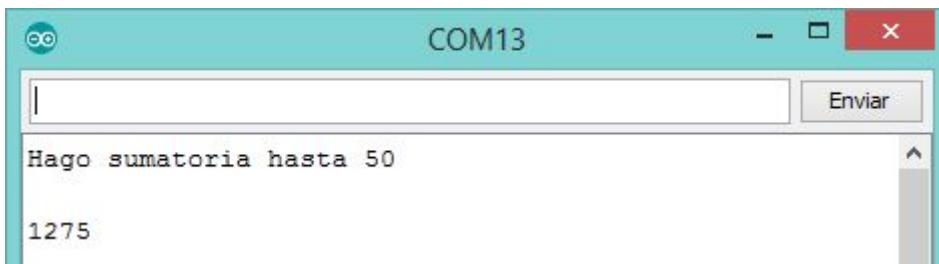
```

int i = 0;          //declara contador i
int sumatorio = 0; //declara el valor de la sumatoria

void setup() {
  Serial.begin(9600); //velocidad serial
  Serial.println("Hago sumatoria hasta 50 ");
  Serial.println("");
  while (i <= 50) {      //contador inicia en 0 hasta 50
    sumatorio = sumatorio + i; //se acumula la suma
    i = i + 1;             //el contador i se incrementa en
                           //1
  }
  Serial.println(sumatorio); //terminado el bucle se muestra
                           //el resultado
}

void loop() {           //loop() no se usa en este
                      //ejercicio
}
  
```

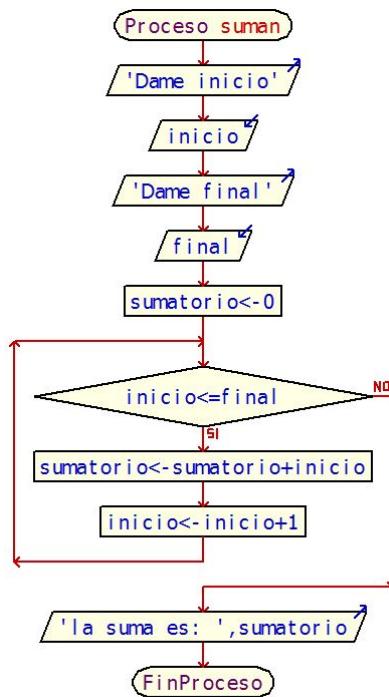
Después de subir el programa a nuestro *Arduino* y de abrir el *Monitor serie*, tendríamos lo siguiente.



Como vimos en este ejercicio, usamos la función *setup()* para solo correr el algoritmo una sola vez, en el siguiente ejercicio vamos hacer la misma sumatoria pero vamos a implementar nuevamente nuestro método para introducir valores por el puerto serial, de tal forma que se pueda ingresar el *inicio* de la sumatoria (desde) y el *final* de la sumatoria (hasta).

33. Hacer un programa que sume números naturales desde “inicio” hasta “final”.

Este ejercicio es parecido al anterior, pero vamos a tener como *entrada* el *inicio* y *final* del proceso de sumatoria, para eso vamos a usar bucles *while* para introducir esos datos, luego se hará las *operaciones matemáticas* de acumulación de la suma mediante un *while* y finalmente se muestra en *salida* la sumatoria y se **pone a cero** el valor de la suma para que no se acumule en nuevas iteraciones del bucle *loop()*. La guía es el siguiente *diagrama de flujo*.



Nuestro código para el *Arduino* sería el siguiente.

```

int inicio = 0; //inicio
int fin = 0; //final
int sumatorio = 0; //declara el valor de la sumatoria
int condicion = 0; //condicion de control de flujo

void setup() {
  Serial.begin(9600); //velocidad serial
  Serial.println("Sumatoria desde inicio a hasta final n ");
  Serial.println("Dame inicio : ");
}

void loop() {
  while (condicion == 0) { // condicion inicial
    if (Serial.available() > 0) {
      inicio = Serial.parseInt(); //entrada de inicio
      Serial.print("inicio = ");
      Serial.println(inicio);
    }
  }
}
  
```

```

Serial.println("Dame final: ");
condicion = 1;           //siguiente condicion
}
}
while (condicion == 1) {
    if (Serial.available() > 0) {
        fin = Serial.parseInt(); //entrada de fin
        Serial.print("fin = ");
        Serial.println(fin);
        condicion = 2;           //salida de while
    }
}

while (inicio <= fin) {           //contador inicio hasta fin
    sumatorio = sumatorio + inicio; //se acumula la suma
    inicio = inicio + 1;           //el contador inicio
se incrementa en 1
}

Serial.println(sumatorio); //terminado el bucle se muestra el
resultado
Serial.println("Dame inicio: ");
sumatorio = 0;           //se resetea la suma para reiniciar el
programa
condicion = 0;           //condicion a cero para reiniciar el
bucle
}

```

Después de subirlo a nuestro *Arduino* y de abrir nuestro *Monitor serie* tendremos lo siguiente.

The screenshot shows a terminal window titled "COM13". The window has a red close button and a grey scroll bar on the right. Inside the window, there is a text input field with a blue border and a "Enviar" (Send) button. Below the input field, the terminal displays the following text:

```

Sumatoria desde inicio a hasta final n
Dame inicio :
inicio = 2
Dame final:
fin = 4
9
Dame inicio:
inicio = 2
Dame final:
fin = 4
9

```

En este ejercicio sino ponemos a cero la sumatoria al final esta se acumularia en la siguiente iteracion del bucle `loop()`, pueden hacer la prueba eliminando la linea.

```

sumatorio = 0;           //se resetea la suma para reiniciar el
programa

```

y ejecutando el programa, verán que el valor de la *sumatoria* va dando valores erróneos esto debido a que esta variable se va acumulando en cada `loop()`.

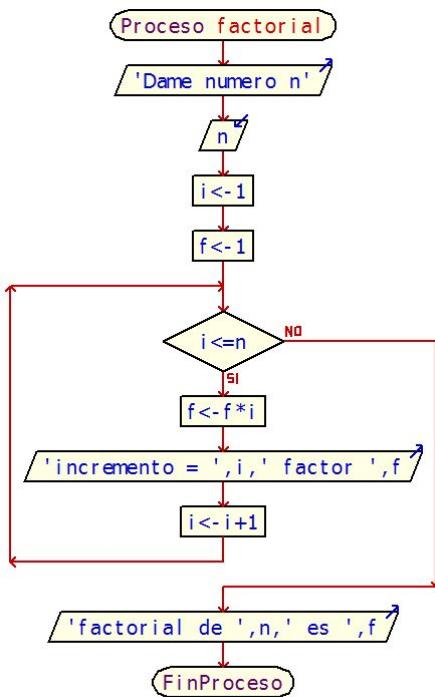
34. Hacer un programa que calcule el factorial de un número “numero”, donde factorial es $n!=1*2*3*...*(n-1)*n$

Para este ejercicio vamos a calcular el factorial de un número, el factorial de un número también se conoce como productoria y está definido como.

$$\text{factorial}(n) = 1 \times 2 \times 3 \times 4 \times \dots \times (n-1) \times n$$

Así que lo que debemos hacer es crear un contador que comience en 1 y que llegue hasta *n*,

además habrá que usar una variable que vaya acumulando el resultado de las multiplicaciones sucesivas. Nos guiaremos del siguiente diagrama de flujo.



El código para este ejercicio es el que sigue a continuación, a escribir código.

```
int n = 0;      //numero a calcular factorial
int i = 1;      //contador inicia en 1
int f = 1;      //valor actual de factorial, 1 por comodidad
int condicion = 0;

void setup() {
    Serial.begin(9600); //velocidad serial
    Serial.println("Factorial hasta n ");
    Serial.println("Dame n : ");
}

void loop() {
    while (condicion == 0) { //condicion inicial
        if (Serial.available() > 0) { //si dato en serial entrar
            n = Serial.parseInt();      //sato en serial es n
            Serial.print("n = ");
        }
    }
}
```

```

Serial.println(n);
condicion = 1;           //cambiar condicion para salir
}
}
while (i <= n) {          //mientras n sea menor a contador
    f = f * i;            //acumula productoria de factorial en
f
    i = i + 1;             //contador incrementandose en 1
}
Serial.print("factorial es ");
Serial.println(f);         //muestra factorial al finalizar el
while anterior
Serial.println("Dame n ");
f = 1;                   //reset al factorial por 1
i = 1;                   //reset al contador por 1
condicion = 0;            //reset de condicion para regresar al
inicio
}

```

Después de subir el programa a nuestro *Arduino* y de abrir el *Monitor serie*, tendremos lo siguiente, por ejemplo para un valor n = 7.



Hay que tomar en cuenta que la multiplicación es un conjunto de sumas y que exige capacidad de cómputo al microcontrolador, ¿Qué sucede cuando usas un valor de n mayor a 7?. ¿Cómo podrías solucionarlo.

El bucle *while* primero analiza la condición, después toma la decisión y finalmente ejecuta código, quiere decir que *si no se cumple la condición no hace nada*.

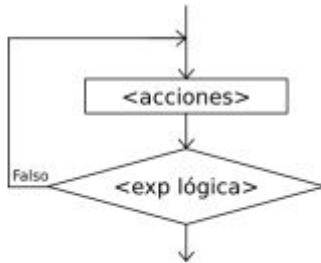
A veces es necesario *hacer algo y después analizar una condición*, de tal manera que se

garantice la ejecución de código **AL MENOS UNA VEZ**. Ese bucle es el bucle *do while*.

EL BUCLE REPETIR HASTA QUE (DO-WHILE)

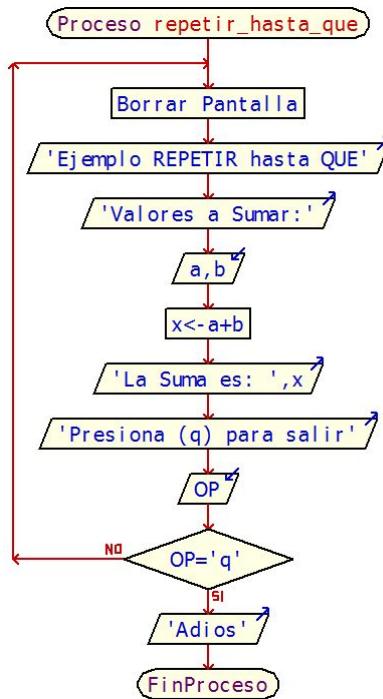
La condicional *Repetir-Hasta Que* ejecuta una secuencia de instrucciones hasta que la condición sea verdadera.

```
Repetir
    <instrucciones>
    Hasta Que <condición>
```



Una diferencia básica con *WHILE* es que la evaluación se evalúa “*al final*” de las “*acciones*” con lo cual se garantiza que se ejecute las acciones **AL MENOS UNA VEZ**.

35. Un programa que suma dos números y del cual solo se salga cuando se presiona una determinada tecla (q)



```

int a = 0;      //numero a
int b = 0;      //numero b
int suma = 0;    //valor actual de suma
int condicion = 0; //para controlar flujo del programa
char comando = 'x';//letra de comando tipo char

void setup() {
  Serial.begin(9600); //velocidad serial

  do                  //inicio de bucle do--while
  {
    Serial.println("Suma de 2 numeros a y b ");
    Serial.println("Dame a : ");

    while (condicion == 0) { //condicion inicial
      if (Serial.available() > 0) { //si dato en serial entrar
        a = Serial.parseInt();      //dato en serial es a
      }
    }
  }
}

void loop() {
  if (condicion != 0) {
    suma = a + b;
    Serial.print("La suma es: ");
    Serial.println(suma);
    condicion = 0;
  }
}
    
```

```
Serial.print("a = ");
Serial.println(a);
Serial.println("Dame b: ");
condicion = 1; //cambiar condicion para salir
}

}

while (condicion == 1) { //condicion actual
if (Serial.available() > 0) { //si dato en serial entrar
b = Serial.parseInt(); //dato en serial es b
Serial.print("b = ");
Serial.println(b);
condicion = 2; //cambiar condicion para salir
}
}

suma = a + b;
Serial.print("suma de a+b es ");
Serial.println(suma); //muestra suma
Serial.println("presione q para salir");
Serial.println("cualquier tecla para continuar");
condicion = 3;
while (condicion == 3) {
if (Serial.available() > 0) { //si dato en serial entrar
comando = Serial.read(); //tipo char en serial es comando
Serial.print("escogio ");
Serial.println(comando);
if (comando == 'q') { //si comando es q entrar
comando = 'q'; //cambiar comando por 'q'
condicion = 4; //cambiar condicion para
salir de bucle
}
if (comando != 'q') { //si es cualquier tecla
Serial.println("Seguimos sumando");
condicion = 0; //regresar al inicio por
condicion cero
}
}
}
```

```

    }
}
}

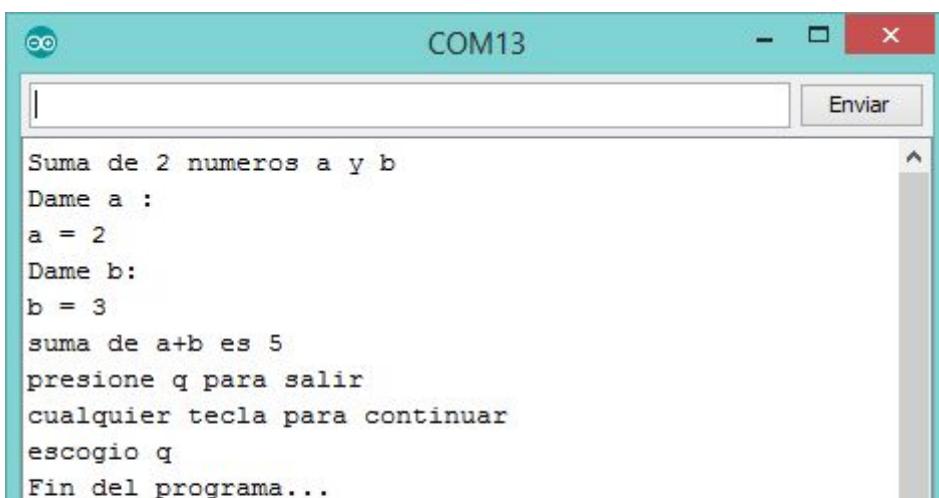
} while (comando != 'q'); //se sale de do-while si comando es
'q'

Serial.println("Fin del programa..."); //adios
}

void loop() { //no usamos loop para que el programa
tenga fin
}

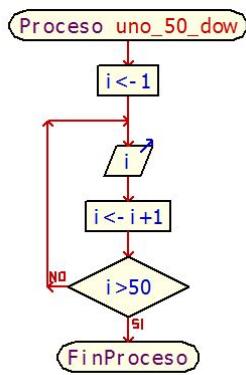
```

Ahora si subimos nuestro programa a nuestro Arduino y abrimos el poderoso *Monitor Serie* tenemos.



Lo más importante de este bucle es que las instrucciones o acciones dentro *AL MENOS SE EJECUTA UNA VEZ* como mínimo.

36. Implemente un programa que imprima números del 0 al 50 usando DO-WHILE



```

int i = 1; //numero i inicia en 1

void setup() {
  Serial.begin(9600); //velocidad serial
  do //inicio de bucle do--while
  {
    Serial.println(i);
    i = i + 1; //contador
  } while (i <= 50); //si i>50 acaba el bucle do-while

  Serial.println("Fin del Proceso"); //sales de bucle y adios
}

void loop() //no usamos loop para que el programa tenga fin
{
}
    
```

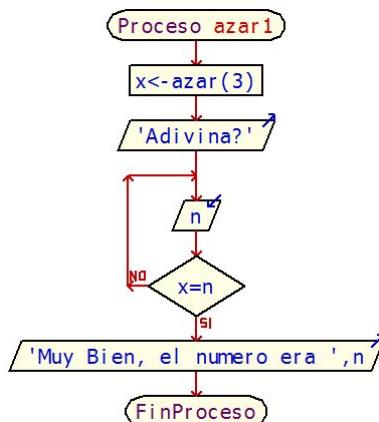
Si subimos el programa a nuestro Arduino y abrimos el Monitor serie tenemos.

```

38
39
40
41
42
43
44
45
46
47
48
49
50
Fin del Proceso
 Autoscroll Sin ajuste de línea ▾ 9600 baudio ▾

```

37. Implemente un juego de “adivinar el número” usando la función AZAR(x) y un bucle DO-WHILE. (USAR 4 números a adivinar)



```
int x = 0;      //numero x que se genera al azar
int n = 0;      //numero que ingresar a la loteria

void setup() {
    Serial.begin(9600); //velocidad serial
    randomSeed(11);     //semilla de azar, cambias este valor para
generar azar
    x = random(3);      //numero al azar
    Serial.println("Adivina hasta 3?");

    do                  //inicio de bucle do--while
    {

        if (Serial.available() > 0) { //si dato en serial se entra
            n = Serial.parseInt();    //dato en serial entero en n
            Serial.println(n);
        }

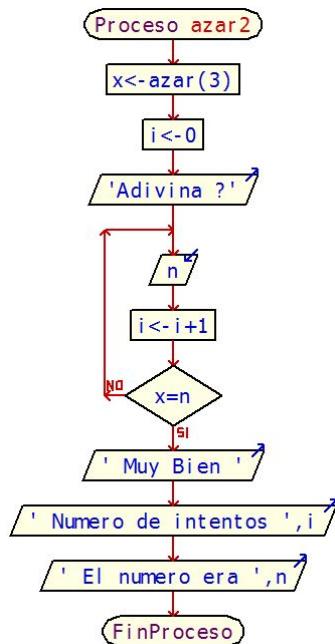
    } while (x != n); //si x es igual a n se sale de bucle do-while

    Serial.print("Adivinaste el numero era "); //sales de bucle y
adios
    Serial.println(x); //sales de bucle y adios
    Serial.println("Prueba cambiando el valor de randomSeed(11)");
}

void loop() {           //no usamos loop para que el programa
tenga fin
}
```

```
Adivina hasta 3?
1
2
Adivinaste el numero era 2
Prueba cambiando el valor de randomSeed(11)
```

38. Implemente un juego de “adivinar el numero” usando la función AZAR(x) y un bucle DO-WHILE, y que muestre el numero de intentos usados para adivinar el numero. (incrementar el número de intentos).



```
int x = 0;      //numero x que se genera al azar
int n = 0;      //numero que ingresar a la loteria
int i = 0;      //numero de intentos usados actualmente en 0

void setup() {
    Serial.begin(9600); //velocidad serial
    randomSeed(11);     //semilla de azar, cambias este valor para
    generar azar
    x = random(3);      //numero al azar
    Serial.println("Adivina hasta 3?");

    do                      //inicio de bucle do--while
    {

        if (Serial.available() > 0) { //si dato en serial se entra
            n = Serial.parseInt();      //dato en serial entero en n
            Serial.println(n);
            i = i + 1;                //aumenta en 1 los intentos
        }

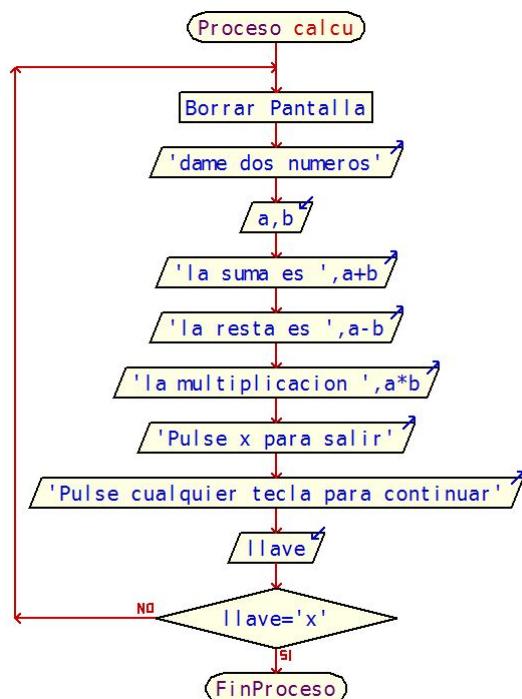
    } while (x != n); //si x es igual a n se sale de bucle do-while

    Serial.print("Adivinaste el numero era "); //sales de bucle y
    adios
    Serial.println(x); //sales de bucle y adios
    Serial.print("Numero de Intentos : ");
    Serial.println(i);
}

void loop() {          //no usamos loop para que el programa
tenga fin
}
```

```
Adivina hasta 3?
1
3
2
Adivinaste el numero era 2
Intentos : 3
```

39. Implementar un programa que sume, reste y multiplique dos números y que se mantenga continuamente operativo sin necesidad de ejecutar el programa nuevamente.



```
int a = 0;          //numero a
int b = 0;          //numero b
```

```
int suma = 0;      //variable de suma
int resta = 0;     //variable de resta
int multiplica = 0; //variable de multiplicacion
char llave = '0';  //numero de intentos usados actualmente en 0
int condicion = 0; //para controlar el flujo del programa

void setup() {
  Serial.begin(9600); //velocidad serial

  do                  //inicio de bucle do--while
  {
    Serial.println("Suma, resta multiplicacion de a y b ");
    Serial.println("Dame a : ");

    while (condicion == 0) { //condicion inicial
      if (Serial.available() > 0) { //si dato en serial entrar
        a = Serial.parseInt();      //dato en serial es a
        Serial.print("a = ");
        Serial.println(a);
        Serial.println("Dame b: ");
        condicion = 1;             //cambiar condicion para salir
      }
    }

    while (condicion == 1) { //condicion actual
      if (Serial.available() > 0) { //si dato en serial entrar
        b = Serial.parseInt();      //dato en serial es b
        Serial.print("b = ");
        Serial.println(b);
        condicion = 2;             //cambiar condicion para salir
      }
    }

    suma = a + b;
    resta = a - b;
    multiplica = a * b;
  }
}
```

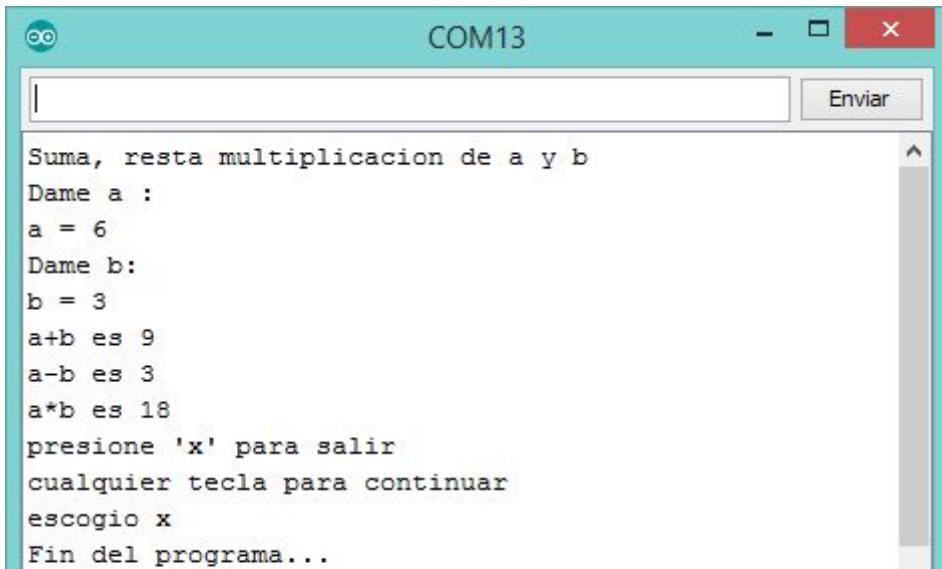
```

Serial.print("a+b es ");
Serial.println(suma);           //muestra suma
Serial.print("a-b es ");
Serial.println(resta);         //muestra resta
Serial.print("a*b es ");
Serial.println(multiplica);     //muestra multiplica
Serial.println("presione 'x' para salir");
Serial.println("cualquier tecla para continuar");
condicion = 3;
while (condicion == 3) {
    if (Serial.available() > 0) { //si dato en serial entrar
        llave = Serial.read();   //tipo char en serial es comando
        Serial.print("escogio ");
        Serial.println(llave);
        if (llave == 'x') {      //si comando es q entrar
            llave = 'x';          //cambiar comando por 'q'
            condicion = 4;          //cambiar condicion para
salir de bucle
        }
        if (llave != 'x') {      //si es cualquier tecla
            Serial.println("Seguimos operando...");
            condicion = 0;          //regresar al inicio por
condicion cero
        }
    }
}
} while (llave != 'x'); //se sale de do-while si llave es 'x'

Serial.println("Fin del programa..."); //adios
}

void loop() {                      //no usamos loop para que el programa
tenga fin
}

```



The screenshot shows a terminal window titled "COM13". The window has a teal header bar with the title and standard window controls (minimize, maximize, close). Below the header is a white text area containing the following text:

```
Suma, resta multiplicacion de a y b
Dame a :
a = 6
Dame b:
b = 3
a+b es 9
a-b es 3
a*b es 18
presione 'x' para salir
cualquier tecla para continuar
escogio x
Fin del programa...
```

40. Implementar un programa de semáforo en puerto serie



```
int i = 0;
void setup() {
    Serial.begin(9600);

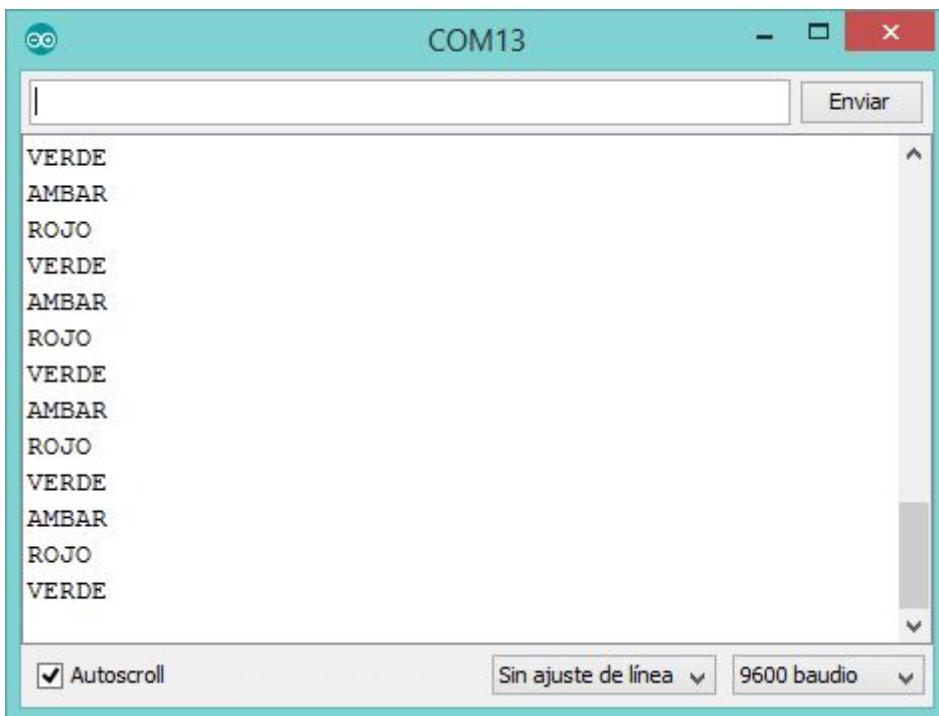
    do          //inicio de bucle do--while
    {
        Serial.println("ROJO"); //semaforo en rojo
        delay(2000);           //por dos segundos

        Serial.println("VERDE"); //semaforo en verde
        delay(2000);           //por dos segundos

        Serial.println("AMBAR"); //semaforo en ambar
        delay(2000);           //por dos segundos

    } while (i == 0); //sale de do-while si i es diferente a cero
                      //imposible
                      //bucle infinito
}

void loop() {          //no usamos loop
}
```



ESTRUCTURAS DE CONTROL

Conocimientos previos

Entrada: recibir datos del teclado, o de un archivo o de otro aparato.

Salida: mostrar datos en el monitor o enviar datos a un archivo a otro aparato.

Matemáticas: ejecutar operaciones básicas, como la adición y multiplicación.

Operación Condicional: probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.

Repetición: ejecutar alguna acción repetidas veces, usualmente con alguna variación.

Recordando Variables (incremento)

¿Qué sucede cuando ejecutamos lo siguiente?

$X=1$

$X=X+1$

EL BUCLE FOR-IN

“Para todo elemento de una serie, hacer...”

FOR variable **IN** serie de valores:

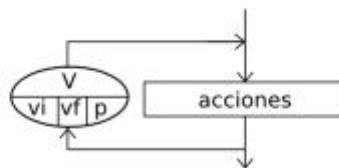
 acción;

 acción;

...

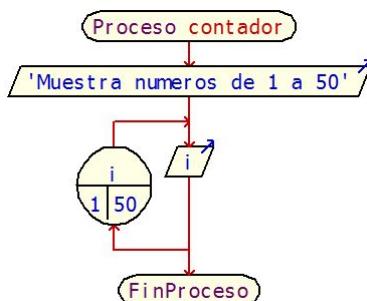
 acción;

La instrucción **Para** ejecuta una secuencia de instrucciones un número determinado de veces.



**Para <variable> <- <inicial> Hasta <final> Con Paso <paso> Hacer
<instrucciones>
FinPara**

41. Imprimir los números desde 1 hasta 50 usando un bucle FOR (Contador).



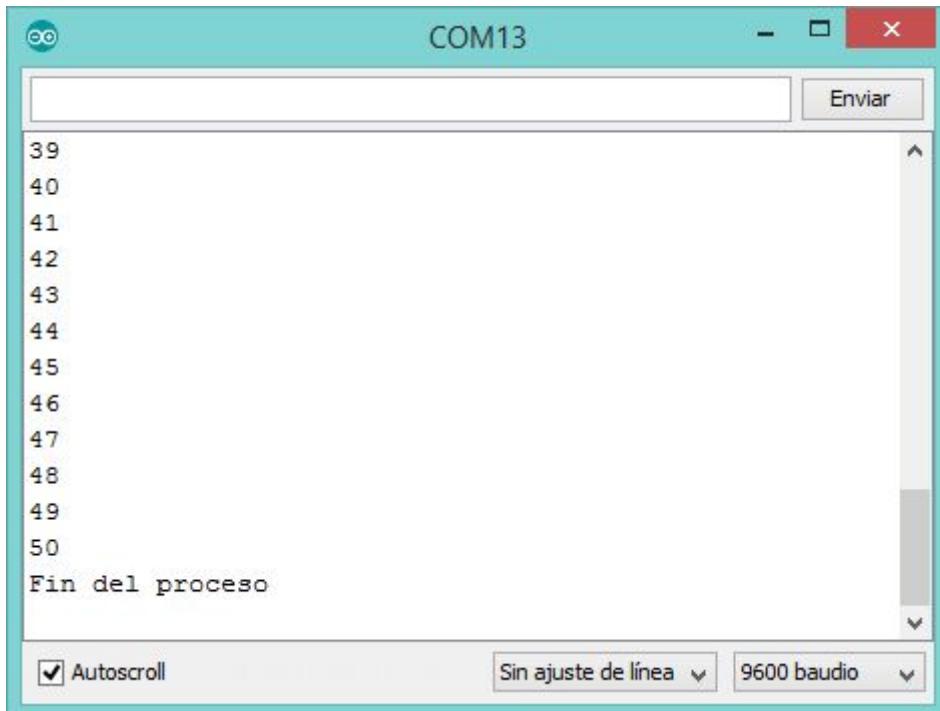
```

void setup() {
    Serial.begin(9600); //velocidad serial 9600
    for (int i = 1; i <= 50; i++) { //desde un inicio i=1 hasta 50
        incrementa i++
        Serial.println(i); //salida numero i
    }
    Serial.println("Fin del proceso"); //al final del for muestra
    este mensaje
}

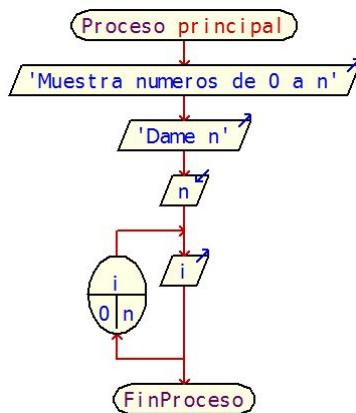
```

```
}
```

```
void loop() { //no usamos loop
}
```



42. Imprimir los números desde “0” hasta “n” usando bucle FOR.



```

int n = 0;
int condicion = 0;

void setup() {
  Serial.begin(9600);           //velocidad serial 9600
  Serial.println("Dame valor de n");

  while (condicion == 0) {      //condicion inicial espera dato
    if (Serial.available() > 0) { //si hay dato en serial
      n = Serial.parseInt();     //guarda en n
      condicion = 1;            //cambia condicion para salir
    }
  }

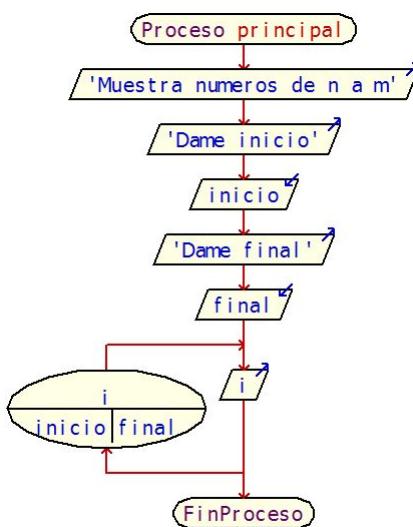
  for (int i = 0; i <= n; i++) { //desde un inicio i=0 hasta 'n'
    Serial.println(i);          //salida numero i
  }

  Serial.println("Fin del proceso"); //al final del for muestra
  este mensaje
}
  
```

```
void loop() { //no usamos loop  
}
```



43. Imprimir los números desde “n” hasta “m” usando bucle FOR.



```

int inicio = 0; //variable inicio
int fin = 0; //variable final
int condicion = 0; // control de flujo

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Dame valor de inicio");

  while (condicion == 0) { //condicion inicial espera dato
    if (Serial.available() > 0) { //si hay dato en serial
      inicio = Serial.parseInt(); //guarda en n
      Serial.println("Dame valor de final");
      condicion = 1; //condicion 1 para ir siguiente bucle
    }
  }

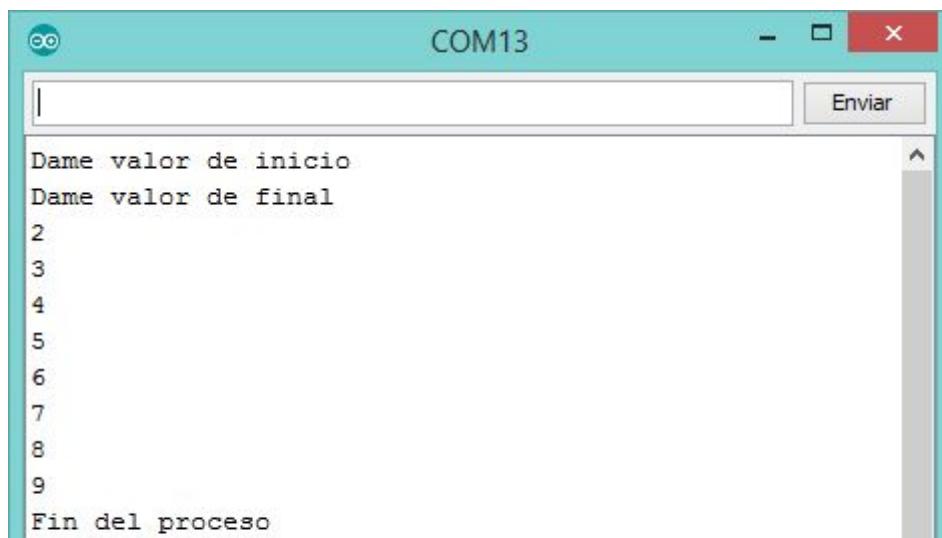
  while (condicion == 1) { //condicion 1 espera dato
    if (Serial.available() > 0) { //si hay dato en serial
      fin = Serial.parseInt(); //guarda en m
    }
  }
}
  
```

```
condicion = 2; //cambia condicion 2 para salir
}
}

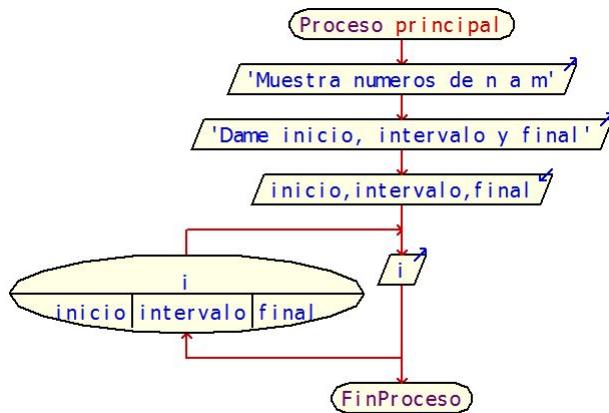
for (inicio; inicio <= fin; inicio++) { //desde un inicio i=0
hasta 'n' incrementa i++
    Serial.println(inicio); //salida numero i
}

Serial.println("Fin del proceso"); //al final del for muestra
este mensaje
}

void loop() { //no usamos loop
}
```



44. Imprimir los números desde “n” hasta “m” con intervalos “i” usando bucle FOR (Contador).



```
int inicio = 0; //variable inicio
int fin = 0; //variable final
int intervalo = 0; //variable intervalo
int condicion = 0; // control de flujo

void setup() {
    Serial.begin(9600); //velocidad serial 9600
```

```

Serial.println("Dame valor de inicio");

while (condicion == 0) { //condicion inicial espera dato
    if (Serial.available() > 0) { //si hay dato en serial
        inicio = Serial.parseInt(); //guarda en inicio
        Serial.println("Dame valor de final");
        condicion = 1; //condicion 1 para ir siguiente
    }
}

while (condicion == 1) { //condicion 1 espera dato
    if (Serial.available() > 0) { //si hay dato en serial
        fin = Serial.parseInt(); //guarda en fin
        Serial.println("Dame valor de intervalo");
        condicion = 2; //cambia condicion 2 para salir
    }
}

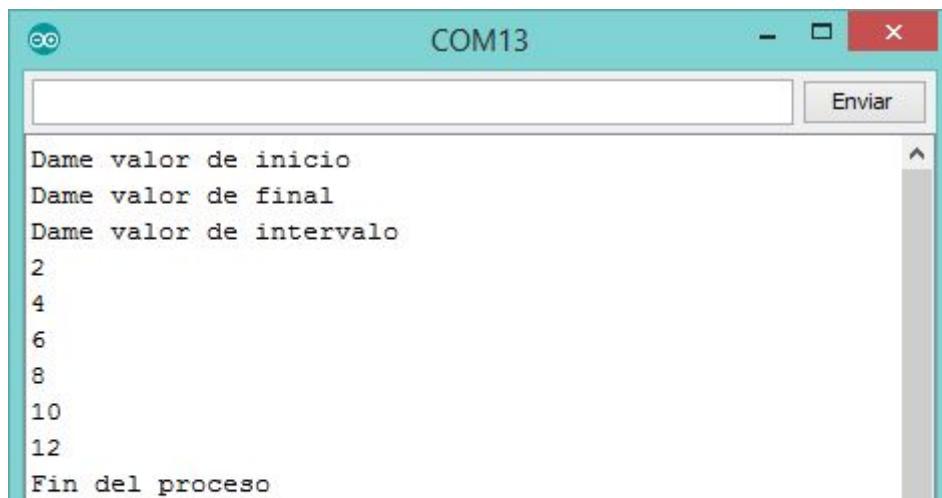
while (condicion == 2) { //condicion 1 espera dato
    if (Serial.available() > 0) { //si hay dato en serial
        intervalo = Serial.parseInt(); //guarda en intervalo
        condicion = 3; //cambia condicion 2 para salir
    }
}

for (inicio; inicio <= fin; inicio = inicio + intervalo) {
    //desde un inicio hasta fin incrementa inicio+intervalo
    Serial.println(inicio); //salida numero i
}

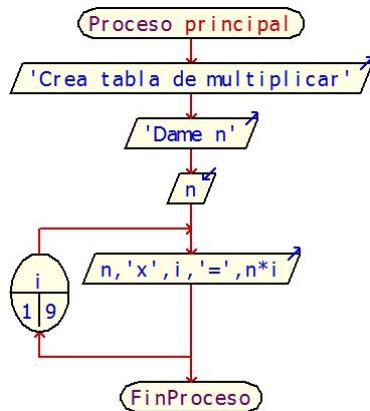
Serial.println("Fin del proceso"); //al final del for muestra
este mensaje
}

```

```
void loop() { //no usamos loop  
}
```



45. Hacer un programa que muestre la tabla de multiplicar de un número “N”



```

int n = 0; //variable de n
int condicion = 0; //condicion para control de flujo

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Crea Tabla de Multiplicar");

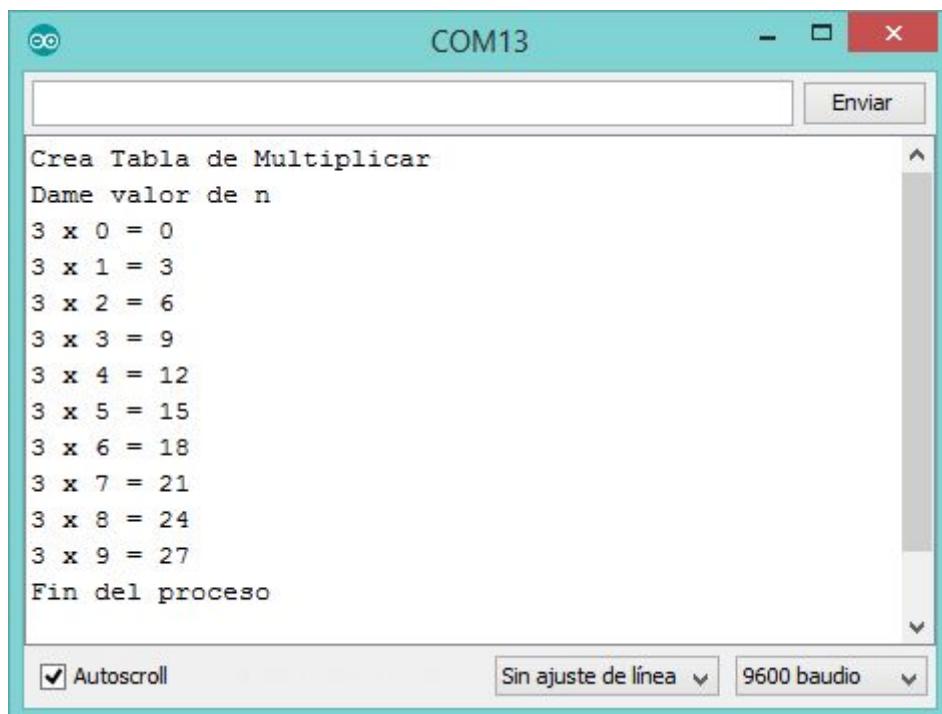
  while (condicion == 0) { //condicion inicial espera dato
    if (Serial.available() > 0) { //si hay dato en serial
      n = Serial.parseInt(); //guarda en n
      Serial.println("Dame valor de n ");
      condicion = 1; //condicion 1 para ir salir
    }
  }

  for (int i = 0; i <= 9; i = i + 1) { //desde un inicio 0 hasta n, incrementa i=i+1
    Serial.print(n); //muestra la tabla
    Serial.print(" x ");
    Serial.print(i);
  }
}
  
```

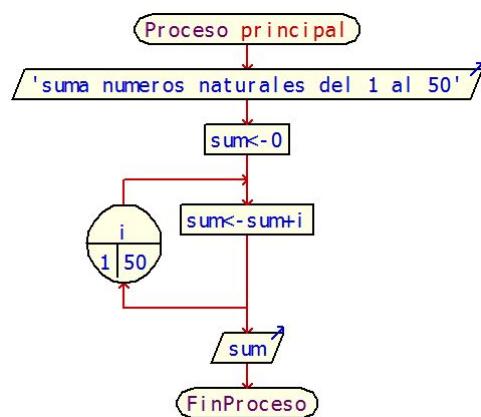
```
Serial.print(" = ");
Serial.println(n * i);           //final de linea con
println
}

Serial.println("Fin del proceso"); //al final del for muestra
este mensaje
}

void loop() {                   //no usamos loop
}
```



46. Hacer un programa que sume los números naturales del 1 al 50 usando FOR

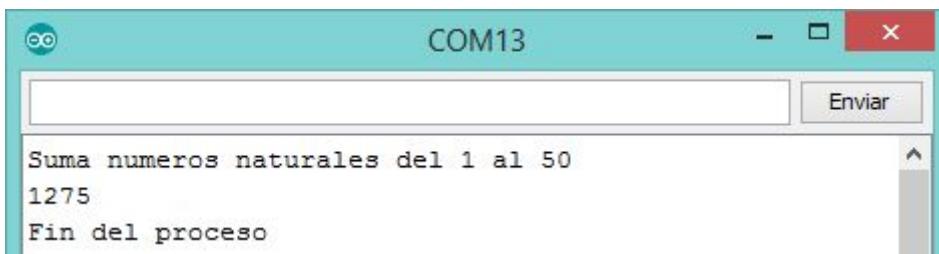


```

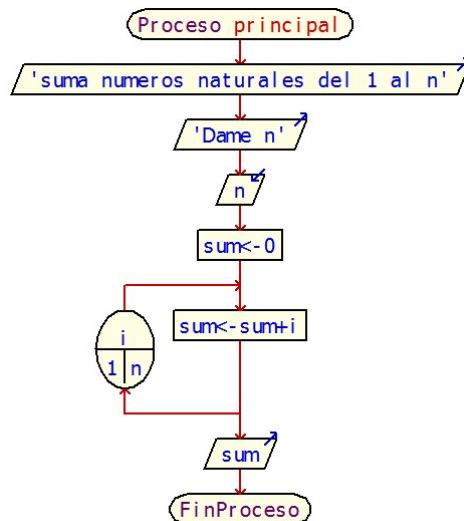
int suma = 0; //suma actual

void setup() {
    Serial.begin(9600);           //velocidad serial 9600
    Serial.println("Suma numeros naturales del 1 al 50");
    for (int i = 1; i <= 50; i = i + 1) { //i desde un inicio 1
        hasta n, incrementa i=i+1
        suma = suma + i;      //acumula suma para cada valor del 0 al
50
    }
    Serial.println(suma);      //salida suma de 0 al 50
    Serial.println("Fin del proceso"); //al final del for muestra
este mensaje
}

void loop() {                  //no usamos loop
}
    
```



47. Hacer un programa que sume los números naturales del 1 al “n” usando FOR



```

int suma = 0; //suma actual
int n = 0; //valor de n
int condicion = 0; //condicion para control de flujo

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Suma numeros naturales del 1 a n");
  Serial.println("Dame valor de n");
  while (condicion == 0) {
  }
}
  
```

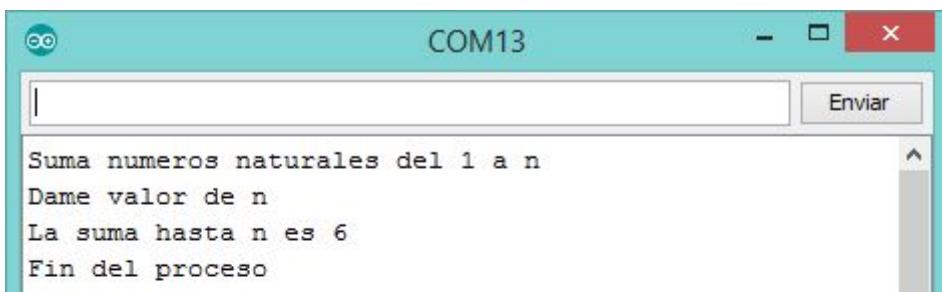
```

if (Serial.available() > 0) {
n = Serial.parseInt();
condicion = 1;
}
}

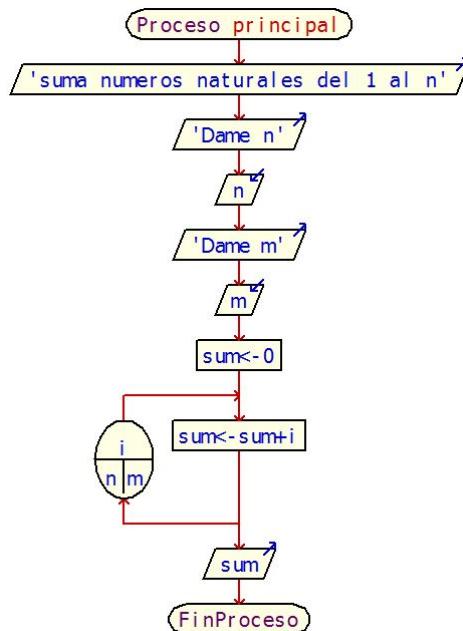
for (int i = 1; i <= n; i = i + 1) { //i desde un inicio 1
hasta n, incrementa i=i+1
suma = suma + i;    //acumula suma para cada valor del 0 al
50
}
Serial.print("La suma hasta n es ");
Serial.println(suma);    //salida suma de 0 al 50
Serial.println("Fin del proceso"); //al final del for muestra
este mensaje
}

void loop() {           //no usamos loop
}

```



48. Hacer un programa que sume los números naturales del “n” al “m” usando FOR



```

int suma = 0; //suma actual
int n = 0; //valor de n
int m = 0; //valor de m
int condicion = 0; //condicion para control de flujo

void setup() {
    Serial.begin(9600);           //velocidad serial 9600
    Serial.println("Suma numeros naturales del 1 a n");
    Serial.println("Dame valor de n");

    while (condicion == 0) {
        if (Serial.available() > 0) {
            n = Serial.parseInt(); //valor de n
            Serial.println("Dame valor de m");
    
```

```
condicion = 1;
}
}

while (condicion == 1) {
    if (Serial.available() > 0) {
        m = Serial.parseInt(); //valor de m
        condicion = 2;
    }
}

for (int i = n; i <= m; i = i + 1) { //i desde un inicio n
hasta m, i=i+1
    suma = suma + i; //acumula suma para cada valor del n al
m
}
Serial.print("La suma de n hasta m es ");
Serial.println(suma); //salida suma
Serial.println("Fin del proceso"); //al final del for muestra
este mensaje
}

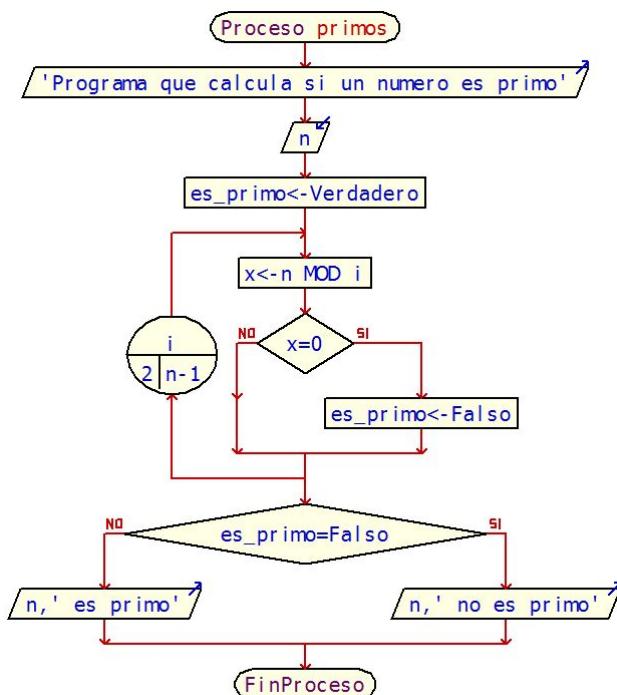
void loop() { //no usamos loop
}
```

```

COM13
Suma numeros naturales del 1 a n
Dame valor de n
Dame valor de m
La suma de n hasta m es 9
Fin del proceso

```

49. Cálculo de números Primos.



```

int n = 0;           //numero n entero
int condicion = 0;   //condicion control de flujo

```

```

bool es_primo = true; //inicialmente es primo "es verdadero"
int x = 0; //variable para iterar

void setup() {
    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Calculo de primos, Dame n");

    while (condicion == 0) { //condicion inicial para entrada de dato
        if (Serial.available() > 0) { //si dato en serial entra
            n = Serial.parseInt(); //valor de n entero
            Serial.println(n); //muestra n
            condicion = 1; //sale de condicion
        }
    }

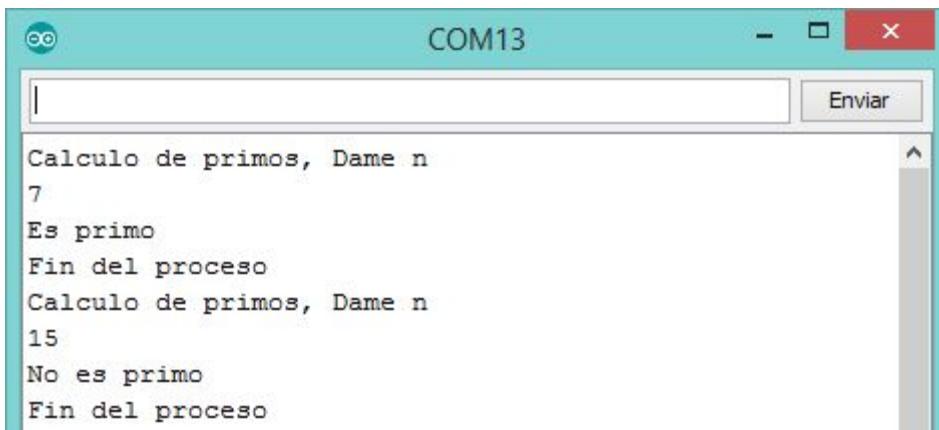
    for (int i = 2; i <= n - 1; i = i + 1) { //i desde 2 hasta n-1, incrementa i=i+1
        x = n % i; //verifica que no haya residuo con MODULO
        if (x == 0) { //si aparece un modulo CERO, no es primo
            es_primo = false; //basta que cumpla una ya no es primo osea "false"
        }
    }

    if (es_primo == false) { //saliendo del for si es false no es primo
        Serial.println("No es primo");
    }
    else {
        Serial.println("Es primo"); //si es true "verdadero" si es primo
    }
}

```

```
Serial.println("Fin del proceso, presiona RESET en Arduino");
}

void loop() { //no usamos loop, presionar RESET en
Arduino
}
```



ALGORITMOS, DIAGRAMAS DE FLUJO, PSEUDOCODIGO (CONTINUACIÓN)

Conocimientos previos

Entrada: recibir datos del teclado, o de un archivo o de otro aparato.

Salida: mostrar datos en el monitor o enviar datos a un archivo a otro aparato.

Matemáticas: ejecutar operaciones básicas, como la adición y multiplicación.

Operación Condicional: probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.

Repetición: ejecutar alguna acción repetidas veces, usualmente con alguna variación.

Forma General de un Algoritmo con Funciones en PSEUDOCODIGO

FUNCIONES

Secuencia de Sentencias que ejecuta alguna operación útil y tiene un nombre definido. Las funciones pueden o no tomar *parámetros*, y pueden entregar o no entregar un *resultado*.

Ejemplo de Funciones (Subprocesos)

```
// función que no recibe ni devuelve nada  
SubProceso Saludar  
    Escribir "Hola mundo!";  
FinSubProceso
```

```
// función que saluda  
SubProceso Saludarconnombre  
    Leer nombre;  
    Escribir "Hola ",nombre;  
FinSubProceso
```

```
// función que recibe un argumento por valor, y devuelve su doble  
SubProceso res <- CalcularDoble(num)  
    res <- num*2; // retorna el doble  
FinSubProceso
```

```
// función que recibe un argumento por referencia, y lo modifica  
SubProceso Triplicar(num por referencia)  
    num <- num*3; // modifica la variable duplicando su valor  
FinSubProceso
```

```
// proceso principal, que invoca a las funciones antes declaradas  
Proceso PruebaFunciones
```

Escribir "Llamada a la función Saludar:";
Saludar; // como no recibe argumentos pueden omitirse los paréntesis vacíos

Escribir "Llamada a la función Saludar con Nombre:";
Saludarconnombre;

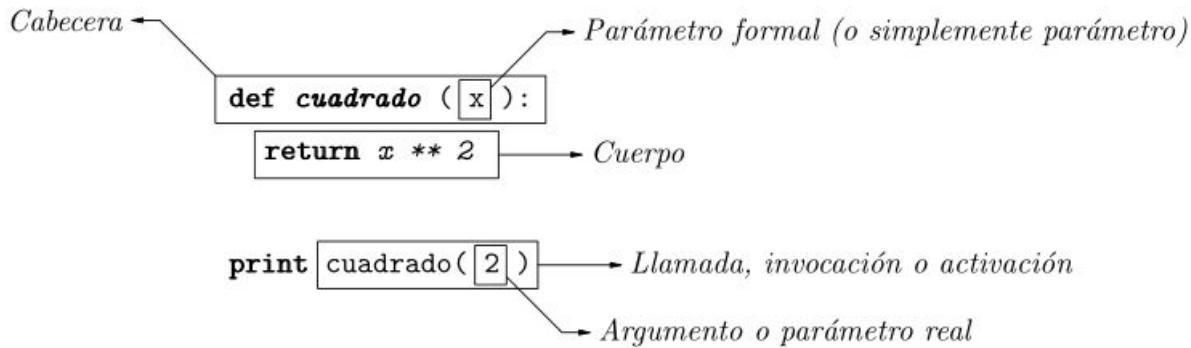
Escribir "Ingrese un valor numérico para x:";
Leer x;

Escribir "Llamada a la función CalcularDoble (pasaje por valor)";
Escribir "El doble de ",x," es ", CalcularDoble(x);
Escribir "El valor original de x es ",x;

Escribir "Llamada a la función Triplicar (pasaje por referencia)";
Triplicar(x);
Escribir "El nuevo valor de x es ", x;

FinProceso

DEFINICIÓN DE FUNCIONES



Definición de Funciones en Arduino.

```

void setup(){
  Serial.begin(9600);
}

void loop() {
  int i = 2;
  int j = 3;
  int k;

  k = myMultiplyFunction(i, j); // k now contains 6
  Serial.println(k);
  delay(500);
}

int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}

```

TIPOS DE FUNCIONES

Función que no recibe ni devuelve nada***SubProceso Saludar******Escribir "Hola mundo!";******FinSubProceso*****Función que recibe un argumento por valor, y devuelve otro valor*****SubProceso res <- CalcularDoble(num)******res <- num*2; // retorna el doble******FinSubProceso*****Función que recibe un argumento por referencia, y lo modifica*****SubProceso Triplicar(num por referencia)******num <- num*3; // modifica la variable duplicando su valor******FinSubProceso***

NOTA, una función puede recibir más de un argumento, dependiendo de su definición.

50. Declaración de una función en Arduino.

```

void setup(){
    Serial.begin(9600); //Serial a 9600
}

void loop() {          //bucle loop
    int i = 2;           //i es entero valor 2
    int j = 3;           //j es entero valor 3
    int k;              //k es entero

    k = miFuncionMultiplicacion(i, j); // k=6, invoca funcion con
    parametros i,j
    Serial.println(k);        // muestra k en serial
    delay(500);           // retardo de medio segundo
}

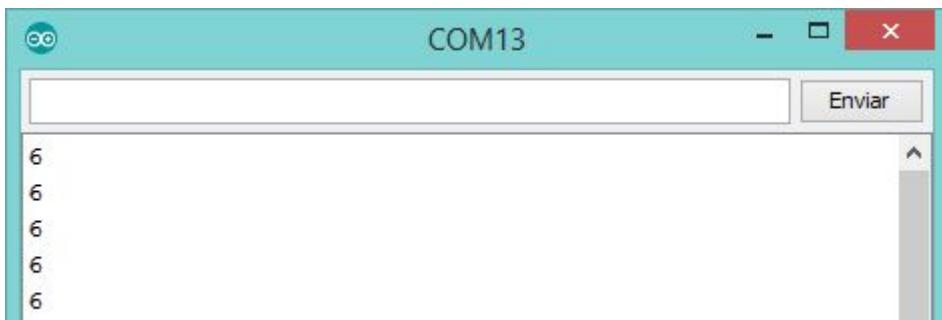
int miFuncionMultiplicacion(int x, int y){ //definicion funcion
    int resultado;         //resultado es entero
}

```

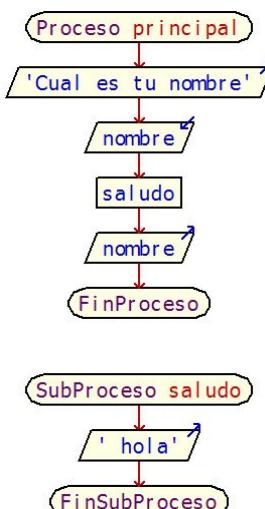
```

resultado = x * y;                                //es producto de los
parametros x y
return resultado;                                 //retorna un valor
resultado
}

```



51. Definir una función que “SALUDE” a la entrada de un nombre. (Función Saludo)



```

int condicion = 0;
String nombre = "alguien";
void setup() {

```

```

Serial.begin(9600);
Serial.println("Cual es tu nombre?");
}

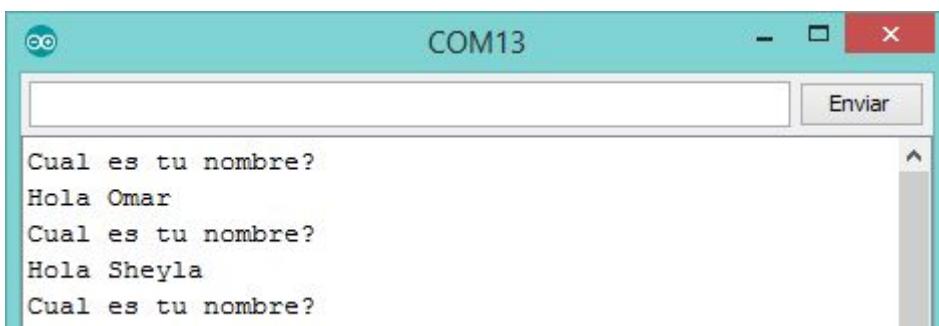
void loop() {

    while (condicion == 0) { //condicion inicial entra es 0
        if (Serial.available() > 0) { //si hay dato en serial
            entra
            nombre = Serial.readString(); //lee el string del serial
            condicion = 1;
        }
    }

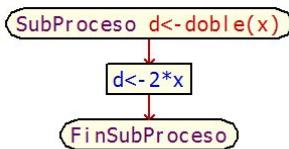
    saludo(); //invoca funcion de saludo
    Serial.println(nombre); //muestra variable nombre
    Serial.println("Cual es tu nombre?"); //repite pide nombre
    condicion = 0; //cambia condicion para recibir nombre con while
}

void saludo() { //define funcion de saludo
    Serial.print("Hola ");
}

```



52. Implementar un programa que calcule el doble de un número, usando funciones.



```

int condicion = 0; //condicion control de flujo
int n = 0; //numero a leer

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Calculo doble, dame n");
}

void loop() {

  while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial
      entra
      n = Serial.parseInt(); //lee el entero del serial
      Serial.println(n); //muestra n
      condicion = 1; //cambia condicion sale de bucle ya se
tiene dato
    }
  }

  Serial.println(doble(n)); //muestra la funcion doble del valor
ingresado
  Serial.println("Calculo doble, dame n");
  condicion = 0; //cambia condicion regresar a bucle
inicial
}

int doble(int x) { //define funcion de doble
  ...
}
  
```

```

int doble;      //tipo de dato de doble
doble = 2 * x;  //operacion de doble
return doble;  //devuelve valor de doble
}

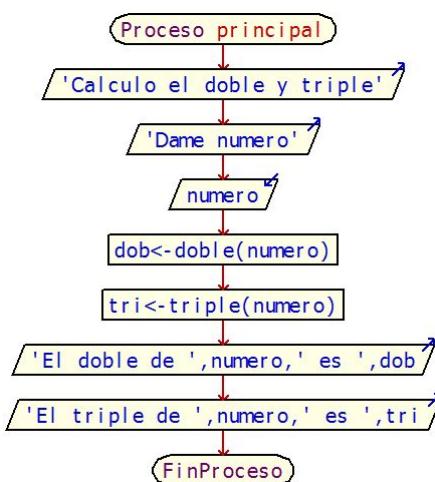
```

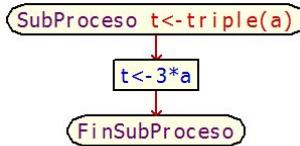
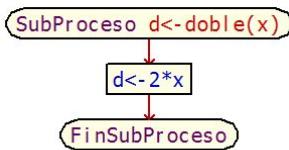
```

Calculo doble, dame n
5
10
Calculo doble, dame n
18
Calculo doble, dame n
153
306
Calculo doble, dame n
99
198
Calculo doble, dame n

```

53. Implementar un programa que calcule el doble y el triple de un número, usando funciones.





```

int condicion = 0; //condicion control de flujo
int n = 0; //numero a leer

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Calculo doble y triple, dame n");
}

void loop() {

  while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial entra
      n = Serial.parseInt(); //lee el entero del serial
      Serial.println(n); //muestra n
      condicion = 1; //cambia condicion sale de bucle ya se tiene dato
    }
  }

  int dob = doble(n);
  int tri = triple(n);
  Serial.print("El doble es "); Serial.println(dob); //muestra la
}
  
```

```
funcion doble del valor ingresado
  Serial.print("El triple es "); Serial.println(tri); //muestra
la funcion doble del valor ingresado
  Serial.println("Calculo doble y triple , dame n");
  condicion = 0;           //cambia condicion regresar a bucle
inicial

}

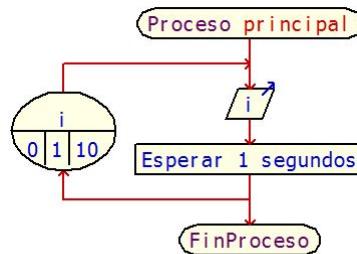
int doble(int x) { //define funcion de doble
  int d;           //tipo de dato de doble
  d = 2 * x;      //operacion de doble
  return d;        //devuelve valor de doble
}

int triple(int a) { //define funcion de doble
  int t;           //tipo de dato de triple
  t = 3 * a;      //operacion de triple
  return t;        //devuelve valor de doble
}
```

```
Calculo doble y triple, dame n
2
El doble es 4
El triple es 6
Calculo doble y triple , dame n
4
El doble es 8
El triple es 12
Calculo doble y triple , dame n
8
El doble es 16
El triple es 24
Calculo doble y triple , dame n
```

Autoscroll Sin ajuste de línea 9600 baudio

54. Implementar un programa que haga un conteo progresivo desde 0 hasta 10 cada 1 segundo



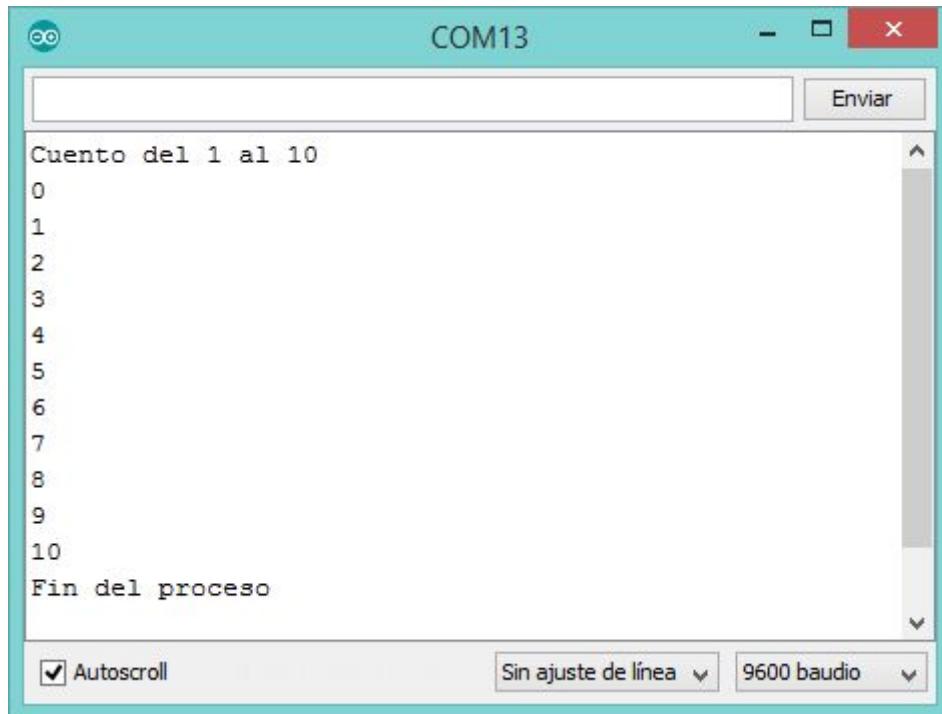
```

void setup() {                                     //funcion setup() se ejecuta una
vez
  Serial.begin(9600);                          //serial a 9600
  Serial.println("Cuento del 1 al 10");

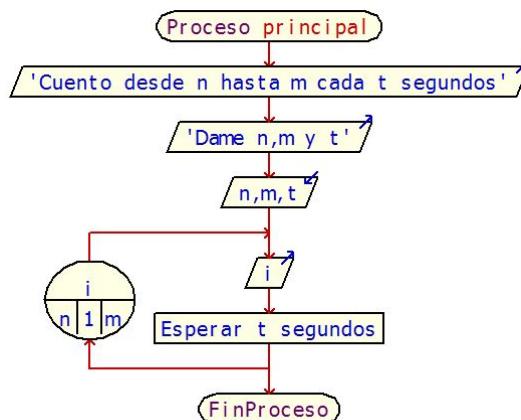
  for (int i = 0; i <= 10; i = i + 1) { //variable i iterar de
for
    Serial.println(i);                         //muestra la variable i
    delay(1000);                            //retraso de 1 seg
  }

  Serial.println("Fin del proceso");
}
  
```

```
void loop() {
}
```



55. Implementar un programa que haga un conteo progresivo desde “n” hasta “m” cada “t” segundo



```
int n = 0;          //declara variable n
int m = 0;          //declara variable m
int t = 0;          //declara variable t
int condicion = 0; //declara condicion inicial

void setup() {                                //funcion setup() se ejecuta una
vez

    Serial.begin(9600);                      //serial a 9600
    Serial.println("Cuento del de 'n' a 'm' cada 't' segundos");
    Serial.println("Dame n");

    while (condicion == 0) {
        if (Serial.available() > 0) {
            n = Serial.parseInt();
            Serial.println("Dame m");
            condicion = 1;
        }
    }

    while (condicion == 1) {
        if (Serial.available() > 0) {
            m = Serial.parseInt();
            Serial.println("Dame t");
            condicion = 2;
        }
    }

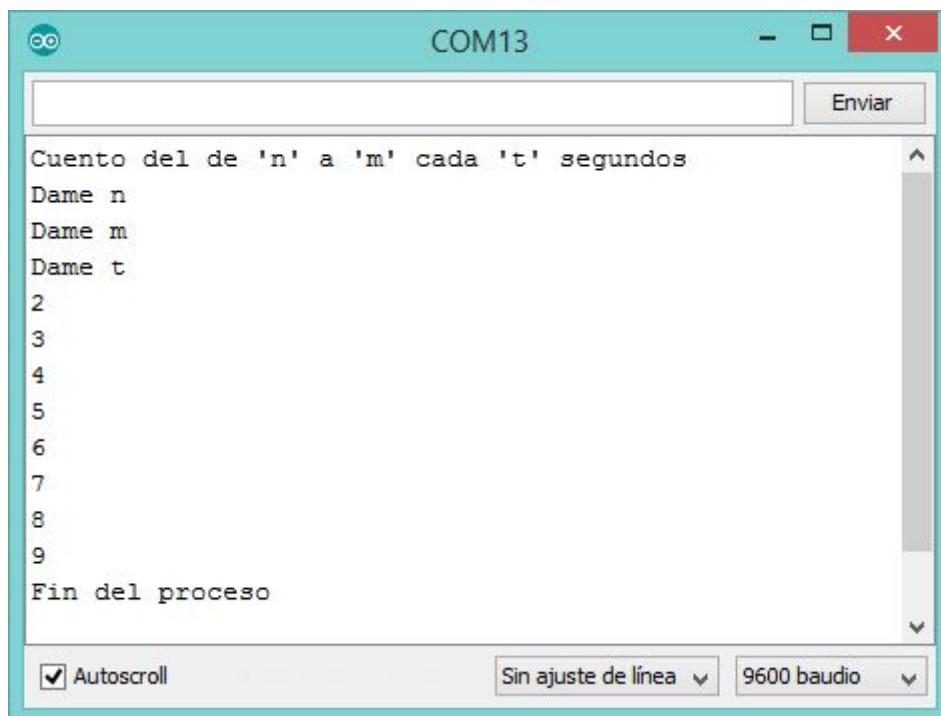
    while (condicion == 2) {
        if (Serial.available() > 0) {
            t = Serial.parseInt();
            condicion = 3;
        }
    }
}
```

```
}

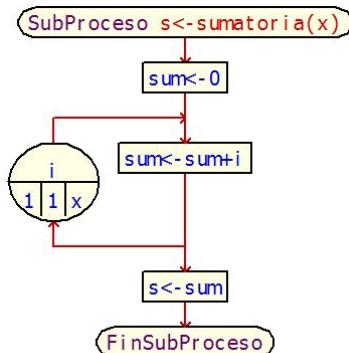
for (int i = n; i <= m; i = i + 1) { //itera for con n m
    Serial.println(i);           //muestra la variable i
    delay(t*1000);             //por mil ya que es milisegundos
}

Serial.println("Fin del proceso");
}

void loop() { //no la usamos ahora
}
```



56. Implementar un programa que haga sumatorias usando funciones



```

int condicion = 0; //condicion control de flujo
int n = 0; //numero a leer

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Calculo sumatorias usando funciones");
  Serial.println("Dame n");
}

void loop() {
}
    
```

```
while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial
        entra
        n = Serial.parseInt(); //lee el entero del serial
        Serial.println(n); //muestra n
        condicion = 1; //cambia condicion sale de bucle ya se
tiene dato
    }
}

Serial.println(sumatoria(n)); //muestra la funcion doble del
valor ingresado
Serial.println("Calculo sumatorias usando funciones");
Serial.println("Dame n");
condicion = 0; //cambia condicion regresar a bucle
inicial

}

int sumatoria(int x) { //define funcion de sumatoria
    int sumatoria = 0; //tipo de dato de sumatoria

    for (int i = 1; i <= x; i = i + 1) { //for para iterar la suma
        sumatoria = sumatoria + i; //se acumula la sumatoria
    }

    return sumatoria; //devuelve valor de sumatoria
}
```

```

COM13
Enviar

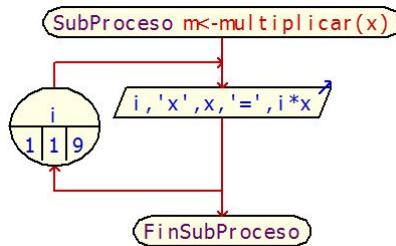
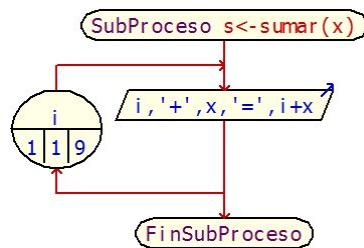
Calculo sumatorias usando funciones
Dame n
6
21
Calculo sumatorias usando funciones
Dame n
45
1035
Calculo sumatorias usando funciones
Dame n
99
4950
Calculo sumatorias usando funciones
Dame n

```

Autoscroll Sin ajuste de línea 9600 baudio

57. Implementar un programa de tabla de multiplicar y tabla de sumar usando funciones





```

int condicion = 0; //condicion control de flujo
int n = 0; //numero a leer

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Tabla de sumas y multiplicaciones");
  Serial.println("Dame n");
}

void loop() {

  while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial
      entra
      n = Serial.parseInt(); //lee el entero del serial
      Serial.println(n); //muestra n
      condicion = 1; //cambia condicion sale de bucle ya se
tiene dato
    }
  }
}
  
```

```
}

sumar(n);      //invoca funcion tabla de +
multiplicar(n); //invoca funcion tabla de x

Serial.println("Tabla de sumas y multiplicaciones");
Serial.println("Dame n");
condicion = 0;           //cambia condicion regresar a bucle
inicial

}

int sumar(int x) { //define funcion de sumar

for (int i = 1; i <= 9; i = i + 1) {
    Serial.print(x); Serial.print(" + "); Serial.print(i);
    Serial.print(" = "); Serial.println(x + i); //arma la tabla
}

}

int multiplicar(int y) { //define funcion de multiplicar

for (int i = 1; i <= 9; i = i + 1) {
    Serial.print(y); Serial.print(" x "); Serial.print(i);
    Serial.print(" = "); Serial.println(y * i); //arma la tabla
}

}
```

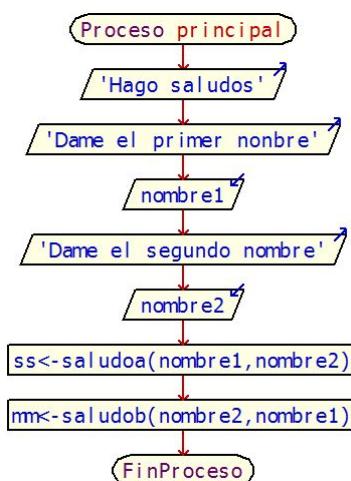
The screenshot shows a terminal window titled "COM13" with the following text output:

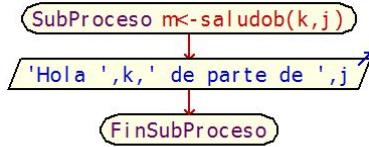
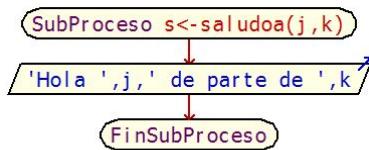
```

Tabla de sumas y multiplicaciones
Dame n
5
5 + 1 = 6
5 + 2 = 7
5 + 3 = 8
5 + 4 = 9
5 + 5 = 10
5 + 6 = 11
5 + 7 = 12
5 + 8 = 13
5 + 9 = 14
5 x 1 = 5
5 x 2 = 10
  
```

At the bottom of the window, there are three buttons: "Autoscroll" (checked), "Sin ajuste de línea" (unchecked), and "9600 baudio" (selected).

58. Se tienen dos nombres de entrada, Definir una función que haga que un nombre salude a el otro nombre y viceversa (2 funciones).





```

int condicion = 0; //condicion control de flujo
String nombre1 = "0"; //primer nombre
String nombre2 = "0"; //segundo nombre

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Hago saludos");
  Serial.println("Dame nombre1");
}

void loop() {

  while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial entra
      nombre1 = Serial.readString(); //lee el texto del serial
      Serial.println(nombre1); //muestra texto
      Serial.println("dame nombre2"); //pide nombre 2
      condicion = 1; //cambia condicion sale de bucle ya se
tiene dato
    }
  }

  while (condicion == 1) { //condicion inicial entra es 1
    if (Serial.available() > 0) { //si hay dato en serial entra
      nombre2 = Serial.readString(); //lee el texto del serial
    }
  }
}
  
```

```
Serial.println(nombre2); //muestra texto
condicion = 2; //cambia condicion sale de bucle ya se
tiene dato
}
}

saludoa(nombre1, nombre2); //invoca funcion de saludo a
saludob(nombre1, nombre2); //invoca funcion de saludo b
Serial.println("dame nombre1"); //pide nombre1
condicion = 0; //cambia condicion para reiniciar
programa

}

void saludoa(String x, String y) { //definicion de funcion de
saludo a
    Serial.print("Hola "); Serial.print(x); Serial.print(" de parte
de ");
    Serial.println(y);
}

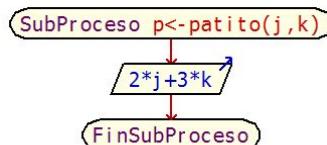
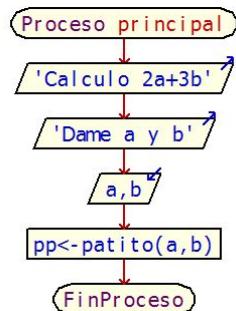
void saludob(String x, String y) { //definicion de saludo b
    Serial.print("Hola "); Serial.print(y); Serial.print(" de parte
de ");
    Serial.println(x);
}
```

```

Hago saludos
Dame nombre1
Omar
dame nombre2
Sheyla
Hola Omar de parte de Sheyla
Hola Sheyla de parte de Omar
dame nombre1

```

59. Implementar un programa que calcule la suma del doble del primero por el triple del segundo. A esa operación la llamaremos “patito”.



```

int condicion = 0; //condicion control de flujo
int a = 0; //primer a
int b = 0; //segundo b

```

```
void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Calculo 2*a + 3*b");
  Serial.println("Dame a");
}

void loop() {

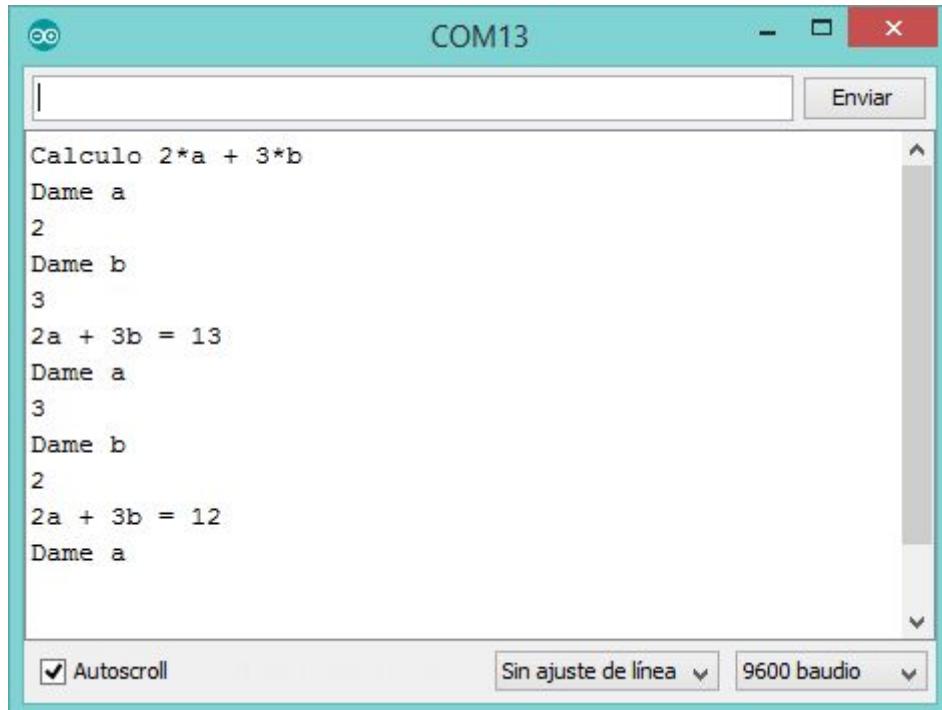
  while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial entra
      a = Serial.parseInt(); //lee el entero del serial
      Serial.println(a); //muestra a
      Serial.println("Dame b"); //pide nombre 2
      condicion = 1; //cambia condicion sale de bucle ya se
tiene dato
    }
  }

  while (condicion == 1) { //condicion inicial entra es 1
    if (Serial.available() > 0) { //si hay dato en serial entra
      b = Serial.parseInt(); //lee el entero del serial
      Serial.println(b); //muestra b
      condicion = 2; //cambia condicion sale de bucle ya se
tiene dato
    }
  }

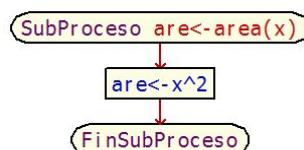
  Serial.print("2a + 3b = ");
  Serial.println(patito(a, b));
  Serial.println("Dame a");
  condicion = 0;
}

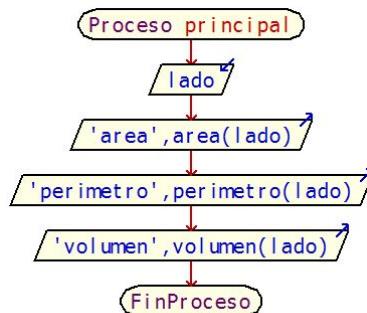
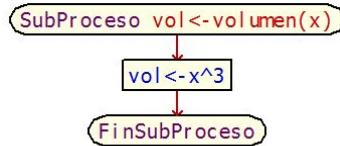
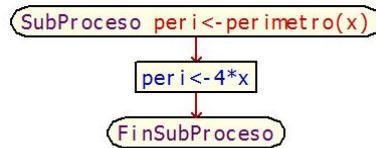
int patito(int j, int k) { //definicion de funcion de saludo a
  int patito = 2 * j + 3 * k;
```

```
return patito;
}
```



60. Implementar un programa que calcule el área, perímetro, volumen de un cuadrado, y cubo usando solo el lado como entrada.





```

int condicion = 0; //condicion control de flujo
int lado = 0; //lado cuadrado

void setup() {
  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Calculo area, perimetro y volumen");
  Serial.println("Dame lado");
}

void loop() {

  while (condicion == 0) { //condicion inicial entra es 0
    if (Serial.available() > 0) { //si hay dato en serial entra
      lado = Serial.parseInt(); //lee el entero del serial
      Serial.println(lado); //muestra lado
    }
  }
}
  
```

```
    condicion = 1; //cambia condicion sale de bucle ya se
tiene dato
}
}

Serial.print("Area "); Serial.println(area(lado));
Serial.print("Perimetro "); Serial.println(perimetro(lado));
Serial.print("Volumen "); Serial.println(volumen(lado));
Serial.println("Dame lado");
condicion = 0;

}

int area(int x) { //definicion de funcion de area
int area = x * x;
return area;
}

int perimetro(int x) { //definicion de funcion de perimetro
int perimetro = 4 * x;
return perimetro;
}

int volumen(int x) { //definicion de funcion de volumen
int volumen = x * x * x;
return volumen;
}
```

The screenshot shows a Windows-style serial monitor window titled "COM13". The window has a teal header bar with the title and standard window controls (minimize, maximize, close). Below the header is a text input field and a "Enviar" (Send) button. The main body of the window contains the following text output:

```
Calculo area, perimetro y volumen
Dame lado
3
Area 9
Perimetro 12
Volumen 27
Dame lado
4
Area 16
Perimetro 16
Volumen 64
Dame lado
```

At the bottom of the window, there are three settings: a checked checkbox for "Autoscroll", a dropdown for "Sin ajuste de línea" (No line adjustment), and another dropdown for "9600 baudio" (9600 baud).

61. Implementar un programa que haga un conteo progresivo desde 0 hasta 10 cada 1 segundo y que luego haga el conteo de 10 a 1 cada segundo (efecto auto fantástico) y que se repita continuamente.

0 (espera un seg)

1 (espera un seg)

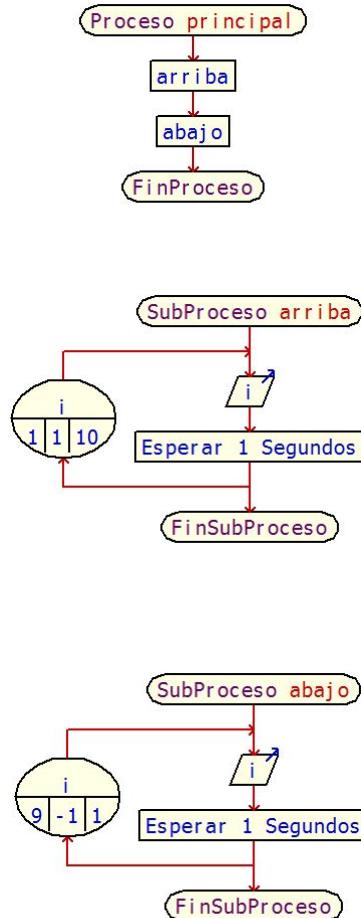
2 (espera un seg)

...

7 (espera un seg)

8 (espera un seg)

9 (espera un seg)
 10 (espera un seg)
 9 (espera un seg)
 8 (espera un seg)
 7 (espera un seg)
 ...
 2 (espera un seg)
 1 (espera un seg)
 0 (espera un seg)



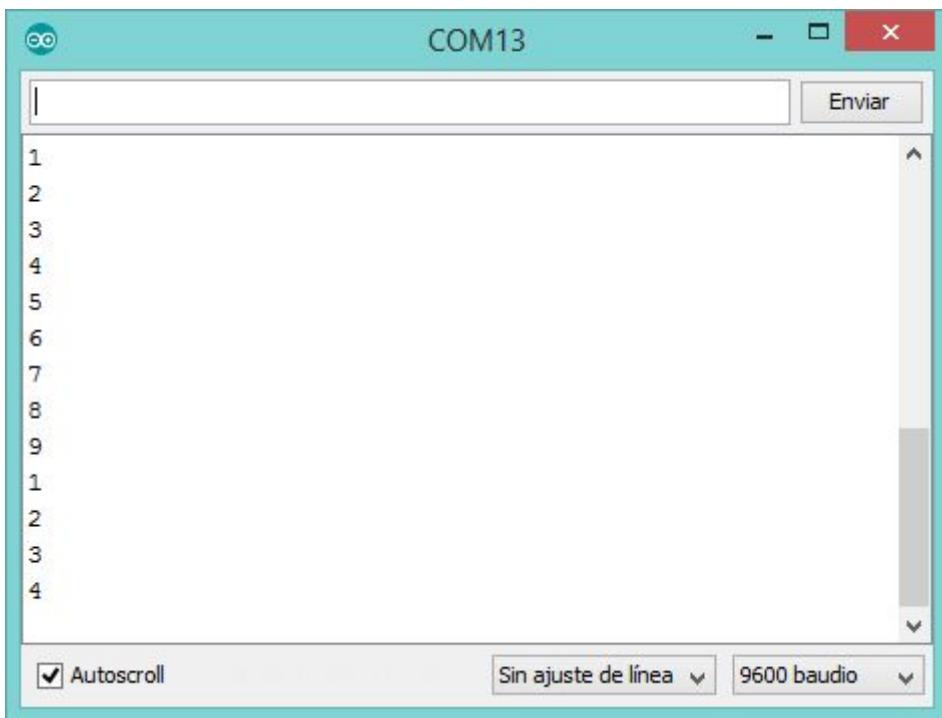
```

void setup() {
  Serial.begin(9600); //velocidad serial 9600
}
  
```

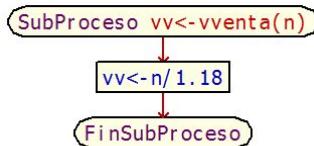
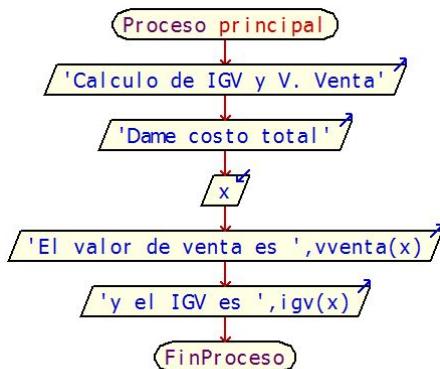
```
void loop() { //repite continuamente
    arriba(); //invoca funcion arriba
    abajo(); //invoca funcion abajo
}

void arriba() { //definicion de funcion de arriba
    for (int i = 1; i <= 9; i = i + 1) { //for incremental
        Serial.println(i); //muestra i
        delay(1000); //retraso de 1 segundo
    }
}

void abajo() { //definicion de funcion de abajo
    for (int i = 9; i <= 1; i = i - 1) { //for decremental
        Serial.println(i); //muestra i
        delay(1000); //retraso de 1 segundo
    }
}
```



62. Implementar un programa que reciba el gasto completo de una compra y calcule el IGV, Valor de venta usando funciones.



```

int condicion = 0;
float x = 0;

void setup() {
    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Calculo IGV y Valor de Venta");
    Serial.print("Dame costo total");
}

void loop() { //repite continuamente
    while (condicion == 0) {
        if (Serial.available() > 0) {
    
```

```
x = Serial.parseFloat();
Serial.println(x);
condicion = 1;
}

Serial.print("Valor de venta = ");
Serial.println(vventa(x)); //imprime la invocacion de vventa()

Serial.print("IGV = ");
Serial.println(igv(x));    //imprime la invocacion de igv()

Serial.println("Dame costo total");
condicion = 0;           //reinicia la condicion para pedir dato
}

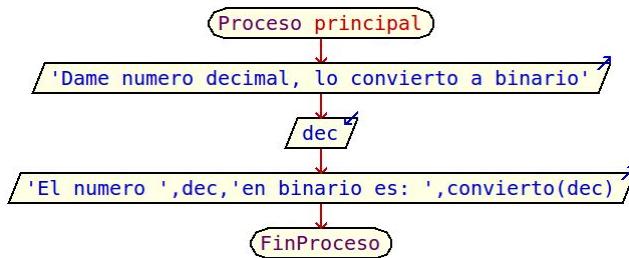
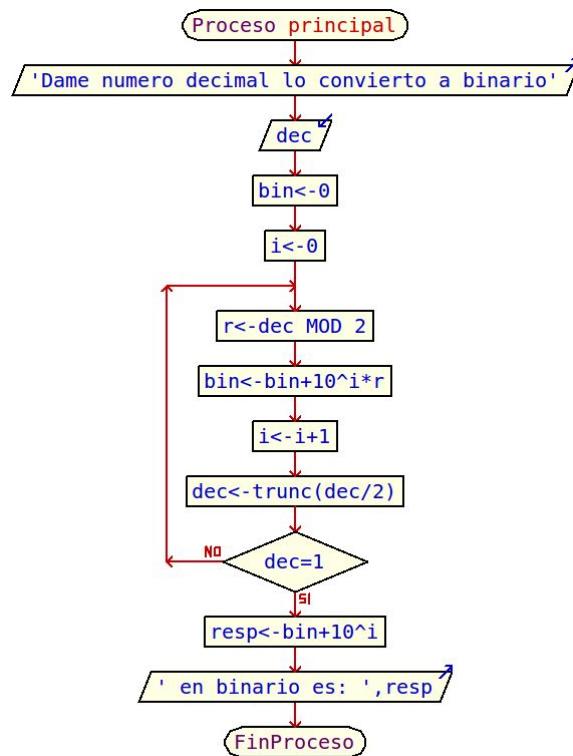
float vventa(float n) {   //definicion de funcion de vventa
  float vventa = n / 1.18; //valor de venta es costo/1.18
  return vventa;
}

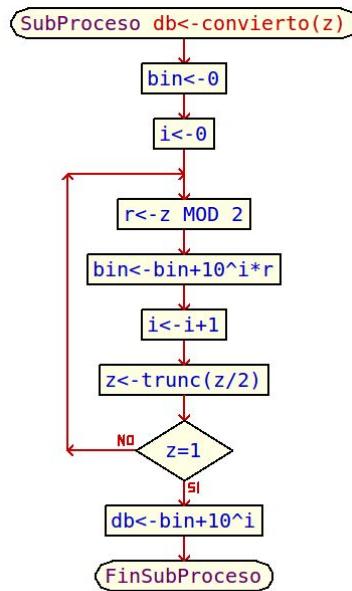
float igv(float n) {     //definicion de funcion de igv
  float igv = vventa(n) * 0.18; //igv es valor de venta * 0.18
  return igv;
}
```

The screenshot shows a terminal window titled "COM13". The window has a light blue header bar with the title and standard window controls (minimize, maximize, close). Below the header is a text input field and a "Enviar" (Send) button. The main area contains the following text output:

```
Calculo IGV y Valor de Venta
Dame costo total11.00
Valor de venta = 9.32
IGV = 1.68
Dame costo total
100.00
Valor de venta = 84.75
IGV = 15.25
Dame costo total
```

63. Implementar un programa de conversión de decimal a binario usando funciones





```

int condicion = 0; //control de flujo
int dec = 0; //declaracion variable decimal

void setup() {

  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Convierzo a binario");
  Serial.println("Dame numero decimal ");

}

void loop() { //repite continuamente

  while (condicion == 0) { //condicion inicial

    if (Serial.available() > 0) { //si dato en serial entra
      dec = Serial.parseInt(); //entra entero como dec
      Serial.println(dec); //muestra valor de dec
      condicion = 1; //sale de condicion
    }
  }
}
  
```

```
}

Serial.print("En binario es: ");
float salida = convierte(dec); //se invoca la conversion
Serial.println(salida);      //se muestra la conversion en
float
Serial.println("Dame numero decimal");
condicion = 0;              //reinicia la condicion para pedir dato
}

float convierte(int z) {    //definicion de funcion de conversion

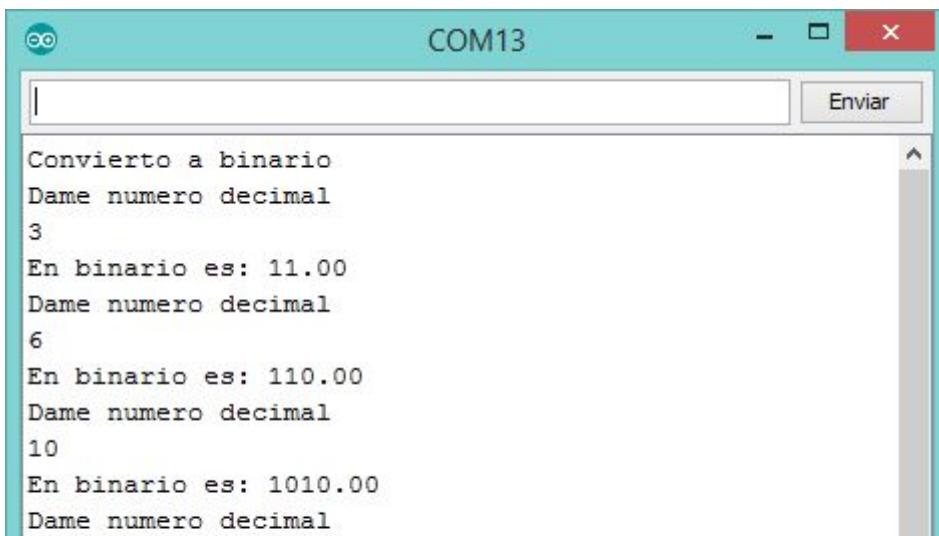
    float bin;    //se declara
    float i;      //variables auxiliares
    float db;
    float r;
    bin = 0;      //bin inicia en 0
    i = 0;        //iterador inicia en 0

    do {          //bucle do while

        r = z % 2;           //iteracion para conversion (modulo
        sucesivas)
        bin = bin + pow(10, i) * r; //pow() necesita float
        i = i + 1;
        z = int(z / 2);

    } while (z != 1);       //mientras z sea diferente a 1

    db = bin + pow(10, i); //db debe ser float
    return db;             //retornar valor de conversion
}
```



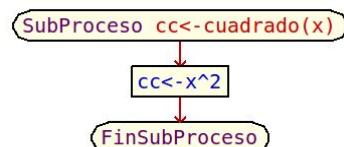
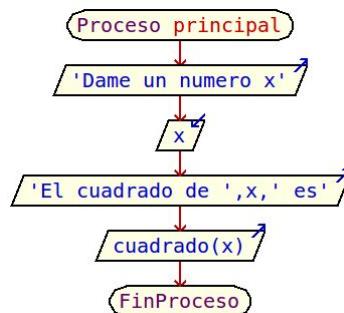
The screenshot shows the Arduino Serial Monitor window titled "COM13". It displays the following text output:

```

Convierbo a binario
Dame numero decimal
3
En binario es: 11.00
Dame numero decimal
6
En binario es: 110.00
Dame numero decimal
10
En binario es: 1010.00
Dame numero decimal

```

64. Definir una función que halle el cuadrado de un número x



```
int condicion = 0; //control de flujo
int x = 0;          //declaracion variable decimal

void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Calculo cuadrado");
    Serial.println("Dame numero ");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial

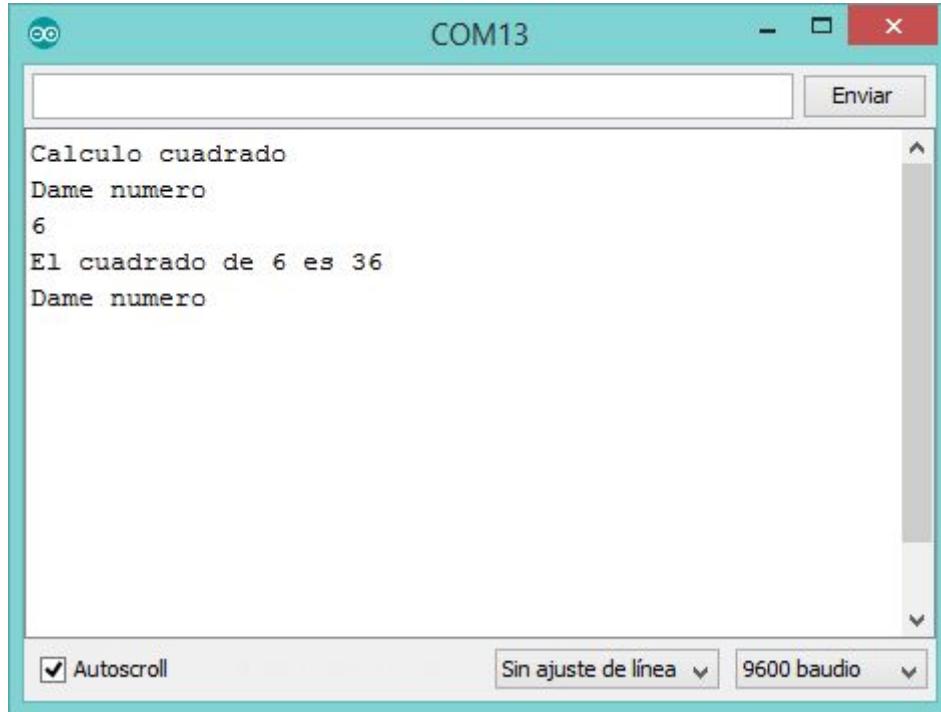
        if (Serial.available() > 0) { //si dato en serial entra
            x = Serial.parseInt(); //entra entero como dec
            Serial.println(x);     //muestra valor de dec
            condicion = 1;         //sale de condicion
        }

    }

    Serial.print("El cuadrado de "); Serial.print(x);
    Serial.print(" es "); Serial.println(cuadrado(x));

    Serial.println("Dame numero ");
    condicion = 0;           //reinicia la condicion para pedir dato
}

int cuadrado(int x) {
    int cc = pow(x, 2);
    return cc;
}
```



ALGORITMOS, DIAGRAMAS DE FLUJO, PSEUDOCODIGO (CONTINUACIÓN)

Conocimientos previos

Entrada: recibir datos del teclado, o de un archivo o de otro aparato.

Salida: mostrar datos en el monitor o enviar datos a un archivo a otro aparato.

Matemáticas: ejecutar operaciones básicas, como la adición y multiplicación.

Operación Condicional: probar la veracidad de alguna condición y ejecutar una secuencia de instrucciones apropiada.

Repetición: ejecutar alguna acción repetidas veces, usualmente con alguna variación.

Forma General de un Algoritmo en PSEUDOCODIGO

Proceso SinTitulo
 acción 1;

acción 1;

...

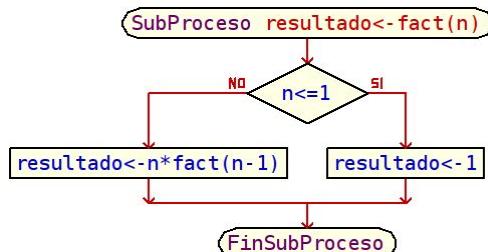
acción n;
FinProceso

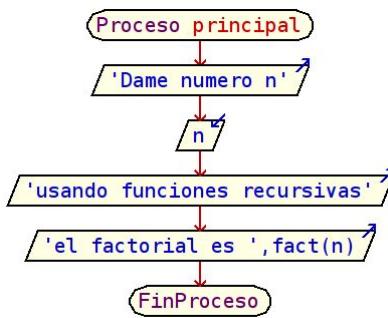
FUNCIONES RECURSIVAS

Una función que se llama a sí misma, directa o indirectamente, es una función recursiva.

65. Definir una función que halle el factorial de un número usando una función recursiva

$$n! = \begin{cases} 1, & \text{si } n = 0 \text{ o } n = 1; \\ n \cdot (n-1)!, & \text{si } n > 1. \end{cases}$$





```

int condicion = 0; //control de flujo
int n = 0; //declaracion variable decimal

void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Factorial Recursiva");
    Serial.println("Dame numero ");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial

        if (Serial.available() > 0) { //si dato en serial entra
            n = Serial.parseInt(); //entra entero como n
            Serial.println(n); //muestra valor de n
            condicion = 1; //sale de condicion
        }
    }
}
  
```

```
Serial.print("el factorial de "); Serial.print(n);
Serial.print(" es "); Serial.println(fact(n));
Serial.println("Dame numero n");
condicion = 0;

}

int fact(int x) { //funcion recursiva

    int resultado;
    if (x <= 1) {
        resultado = 1;
    }
    else {
        resultado = x * fact(x - 1); //se llama a si misma
    }
    return resultado;    //retornar el valor de resultado

}
```

```

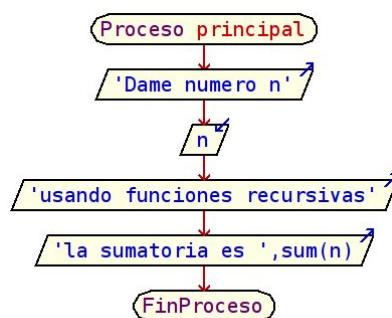
Calculo cuadrado
Dame numero
6
el factorial de 6 es 720
Dame numero n

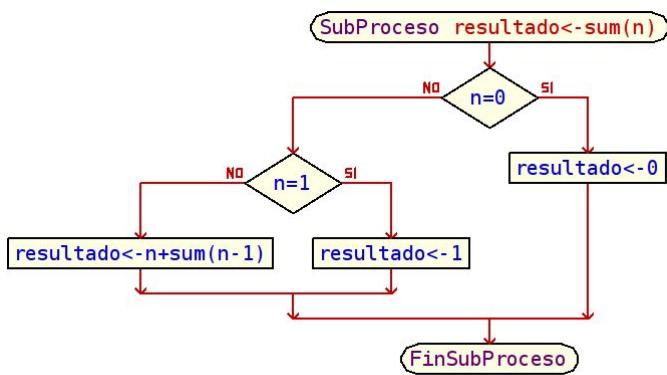
```

Autoscroll Sin ajuste de línea 9600 baudio

66. Implementar una función recursiva que halle la sumatoria de 0 hasta n.

$$\sum_{i=1}^n i = \begin{cases} 1, & \text{si } n = 1; \\ n + \sum_{i=1}^{n-1} i, & \text{si } n > 1. \end{cases}$$





```

int condicion = 0; //control de flujo
int n = 0;          //declaracion variable decimal

void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Sumatoria Recursiva");
    Serial.println("Dame numero ");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial

        if (Serial.available() > 0) { //si dato en serial entra
            n = Serial.parseInt(); //entra entero como n
            Serial.println(n);     //muestra valor de n
            condicion = 1;         //sale de condicion
        }

    }

    Serial.print("la sumatoria de "); Serial.print(n);
    Serial.print(" es "); Serial.println(sum(n));
}

```

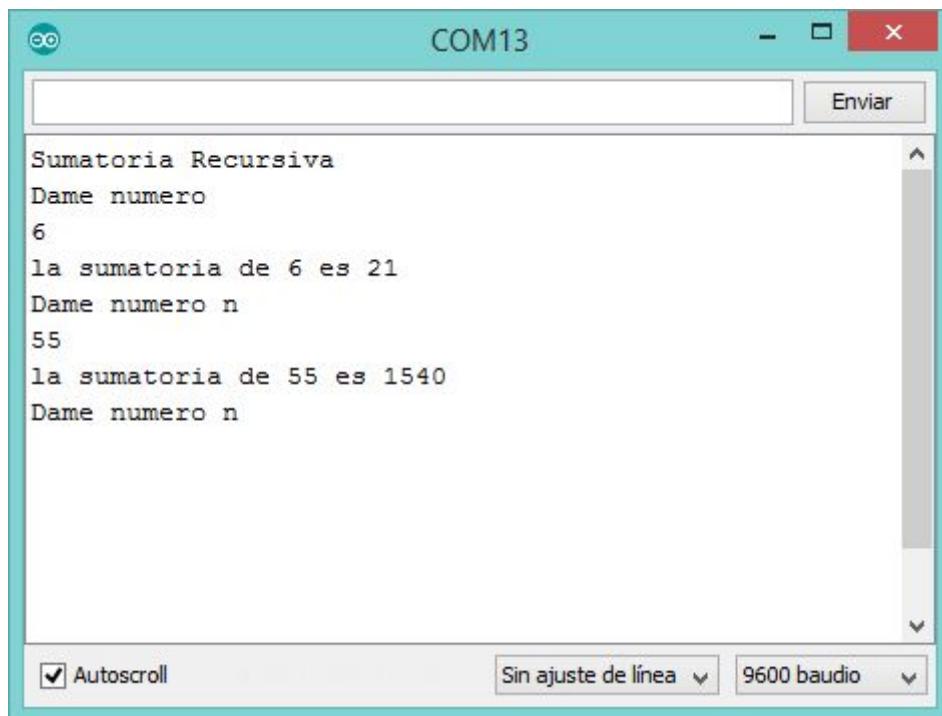
```
Serial.println("Dame numero n");
condicion = 0;

}

int sum(int n) { //funcion recursiva de suma

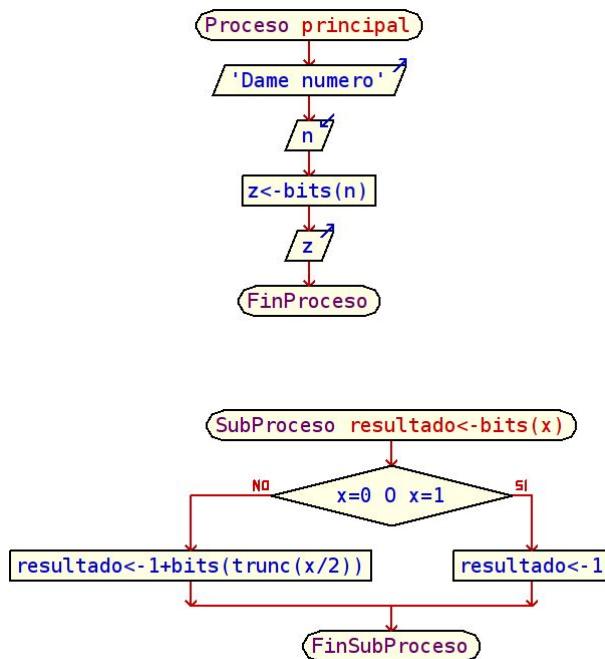
int resultado;
if (n == 0) {
    resultado = 0;
}
else {
    if (n == 1) {
        resultado = 1;
    }
    else {
        resultado = n + sum(n - 1);
    }
}
return resultado; //retornar el valor de resultado

}
```



67. Cálculo recursivo del número de bits necesarios para representar un número

```
def bits(n):
    if n == 0 or n == 1:
        resultado = 1
    else:
        resultado = 1 + bits(n / 2)
    return resultado
```



```

int condicion = 0; //control de flujo
int n = 0;          //declaracion variable decimal

void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Bits Necesarios Recursivo");
    Serial.println("Dame numero ");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial
    }
}
    
```

```
if (Serial.available() > 0) { //si dato en serial entra
    n = Serial.parseInt(); //entra entero como n
    Serial.println(n); //muestra valor de n
    condicion = 1; //sale de condicion
}

}

Serial.print("Numero bits necesarios para "); Serial.print(n);
Serial.print(" son "); Serial.println(bits(n));
Serial.println("Dame numero ");
condicion = 0;

}

int bits(int x) { //funcion recursiva de bits necesarios

    int resultado;
    if (x == 0 || x == 1) { //operador or
        resultado = 1;
    }
    else {
        resultado = 1 + bits(int(x / 2)); //se llama a si misma
    }
    return resultado; //retornar el valor de resultado

}
```

```

Bits Necesarios Recursivo
Dame numero
3
Número bits necesarios para 3 son 2
Dame numero
8
Número bits necesarios para 8 son 4
Dame numero

```

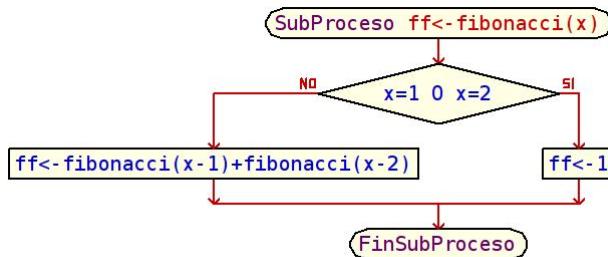
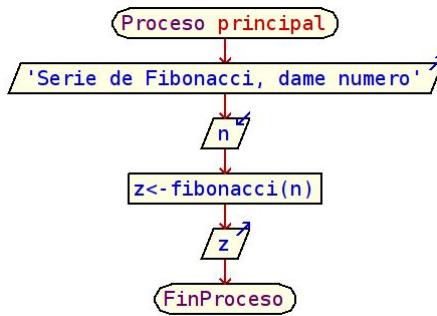
68. NÚMEROS DE FIBONACCI

Definir una función que calcule los números de Fibonacci.

F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
1	1	2	3	5	8	13	21	34	55

Función Recursiva de Fibonacci

$$F_n = \begin{cases} 1, & \text{si } n = 1 \text{ o } n = 2; \\ F_{n-1} + F_{n-2}, & \text{si } n > 2. \end{cases}$$



```

int condicion = 0; //control de flujo
int n = 0; //declaracion variable n

void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("/**Fibonacci**");
    Serial.println("Dame numero ");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial

        if (Serial.available() > 0) { //si dato en serial entra
            n = Serial.parseInt(); //entra entero como n
            Serial.println(n); //muestra valor de n
    }
}

```

```
condicion = 1;           //sale de condicion
}

}

Serial.print("Fibonacci de "); Serial.print(n);
Serial.print(" es "); Serial.println(fibonacci(n)); //salida
invocar funcion
Serial.println("Dame numero ");
condicion = 0;

}

int fibonacci(int x) { //funcion recursiva de fibonacci
    int ff;
    if (x == 1 || x == 2) { //condicion or
        ff = 1;
    }
    else {
        ff = fibonacci(x - 1) + fibonacci(x - 2); //se llama a si
mismo
    }
    return ff;      //retorna valor ff de la funcion
}
```

The screenshot shows a terminal window titled "COM13". The window contains the following text:

```
***Fibonacci***
Dame numero
3
Fibonacci de 3 es 2
Dame numero
10
Fibonacci de 10 es 55
Dame numero
```

69. El Algoritmo de Euclides

Calcula el resto de dividir el mayor de los dos números por el menor de ellos.

Si el resto es cero, entonces el máximo común divisor es el menor de ambos números. Si el resto es distinto de cero, el máximo común divisor de n y m es el máximo común divisor de otro par de números: el formado por el menor de n y m y por dicho resto.

Resolvamos un ejemplo a mano. **Calculemos el mcd de 500 y 218 paso a paso:**

Queremos calcular el mcd de 500 y 218. Empezamos calculando el resto de dividir 500 entre 218: es 64. Como el resto no es cero, aun no hemos terminado. Hemos de calcular el mcd de 218 (el menor de 500 y 218) y 64 (el resto de la división).

Ahora queremos calcular el mcd de 218 y 64, pues ese valor sera también la solución al problema original. El resto de dividir 218 entre 64 es 26, que no es cero. Hemos de calcular el mcd de 64 y 26.

Ahora queremos calcular el mcd de 64 y 26, pues ese valor sera también la solución al problema original. El resto de dividir 64 entre 26 es 12, que no es cero. Hemos de calcular el mcd de 26 y 12.

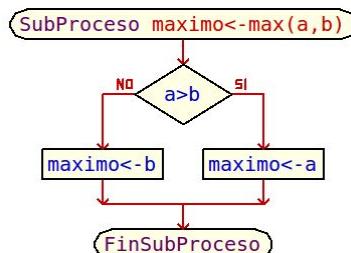
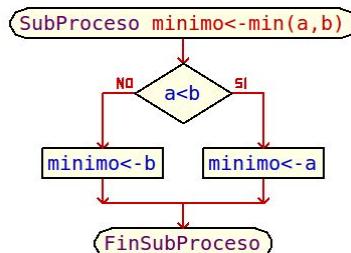
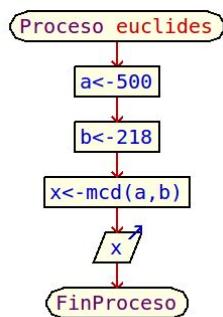
Ahora queremos calcular el mcd de 26 y 12, pues ese valor sera también la solución al problema original. El resto de dividir 26 entre 12 es 2, que no es cero. Hemos de calcular el mcd de 12 y 2.

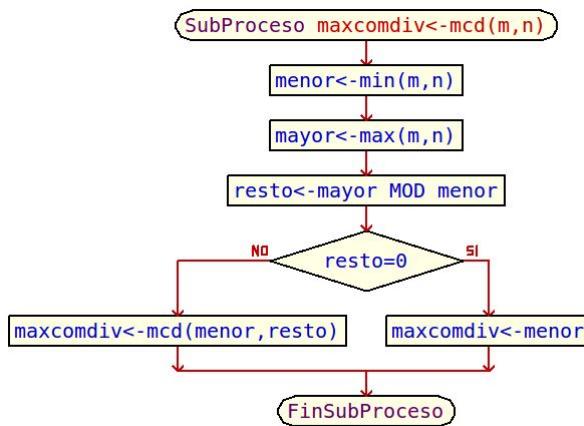
Ahora queremos calcular el mcd de 12 y 2, pues ese valor sera también la solución al problema original. El resto de dividir 12 entre 2 es 0. Por fin: el resto es nulo.

El mcd de 12 y 2, que es el mcd de 26 y 12, que es el mcd de 64 y 26, que es el mcd de 218 y 64, que es el mcd de 500 y 218, es 2.

El caso base es aquel en el que el resto de la división es 0, y el caso general, cualquier otro.

Necesitaremos calcular el mínimo y el máximo de dos números, por lo que nos vendrá bien definir antes funciones que hagan esos cálculos.





```

int a = 500;           //declaracion a
int b = 218;          //declaracion b

void setup() {

  Serial.begin(9600); //velocidad serial 9600
  Serial.println("Algoritmo de Euclides");
  Serial.print("de 500 y 218 es ");
  Serial.println(mcd(a,b)); //salida invocar funcion

}

void loop() { //no se usa

}

int mcd(int m, int n) { //funcion recursiva de mcd
  int maxcomdiv;
  int mayor;
  int menor;
  int resto;
  menor = mini(m, n);
  mayor = maxi(m, n);
  resto = mayor % menor;
}
  
```

```
if (resto == 0) { //evalua si es igual a 0
    maxcomdiv = menor;
}
else {
    maxcomdiv = mcd(menor, resto);
}
return maxcomdiv;    //retorna valor mcd de la funcion

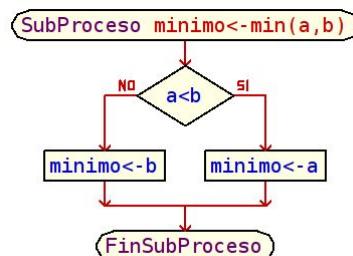
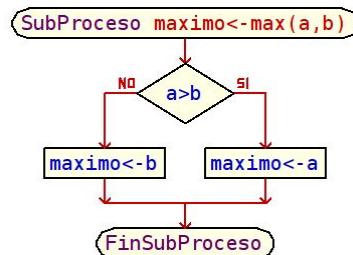
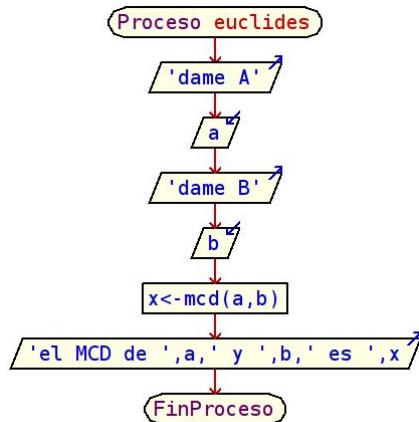
}

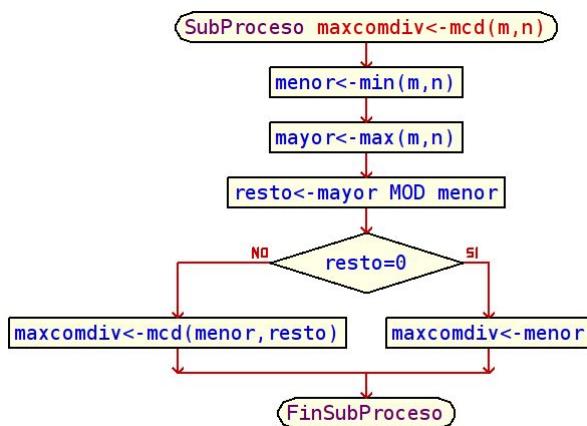
int mini(int a, int b) {
    int minimo;
    if (a < b) {
        minimo = a;
    }
    else {
        minimo = b;
    }
    return minimo;
}

int maxi(int a, int b) {
    int maximo;
    if (a > b) {
        maximo = a;
    }
    else {
        maximo = b;
    }
    return maximo;
}
```

```
COM13
Algoritmo de Euclides
de 500 y 218 es 2
```

70. Desarrollar un programa que calcule el MCD para cualquier par de números





```

int a = 0;           //declaracion a
int b = 0;           //declaracion b
int condicion = 0;   //control flujo

void setup() {

  Serial.begin(9600); //velocidad serial 9600
  Serial.println("MCD");
  Serial.print("Dame el numero a ");
  while (condicion == 0) { //condicion inicial
    if (Serial.available() > 0) { //si dato en serial entrar
      a = Serial.parseInt(); //dato entero en a
      Serial.println(a);
      Serial.print("Dame el numero b ");
      condicion = 1;          //cambiar de bucle
    }
  }

  while (condicion == 1) { //otro while para entrada de b
    if (Serial.available() > 0) {
      b = Serial.parseInt(); //dato entero en b
      Serial.println(b);
    }
  }
}
  
```

```
condicion = 2;           //salir de bucle

}

}

int x = mcd(a, b);      //invoca la funcion de MCD
Serial.print("El MCD es ");
Serial.println(x);
Serial.println("Dame el numero a");
condicion = 0;

}

void loop() { //no se usa

}

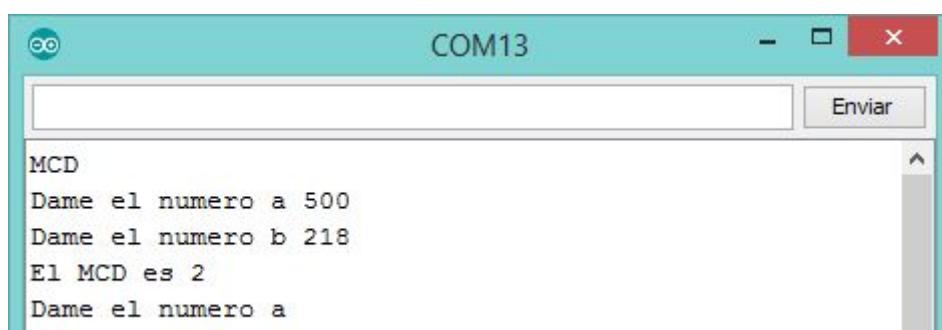
int mcd(int m, int n) { //funcion recursiva de mcd
    int maxcomdiv;
    int mayor;
    int menor;
    int resto;
    menor = mini(m, n);
    mayor = maxi(m, n);
    resto = mayor % menor;
    if (resto == 0) { //evalua si es igual a 0
        maxcomdiv = menor;
    }
    else {
        maxcomdiv = mcd(menor, resto);
    }
    return maxcomdiv;    //retorna valor mcd de la funcion
}
```

```

int mini(int a, int b) {
    int minimo;
    if (a < b) {
        minimo = a;
    }
    else {
        minimo = b;
    }
    return minimo;
}

int maxi(int a, int b) {
    int maximo;
    if (a > b) {
        maximo = a;
    }
    else {
        maximo = b;
    }
    return maximo;
}

```



TIPOS DE DATOS

ENTERO,

Ocupa menos memoria

Operaciones entre enteros son mas rápidas

FLOTANTE

Estándar IEEE 754 para coma flotante.

Sirve para representar decimales.

VALORES LOGICOS O BOOLEANOS

True, para valores “verdaderos”

False, para valores “falsos”

OPERADORES LOGICOS Y DE COMPARACION

AND, similar a multiplicar

OR, similar a sumar

NOT, negación

SENTENCIAS ITERATIVAS

71. El número de combinaciones que podemos formar tomando m elementos de un conjunto con n elementos es: osea n es mayor que m

$$C = \frac{n!}{(n-m)!(m!)}$$

Combinaciones

Cuando se trata de contar el número de subconjuntos de x elementos en un conjunto de n elementos tenemos lo que se denomina *combinaciones de x elementos en un conjunto de n*. El cálculo del conteo se hace mediante el número combinatorio, de la manera siguiente:

$$C_{n}^x = \binom{n}{x} = \frac{n!}{x!(n-x)!}$$

Ejemplo:

¿De cuántas maneras podemos elegir, en la urna anterior (recordemos que había cinco bolas), tres bolas en una única extracción?

Respuesta:

Serán combinaciones de cinco elementos tomados de tres en tres, por tanto, tendremos:

$$C_5^3 = \binom{5}{3} = \frac{5!}{3!(5-3)!} = 10$$

Para el presente ejercicio el código en *Arduino* sería el siguiente.

```
int condicion = 0; //control de flujo
```

```
int n = 0;          //declaracion variable n
int m = 0;          //declaracion variable m
int c = 0;          //combinaciones posibles
void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Numero de Combinaciones ");
    Serial.println("Dame numero n");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial

        if (Serial.available() > 0) { //si dato en serial entra
            n = Serial.parseInt(); //entra entero como n
            Serial.println(n);     //muestra valor de n
            condicion = 1;         //sale de condicion
        }

    }

    while (condicion == 1) { //condicion inicial

        if (Serial.available() > 0) { //si dato en serial entra
            m = Serial.parseInt(); //entra entero como m
            Serial.println(m);     //muestra valor de m
            condicion = 2;         //sale de condicion
        }

    }

    c = fact(n) / (fact(n - m) * fact(m)); //formula de
    combinatoria
    Serial.print("Combinaciones posibles de "); Serial.print(m);
    Serial.print(" en "); Serial.print(n); Serial.print(" son ");

}
```

```

Serial.println(c);
Serial.println("Dame numero n");
condicion = 0;

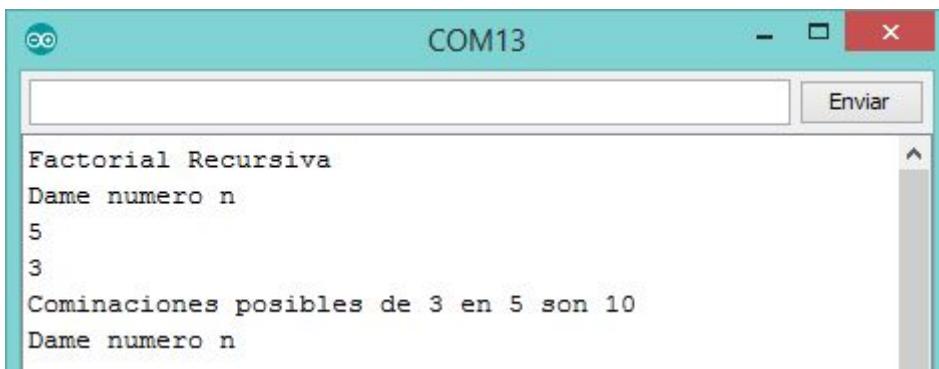
}

int fact(int x) { //funcion recursiva de factorial

int resultado;
if (x <= 1) {
    resultado = 1;
}
else {
    resultado = x * fact(x - 1); //se llama a si misma
}
return resultado; //retornar el valor de resultado

}

```



72. Cálculo de Raíz cuadrado con condición de número positivo.

```

int condicion = 0; //control de flujo

```

```
float n = 0;           //declaracion variable n

void setup() {

    Serial.begin(9600); //velocidad serial 9600
    Serial.println("Dame numero positivo ");
    Serial.println("Calculo raiz cuadrada ");

}

void loop() { //repite continuamente

    while (condicion == 0) { //condicion inicial

        if (Serial.available() > 0) { //si dato en serial entra
            n = Serial.parseFloat(); //entra entero como float
            Serial.println(n);      //muestra valor de n
            if (n < 0) {           //si es negativo error
                Serial.println("Error dato negativo, introducir dato
positivo");
                condicion = 0;
            }
            else {                 //si no salir de bucle
                condicion = 1;      //sale de condicion
            }
        }
    }

    float rc = sqrt(n);
    Serial.println(rc);
    Serial.println("Dame numero positivo ");
    condicion = 0;

}
```

```
Dame numero positivo
Calculo raiz cuadrada
-99.00
Error dato negativo, introducir dato positivo
45.00
6.71
Dame numero positivo
9.00
3.00
Dame numero positivo
4.00
2.00
Dame numero positivo
```

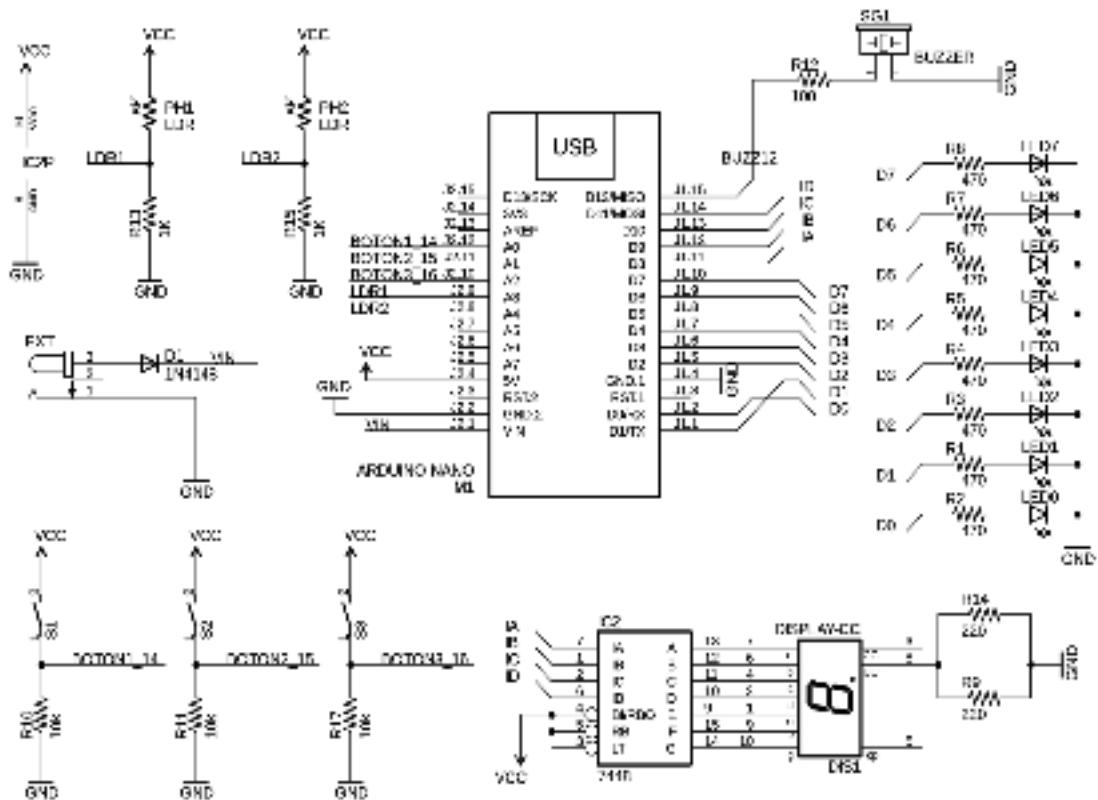
Autoscroll Sin ajuste de línea 9600 baudio

LA PLACA DEL LABORATORIO de IAN Labs

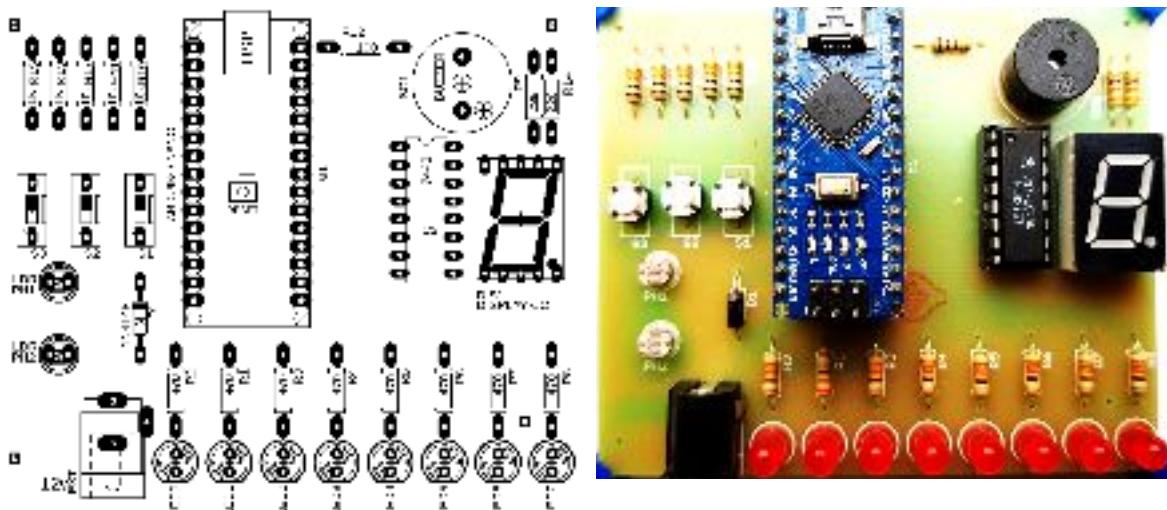
Los siguientes ejercicios están hechos para la placa de entrenamiento IAN Labs, pero también los puedes implementar con cualquier Arduino y protoboard con componentes externos como resistencias y leds, así como pulsadores.

La PCB está diseñada para ser implementada en baquelita o en PCB de fibra de vidrio, y tiene solo una cara, lo que la hace barata y simple, los archivos del PCB diseñados son *Creative Commons* (puedes hacerte la tuya, libremente, respetando al autor original) y pueden ser descargados de www.ianlabs.org, así como los ejercicios de este texto.

El circuito del trainer IAN Labs es el siguiente.



y la placa PCB con sus componentes es la siguiente.



Para armarla es mejor empezar por la parte inferior donde están las resistencias y leds, una buena parte de este texto solo utiliza las *salidas digitales del 0 al 7*, es decir los *LED0 al LED7*.

Siguiendo si lo que quieras es tenerla completa, puedes soldar el resto de componentes, las salidas 8-11 manejan un decodificador de 7 segmentos 7448 para display de cátodo común. Luego para manejar el display en vez de 8 resistencias, una por cada segmento se usan 2 resistencias en paralelo conectadas a tierra.

La parte izquierda de la placa tiene 3 pulsadores en las entradas analogicas (usadas como entradas digitales) las A0, A1 y A2, luego las entradas analogicas A3 y A4 son usadas para 2 resistencias variables con luz, también llamadas LDR, eso para practicar con valores analogicos (luz).

En la parte inferior izquierda vemos un jack para conectar una fuente externa (si quieras usarla) como una batería en caso quieras que tus programas sean independientes de una conexión USB del arduino nano, el diodo más arriba del jack protege la placa en caso la fuente externa tenga una polaridad inversa o incorrecta. Como este conector va al regulador del Arduino nano, lo mejor es energizarla desde 7 voltios hasta 12 voltios como máximo, esto para evitar calentar el regulador que tiene el Arduino NANO.

Como vemos la placa es muy simple y sencilla, esto para facilitar la manufactura en forma casera, es decir la puedes hacer tú mismo en casa, para adquirir la placa ya manufacturada puedes visitar www.ianlabs.org, En la foto también se puede ver que tiene una base de plástico, puedes imprimirla en 3D, el diseño también está en la página web.

Si no quieres adquirir la placa de desarrollo y quieres hacerla tu mismo desde cero puedes leer los anexos de este texto, donde se explica cómo “quemar” una placa de PCB con el método del papel satinado.

73. Luces ida y vuelta

```
int tiempo = 100; //variable entera de tiempo que retarda

void setup() { //bucle for para configurar salidas

    for (int i = 0; i <= 7; i = i + 1) {
        pinMode(i, OUTPUT); //el for que configura las salidas
    }
}

void loop() { //este bucle se repite siempre

    for (int i = 0; i <= 7; i = i + 1) { //for para prender y
```

```
apagar
```

```
    digitalWrite(i, LOW); //apagar 1 a 1 las salidas
    delay(tiempo); //retardo de apagado
    digitalWrite(i, HIGH); //prender 1 a 1 las salidas
}
```

```
}
```



El led apagado *circula de izquierda a derecha* y regresa nuevamente a *repetir* el bucle *for*.

74. Destello de luces

El código fuente es el siguiente.

```
void setup() {
  for (int i = 0; i <= 7; i++) { // desde 0 a 7 como salidas
    pinMode(i, OUTPUT);
  }
}

void loop() { //repite siempre

  for (int i = 0; i <= 7; i = i + 1) { //for desde 0 hasta 7
    digitalWrite(i, HIGH); //prende led
    delay(100);           //por 0.1 seg
    digitalWrite(i, LOW); //apaga led
    delay(100);           //por 0.1 seg
}
```

}



la anterior imagen muestra cómo deberían encenderse las luces, en secuencia desde el led 1 al led 8 y de ahí de regreso al led 1 al led 8 (bucle for), siempre se repite porque está en el bucle *loop()* del *Arduino*.

75. Destello de luces ida y vuelta con for

EL código sería el siguiente

```
int tiempo = 100; //100 milisegundos de variable tiempo

void setup() {    //bucle de configuracion corre una vez
  for (int i = 0; i <= 7; i++) { // desde 0 a 7 como salidas
    pinMode(i, OUTPUT);           //definidos con un bucle for
  }

}

void loop() { //repite siempre bucle loop()

  for (int i = 0; i <= 7; i = i + 1) { //for desde 0 hasta 7.
  incremento
    digitalWrite(i, HIGH);      //prende led
    delay(tiempo);             //por 'tiempo' segundos
    digitalWrite(i, LOW);       //apaga led
    delay(tiempo);             //por 'tiempo' seg
  }

  for (int i = 7; i >= 0; i = i - 1) { //for desde 7 hasta 0.
```

```
decremento
```

```
    digitalWrite(i, HIGH);      //prende led
    delay(tiempo);            //por 'tiempo' seg
    digitalWrite(i, LOW);       //apaga led
    delay(tiempo);            //por 'tiempo' seg
```

```
}
```

```
}
```



76. Visualizar Binarios

Para este ejercicio vamos a introducir un número entero por el puerto serial y luego vamos a ver como este se muestra en las salidas digitales de los LEDS. El código sería el siguiente.

```
int condicion = 0;
int n = 0;

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 8 ; i++) { //i++ incremento, del 0 al 7
    salidas
    pinMode(i, OUTPUT);
  }
}

void loop() {
  Serial.println("Dame numero:");
  
```

```

while (condicion == 0) { //condicion inicial

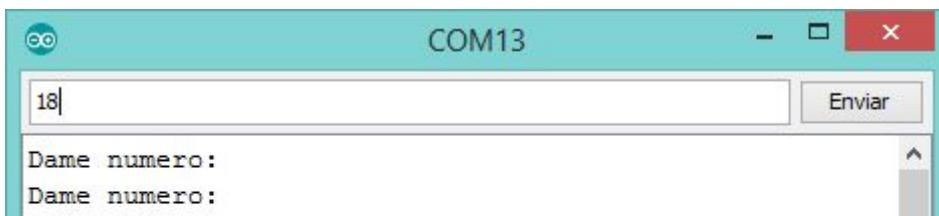
    if (Serial.available() > 0) { //si hay dato en serial entrar
        n = Serial.parseInt(); //dato entero en n
        condicion = 1; //cambiar condicion para salir
    }

}

Serial.end(); //cerrar conexion serial para usar LED1 y LED2
PORTD = n; //mostrar valor de n en el puerto D (LED1 al
LED8)
delay(5000); //por 5 segundos

Serial.begin(9600); //iniciar el puerto serial nuevamente
delay(500); //temporizado para estabilizar conexion serial
condicion = 0; //cambiar condicion para reiniciar entrada de
dato
}

```



Podemos ver a la salida del puerto D el número introducido en forma binaria, por 5 segundos, luego cambia la condición a 0 y vuelve a pedir numero a convertir.



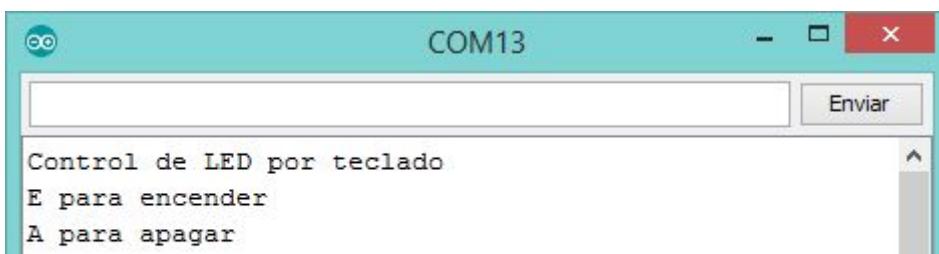
La anterior imagen muestra el valor binario de 18.

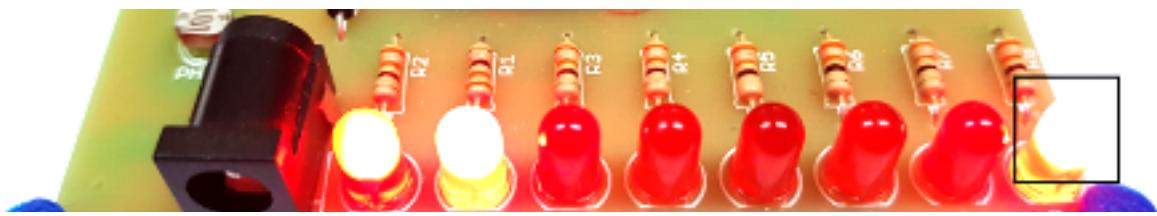
77. Encender un Led por teclado

```
char dato = 0; //declara dato como caracter char
int pin = 7; //se usa pin 7 se puede cambiar

void setup() {
    Serial.begin(9600); //serial a 9600
    pinMode(pin, OUTPUT);
    Serial.println("Control de LED por teclado");
    Serial.println("E para encender");
    Serial.println("A para apagar");
}

void loop() {
    if (Serial.available() > 0) {
        dato = Serial.read(); //Serial.read() lee datos 'char'
    }
    if (dato == 'E' or dato == 'e') { //si es e o E
        digitalWrite(pin, HIGH); //encender pin
    }
    if (dato == 'A' or dato == 'a') { //si es a o A
        digitalWrite(pin, LOW); //apagar
    }
}
```





Como podemos ver cuando enviamos la letra **e** o la letra **E**, el *led 7* se enciende, y cuando enviamos la letra **a** o la letra **A**, el *led 8* se apaga. Se puede usar para muchas cosas este pequeño programa.

78. Led incrementa y regresa

```

int tiempo = 100; //tiempo de retardo en milis

void setup() {      //setup corre una vez

    for (int i = 0; i < 8; i++) { //declara
        pinMode(i, OUTPUT);      //0 al 7 como salidas
    }
    Serial.end();                //da de baja el puerto serie para
    usar                          //pines 0 y 1
}

void loop() {          //se repite siempre el loop

    for (int i = 1; i < 8; i++) { //for ascendente enciende y apaga
        digitalWrite(i, HIGH);    //enciende led i
        delay(tiempo);           //por tiempo milisegundos
        digitalWrite(i, LOW);     //apaga led i
        digitalWrite(0, HIGH);    //enciende led 0
        delay(tiempo);           //por tiempo milisegundos
        digitalWrite(0, LOW);     //apaga led i
    }

    for (int i = 7; i > 0; i--) { //for descendente enciende apaga
        digitalWrite(i, HIGH);    //enciende i
    }
}

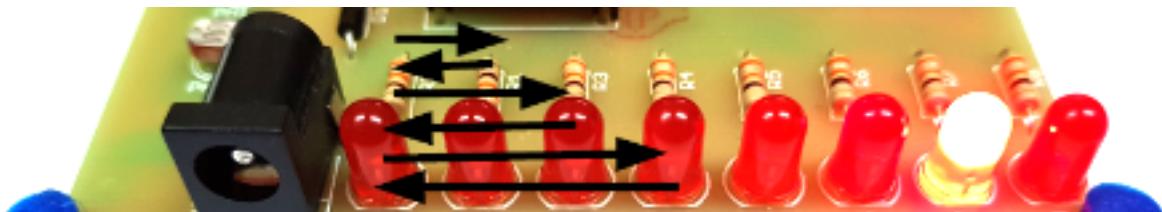
```

```

delay(tiempo);           //por tiempo milisegundos
digitalWrite(i, LOW);    //apaga led i
digitalWrite(0, HIGH);   //enciende led 0
delay(tiempo);           //por tiempo milisegundos
digitalWrite(0, LOW);    //apaga led 0
}

}

```



79. Auto fantastico con while

```

int tiempo = 100; //tiempo de retardo en milis

void setup() {      //setup corre una vez

  int contador = 0; //variable contador auxiliar
  while (contador < 8) {
    pinMode(contador, OUTPUT);
    contador = contador + 1;
  }

  Serial.end(); //da de baja el puerto serie para usar pines 0 y
1
}

void loop() {          //se repite siempre el loop

  int x = 0;           //contador auxiliar x=0

```

```

while (x < 7) {           //si 0<7 entrar
  digitalWrite(x, HIGH);   //encender x
  delay(tiempo);          //pot tiempo en milis
  digitalWrite(x, LOW);    //apagar leds
  x = x + 1;              //aumentar x en 1
}

int y = 7;                //contador auxiliar y=0
while (y > 0) {           //si 7>0 entrar
  digitalWrite(y, HIGH);   //encender y
  delay(tiempo);          //por tiempo milis
  digitalWrite(y, LOW);    //apagar y
  y = y - 1;              //restar y en 1
}

}

```

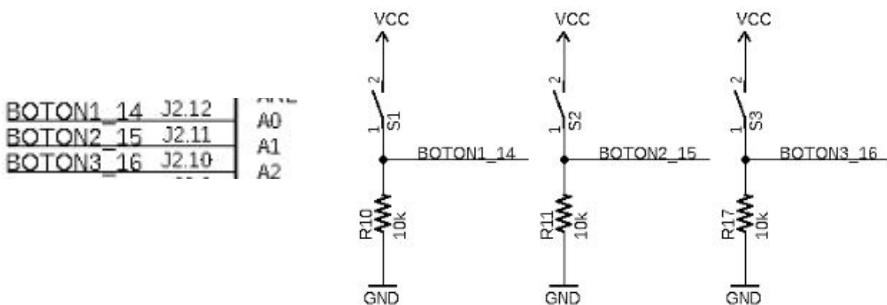


A lo largo de este texto hemos visto *entradas*, *salidas*, *operaciones matemáticas* (incremento), *condicionales* (if) y *repeticiones* (while y for), justo lo que recalcamos, las partes componentes de un programa, es decir ***si has hecho tus ejercicios en forma rigurosa ya sabes programar.***

Ahora vamos a jugar con nuestras entradas digitales, es decir nuestros botones.

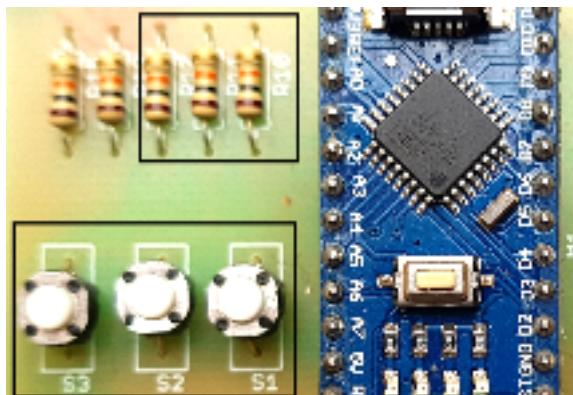
80. Encender un led mediante un pulsador

Si volvemos a nuestro esquema original, vamos a ver que los botones están en las entradas analogicas A0,A1 y A2, estas entradas también se pueden usar en forma digital, veamos las equivalencias.



El diagrama anterior muestra el esquema electrónico de los botones de la placa de entrenamiento. A continuación una imagen que muestra en detalle la ubicación de los botones en la placa de entrenamiento.

Entradas Analogicas	I/O Digital
A0	14
A1	15
A2	16



La imagen anterior muestra los botones y sus correspondientes resistencias. Cuando se presiona un botón, se genera un estado lógico de *Verdadero*, también se podría decir que entra un valor True o un valor de “1”. Cuando no se presiona el botón (es decir al aire) se tiene el estado lógico *Falso* o False o el valor “0”.

La siguiente tabla muestra las equivalencias entre pines I/O digitales y entradas analógicas. Como vemos los botones están definidos en los pines 14, 15 y 16. Ahora que ya entendemos mejor las características de los botones en la placa, continuemos con el ejercicio.

```

int boton = 14; //declara el pin del boton
int led = 7; //declara el pin del led

void setup() { //solo corre una vez
  pinMode(boton, INPUT); //configura boton como entrada
  for (int i = 0; i < 8; i = i + 1) { //0a7 como salida
    
```

```

pinMode(i, OUTPUT); //mediante for
}

}

void loop() {

if (digitalRead(boton) == 1) { //si el valor del boton es 1
  digitalWrite(led, HIGH); //encender el led

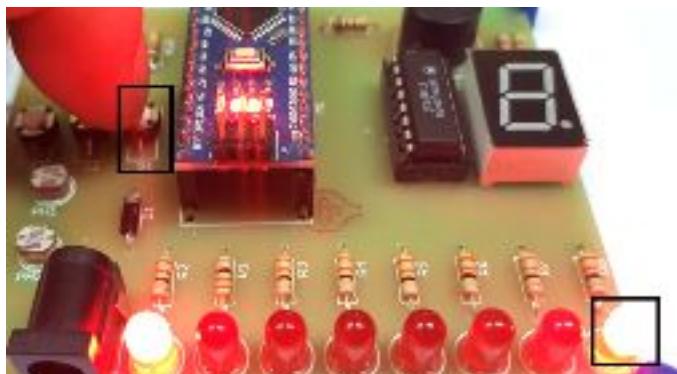
}
else { //o si no
  digitalWrite(led, LOW); //apagar el led
}

}

```

Después de subir el programa a nuestro Arduino y conectado a nuestra placa tendremos lo siguiente.

Cuando pulsamos el botón en el *pin14* se enciende el led en el *pin7*, además en la imagen vemos que el *pin0* está encendido, esto ocurre porque el *Arduino* está energizado por el puerto USB, si usamos una fuente externa el *pin0* estará apagado. Para este ejercicio si no tienes la placa puedes hacer la prueba con el *pin13*, que está incorporado en el mismo *Arduino*.



81. Encender un led con enclavamiento

Enclavamiento es un término eléctrico que significa “mantener”, es decir si presionamos el botón se enciende el led y si lo volvemos a presionar se apaga.

Este comportamiento es muy usado en industria, especialmente en arranque de motores donde estos inician su funcionamiento mediante un simple “pulsador” de enclavamiento. Nuestro código sería el siguiente.

```
int boton = 14; //declara el pin del boton
int led = 7; //declara el pin del led
int encender = 0; //estado para encender
int anterior = 0; //estado anterior
int actual = 0; //estado actual

void setup() {
    pinMode(boton, INPUT); // Pin digital del boton como entrada
    pinMode(led, OUTPUT); // Pin digital del led como salida
    digitalWrite(led, LOW); // led inicia apagado
}

void loop() {

    actual = digitalRead(boton); // estado actual del boton

    if (actual && anterior == 0) { //Comparamos el estado actual y
        el anterior del pulsador
        encender = 1 - encender; //si uno es diferente entrar a
        condicional
        // "encender" cambia en cada interaccion con esta condicion if
        delay(200); // evita rebotes del boton
    }

    anterior = actual; //si presionamos boton, "anterior" es 1, es
    decir V
    //sin presionar boton este estado cambia de 1 a 0

    if (encender == 1) { // Si el estado interno del pulsador pasa
        de "0" a "1".
        digitalWrite(led, HIGH); // led encendido
    }

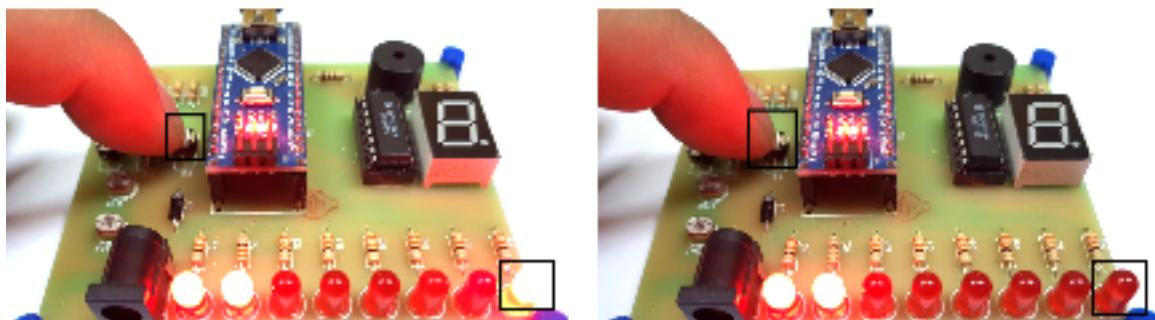
    else { // o si no el estado del boton pasa de "1" a "0".
        digitalWrite(led, LOW); // led apagado
    }
}
```

```

    }
}

```

La salida del programa sería según las siguientes imágenes, si presionas el botón, este se enciende, si vuelves a presionar este se apaga, es lo que se denomina pulsador de enclavamiento.



Si pulsamos botón se enciende y se mantiene encendido, si pulsamos de nuevo se apaga y se queda apagado, y así continúa el ciclo en `loop()`.

82. Generar un tono con el buzzer

Nuestra placa tiene un buzzer conectado al *pin12*, para el presente ejercicio vamos a generar un sonido en ese buzzer, para esto vamos a usar la función `tone`.

La función `tone()`, puede recibir hasta 3 argumentos.
`tone(argumento1,argumento2,argumento3)`.

- *argumento1*: Pin donde va a ejecutar
- *argumento2*: Frecuencia a generar
- *argumento3*: Duración en milisegundos, hay que notar que este argumento no genera ningún tipo de retraso o delay, es decir el programa continúa “aunque suene en el fondo”.

Veamos por ejemplo.

```
tone(12,1000);
```

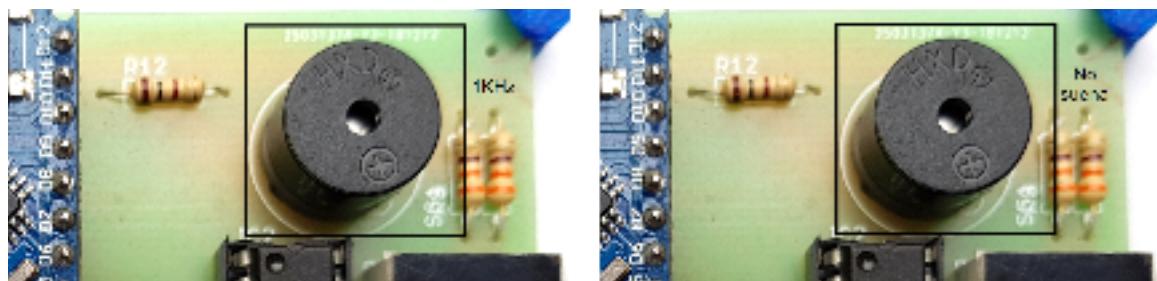
```
tone(12,1000,1000);
```

La primera línea genera una frecuencia de *1KHz* en el *pin12*, y se mantiene el sonido.
La segunda línea genera igual *1KHz* en el *pin12* pero por solo 1 segundo (1000 milis)

Continuando con este ejercicio vamos a generar un tono de *1KHz* en el buzzer conectado en el pin12, por un tiempo de 1 segundo, luego se callara por 1 segundo y así continuamente. El código es el siguiente.

```
void setup() {    //configuracion
  pinMode(12, OUTPUT); //pin12 es salida
}

void loop() {    //se repite siempre
  tone(12, 1000, 1000); //tono de 1kh en pin12 por 1 seg
  delay(1000); //no hacer nada 1 seg
  noTone(12); //apagar tono
  delay(1000); //no hacer nada 1 seg, y regresar al loop
}
```



83. Sirena con Buzzer con while

```
//Sirena con bucle while
void setup() {
  pinMode(12, OUTPUT);           //pin12 salida
  for (int i = 0; i < 8; i++) { //pin0 al pin7
    pinMode(i, OUTPUT);         //salidas
  }
  Serial.end();
}
```

```

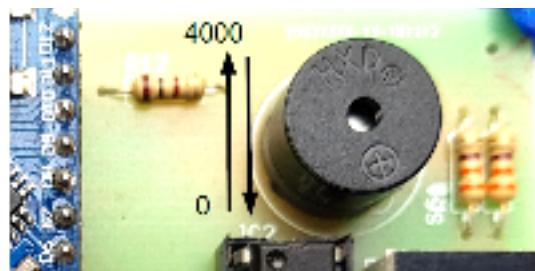
void loop() {

    int contador = 0; //contador inicia en cero
    while (contador < 4000) { //contador cero menor a 4000
        tone(12, contador); //generar tono en pin12, frecuencia
        contador
        PORTD = contador / 16; //mostrar binario de contador/16
        contador = contador + 1; //incrementar contador en 1
    }
    //saliendo del bucle contador tiene 4000
    while (contador > 0) { //contador mayor a cero
        tone(12, contador); //generar tono en pin12 frecuencia
        contador
        PORTD = contador / 16; //mostrar binario de contador/16
        contador = contador - 1; //decrementa contador en 1
    }
    //saliendo del bucle contador es cero y regresar a loop
}

```

La frecuencia del sonido inicia en *contador* 0 hasta 4000 y luego desciende el contador desde 4000 hasta 0, este proceso se repite gracias al bucle *loop()*.

A la vez el valor de contador se muestra en el puerto D del *Arduino*, es decir la barra de leds, esto en formato binario.



84. Números del 0 al 9 en display

Para este ejercicio, vamos a usar el display de 7 segmentos, y vamos a mostrar números desde el 0 hasta el 9 y luego desde el 9 hasta el 0 con intervalos de medio segundo haciendo uso del bucle *for*.

```

void setup() {

```

```

pinMode(14, INPUT); //BOTON S1
pinMode(15, INPUT); //BOTON S2
pinMode(16, INPUT); //BOTON S3

pinMode(8, OUTPUT); //salidas
pinMode(9, OUTPUT); //puerto B
pinMode(10, OUTPUT); //display
pinMode(11, OUTPUT); //del 8 al 11

}

void loop() {
  for (int i = 0; i < 10; i++) { //subida de 0 al 9
    PORTB = i;
    delay(500);
  }

  for (int i = 9; i >= 0; i--) { //bajada de 9 a 0
    PORTB = i;
    delay(500);
  }
}

```

85. Botones sube y baja binarios

Para este ejercicio vamos a usar los 3 botones de la placa de entrenamiento, los cuales irán cambiando los valores binarios presentes en los led del de los pines 0 al 7, es decir el puerto D. Esto lo hará de la siguiente forma, si presionas el botón S1 el valor sube, si presionas el botón S2 el valor baja, si presionas el último botón S3 el valor vuelve a cero. Veamos el código fuente del programa.

```

int valor = 0; //declaracion de variable valor entero

void setup() {

```

```
pinMode(14, INPUT); //boton S1
pinMode(15, INPUT); //boton S2
pinMode(16, INPUT); //boton S3

pinMode(8, OUTPUT); //pin 8 al 11 salidas
pinMode(9, OUTPUT); //para usar en puerto B
pinMode(10, OUTPUT); //PORTB
pinMode(11, OUTPUT); //el display esta en puerto B

}

void loop() {

    PORTB = valor; //muestra valor en puerto B (pin0-pin7)
    if (digitalRead(14) == 1) { //si boton S1 es Verdad
        delay(300); //antirebote
        valor = valor + 1; //incremento en 1

        if (valor > 9) { //si valor es 9
            valor = 9; //ya no sube el valor si es 9
        }
    }

    if (digitalRead(15) == 1) { //si boton S2 es verdad
        delay(300); //antirebote
        valor = valor - 1; //decremento en 1

        if (valor < 0) { //si valor es menor a cero
            valor = 0; //valor ya no puede bajar
        }
    }

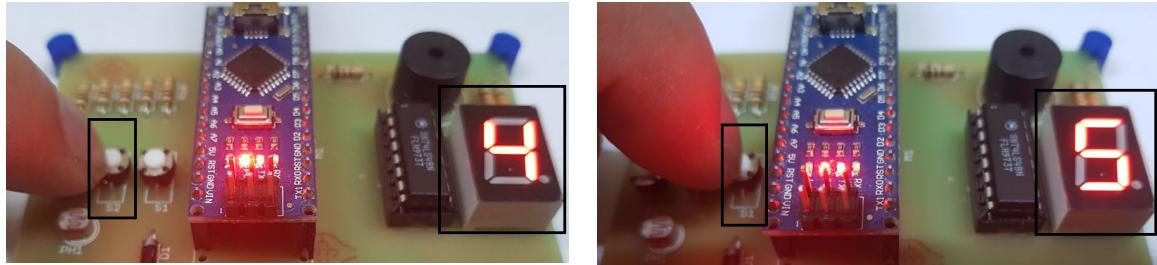
    if (digitalRead(16) == 1 ) { //si boton S3 es verdad
        delay(300); //antirebote
    }
}
```

```

valor = 0;           //valor es cero (reset)
}

}

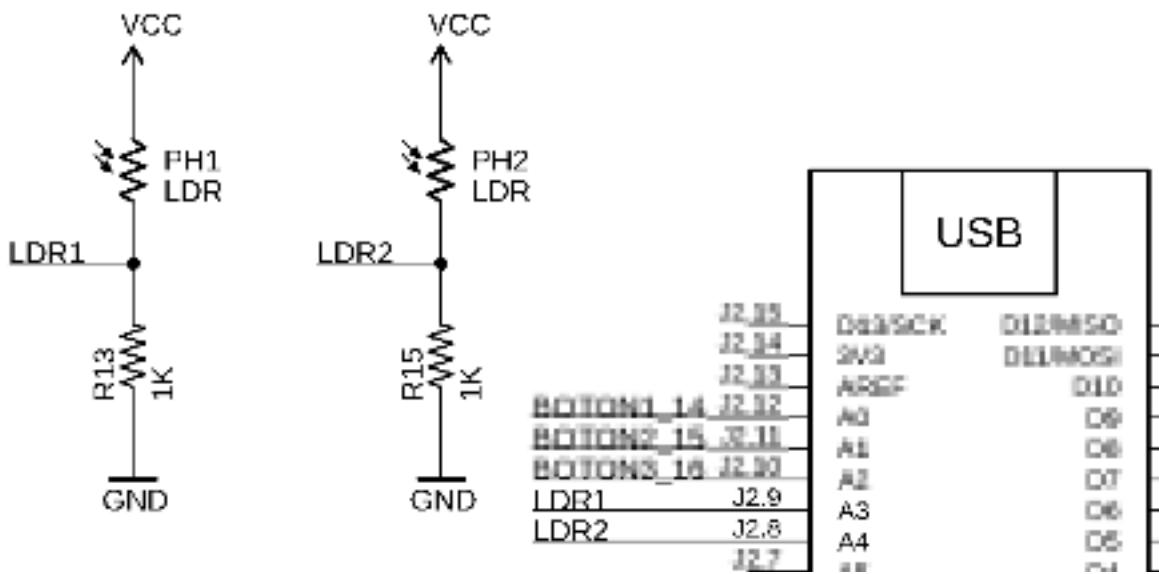
```



Si presionamos el botón *S1* el valor aumenta y si presionamos el botón *S2* el valor disminuye, finalmente si presionamos el botón *S3* el valor vuelve a *cero*.

86. Lectura de valores analogicos LDR

Para este ejercicio vamos a usar una resistencia variable con la luz, la placa tiene 2, una conectada a la entrada analoga A3 y la otra a A4, según la siguiente figura.



El convertidor analógico digital del Arduino nano es de 10 bits y de 0 a 5 voltios, esto quiere decir que puede leer valores de voltaje desde 0 hasta 5 voltios y puede guardar esas

variaciones de voltaje en 1023 posibles valores (1023 es el valor máximo que puede alcanzar 10 bits). Volviendo al ejercicio vamos a leer el valor analogico presente en la entrada analogica A3 es decir el LDR1 (PH1) y vamos a mostrar el valor leido en dos formas, una mediante el Monitor serie y otra mediante el Serial Plotter. EL código sería el siguiente.

```

int dato = 0; //declaracion de dato a leer
int pin = 3; //pin analogico a usar (tambien puede ser el 4)

void setup() {//funcion setup solo corre una vez

    Serial.begin(9600); //puerto serial a 9600 velocidad

}

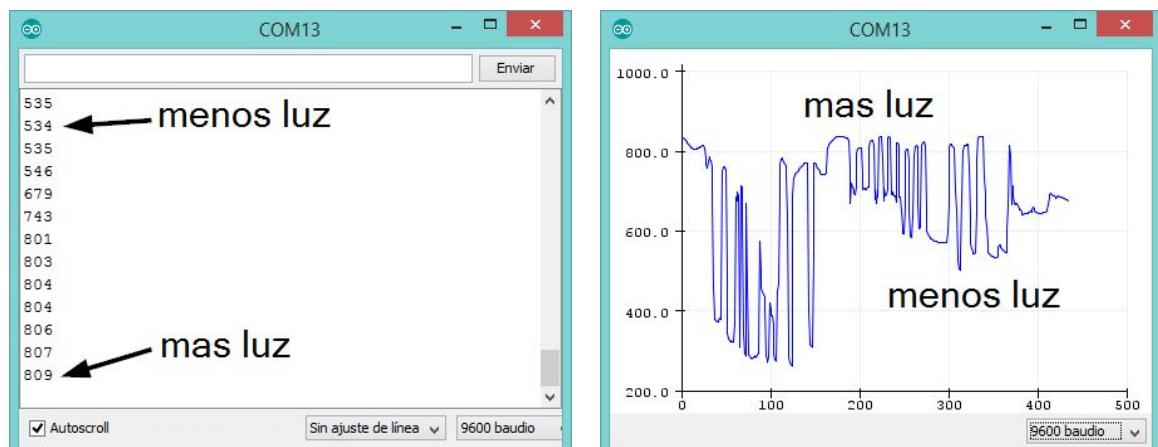
void loop() { //bucle loop se repite siempre

    dato = analogRead(pin); //leer analogico en "pin", guardar en
    "dato"
    Serial.println(dato); //salida de "dato" por puerto serial
    delay(100); //retardo de 100 milis para estabilizar la medida

}

```

Después de subir nuestro programa a nuestro Arduino, abramos primero el Monitor serie y el Plotter serial del menú de Herramientas, se tendrá lo siguiente.



Las imágenes anteriores muestran la salida por el Monitor serie y la salida por el Plotter serial.

87. Calibración de LDR

Los valores del LDR en el puerto analogico varian de acuerdo a las condiciones de luz. En este ejercicio se desea que este sensor se pueda calibrar de acuerdo a cada condición lumínica, es decir.

Valores en el Conversor ADC	Valores a usar de 0 a 255 (8 bits)
830 aprox (con luz)	255
350 aprox (sin luz)	0

Para esto vamos a calibrar por el puerto serial que el valor máximo del ADC (830) sea equivalente a 255 y el valor mínimo del ADC (350) sea equivalente a 0. Para esto vamos a usar la función *map()*.

Función *map()*

```
valor = map(valor, minimo, maximo, 0, 255); //map()
equivalencias
```

y = (valor que recibo, de Mínimo, de Máximo, a Mínimo, a Máximo)

El código fuente del programa sería el siguiente.

```
int ldr = 4; //pin analogico a usar ldr PH2
int condicion = 0; //control de flujo para configurar
int maximo = 0; //maximo valor posible en el ldr
int minimo = 0; //minimo valor posible en el ldr
int valor = 0; //valor que se sensa en el ldr
int s1 = 14; //boton s1
int lampara = 7; //lampara indicador configuracion

void setup() {//funcion setup solo corre una vez
```

```

pinMode(s1, INPUT); //boton s1 entrada
pinMode(lampara, OUTPUT); //lampara es salida

while (condicion == 0) { //condicion inicial
    digitalWrite(lampara, HIGH); //indica espera config
    if (digitalRead(s1) == 1) { //lee boton 1
        delay(100); //retraso antirebote
        maximo = analogRead(ldr); //maximo es valor analogico es
ldr
        digitalWrite(lampara, LOW); //indica que funciona
        delay(1000); //apagado por 1 segundo
        condicion = 1; //sale de condicion 0 y while
    }
}

while (condicion == 1) { //nueva condicion
    digitalWrite(lampara, HIGH); //indica espera config
    if (digitalRead(s1) == 1) { //lee boton 1
        delay(100); //retraso antirebote
        digitalWrite(lampara, LOW); //indica que funciona
        minimo = analogRead(ldr); //valor analogico de ldr es
minimo
        digitalWrite(lampara, LOW); //indica que funciona
        delay(1000); //apagado por 1 segundo
        condicion = 2; //salir de condicion 1
    }
}
Serial.begin(9600); //puerto serial a 9600 velocidad
}

void loop() { //bucle loop se repite siempre

    valor = analogRead(ldr); //leer valor analogico en ldr
}

```

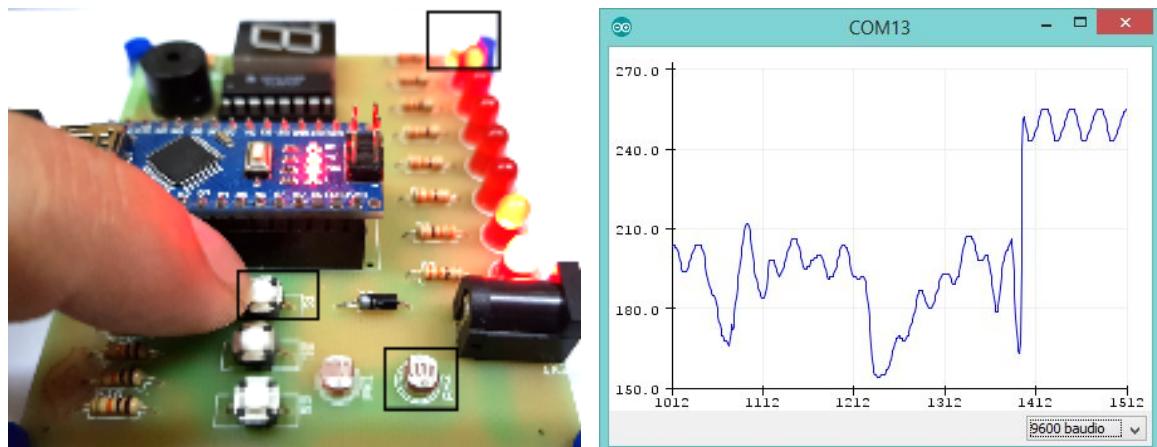
```

valor = map(valor, minimo, maximo, 0, 255); //map()
equivalencias
if (valor > 255) { // si valor>255 entonces no sube mas
    valor = 255;      //es decir valor =255
}
if (valor < 0) {     //si valor<0 no puede ser menor
    valor = 0;        //valor =0
}
Serial.println(valor); //salida de valor por serial
delay(50);           //retardo (20 medidas por segundo)

}

```

Una vez subido el código a nuestro Arduino nano, abrimos nuestro *Serial Plotter* (Herramientas--Serial Plotter) a 9600 de velocidad, vemos que no hay nada, ya que el Arduino está esperando la configuración (*led7*), ahora presionamos el botón *S1*, después de eso el *led7* hará un *parpadeo* señalando que la configuración tuvo éxito, eso quiere decir que el *valor máximo* de luz está seteado. Ahora tapamos el *LDR* (sensor de luz) con un dedo y tapandolo presionamos el botón *S1* de nuevo, nuevamente el *led7* hará un *parpadeo*, eso querrá decir que el *valor mínimo* también fue seteado y ahora si el *Serial Plotter* empezará a mostrar datos. Si nos fijamos bien ahora los valores de luz van a variar entre **0** y **255**.



Con el botón *S1* se configura el valor máximo y mínimo y a la salida tendremos el valor de luz entre *0* y *255*. Este programa con ligeras variaciones puede servir para calibrar robots, o quizás otro tipo de sensores.

88. Detección de las manos

Para el presente ejercicio vamos a escribir un programa que detecte cuando pases la mano por encima de la placa de entrenamiento.

La placa *Arduino* de *IAN Labs* tiene 2 *LDR* para poder hacer aplicaciones de detección de gestos. Para este ejercicio vamos a detectar cuando la mano pasa de un lado hacia el otro, es decir, cuando movemos nuestra mano por encima de los dos sensores, uno reaccionara antes que el otro, ya que la mano no obstruye la luz de los sensores al mismo tiempo, la imagen del costado nos da idea de este escenario.



Ahora la lógica del programa sería la siguiente. Al inicio los sensores están funcionando y detectan una luz ambiental constante, ahora si deslizamos la mano de izquierda a derecha el sensor *PH1* será el primero en cambiar y luego el *PH2*. Y si deslizamos desde la derecha a la izquierda el primero en reaccionar será el *PH2* y luego el *PH1*. Ahora que ya tenemos claro el proceso físico escribamos el programa.

```

int izquierda = 7; //led indicador izquierda
int derecha = 6; //led indicador derecha
int ldr1 = 3; //sensor ldr1
int ldr2 = 4; //sensor ldr2
int condicion = 0; //variable condicion control de flujo

void setup() { //funcion setup solo corre una vez
  Serial.begin(9600); //perto serial a 9600
  pinMode(izquierda, OUTPUT); //izquierda es salida
  pinMode(derecha, OUTPUT); //derecha es salida
}

void loop() { //bucle loop se repite siempre

  int a = analogRead(ldr1); //a es valor de ldr1
  int b = analogRead(ldr2); //b es valor de ldr2
}

```

```
if (condicion == 0) { //condicion por defecto
    digitalWrite(izquierda, LOW); //todo apagado
    digitalWrite(derecha, LOW);

}

if (condicion == 1) { //si condicion es 1
    digitalWrite(izquierda, LOW); //encender led 6
    digitalWrite(derecha, HIGH);
    Serial.println("derecha");

}

if (condicion == 2) { //si condicion es 2
    digitalWrite(izquierda, HIGH);
    digitalWrite(derecha, LOW); //encender led 7
    Serial.println("izquierda");

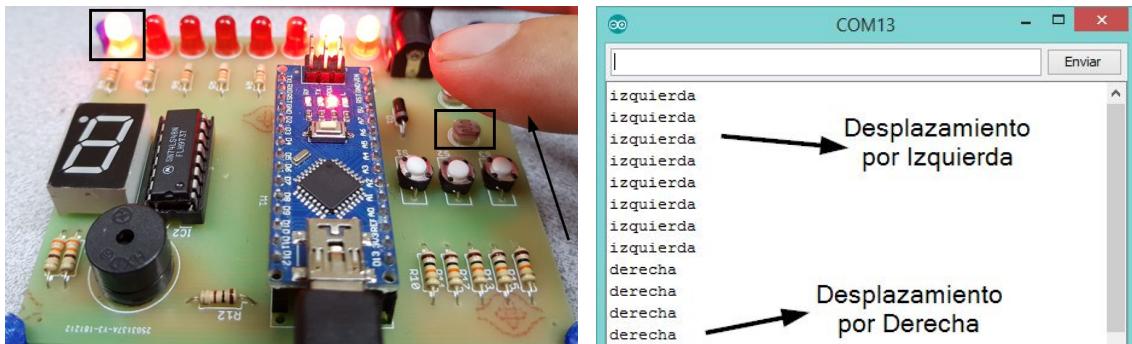
}

if ((a - b) > 30) { //si la resta de sensores<30
    condicion = 1;
}

if ((a - b) < -30) { //si la resta de sensores<30
    condicion = 2;
}

if (abs(a - b) < 20 ) { //si absoluto de la resta<20 apagar todo
    condicion = 0;           //verificar este valor mediante pruebas
}
delay(100);
}
```

La salida del programa en nuestro Arduino seria de la siguiente forma.



Este proyecto aunque sencillo, si lo unimos a un lenguaje de alto nivel como Python, podríamos hacer desde control de volumen hasta scroll de una pantalla, es decir podríamos controlar la PC mediante gestos.³

89. Subir y Bajar binarios

```

int mas = 14; //boton incrementar
int menos = 15; //boton decrementar
int nada = 16; //boton reset
int numero = 0; //variable numero

void setup() { //funcion setup solo corre una vez

    for (int i = 0; i < 8; i++) { //pines 0-7 salidas
        pinMode(i, OUTPUT); //puerto D
    }
}

void loop() { //bucle loop se repite siempre

    PORTD = numero; //dato "numero" en el puerto D (led0-led7)

    if ((digitalRead(mas) == 1)) { //si S1 es Verdadero
        delay(300); //antirebote
        numero = numero + 1; //incrementar numero en 1
    }
}

```

³ Para control de PC por gestos podríamos usar *pyserial* y *system*, ambas librerías del lenguaje python.

```

if (numero > 255) { //hasta un maximo de 255
    numero = 255;
}
}
if ((digitalRead(menos) == 1)) { //si S2 es Verdadero
    delay(300); //antirebote
    numero = numero - 1; //decrementar numero en 1
    if (numero < 0) { //hasta un minimo de 0
        numero = 0;
    }
}
if ((digitalRead(nada) == 1)) { //si S3 es Verdadero
    delay(300); //antirebote
    numero = 0; //resetear numero a 0
}

}

```

90. Alarma programada

Se tiene un sistema de alarma listo para activarse, se requiere introducir una clave numérica ya programada, de presentarse un error, deberá activarse un *LED* que representa la alarma del sistema.

Solución:

Implementar un sistema para introducir la clave, si el usuario se equivoca se activará las luces de emergencia similares a ejercicios previos.

Objetivos:

1. Crear el sistema de alarma.
2. Crear una función de alarma similar a luces de emergencia.
3. Implementar un sistema para introducir la contraseña de acceso.
4. Comparar la clave interna con la que tenemos.
5. Si las claves son las mismas *DESACTIVAR* la alarma.
6. Si las claves son diferentes *ACTIVAR* la alarma

Procedimiento:

El sistema de alarma.

- El sistema de alarma tiene la siguiente forma:
 - Se tiene configurada una “clave” de antemano, esta clave es secreta y sirve para desactivar la alarma.
 - Se debe poder introducir la alarma de algún modo, en este caso usamos el teclado.
 - Si la clave se introduce en forma equivocada, las luces de emergencia se activan y se visualiza un mensaje de que la alarma se ha activado.
- **Las declaraciones de variable.**
 - i. Para el siguiente ejercicio se tiene una “clave en el arduino” y una “clave en mi cabeza”, se supone que si los dos son iguales la alarma se desactiva.
 - ii. Estas “claves” deben declararse previamente, para que puedan ser usadas en el programa, la forma de hacerlo es.

```
String entrada;
String clave = "1234";
```

- iii. String significa que son una “cadena de caracteres”, ahora clave tiene el valor de 1234 y entrada no tiene un valor definido.
- **La función *setup()***
 - i. Para este proyecto la función setup tiene más configuraciones, ya que queremos comunicación, por lo tanto añadimos la comunicación serial mediante *Serial.begin(9600)*, así como que el pin 13 sea OUTPUT, también mostramos un mensaje de que la alarma está iniciada con *Serial.println()*.

```
void setup() {
```

```

Serial.begin(9600);
pinMode(13, OUTPUT);
Serial.println("Sistema Armado");
Serial.println("Introduzca la Contrasena: ");
}

```

Procedimiento

- Creamos una función tipo void que haga el blink. (parpadeo).
- while (1) crea un bucle infinito para que una vez activado no se pueda apagar las luces de emergencia.

```

void alarma() {
    while (1) {
        digitalWrite(13, HIGH);
        delay(100);
        digitalWrite(13, LOW);
        delay(100);
    }
}

```

Introducir la clave al sistema de alarma

- Si el puerto serial está available (disponible), leer y guardar el dato como String en la variable entrada.

```

if (Serial.available() > 0) {
    entrada = Serial.readString();
}

```

Comparar la clave entrada con la clave original

- Continuando el proyecto tenemos que comparar la entrada por teclado con la clave de la alarma para eso usamos las condicionales.

- Si una condición se cumple se ejecuta una acción, si se cumple otra se ejecuta esa otra acción y si no se cumple la ninguna condición (else), se ejecuta una última acción.
- Para este proyecto como evaluamos una sola condición, la entrada es igual a la clave, si te equivocas se activa la función alarma() y empiezan las luces de emergencia.

```
if (entrada == clave) {  
    Serial.print("exito, sistema desactivado");  
}  
else {  
    Serial.print("error, alarma activada");  
    alarma();  
}
```

Si las claves son iguales desactivar el sistema de alarma.

```
if (entrada == clave) {  
    Serial.print("exito, sistema desactivado");  
}
```

Si las claves son diferentes activar el sistema de alarma.

```
else {  
    Serial.print("error, alarma activada");  
    alarma();  
}
```

El código completo:

```
/*declaraciones*/  
String entrada ;  
String clave = "1234";
```

```
/*-----*/  
  
void setup() {  
    Serial.begin(9600);  
    for (int i = 0; i < 8; i++) {  
        pinMode(i, OUTPUT);  
    }  
    pinMode(12, OUTPUT);  
    Serial.println("Sistema Armado");  
    Serial.println("Introduzca la Contrasena: ");  
}  
  
void loop() {  
  
    if (Serial.available() > 0) {  
        entrada = Serial.readString();  
        Serial.print("clave es = ");  
        Serial.println(entrada);  
        if (entrada == clave) {  
            Serial.print("exito, sistema desactivado");  
            int contador = 0;  
            while (contador < 2) { //while de 2 ciclos  
                tone(12, 1000); //tono de 1K en buzzer 12  
                delay(250); //por 1/4 segundo  
                noTone(12); //apagar buzzer  
                delay(250); //por 1/4 segundo  
                contador = contador + 1; //aumentar en 1  
            }  
        }  
        else { //o si no te equivocaste  
            Serial.print("error, alarma activada");  
            Serial.end();  
            alarma(); //ejecutar funcion alarma  
        }  
    }  
}
```

```

void alarma() { //funcion de alarma
    while (1) { //bucle infinito, no tiene salida

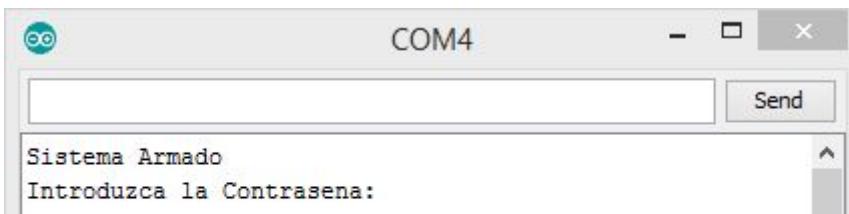
        for (int i = 0; i < 8; i++) { //for 0 hasta el 7
            digitalWrite(i, HIGH); //encender las salidas 0-7
        }
        tone(12, 1500); //tone pin12 1500 hertz
        delay(1000); //por un segundo

        for (int i = 0; i < 8; i++) { //for 0 hasta el 7
            digitalWrite(i, LOW); //apagar las salidas 0-7
        }
        tone(12, 500); //tone pin12 500 hertz
        delay(1000); //por un segundo
    }
}

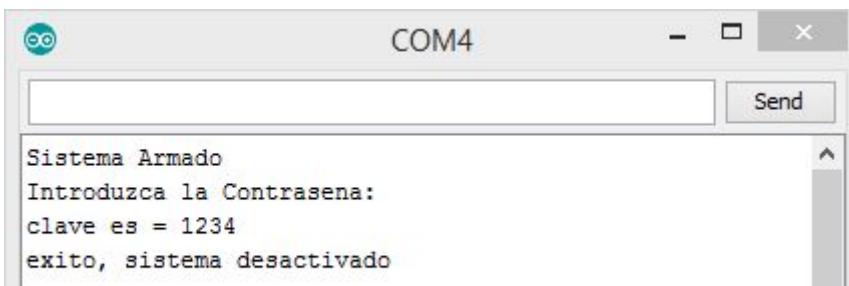
```

En funcionamiento

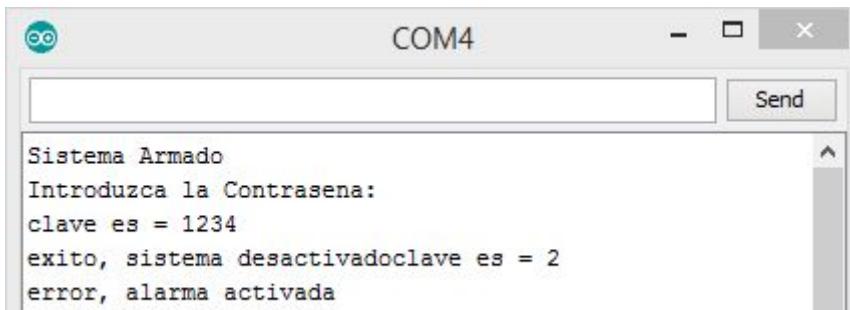
Si usamos el *Monitor serial* del *Arduino* se tiene lo siguiente:



Si enviamos un dato correcto, se desactiva la alarma.



y si nos equivocamos... se activa la alarma.



91. Dado Electronico

Para el presente ejercicio vamos a implementar un dado electrónico para jugar juegos de mesa, es decir:

- Vamos a mostrar valores desde el 1 al 6 en el display de 7 segmentos
- Pulsamos un botón *S1* y empieza el “Chocolateo” de valores, parpadea la barra de leds y además hace un pitido.
- Pulsamos otro botón *S2* y se fija un valor al azar entre 1 al 6 (lados del dado), la barra de leds se detiene, también hay otro pitido indicando que se detuvo el dado.

El código sería el siguiente.

```
int s1 = 14; //boton s1
int s2 = 15; //boton s2
int s3 = 16; //boton s3
int buzzer = 12; //pin del buzzer
int azar = 0; //variable al azar
int condicion = 0; //control de flujo de programa

void setup() {
  pinMode(0, OUTPUT); //leds del 0-7
  pinMode(1, OUTPUT); //trata de configuralos
  pinMode(2, OUTPUT); //con un while
  pinMode(3, OUTPUT); //para practicar
  pinMode(4, OUTPUT); //o un for
  pinMode(5, OUTPUT); //esta declaracion explicita es
  pinMode(6, OUTPUT); //para repaso nomas
  pinMode(7, OUTPUT); //
```

```
pinMode(8, OUTPUT);//los 4 bits 8-11
pinMode(9, OUTPUT);//para el display
pinMode(10, OUTPUT);//de 7 segmentos
pinMode(11, OUTPUT);//son salidas

pinMode(buzzer, OUTPUT); //buzer en pin12
pinMode(s1, INPUT);//entrada boton s1-14
pinMode(s2, INPUT);//entrada boton s2-15
pinMode(s3, INPUT);//entrada boton s3-16
Serial.end();//desactiva el puerto serial para usar pin0 y pin1
randomSeed(analogRead(3) + analogRead(4)); //la "semilla" para
el azar
                                         //viene de los ldr
(A3-A4)
}

void loop() {

    if (digitalRead(s1) == 1) { //lanza dado
        delay(50);
        int contador = 0;
        while (contador < 2) {
            tone(buzzer, 1500);
            delay(100);
            noTone(buzzer);
            delay(100);
            contador = contador + 1;
        }
        condicion = 1;
    }

    if (digitalRead(s2) == 1) { //para el dado
        delay(50);
        tone(buzzer, 2500);
        delay(100);
    }
}
```

```
noTone(buzzer);
delay(100);
condicion = 2;
}

if (digitalRead(s3) == 1) { //reset de dado
    delay(50);
    tone(buzzer, 1500);
    delay(100);
    noTone(buzzer);
    delay(100);
    condicion = 0;
}

if (condicion == 0) { //reset dado
    for (int i = 0; i < 8; i = i + 1) {
        digitalWrite(i, LOW);
    }
    PORTB = 0;
}

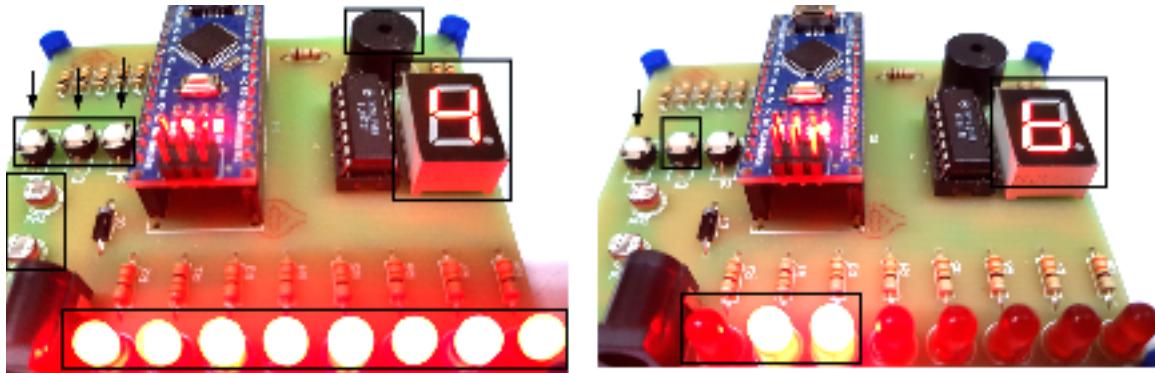
if (condicion == 1) { //lanza dado
    azar = random(1, 7); //genera un numero aleatorio del 1 al 7
    delay(50); //retraso para efecto de destello
    PORTB = azar; //muestra "azar" en display
    PORTD = map(azar, 0, 6, 0, 255); //muestra "azar" en barra de
leds
    //map() "amplia" numero de 0-255

}

if (condicion == 2) { //para el dado
    PORTB = azar;
    PORTD = azar;
}
```

}

Después de subir nuestro código a nuestro *Arduino*, tendremos lo siguiente. Al inicio todo es 0. Si presionamos *S1* inicia el lanzador de dados, luego podemos presionar el *S2* y el dado se detendrá, ahora si queremos resetear el lanzador de dados presionamos el *S3* y el dado se hace cero.



Las anteriores imágenes muestran inicialmente el proceso de *azar* del dado, luego a la derecha se observa el número que se obtuvo, el 6 y en la *barra de leds* lo mismo pero en *binario*.

92. Sonido controlado por luz.

Como hemos visto en anteriores ejercicios se puede generar “tonos” de una frecuencia determinada en el buzzer.

Para este ejercicio vamos a usar una variante del ejemplo *tonePitchFollow* del *IDE Arduino* escrito por Tom Igoe y Michael Flynn el cual está en el dominio público, el código es el siguiente.

```
int condicion = 0; //condicion para control de flujo

void setup() {
  Serial.begin(9600);
  pinMode(14, INPUT);
  pinMode(15, INPUT);
```

```
}

void loop() {

    if (digitalRead(14) == 1) { //si S1 es 1, condicion es 1
        condicion = 1;           //inicia musica
    }

    if (digitalRead(15) == 1) { //si S2 es 1, condicion es 0
        condicion = 0;           //se calla la musica
    }

    if (condicion == 0) { //si condicion es 0 hacer
        noTone(12);
    }

    if (condicion == 1) { //si condicion es 1 hacer
        musica(); //invoca la funcion musica()
    }

}

void musica() { //define la funcion void musica

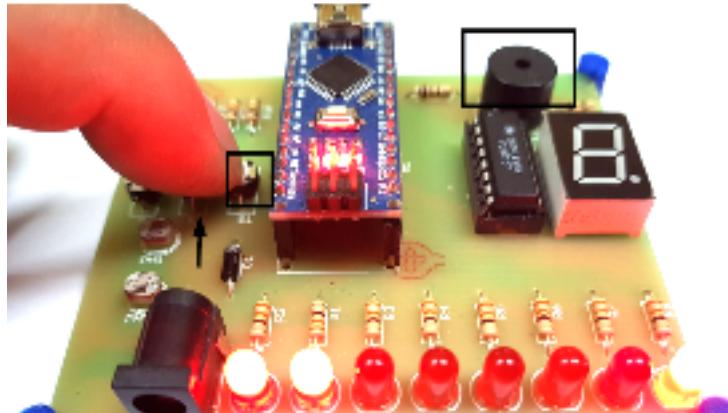
    int sensorLuz = analogRead(A3); // leer sensor PH1 -A3
    Serial.println(sensorLuz); //muestra lectura de sensor
    // map() del LDR a valor de salida de pitch (120-1500Hz)
    // cambiar valores de ser necesario (400-1000)
    // para nuestro caso usamos 350-840 ya que el divisor
    // del ldr es de 10Kohms
    int tono = map(sensorLuz, 350, 840, 120, 1500);

    // play the pitch:
    tone(12, tono, 10); //funcion tone(pin,frecuencia,tiempo)
    delay(1);           // delay mejora la medida
}
```

}

Si presionamos el S1, se inicia el sonido variante con la luz, si cubrimos o dejamos descubierto el sensor de luz *PH1* la frecuencia del sonido cambiará, puedes experimentar con la función *map()* en el código fuente para tonos más agudos o graves.

Tambien ademas se muestra por el puerto serial la frecuencia actual que suena en el buzzer.



93. Estudio Musical

Como vimos en el ejercicio anterior, se pueden generar sonidos con el buzzer, estos sonidos pueden ser definidos mediante frecuencia y tiempo, las notas musicales también tienen esa característica, por lo tanto podemos definir notas musicales también en el *Arduino*.

Ahora, una nota musical está definida por la frecuencia del tono así como la duración del mismo.

Supongamos que una negra dura un segundo, se tendría la siguiente tabla para distintas figuras musicales.

Figura	Tiempo(milis)
Redonda	4000 (4 seg)
Blanca	2000
Negra	1000 (1 seg)
Corchea	500
Semicorchea	250

El procedimiento para implementar musica con el buzzer es:

- Definir la equivalencia entre las frecuencias y las notas musicales (*pitches.h*).
- Crear una lista o array con las notas de la canción
- Crear una lista o array con el duración de cada nota.
- Definir un tiempo de silencio entre notas.
- Implementar un for que recorra la lista de notas y a cada nota le corresponda también una duración, esto se hace con la función *tone(pin,nota,duracion)*.

Una vez visto el procedimiento, vamos a implementar una melodía conocida.

```
*****
Equivalencia NOTA Frecuencia (pitches.h)
*****  

//estas definiciones van en una sola columna,
//se muestran en 3 columnas para ahorrar papel.  

/*  

   Equivalencia de Notas S=Sostenido  

   A ,B, C, D, E, F, G  

   La, Si, Do, Re, Mi, Fa, Sol  

*/  

#define NOTE_B0 31  #define NOTE_B3 247  #define NOTE_B6
#define NOTE_C1 33  #define NOTE_C4 262  1976
#define NOTE_CS1 35 #define NOTE_CS4 277  #define NOTE_C7
#define NOTE_D1 37  #define NOTE_D4 294  2093
#define NOTE_DS1 39 #define NOTE_DS4 311  #define NOTE_CS7
#define NOTE_E1 41  #define NOTE_E4 330  2217
#define NOTE_F1 44  #define NOTE_F4 349  #define NOTE_D7
#define NOTE_FS1 46 #define NOTE_FS4 370  2349
#define NOTE_G1 49  #define NOTE_G4 392  #define NOTE_DS7
#define NOTE_GS1 52 #define NOTE_GS4 415  2489
#define NOTE_A1 55  #define NOTE_A4 440  #define NOTE_E7
#define NOTE_AS1 58 #define NOTE_AS4 466  2637
#define NOTE_B1 62  #define NOTE_B4 494  #define NOTE_F7
#define NOTE_C2 65  #define NOTE_C5 523  2794
#define NOTE_CS2 69 #define NOTE_CS5 554  #define NOTE_FS7
```

```

#define NOTE_D2 73   #define NOTE_D5 587 2960
#define NOTE_DS2 78  #define NOTE_DS5 622  #define NOTE_G7
#define NOTE_E2 82   #define NOTE_E5 659  3136
#define NOTE_F2 87   #define NOTE_F5 698  #define NOTE_GS7
#define NOTE_FS2 93  #define NOTE_FS5 740  3322
#define NOTE_G2 98   #define NOTE_G5 784  #define NOTE_A7
#define NOTE_GS2 104 #define NOTE_GS5 831  3520
#define NOTE_A2 110 #define NOTE_A5 880  #define NOTE_AS7
#define NOTE_AS2 117 #define NOTE_AS5 932  3729
#define NOTE_B2 123 #define NOTE_B5 988  #define NOTE_B7
#define NOTE_C3 131 #define NOTE_C6 1047 #define NOTE_C8
#define NOTE_CS3 139 #define NOTE_CS6 1109 #define NOTE_CS8
#define NOTE_D3 147 #define NOTE_D6 1175 #define NOTE_D8
#define NOTE_DS3 156 #define NOTE_DS6 1245 #define NOTE_DS8
#define NOTE_E3 165 #define NOTE_E6 1319
#define NOTE_F3 175 #define NOTE_F6 1397
#define NOTE_FS3 185 #define NOTE_FS6 1480
#define NOTE_G3 196 #define NOTE_G6 1568
#define NOTE_GS3 208 #define NOTE_GS6 1661
#define NOTE_A3 220 #define NOTE_A6 1760
#define NOTE_AS3 233 #define NOTE_AS6 1865

// notas en la melodía:
int melodía[] = {
    NOTE_C5, NOTE_C5, NOTE_F5, NOTE_C5, 0,

```

```

NOTE_C5, NOTE_D5, NOTE_A4, 0,
NOTE_C5, NOTE_AS4, NOTE_A4, NOTE_A4, NOTE_G4, NOTE_F4, NOTE_F4,
NOTE_C5, 0,
NOTE_C5, NOTE_C5, NOTE_C5, NOTE_C5

/* Notas del Himno Nacional para Lira
Do* Do* Fa* Do*
Do* Re* La
Do* Sib La La SoI Fa Fa Do*
Do* Do* Do*
*/
};

// duracion de nota: 4 = cuarto, 8 = octavo de nota, etc:
//para cada nota le corresponde un tiempo

int duracionNotas[] = {
  4, 4, 2, 2, 4,
  4, 2, 2, 4,
  4, 2, 4, 4, 2, 4, 4, 2, 8,
  8, 8, 8, 4
};

int pin = 12; //pin del buzzer

void setup() { //funcion setup se ejecuta una vez
  Serial.begin(9600);
  // iteracion sobre las notas en la melodía:
  //blanca= 4000, redonda = 2000, negra = 1000
  //corchea=500,semicorchea/250,fusa=125
  //si negra es = 1000 , fusa = 1000/8

  for (int estaNota = 0; estaNota < 22; estaNota = estaNota + 1)
{

```

```
int duracionNota = 1100 / duracionNotas[estaNota];
tone(pin, melodia[estaNota], duracionNota);
Serial.println(melodia[estaNota]);

// Para distinguir las notas se necesita un minimo de
silencio entre ellas
// usando 1.3 se tiene una separacion de 30%, valor estandar.
// se usa 1.3 porque estamos usando la variable de tiempo de
la funcion
// tone, sin esto las notas se solaparian.

int pausaEntreNotas = duracionNota * 1.30;
delay(pausaEntreNotas);
// Detener el sonido:
noTone(pin);
}

}

void loop() {
// la melodia solo se ejecuta una sola vez ya que no hay nada
en loop
}
```

ÍNDICE DE EJERCICIOS

Componentes Básicos de un Programa	1
Mostrar “Hola Mundo” en la pantalla usando Arduino	4
Mostrar “Hola Mundo” en la pantalla usando Arduino y la función setup.	5
Mostrar “Hola Mundo” usando el led incorporado en el Arduino.	6
Calcular la suma de dos números y mostrarla usando el monitor serial.	7
Un programa que calcule el área de un cuadrado	8
El área de un cuadrado con entrada de lado por teclado	9
Hola amigo humano.	11
Cálculo del volumen de una esfera con entrada de datos por teclado.	12
Suma de dos números introducidos por teclado	13
Recibe altura y base de un triángulo y muestra el área del mismo.	15
Convertir grados sexagesimales a radianes	17
Calcular la corriente si se tiene voltaje y resistencia	18
Resolver una ecuación de primer grado.	19
Leer un número y ver si es Positivo o Negativo	26
Compara las edades de dos personas, cual es mayor, menor o si son de la misma edad.	
29	
En que cuadrante trigonométrico esta un grado sexagesimal	32

Cuando un número es PAR o IMPAR	35
Desglosar una cantidad de dinero en billetes y monedas de circulación común.	36
Generar un número al azar, y adivinar qué número es	38
Reciba 3 números y muestra el número máximo introducido	40
Reciba 3 números y muestra el número máximo introducido (otro)	42
Recibe 5 números y muestra el número mayor introducido.	44
Recibe un mes y dice en qué estación estamos.	48
Calcula el área, perímetro y circunferencia de un círculo, mediante un menú	50
El bucle while	54
Muestra los números naturales del 1 al 50.	56
Muestra los números naturales del 1 al 50 de dos en dos	57
Muestra los números naturales del 16 al 72	58
Muestra los números naturales de 0 hasta “n”	60
Muestra números naturales desde un inicio “a”, hasta un final “n”	61
Hacer un programa que imprima números naturales desde un valor “inicio”, hasta un valor “final” y con intervalos de “intervalo”	63
Cálculo de sumatorios, Hacer un programa que sume los primeros 50 números naturales.	66
Hacer un programa que sume números naturales desde “inicio” hasta “final”.	67
Hacer un programa que calcule el factorial de un número “numero”, donde factorial es $n!=1*2*3*...*(n-1)*n$	70
Un programa que suma dos números y del cual solo se salga cuando se presiona una determinada tecla (q)	73
Implemente un programa que imprima números del 0 al 50 usando DO-WHILE	75
Implemente un juego de “adivinar el número” usando la función AZAR(x) y un bucle DO-WHILE. (USAR 4 números a adivinar)	77

Implemente un juego de “adivinar el numero” usando la función AZAR(x) y un bucle DO-WHILE, y que muestre el numero de intentos usados para adivinar el numero. (incrementar el número de intentos).	79
Implementar un programa que sume, reste y multiplique dos números y que se mantenga continuamente operativo sin necesidad de ejecutar el programa nuevamente.	80
Implementar un programa de semáforo en puerto serie	83
Imprimir los números desde 1 hasta 50 usando un bucle FOR (Contador).	87
Imprimir los números desde “0” hasta “n” usando bucle FOR.	88
Imprimir los números desde “n” hasta “m” usando bucle FOR.	90
Imprimir los números desde “n” hasta “m” con intervalos “i” usando bucle FOR (Contador).	92
Hacer un programa que muestre la tabla de multiplicar de un número “N”	94
Hacer un programa que sume los números naturales del 1 al 50 usando FOR	96
Hacer un programa que sume los números naturales del 1 al “n” usando FOR	97
Hacer un programa que sume los números naturales del “n” al “m” usando FOR	99
Cálculo de números Primos.	101
Declaración de una función en Arduino.	106
Definir una función que “SALUDE” a la entrada de un nombre. (Función Saludo)	107
Implementar un programa que calcule el doble de un número, usando funciones.	108
Implementar un programa que calcule el doble y el triple de un número, usando funciones.	109
Implementar un programa que haga un conteo progresivo desde 0 hasta 10 cada 1 segundo	113
Implementar un programa que haga un conteo progresivo desde “n” hasta “m” cada “t” segundo	114
Implementar un programa que haga sumatorias usando funciones	116

Implementar un programa de tabla de multiplicar y tabla de sumar usando funciones	118
Se tienen dos nombres de entrada, Definir una función que haga que un nombre salude a el otro nombre y viceversa (2 funciones).	121
Implementar un programa que calcule la suma del doble del primero por el triple del segundo. A esa operación la llamaremos “patito”.	123
Implementar un programa que calcule el área, perímetro, volumen de un cuadrado, y cubo usando solo el lado como entrada.	125
Implementar un programa que haga un conteo progresivo desde 0 hasta 10 cada 1 segundo y que luego haga el conteo de 10 a 1 cada segundo (efecto auto fantástico) y que se repita continuamente.	128
Implementar un programa que reciba el gasto completo de una compra y calcule el IGV, Valor de venta usando funciones.	131
Implementar un programa de conversión de decimal a binario usando funciones	133
Definir una función que halle el cuadrado de un número x	136
Implementar una función recursiva que halle la sumatoria de 0 hasta n.	141
Cálculo recursivo del número de bits necesarios para representar un número	144
NÚMEROS DE FIBONACCI	146
El Algoritmo de Euclides	149
El número de combinaciones que podemos formar tomando m elementos de un conjunto con n elementos es: osea n es mayor que m	156
Cálculo de Raíz cuadrado con condición de número positivo.	158
Luces ida y vuelta	161
Destello de luces	162
Destello de luces ida y vuelta con for	163
Visualizar Binarios	164
Encender un Led por teclado	165
Led incrementa y regresa	166

Auto fantastico con while	167
Encender un led mediante un pulsador	168
Encender un led con enclavamiento	170
Generar un tono con el buzzer	171
Sirena con Buzzer con while	172
Números del 0 al 9 en display	173
Botones sube y baja binarios	174
Lectura de valores analogicos LDR	175
Calibración de LDR	177
Detección de las manos	179
Subir y Bajar binarios	182
Alarma programada	182
Dado Electronico	187
Sonido controlado por luz.	190
Estudio Musical	192