# *Memoria Práctica 1*

Carlos Moreno Antorrena
Marcos Malandro Pérez
Pablo Ruiz Calvo

# Sección 1

## Ejercicio 1. NIA.

Si A es el primer dígito de su NIA y F el último, diseñe un programa que actúe como un indicador digital de su número de NIA haciendo parpadear el LED43 con cada uno de los valores del NIA, para contar cada número realice parpadeos cortos de 100mS y entre dígitos retrase medio segundo.  Se valorará el uso de funciones para optimizar el código. Mejora1: no use la función delay. Mejora2: haga que el LED44 parpadee con una frecuencia fija de 400mS.

```C/C++
#define NIPLENGHT 6
#define BLUELED 44
#define REDLED 43

int NIP[NIPLENGHT] = {8,2,2,10,10,2};

void setup()
{

  Serial.begin(115200);
  pinMode(BLUELED,OUTPUT);

  Serial.println("BEGIN");
  for (int i =0 ; i<NIPLENGHT ;i++)
  {
    for (int j = 0; j<NIP[i];j++)
    {
      digitalWrite(BLUELED,LOW);
      delay(100);
      digitalWrite(BLUELED,HIGH);
      delay(100);
    }
    delay(1000);
  }

  Serial.println("FIN");
}

void loop()
{
}
```

## Ejercicio 2. Lampara menos relajante.

Algunas de las salidas de arduino se pueden convertir en analógicas usando la técnica PWM. La función utilizada es analogWrite(pin, value) para Arduino Uno o la clase ledc para esp32. Diseñe un programa que haga que el LED43 pase de apagado a encendido de forma progresiva en un tiempo total de B segundos, se mantenga C segundos encendido, se apague de forma progresiva en F segundos y vuelva a empezar. En la salida tendrá que escribir de 0 a 255 usando la función analogWrite o ledc para modular la intensidad del led. Haga que el LED44 luzca de forma inversa al LED43

```C/C++
#include "driver/ledc.h"

#define BLUELED 44
#define GREENLED 43
#define LEDCHANNELBLUE 0
#define LEDCHANNELGREEN 1
#define FREQUENCY 5000
#define RESOLUTION 8



void setup() {
  // put your setup code here, to run once:
  ledcSetup(LEDCHANNELBLUE, FREQUENCY, RESOLUTION);
  ledcSetup(LEDCHANNELGREEN, FREQUENCY, RESOLUTION);
  ledcAttachPin(BLUELED, LEDCHANNELBLUE);
  ledcAttachPin(GREENLED, LEDCHANNELGREEN);
}

void loop() {
  // put your main code here, to run repeatedly:
  for (int dutyCycle=0; dutyCycle<=255; dutyCycle++)
  {
    ledcWrite(LEDCHANNELBLUE, dutyCycle);
    ledcWrite(LEDCHANNELGREEN,255-dutyCycle);
    delay(10);
  }


  for (int dutyCycle=255; dutyCycle>=0; dutyCycle--)
  {
    ledcWrite(LEDCHANNELBLUE, dutyCycle);
    ledcWrite(LEDCHANNELGREEN,255-dutyCycle);
    delay(10);
  }
}
```

# Ejercicio 3. Lampara relajante

La función random devuelve un valor aleatorio https://programarfacil.com/blog/random-en-arduino-como-usar-numeros-aleatorios/ , haga una llamada de random para que le devuelva un valor de 0 a 10 y si el valor devuelto es mayor que 5 incremente la intensidad del led y si es menor que 5 decremente esa intensidad. Tenga en cuenta que la intensidad aplicada al led con la función analogWrite o ledc no puede ser ni superior a 255 ni inferior a 0. Ajuste el valor en que se incrementa / decrementa de la intensidad del led y el tiempo transcurrido entre dos llamadas a random para conseguir cambios de iluminación continuos y no demasiado rápidos. Si tiene un led tricolor, aplique el algoritmo a cada una de los tres leds para obtener una lámpara relajante que vaya modulando su color e intensidad. Los leds externos deben colocarse con una resistencia en serie de 220 ohm

C/C++

```
int R=random(0,256);
int G=random(0,256);
int B=random(0,256);
int randomFlag, ttw, ttd, iR, iG, iB, mult, counter=0;

void setup() {
  // put your setup code here, to run once:

}

void loop() {

  if (millis()-counter>=ttw)
  {
    ttw = random(1,3)*1000/random(1,4);
    ttd = random(2,6);

    if (random(1,11) > 5)
      mult=1;
    else
      mult=-1;

    iR = random(0,3);
    iG = random(0,3);
    iB = random(0,3);

    counter = millis();
  }

  R+=iR*mult;
  G+=iG*mult;
```

```
  B+=iB*mult;

  R=constrain(R,20,255);
  G=constrain(G,20,255);
  B=constrain(B,20,255);

  neopixelWrite(RGB_BUILTIN,R,G,B);

  delay(ttd);
}
```

## Ejercicio 4. Señal sinusoidal.

Use el Arduino como un generador de una señal sinusoidal F*sen(w*t) donde F es el dígito F de su NIA en voltios y w = 1/C rad/s frecuencia de la señal sinusoidal. Para representar la salida utilice SerialPlotter de Arduino. https://www.luisllamas.es/arduino-serial-plotter/

```C/C++
#include <math.h>

const float F = 903699.0;
const float C = 1.0;
const float w = 1.0 / C;

void setup() {
  Serial.begin(115200);
  delay(20);
}

void loop() {
  static unsigned long tiempopasa = 0;
  unsigned long currentTime = millis();

  if (currentTime - tiempopasa >= 1) {
  tiempopasa = currentTime;

    float t = currentTime / 1000.0;
    float signal = F * sin(w * t);

    Serial.println(signal);
  }
}
```

# Sección 2

## Ejercicio 5. Led44.

Diseñe un programa que lea por teclado una secuencia de letras. Cuando encuentre una a, encienda el led44, cuando encuentre una s apague el led44, una b, encienda el led43, cuando encuentre una c apague el led43, cuando encuentre una d haga parpadear el led43 E veces, cuando encuentre una f haga parpadear el led 44 C veces. Tenga en cuenta que el sistema operativo no envía nada al Arduino por el puerto serie hasta que no pulse la tecla Enter y que los caracteres ASCII del Enter también serán enviados al Arduino.

```
C/C++
#define BLUELED 44
#define GREENLED 43


const int E = 3, C = 4;

int serialIndex, i;
char serialBuffer[200];

void setup()
{
  digitalWrite(GREENLED, HIGH);
  Serial.begin(115200);
  Serial.print("BEGIN");
  delay(20);
}

void loop()
{
  i=0;
  serialBuffer[200]={};
  Serial.print("AA");
  digitalWrite(GREENLED, HIGH);

  while(Serial.available())
  {
    serialBuffer[i]=Serial.read();
    delay(10);

    if (serialBuffer[i] == '\n' || serialBuffer[i] == '\r')
    {
      digitalWrite(GREENLED, LOW);
```

```
      serialBuffer[i]='\0';
      break;
    }
    i++;

    if (i>=200)
      i=0;
  }

  for (int j=0; j<200;j++)
  {
    Serial.print(serialBuffer[j]);
  }
  Serial.print('\n');
}
```

## Ejercicio 6. Saludos.

Diseñe un programa que solicita al usuario su nombre y responda saludando al usuario por su nombre

```
C/C++
/*
Hola
¿Cómo te llamas?
Julia
Salida: Hola Julia que tengas un gran día
*/
/* IO with Serial clases */
#define MAXBUF 20
char buffer[MAXBUF];
boolean myReadString(int timeout);
void setup() {
  delay(700);
  Serial.begin(9600);
  delay(700);
  Serial.print("¿Cómo te llamas? \n");
}
void loop(){
  if(myReadString(50)){
       Serial.print("Hola ");
       Serial.print(buffer);
       Serial.println(" que tengas un gran día");
  }
}
```

```
//Working on Arduino Uno
boolean myReadString(int timeout) { //ReadString with timeout limit, return
true if a string is readed within timeout limits
  static int i=0; //Número de caracteres leidos
  static long tLimit; //Reloj interno de timeout
  static boolean reading=false; //Marca si hay una lectura en curso
  while((Serial.available()>0) && (i<(MAXBUF-1))){ //Hay caracteres
disponibles y no hemos desbordado el buffer
       if(!reading){ //First character
       reading=true; //Estamos en un proceso de lectura
       tLimit = millis(); //Ajustamos el reloj de timeout
       i=0; //Iniciamos el índice del buffer
       }
       buffer[i]=Serial.read(); //read next caracter from buffer
       if((buffer[i]=='\n') || (buffer[i]=='\r')){ //Fin de mensaje
       reading = false; //Fin de lectura
       if(i>0){ //No se trata de una linea vacía
       buffer[i] = 0; //Pon la marca de fin de cadena
       return true; //Lectura exitosa
       }
       }
       i=i+1;
       if(millis()-tLimit > timeout){ //Superamos el timeout de la lectura
86uS por caracter a 115200
       Serial.print("TimeOut"); //Mensaje de error
       reading = false; //Fin de lectura
       return false; //Fallo en la lectura
       }
  }
  return false;
}
```

## Ejercicio 7. Suma lista.

Diseñe un programa al que introduzca una lista de números y devuelva la suma de todos ellos, la lista es de longitud variable.

```
C/C++
/*
Entrada: 1 34 56 90 47
Salida: 228
*/
/* IO with Serial clases */
#define MAXBUF 20
```

```
#define MAXVECT 10
char buffer[MAXBUF];
boolean myReadString(int timeout);
void setup() {
  Serial.begin(9600);
  //Serial.printf("Introduce una lista de numeros a sumar separados por
:\n");
  Serial.print("Entrada:\n");
}
void loop(){
  int v[MAXVECT];
  if(myReadString(50)){
       char *p;
       int j=0;
       p=strtok(buffer," "); //Divide the string in numbers tokenized by el
espacio
       while((p!=NULL)&&(j<MAXVECT)){ //This mark the end of string
       v[j++]=atoi(p); //Convert string to int
       p=strtok(NULL," "); //Try to found a new value
       }
       if(j>0){ //Found numbers to add
       int result = 0;
       for(int i=0; i<j;i++){
       result += v[i];
       }
       //Serial.printf("El resultado de la suma es %d\n", result);
       //Serial.printf("Intenta otro grupo de numeros\n");
       Serial.print("Salida:");
       Serial.println(result);
       Serial.print("Entrada:\n");
       }
  }
}
//Working on Arduino Uno
boolean myReadString(int timeout) { //ReadString with timeout limit, return
true if a string is readed within timeout limits
  static int i=0; //Número de caracteres leidos
  static long tLimit; //Reloj interno de timeout
  static boolean reading=false; //Marca si hay una lectura en curso
  while((Serial.available()>0) && (i<(MAXBUF-1))){ //Hay caracteres
disponibles y no hemos desbordado el buffer
       if(!reading){ //First character
       reading=true; //Estamos en un proceso de lectura
       tLimit = millis(); //Ajustamos el reloj de timeout
       i=0; //Iniciamos el índice del buffer
       }
       buffer[i]=Serial.read(); //read next caracter from buffer
       //Serial.print(buffer[i]);
```

```
      if((buffer[i]=='\n') || (buffer[i]=='\r')){ //Fin de mensaje
      reading = false; //Fin de lectura
      if(i>0){ //No se trata de una linea vacía
      buffer[i] = 0; //Pon la marca de fin de cadena
      //Serial.print("Read "); //Depuración
      //Serial.print(i+1);
      //Serial.print(" characters\n");
      return true; //Lectura exitosa
      }
      }
      i=i+1;
      if(millis()-tLimit > timeout){ //Superamos el timeout de la lectura
86uS por caracter a 115200
      Serial.print("TimeOut"); //Mensaje de error
      reading = false; //Fin de lectura
      return false; //Fallo en la lectura
      }
  }
  return false;
}
```

## Ejercicio 8. Primo bool.

Solicite un número entero como entrada y responda indicando si el número es primo no

```
C/C++
/* IO with Serial clases */
#define MAXBUF 20
char buffer[MAXBUF];
boolean myReadString(int timeout);
void setup() {
  delay(700);
  Serial.begin(9600);
  delay(700);
  Serial.print("Dame un número: \n");
}
void loop(){
  if(myReadString(50)){
      int numero = atoi(buffer);
      if (esPrimo(numero)) {
      Serial.print("El número ");
      Serial.print(numero);
```

```
        Serial.println(" es primo");
        } else {
        Serial.print("El número ");
        Serial.print(numero);
        Serial.println(" no es primo");
        }
  }
}
//Working on Arduino Uno
boolean myReadString(int timeout) { //ReadString with timeout limit, return
true if a string is readed within timeout limits
  static int i=0; //Número de caracteres leidos
  static long tLimit; //Reloj interno de timeout
  static boolean reading=false; //Marca si hay una lectura en curso
  while((Serial.available()>0) && (i<(MAXBUF-1))){ //Hay caracteres
disponibles y no hemos desbordado el buffer
        if(!reading){ //First character
        reading=true; //Estamos en un proceso de lectura
        tLimit = millis(); //Ajustamos el reloj de timeout
        i=0; //Iniciamos el índice del buffer
        }
        buffer[i]=Serial.read(); //read next caracter from buffer
        if((buffer[i]=='\n') || (buffer[i]=='\r')){ //Fin de mensaje
        reading = false; //Fin de lectura
        if(i>0){ //No se trata de una linea vacía
        buffer[i] = 0; //Pon la marca de fin de cadena
        return true; //Lectura exitosa
        }
        }
        i=i+1;
        if(millis()-tLimit > timeout){ //Superamos el timeout de la lectura
86uS por caracter a 115200
        Serial.print("TimeOut"); //Mensaje de error
        reading = false; //Fin de lectura
        return false; //Fallo en la lectura
        }
  }
  return false;
}
// Nueva función para determinar si un número es primo
boolean esPrimo(int numero) {
  if (numero <= 1) return false;
  for (int i = 2; i <= sqrt(numero); i++) {
        if (numero % i == 0) return false;
  }
  return true;
}
```

# Ejercicio 9. Fibonacci.

9. El programa solicita un número como entrada y proporciona los valores de la serie de Fibonacci inferiores al valor introducido como entrada
https://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci

```
C/C++
/* IO with Serial clases */
#define MAXBUF 20
char buffer[MAXBUF];
boolean myReadString(int timeout);
void setup() {
  delay(700);
  Serial.begin(9600);
  delay(700);
  Serial.print("Dame un número: \n");
}
void loop(){
  if(myReadString(50)){
       int numero = atoi(buffer);
       Serial.print("Serie de Fibonacci menor que ");
       Serial.print(numero);
       Serial.println(":");
       fibonacci(numero);
  }
}
//Working on Arduino Uno
boolean myReadString(int timeout) { //ReadString with timeout limit, return
true if a string is readed within timeout limits
  static int i=0; //Número de caracteres leidos
  static long tLimit; //Reloj interno de timeout
  static boolean reading=false; //Marca si hay una lectura en curso
  while((Serial.available()>0) && (i<(MAXBUF-1))){ //Hay caracteres
disponibles y no hemos desbordado el buffer
       if(!reading){ //First character
       reading=true; //Estamos en un proceso de lectura
       tLimit = millis(); //Ajustamos el reloj de timeout
       i=0; //Iniciamos el índice del buffer
       }
       buffer[i]=Serial.read(); //read next caracter from buffer
       if((buffer[i]=='\n') || (buffer[i]=='\r')){ //Fin de mensaje
       reading = false; //Fin de lectura
       if(i>0){ //No se trata de una linea vacía
       buffer[i] = 0; //Pon la marca de fin de cadena
       return true; //Lectura exitosa
       }
       }
       i=i+1;
```

```
      if(millis()-tLimit > timeout){ //Superamos el timeout de la lectura
86uS por caracter a 115200
      Serial.print("TimeOut"); //Mensaje de error
      reading = false; //Fin de lectura
      return false; //Fallo en la lectura
      }
  }
  return false;
}
// Nueva función para calcular y mostrar la serie de Fibonacci
void fibonacci(int numero) {
  int a = 0, b = 1, c;
  Serial.print(a);
  Serial.print(" ");
  while (b < numero) {
      Serial.print(b);
      Serial.print(" ");
      c = a + b;
      a = b;
      b = c;
  }
  Serial.println();
}
```

## Ejercicio 10. Factores primos.

El programa solicita un número entero como entrada y responde indicando todos los factores primos del número.
Por ejemplo, si el número es 8 -> 2^3, si el número es
24 -> 2^3; 3^1

```
C/C++
/*
Por ejemplo, si el número es 8 -> 2^3, si el número es
24 -> 2^3; 3^1
*/
/* IO with Serial clases */
#define MAXBUF 20
char buffer[MAXBUF];
boolean myReadString(int timeout);
void setup() {
  delay(700);
  Serial.begin(9600);
  delay(700);
```

```
  Serial.print("Dame un número: \n");
}
void loop(){
  if(myReadString(50)){
       int numero = atoi(buffer);
       Serial.print("Factores primos de ");
       Serial.print(numero);
       Serial.println(":");
       factoresPrimos(numero);
  }
}
//Working on Arduino Uno
boolean myReadString(int timeout) { //ReadString with timeout limit, return
true if a string is readed within timeout limits
  static int i=0; //Número de caracteres leidos
  static long tLimit; //Reloj interno de timeout
  static boolean reading=false; //Marca si hay una lectura en curso
  while((Serial.available()>0) && (i<(MAXBUF-1))){ //Hay caracteres
disponibles y no hemos desbordado el buffer
       if(!reading){ //First character
       reading=true; //Estamos en un proceso de lectura
       tLimit = millis(); //Ajustamos el reloj de timeout
       i=0; //Iniciamos el índice del buffer
       }
       buffer[i]=Serial.read(); //read next caracter from buffer
       if((buffer[i]=='\n') || (buffer[i]=='\r')){ //Fin de mensaje
       reading = false; //Fin de lectura
       if(i>0){ //No se trata de una linea vacía
       buffer[i] = 0; //Pon la marca de fin de cadena
       return true; //Lectura exitosa
       }
       }
       i=i+1;
       if(millis()-tLimit > timeout){ //Superamos el timeout de la lectura
86uS por caracter a 115200
       Serial.print("TimeOut"); //Mensaje de error
       reading = false; //Fin de lectura
       return false; //Fallo en la lectura
       }
  }
  return false;
}
// Nueva función para calcular y mostrar los factores primos
void factoresPrimos(int numero) {
  for (int i = 2; i <= numero; i++) { // Itera desde 2 hasta el número
ingresado
       int exponente = 0; // Inicializa el exponente
       while (numero % i == 0) { // Mientras el número sea divisible por i
```

```
        numero /= i; // Divide el número por i
        exponente++; // Incrementa el exponente
        }
        if (exponente > 0) { // Si el exponente es mayor que 0
        Serial.print(i); // Imprime el factor primo
        Serial.print("^"); // Imprime el símbolo de exponente
        Serial.print(exponente); // Imprime el exponente
        if (numero > 1) { // Si el número aún no es 1
        Serial.print("; "); // Imprime un separador
        }
        }
  }
  Serial.println();
}
```

# Sección 3

## Ejercicio 11. Strncat.

strncat es una función de librería que copia una cadena al final de otra
char *strncat(char *dest, char *src, size_t n)
Escriba el código que debe contener la función strncat para que se realice la copia, tenga en cuenta que dest debe tener espacio suficiente para contener la combinación de ambas cadenas.

```C/C++
// Implementación de la función strncat
char *my_strncat(char *dest, const char *src, size_t n) {
  // Guardamos el puntero original para devolverlo al final
  char *original_dest = dest;

  // Encontramos el final de la cadena destino
  while (*dest != '\0') {
       dest++;
  }

  // Copiamos hasta n caracteres de src a dest
  size_t i = 0;
  while (i < n && *src != '\0') {
       *dest = *src;
```

```
        dest++;
        src++;
        i++;
    }

    // Aseguramos que la cadena resultante termine con '\0'
    *dest = '\0';

    // Devolvemos el puntero original
    return original_dest;
}

// Definimos el tamaño máximo del buffer
#define MAX_BUFFER 100

void setup() {
    // Inicializamos la comunicación serial
    Serial.begin(9600);
    while (!Serial) {
        ; // Esperamos a que el puerto serial se conecte
    }

    Serial.println("Prueba de implementación de strncat");

    // Caso 1: Concatenación normal
    char buffer1[MAX_BUFFER] = "Hola ";
    Serial.print("Cadena original: ");
    Serial.println(buffer1);
    my_strncat(buffer1, "mundo!", 6);
    Serial.print("Después de strncat: ");
    Serial.println(buffer1);
    Serial.println();

    // Caso 2: Límite de caracteres
    char buffer2[MAX_BUFFER] = "Prueba: ";
    Serial.print("Cadena original: ");
    Serial.println(buffer2);
    my_strncat(buffer2, "123456789abcd", 10);
    Serial.print("Después de strncat con límite de 10 caracteres: ");
    Serial.println(buffer2);
    Serial.println();

    // Caso 3: Cadena vacía como destino
    char buffer3[MAX_BUFFER] = "";
    Serial.print("Cadena original (vacía): ");
    Serial.println(buffer3);
    my_strncat(buffer3, "Añadiendo a cadena vacía", 15);
    Serial.print("Después de strncat: ");
```

```
  Serial.println(buffer3);
  Serial.println();

  // Caso 4: n = 0 (no debería copiar nada)
  char buffer4[MAX_BUFFER] = "Test ";
  Serial.print("Cadena original: ");
  Serial.println(buffer4);
  my_strncat(buffer4, "no se copia nada", 0);
  Serial.print("Después de strncat con n=0: ");
  Serial.println(buffer4);
  Serial.println();
}

void loop() {
  // No hacemos nada en el loop
}
```

## Ejercicio 12. Strinit.

bool strinit(char *s, char *t) es una función que devuelve 1 si la cadena t representa el principio de la cadena s o 0 en caso contrario. Implemente dicha función.

```C/C++
// Implementación de la función strinit
bool strinit(char *s, char *t) {
  // Si t es una cadena vacía, siempre es el inicio de cualquier cadena
  if (*t == '\0') {
       return true;
  }

  // Comparamos caracter por caracter
  while (*t != '\0') {
       // Si s termina antes que t o los caracteres no coinciden
       if (*s == '\0' || *s != *t) {
       return false;
       }
       // Avanzamos ambos punteros
       s++;
       t++;
  }

  // Si llegamos aquí, significa que recorrimos toda la cadena t
```

```
  // y todos los caracteres coincidieron con el inicio de s
  return true;
}

// Definimos el tamaño máximo del buffer
#define MAX_BUFFER 100

void setup() {
  // Inicializamos la comunicación serial
  Serial.begin(9600);
  while (!Serial) {
       ; // Esperamos a que el puerto serial se conecte
  }

  Serial.println("Prueba de implementación de strinit");

  // Caso 1: t es el inicio de s
  char s1[] = "Hola mundo";
  char t1[] = "Hola";
  Serial.print("¿'" + String(t1) + "' es el inicio de '" + String(s1) + "'?
");
  Serial.println(strinit(s1, t1) ? "Sí" : "No");

  // Caso 2: t no es el inicio de s
  char s2[] = "Hola mundo";
  char t2[] = "mundo";
  Serial.print("¿'" + String(t2) + "' es el inicio de '" + String(s2) + "'?
");
  Serial.println(strinit(s2, t2) ? "Sí" : "No");

  // Caso 3: t es más largo que s
  char s3[] = "Hola";
  char t3[] = "Hola mundo";
  Serial.print("¿'" + String(t3) + "' es el inicio de '" + String(s3) + "'?
");
  Serial.println(strinit(s3, t3) ? "Sí" : "No");

  // Caso 4: t es una cadena vacía
  char s4[] = "Hola";
  char t4[] = "";
  Serial.print("¿'" + String(t4) + "' es el inicio de '" + String(s4) + "'?
");
  Serial.println(strinit(s4, t4) ? "Sí" : "No");

  // Caso 5: s y t son iguales
  char s5[] = "Hola";
  char t5[] = "Hola";
```

```
  Serial.print("¿'" + String(t5) + "' es el inicio de '" + String(s5) + "'?
");
  Serial.println(strinit(s5, t5) ? "Sí" : "No");
}

void loop() {
  // No hacemos nada en el loop
}
```

## Ejercicio 13. Bubble Sort.

implemente una función que se encargue de ordenar los elementos de un vector numérico. void ordena(int v[]), a la función se le pasa el vector desordenado y lo devuelve ordenado de mayor a menor. El algoritmo tradicional para la ordenación es el de la burbuja https://es.wikipedia.org/wiki/Ordenamiento_de_burbuja

```
C/C++
#define MAXBUF 20
char buffer[MAXBUF];
boolean myReadString(int timeout);

// Función para ordenar un vector de enteros de mayor a menor usando el
algoritmo de burbuja
void ordena(int v[], int longitud) {
  // Implementación del algoritmo de burbuja para ordenación descendente
  for (int i = 0; i < longitud - 1; i++) {
       for (int j = 0; j < longitud - i - 1; j++) {
       // Comparamos elementos adyacentes y los intercambiamos si están en el
orden incorrecto
       // Para orden descendente, intercambiamos si el elemento actual es
menor que el siguiente
       if (v[j] < v[j + 1]) {
       // Intercambio de elementos
       int temp = v[j];
       v[j] = v[j + 1];
       v[j + 1] = temp;
       }
       }
  }
}

void setup() {
  delay(700);
  Serial.begin(9600);
```

```
  delay(700);
  Serial.println("Programa de ordenación de vectores");
  Serial.println("Introduce números separados por espacios:");
}

void loop() {
  int v[MAXBUF]; // Vector para almacenar los números

  if (myReadString(50)) {
       char *p;
       int j = 0;

       p = strtok(buffer, " ");
       while ((p != NULL) && (j < MAXBUF)) {
       v[j++] = atoi(p); // atoi = ASCII to Integral
       p = strtok(NULL, " "); // Buscar el siguiente valor
       }

       if (j > 0) { // Si se encontraron números
       Serial.print("Vector original: ");
       for (int i = 0; i < j; i++) {
       Serial.print(v[i]);
       Serial.print(" ");
       }
       Serial.println();

       // Ordenar el vector
       ordena(v, j);

       Serial.print("Vector ordenado (mayor a menor): ");
       for (int i = 0; i < j; i++) {
       Serial.print(v[i]);
       Serial.print(" ");
       }
       Serial.println();

       Serial.println("\nIntroduce más números para ordenar:");
       }
  }
}

// Función para leer una cadena con límite de tiempo
boolean myReadString(int timeout) { // ReadString with timeout limit, return
true if a string is read within timeout limits
  static int i = 0; // Número de caracteres leídos
  static long tLimit; // Reloj interno de timeout
  static boolean reading = false; // Marca si hay una lectura en curso
```

```
  while ((Serial.available() > 0) && (i < (MAXBUF - 1))) { // Hay caracteres
disponibles y no hemos desbordado el buffer
        if (!reading) { // First character
        reading = true; // Estamos en un proceso de lectura
        tLimit = millis(); // Ajustamos el reloj de timeout
        i = 0; // Iniciamos el índice del buffer
        }

        buffer[i] = Serial.read(); // read next character from buffer
        if ((buffer[i] == '\n') || (buffer[i] == '\r')) { // Fin de mensaje
        reading = false; // Fin de lectura
        if (i > 0) { // No se trata de una línea vacía
        buffer[i] = 0; // Pon la marca de fin de cadena
        return true; // Lectura exitosa
        }
        }

        i = i + 1;
        if (millis() - tLimit > timeout) { // Superamos el timeout de la
lectura
        Serial.println("TimeOut"); // Mensaje de error
        reading = false; // Fin de lectura
        return false; // Fallo en la lectura
        }
  }

  return false;
}
```

## Ejercicio 14. Quicksort.

el algoritmo de la burbuja no es muy óptimo, repita el ejercicio anterior usando el algoritmo QuickShort https://es.wikipedia.org/wiki/Quicksort

```C/C++
#define MAXBUF 20
char buffer[MAXBUF];
boolean myReadString(int timeout);

// Función para intercambiar dos elementos
void swap(int* a, int* b) {
  int temp = *a;
  *a = *b;
```

```
  *b = temp;
}

// Función para encontrar la partición
int partition(int arr[], int low, int high) {
  // Seleccionamos el último elemento como pivote
  int pivot = arr[high];
  // Índice del elemento más pequeño
  int i = (low - 1);

  for (int j = low; j <= high - 1; j++) {
       // Si el elemento actual es mayor que el pivote (para orden
descendente)
       if (arr[j] > pivot) {
       i++;
       swap(&arr[i], &arr[j]);
       }
  }
  swap(&arr[i + 1], &arr[high]);
  return (i + 1);
}

// Función principal de QuickSort
void quickSort(int arr[], int low, int high) {
  if (low < high) {
       // pi es el índice de partición
       int pi = partition(arr, low, high);

       // Ordenar elementos antes y después de la partición
       quickSort(arr, low, pi - 1);
       quickSort(arr, pi + 1, high);
  }
}

// Función para ordenar un vector usando QuickSort
void ordena(int v[], int longitud) {
  if (longitud > 1) {
       quickSort(v, 0, longitud - 1);
  }
}

void setup() {
  delay(700);
  Serial.begin(9600);
  delay(700);
  Serial.println("Programa de ordenación de vectores usando QuickSort");
  Serial.println("Introduce números separados por espacios:");
}
```

```
void loop() {
  int v[MAXBUF]; // Vector para almacenar los números

  if (myReadString(50)) {
       char *p;
       int j = 0;

       p = strtok(buffer, " ");
       while ((p != NULL) && (j < MAXBUF)) {
       v[j++] = atoi(p); // atoi = ASCII to Integral
       p = strtok(NULL, " "); // Buscar el siguiente valor
       }

       if (j > 0) { // Si se encontraron números
       Serial.print("Vector original: ");
       for (int i = 0; i < j; i++) {
       Serial.print(v[i]);
       Serial.print(" ");
       }
       Serial.println();

       // Ordenar el vector usando QuickSort
       ordena(v, j);

       Serial.print("Vector ordenado (mayor a menor): ");
       for (int i = 0; i < j; i++) {
       Serial.print(v[i]);
       Serial.print(" ");
       }
       Serial.println();

       Serial.println("\nIntroduce más números para ordenar:");
       }
  }
}

// Función para leer una cadena con límite de tiempo
boolean myReadString(int timeout) { // ReadString with timeout limit, return
true if a string is read within timeout limits
  static int i = 0; // Número de caracteres leídos
  static long tLimit; // Reloj interno de timeout
  static boolean reading = false; // Marca si hay una lectura en curso

  while ((Serial.available() > 0) && (i < (MAXBUF - 1))) { // Hay caracteres
disponibles y no hemos desbordado el buffer
       if (!reading) { // First character
       reading = true; // Estamos en un proceso de lectura
```

```
        tLimit = millis(); // Ajustamos el reloj de timeout
        i = 0; // Iniciamos el índice del buffer
        }

        buffer[i] = Serial.read(); // read next character from buffer
        if ((buffer[i] == '\n') || (buffer[i] == '\r')) { // Fin de mensaje
        reading = false; // Fin de lectura
        if (i > 0) { // No se trata de una línea vacía
        buffer[i] = 0; // Pon la marca de fin de cadena
        return true; // Lectura exitosa
        }
        }

        i = i + 1;
        if (millis() - tLimit > timeout) { // Superamos el timeout de la
lectura
        Serial.println("TimeOut"); // Mensaje de error
        reading = false; // Fin de lectura
        return false; // Fallo en la lectura
        }
  }

  return false;
}
```

## Ejercicio 15. Torres de Hanoi.

El juego, en su forma más tradicional, consiste en tres postes verticales. En uno de los postes se apila un número indeterminado de discos perforados por su centro (elaborados de madera), que determinará la complejidad de la solución. Por regla general se consideran siete discos. Los discos se apilan sobre uno de los postes en tamaño decreciente de abajo arriba. No hay dos discos iguales, y todos ellos están apilados de mayor a menor radio -desde la base del poste hacia arriba- en uno de los postes, quedando los otros dos postes vacíos. El juego consiste en pasar todos los discos desde el poste ocupado (es decir, el que posee la torre) al tercer poste pudiendo usarse el poste central como ayuda.
El programa comenzará solicitando el número de discos que se van a usar y comenzará a indicar los movimientos que se producen entre los postes. Los discos se numeran desde 1 el de radio menor a n el de radio mayor.

```
C/C++
#define MAXBUF 20
char buffer[MAXBUF];
```

```
boolean myReadString(int timeout);

// Función para resolver las Torres de Hanoi mediante recursión
void hanoi(int n, char origen, char auxiliar, char destino) {
  if (n == 1) {
       // Caso base: mover un disco directamente del origen al destino
       Serial.print("Mover disco 1 desde poste ");
       Serial.print(origen);
       Serial.print(" hasta poste ");
       Serial.println(destino);
       return;
  }

  // Mover n-1 discos desde origen a auxiliar usando destino como auxiliar
  hanoi(n-1, origen, destino, auxiliar);

  // Mover el disco n desde origen a destino
  Serial.print("Mover disco ");
  Serial.print(n);
  Serial.print(" desde poste ");
  Serial.print(origen);
  Serial.print(" hasta poste ");
  Serial.println(destino);

  // Mover n-1 discos desde auxiliar a destino usando origen como auxiliar
  hanoi(n-1, auxiliar, origen, destino);
}

void setup() {
  delay(700);
  Serial.begin(9600);
  delay(700);
  Serial.println("Torres de Hanoi");
  Serial.println("Introduce el numero de discos: ");
}

void loop() {
  if (myReadString(50)) {
       int numDiscos = atoi(buffer);

       if (numDiscos > 0) {
       Serial.print("Resolviendo Torres de Hanoi con ");
       Serial.print(numDiscos);
       Serial.println(" discos:");

       // Resolver Torres de Hanoi
       // Origen: 'A', Auxiliar: 'B', Destino: 'C'
       hanoi(numDiscos, 'A', 'B', 'C');
```

```
        // Mostrar el número total de movimientos
        int totalMovimientos = pow(2, numDiscos) - 1;
        Serial.print("Total de movimientos: ");
        Serial.println(totalMovimientos);

        // Solicitar nuevo número de discos
        Serial.println("\nIntroduce el numero de discos: ");
        } else {
        Serial.println("Por favor, introduce un numero positivo de discos.");
        }
  }
}

// Función para leer una cadena de la entrada serial con límite de tiempo
boolean myReadString(int timeout) { // ReadString with timeout limit, return
true if a string is readed within timeout limits
  static int i=0; // Número de caracteres leidos
  static long tLimit; // Reloj interno de timeout
  static boolean reading=false; // Marca si hay una lectura en curso
  while((Serial.available()>0) && (i<(MAXBUF-1))){ // Hay caracteres
disponibles y no hemos desbordado el buffer
        if(!reading){ // First character
        reading=true; // Estamos en un proceso de lectura
        tLimit = millis(); // Ajustamos el reloj de timeout
        i=0; // Iniciamos el índice del buffer
        }
        buffer[i]=Serial.read(); // read next caracter from buffer
        if((buffer[i]=='\n') || (buffer[i]=='\r')){ // Fin de mensaje
        reading = false; // Fin de lectura
        if(i>0){ // No se trata de una linea vacía
        buffer[i] = 0; // Pon la marca de fin de cadena
        return true; // Lectura exitosa
        }
        }
        i=i+1;
        if(millis()-tLimit > timeout){ // Superamos el timeout de la lectura
86uS por caracter a 115200
        Serial.print("TimeOut"); // Mensaje de error
        reading = false; // Fin de lectura
        return false; // Fallo en la lectura
        }
  }
  return false;
}
```