

Tampico, Tamaulipas a 18 de Septiembre de 2022



VERDAD, BELLEZA, PROBIDAD

UAT



Facultad de Ingeniería  
Arturo Narro Siller

# Ejercicios en Clase

## Equipo 1

Nombres: **Gonzalez Saldivar Luis Roberto**

**Guerreo Gamez Francisco Javier**

**Martinez Reyes Fernando**

**Sanchez Ramirez Alan Ariel**

**Villalobos de Leon Juan Carlos**

Profesor: **Dr. García Ruiz Alejandro Humberto**

Asignatura: **Diseño Electronico Basado en Sistemas  
Embebidos**

**8vo. Semestre – Grupo “I”**

**2022-2**

# Índice

Índice .....	2
Ejercicio 1. Leds .....	3
Ejercicio 2. Puerto Serial .....	5
Ejercicio 3. Leds 2 .....	7
Ejercicio 4. Leds 3 .....	12
Ejercicio 5. Suma.....	15
Ejercicio 6. Suma 2.....	17
Ejercicio 7. Área de Rectángulo .....	20
Ejercicio 8. Promedio .....	23

## Ejercicio 1. Leds

**Descripción:** Realizar un programa en Arduino que sea capaz de encender y apagar el led de la placa de Arduino.

### Introducción:

Se requiere conocimiento de las siguientes instrucciones de Arduino:

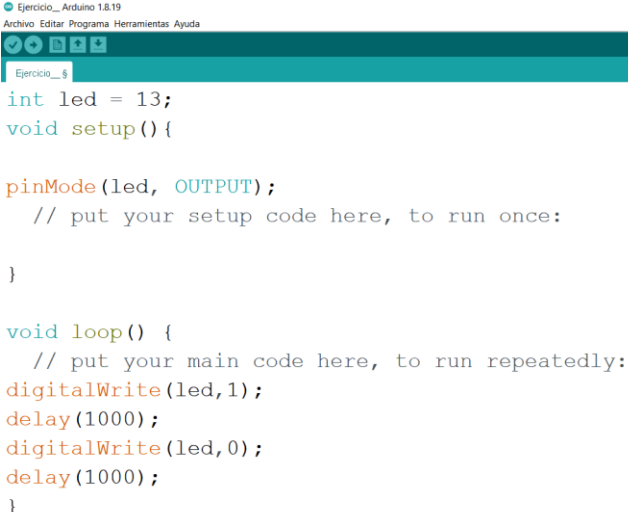
**digitalWrite:** Sirve para leer un valor (o poner en un estado) un pin digital

**delay:** Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.

`delay(1000)` //para esperar un segundo.

**pinMode:** sirve para configurar el modo de trabajo de un pin pudiendo ser INPUT/OUTPUT

### Desarrollo:



```
Ejercicio_ Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda
Ejercicio_s
int led = 13;
void setup() {

  pinMode(led, OUTPUT);
  // put your setup code here, to run once:

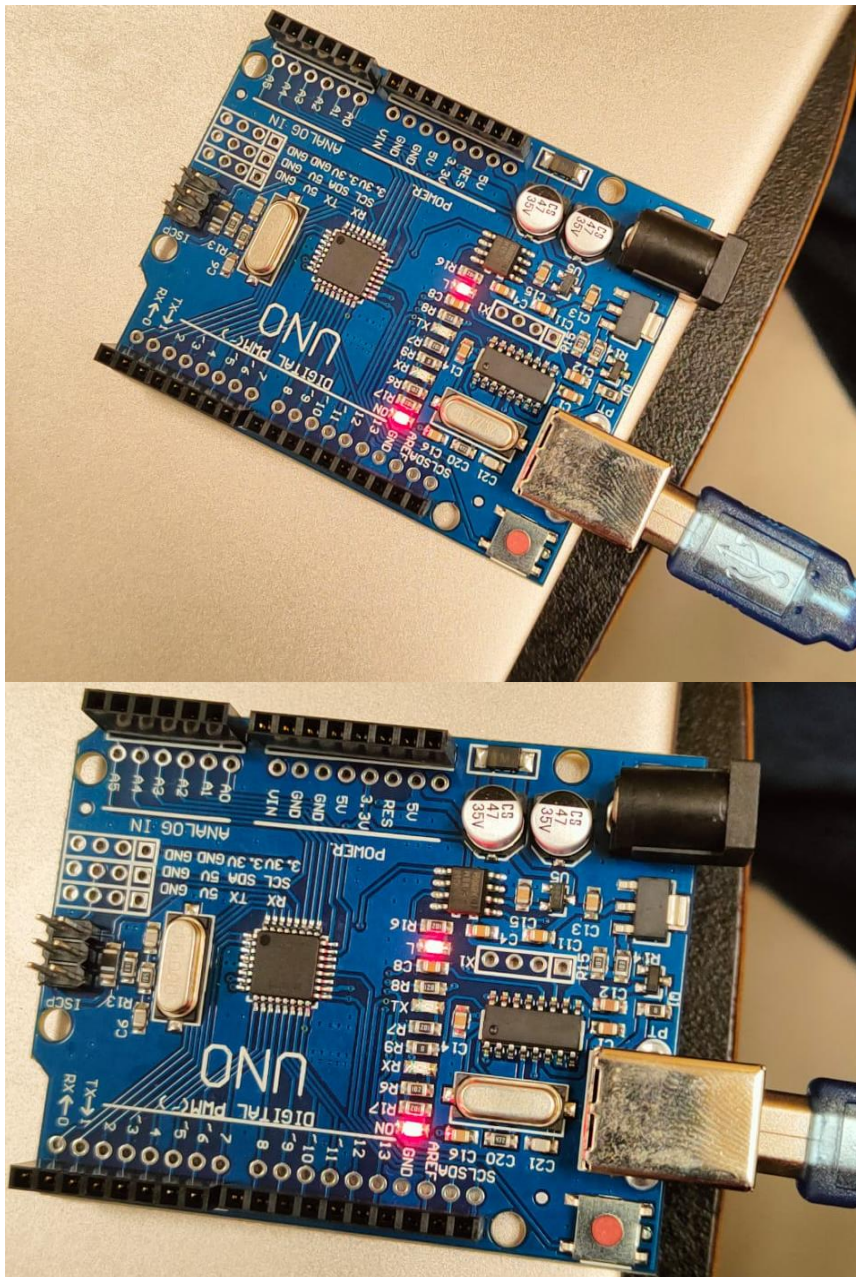
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(led,1);
  delay(1000);
  digitalWrite(led,0);
  delay(1000);
}
```

Primero se declara e instancia un variable entera llamada led con el valor de 13, y en el método setup usamos el pinMode en este caso será OUTPUT, además en el método de loop tenemos la función digitalWrite y se le asigna la variable led, 1, para que prenda el led del

arduino, y después con la función delay, esperamos 1 segundo para ejecutar la segunda instrucción de digitalWrite y poder apagar el led con el valor en 0.

## Resultados:



## **Ejercicio 2. Puerto Serial**

**Descripción:** Realizar un programa en Arduino que sea capaz de imprimir el valor que el usuario ingrese.

### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estandar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función `Serial`.

`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.

`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al el puerto serial.

`delay`: Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.

`delay(1000)` //para esperar un segundo.

### **Desarrollo:**

```
Ejercicio_2 Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda
Ejercicio_2 $
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.setTimeout(1000); //ms por defecto = 1000
}

String v;
void loop() {
    // put your main code here, to run repeatedly:
    if(Serial.available()>0){
        v = Serial.readString();
        Serial.println(v);
    }
}
```

E loop, se creó un if poniendo como condición `Serial.available()>0` esto con el fin de que si el usuario ingreso algún dato `serial.available` dará un valor diferente de 0 por tanto es verdadero y pasara la siguiente instrucción la cual es que la variable `v` se le asigna el valor que será leído a través de la instrucción `Serial.readString`, el cual convertirá el valor recibido a String y después se imprimirá en la pantalla el valor de `v`.

## Resultados:



### **Ejercicio 3. Leds 2**

**Descripción:** Realizar un programa en Arduino que sea capaz de encender y apagar el led de la placa de Arduino, que sea a decisión del usuario.

#### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estandar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función Serial.

`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.

`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al el puerto serial.

`digitalWrite`: Sirve para leer un valor (o poner en un estado) un pin digital

`delay`: Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.  
`delay(1000)` //para esperar un segundo.

#### **Desarrollo:**

```
Ejercicio_3 Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda

Ejercicio_3

int led = 13;
void setup() {
  // put your setup code here, to run once:
  pinMode(led, OUTPUT);
  Serial.begin(9600);
  Serial.setTimeout(1000); //ms por defecto = 1000
}

String v;
int estado_led;
void loop() {
  // put your main code here, to run repeatedly:

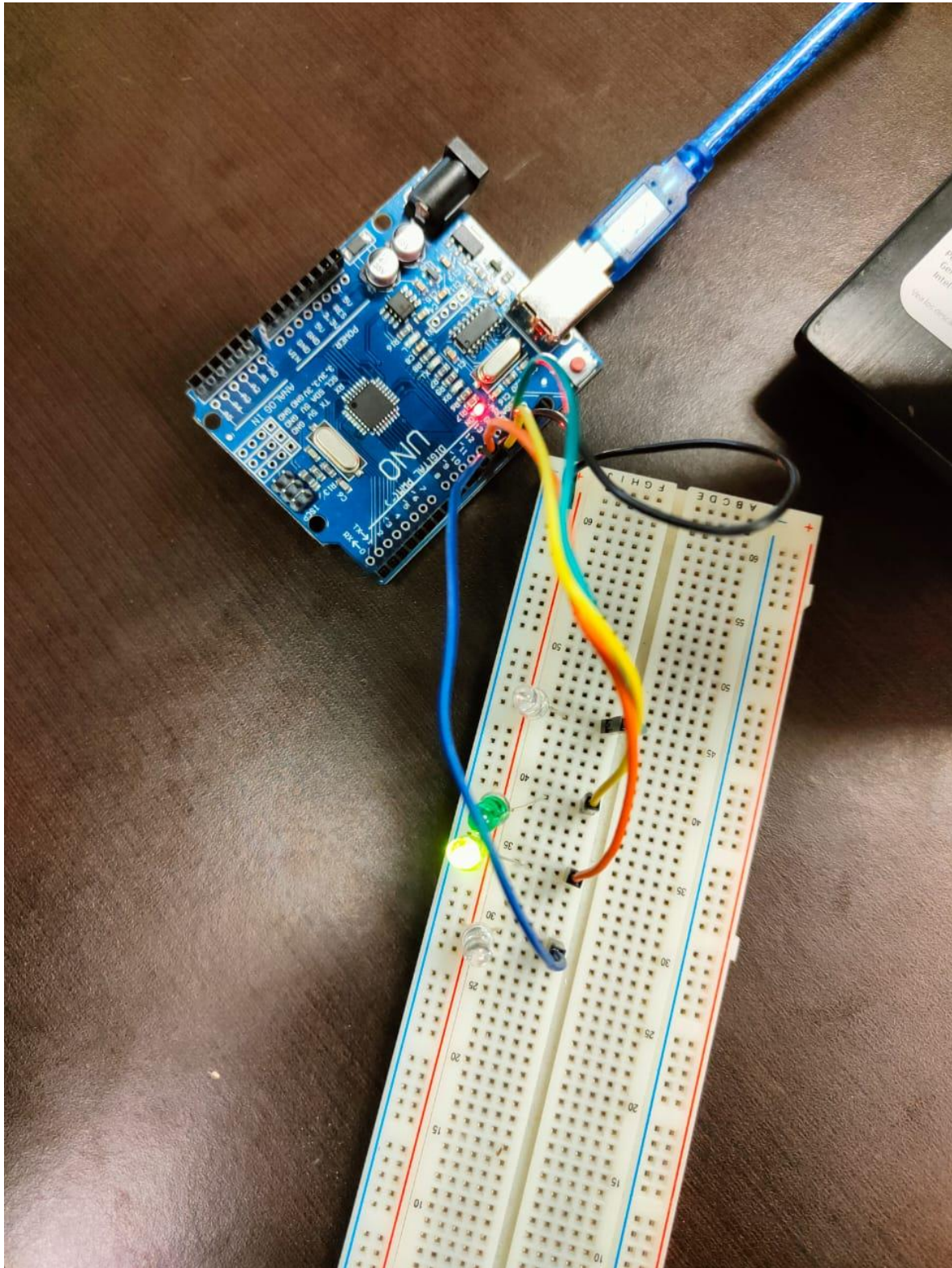
  if(Serial.available() > 0) {
    v = Serial.readString();
    estado_led = v.toInt();
    Serial.println(v);
    digitalWrite(led, estado_led);
  }
}
```

Se inicializa y asigna la variable led de tipo int con valor de 13, posteriormente, se asigna el pinMode con la variable led, y será OUTPUT, se establece el tiempo de recepción de datos con setTimeout con un valor de 1 segundo, posteriormente, se crea una variable de tipo String llamada v, otra variable de tipo int llamada estado\_led, y en el loop, se crea un if si detecta que el usuario ingreso algún valor se cumple la condición con el serial available, después se guarda y convierte el valor de v en string y en estado\_led se convierte el valor de v a int y se guarda para que por ultimo se ejecute la instrucción de digitalWrite que se toma como parámetros led y estado\_led, el led solo se prendera si el valor ingresado es un número mayor o igual a 1 y se apagara si el usuario ingresa letras o el valor 0

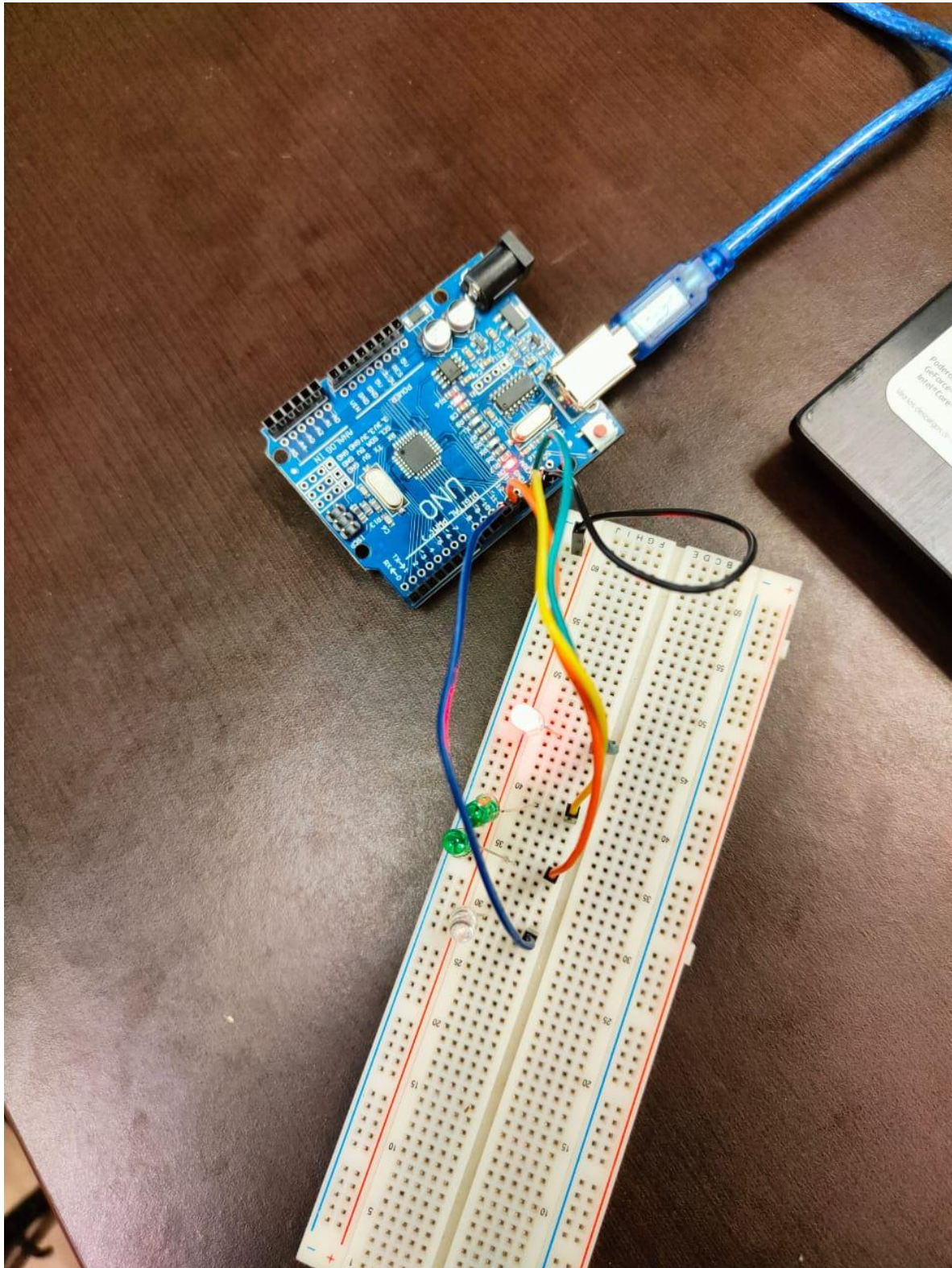
## Resultados:

```
COM4
|
5
|
H o l a
|
1
|
```

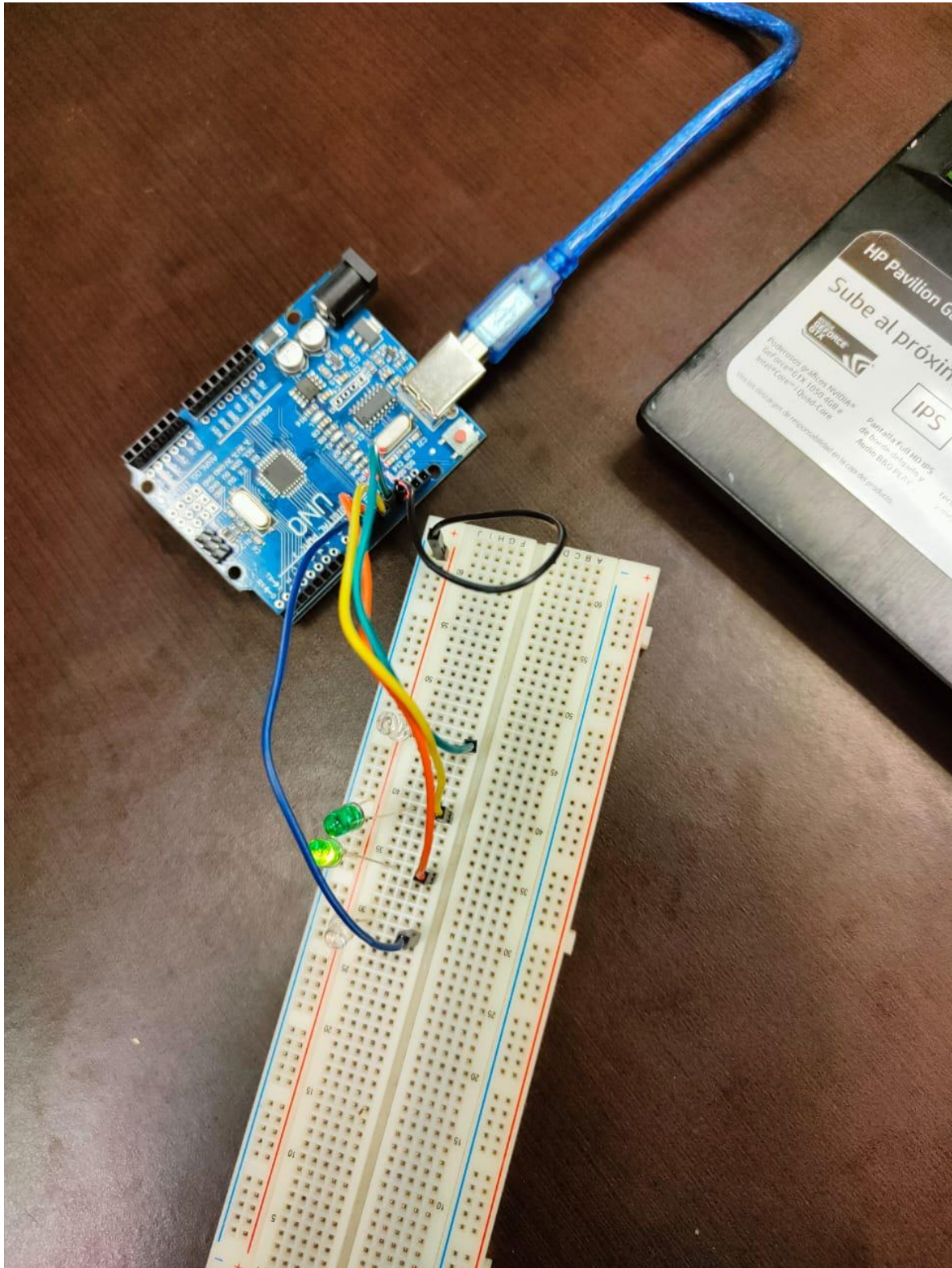












## **Ejercicio 4. Leds 3**

**Descripción:** Realizar un programa en Arduino que sea capaz de encender y apagar el led de la placa de Arduino, que sea a controlado.

### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estandar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función Serial.

`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.

`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al el puerto serial.

`digitalWrite`: Sirve para leer un valor (o poner en un estado) un pin digital

`delay`: Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.  
`delay(1000)` //para esperar un segundo.

`pinMode`: sirve para configurar el modo de trabajo de un pin pudiendo ser INPUT/OUTPUT

### **Desarrollo:**

#### Ejercicio\_4 Arduino 1.8.19

Archivo Editar Programa Herramientas Ayuda



Ejercicio\_4

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

void setup()
{
    Serial.begin(9600);
    Serial.setTimeout(10);

    for(int i = 0; i<4; i++)
    {
        pinMode(leds[i], OUTPUT);
    }
}

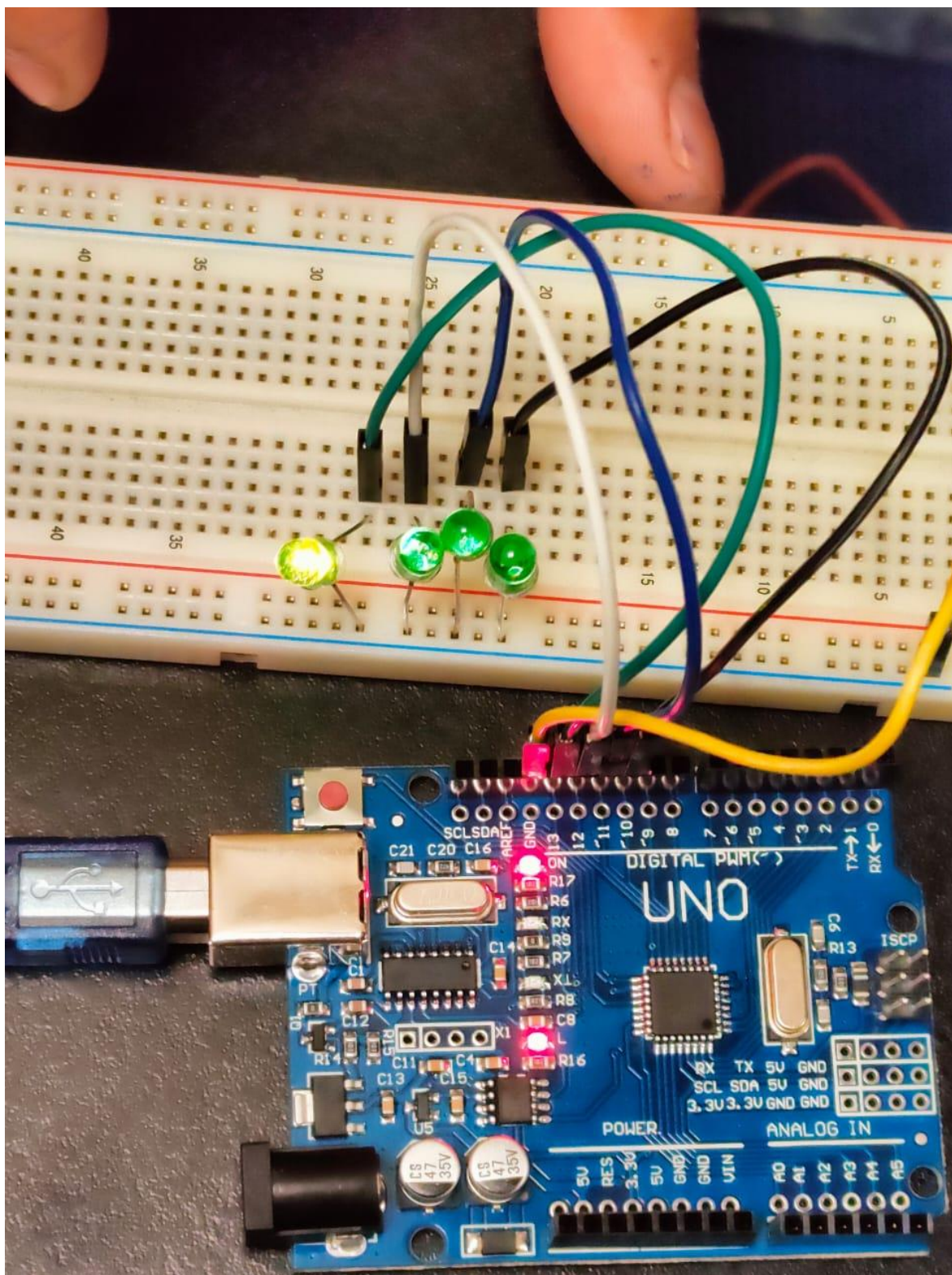
String v;
int estado_led;
void loop() {

    if(Serial.available()>0)
    {
        v = Serial.readString();
        estado_led = v.toInt();
        for(int i = 0; i<4; i++)
        {
            if(estado_led == i)
            {
                digitalWrite(leds[i], HIGH);
            }
            else
            {
                digitalWrite(leds[i], LOW);
            }
        }
    }
}
```

Primero se declara una variable de tipo entero llamada leds que además es un arreglo que se le asignan los valores 10,11,12 y 13, después en el setup se establece el tiempo que esperara el programa para recibir datos, y se crea un bucle for con i inicializado en 0 y con la condición de que se repita hasta que i sea menor que 4 y luego i se incrementa en 1, esto para poner el PinMode en Output, después se crean las dos variables v y estado\_led, string e int respectivamente, y en loop se crea un if para detectar si se ingresó algún dato y si lo hizo se guarda en la variable v como dato de tipo string después se convierte en int y se guarda en la variable estado\_led, después se crea un for exactamente igual como el anterior y se crea un if si estado\_led es igual a el valor de i y con digitalWrite y de parámetros el arreglo de leds en i y HIGH se prendera el led y de no ser iguales los valores, el led se apagará con LOW

#### Resultados:





## **Ejercicio 5. Suma**

**Descripción:** Realizar un programa en Arduino que sea capaz de sumar 2 números ingresados por el usuario.

### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estandar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función Serial.

`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.

`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al puerto serial.

### **Desarrollo:**

```
P_5_SumaDosNums Arduino 1.8.19
Archivo Editar Programa Herramientas Ayuda

P_5_SumaDosNums $
int num1, num2;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.setTimeout(10);
}

int estado = 0;
int v;
void loop() {
  // put your main code here, to run repeatedly:

  if(Serial.available()>0)//Si hay informacion
  {
    v = Serial.readString().toInt();

    if(estado==0)
    {
      num1 = v;
      Serial.println(String(num1));
      estado = 1;
    }else if(estado==1)
    {
      num2 = v;
      Serial.println(String(num2));
      estado = 2;
    }
  }
  if(estado == 2)
  {
    Serial.println(String(num1+num2));
    estado = 0;
  }
}
```

Primero se declaran 2 variables de tipo int llamadas num1 y num2, a continuación se declara otra variable tipo int llamada estado y se le asigna el valor de 0, se declara otra variable tipo int llamada v y en el método loop se crea un if para saber si el usuario ha mandado información y posteriormente se le asigna dicha entrada la variable v, previamente leída con la función Serial.readString y convertida a int por la función toInt, posteriormente se crea un if estado es igual a 0 lo cual es cierto entonces num1 se le asigna el valor de v, se imprime el número ingresado en pantalla y a estado se le asigna el valor de 1, después de crea un else if estado es igual a 1 a la variable num2 se le asigna v, se imprime el número ingresado en pantalla y estado se le asigna 2, saliendo del primer if se crea el ultimo if estado igual a 2, si se cumple la condición de imprime en pantalla el resultado de la suma de num1 + num2 convertido a String, y después a estado se le asigna el valor de 0, para que vuelva a realizar una suma.

## Resultados:

```
COM4
5
4
9
```



## **Ejercicio 6. Suma 2**

**Descripción:** Realizar un programa en Arduino que sea capaz de sumar 2 números ingresados por el usuario.

### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estandar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función Serial.

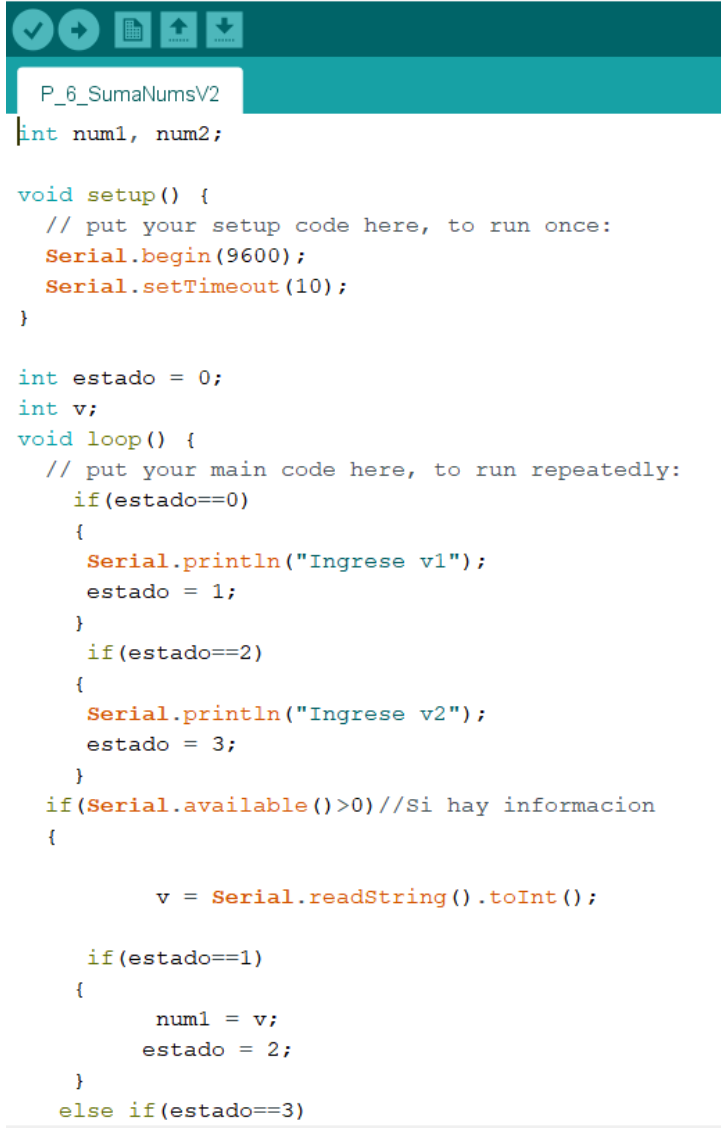
`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.

`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al puerto serial.

`delay`: Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.  
`delay(1000)` //para esperar un segundo.

### **Desarrollo:**



```
int num1, num2;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.setTimeout(10);
}

int estado = 0;
int v;
void loop() {
  // put your main code here, to run repeatedly:
  if(estado==0)
  {
    Serial.println("Ingrese v1");
    estado = 1;
  }
  if(estado==2)
  {
    Serial.println("Ingrese v2");
    estado = 3;
  }
  if(Serial.available()>0)//Si hay informacion
  {

    v = Serial.readString().toInt();

    if(estado==1)
    {
      num1 = v;
      estado = 2;
    }
    else if(estado==3)
```

```

P_6_SumaNumsV2
    estado = 3;
}
if(Serial.available()>0)//Si hay informacion
{
    v = Serial.readString().toInt();

    if(estado==1)
    {
        num1 = v;
        estado = 2;
    }
    else if(estado==3)
    {
        delay(3000);
        num2 = v;
        estado = 4;
    }

}
if(estado == 4)
{
    Serial.println("La suma es igual: " + String(num1+num2));
    estado = 0;
}
}

```

Primero se declaran 2 variables de tipo int llamadas num1 y num2, a continuación se declara otra variable tipo int llamada estado y se le asigna el valor de 0, se declara otra variable tipo int llamada v y en el método loop se crea un if estado es igual a 0 lo cual es cierto entonces, se imprime que ingresa el número 1 en pantalla y a estado se le asigna el valor de 1, después de crea un else if estado es igual a 2 se imprime ingresa el número 2 en pantalla y estado se le asigna 3, posteriormente se crea un if para saber si el usuario ha mandado información y posteriormente se le asigna dicha entrada la variable v, previamente leída con la función Serial.readString y convertida a int por la función toInt.

Se crea un if si estado es igual a 1 a num1 se le asigna el valor de v, y a estado se le asigna el valor de 2, posteriormente se crea un if else si estado es igual a 3, con la función delay espera 3 segundos para que el usuario ingrese la información, a num2 se le asigna el valor de v, y a estado se le asigna el valor de 4

Saliendo del primer if se crea el ultimo if estado igual a 2, si se cumple la condición de imprime en pantalla el resultado de la suma de num1 + num2 convertido a String, y después a estado se le asigna el valor de 0, para que vuelva a realizar una suma.

## Resultados:

```

COM4
Ingrese v1
Ingrese v2
La suma es igual: 10
Ingrese v1

```

## **Ejercicio 7. Área de Rectángulo**

**Descripción:** Realizar un programa en Arduino que sea capaz de calcular el área de un rectángulo.

### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estandar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función Serial.

`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.


`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al puerto serial.

`delay`: Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.  
`delay(1000)` //para esperar un segundo.

### **Desarrollo:**

P\_7\_AreaRectangulo Arduino 1.8.19

Archivo Editar Programa Herramientas Ayuda



```
int b, h;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.setTimeout(10);
}

int estado = 0;
int v;
void loop() {
  // put your main code here, to run repeatedly:
  if(estado==0)
  {
    Serial.println("Ingrese la base");
    estado = 1;
  }
  if(estado==2)
  {
    Serial.println("Ingrese la altura");
    estado = 3;
  }
  if(Serial.available()>0)//Si hay informacion
  {

    v = Serial.readString().toInt();

    if(estado==1)
    {
      b = v;
      estado = 2;
    }
    else if(estado==3)

{
  Serial.println("Ingrese la altura");
  estado = 3;
}
  if(Serial.available()>0)//Si hay informacion
  {

    v = Serial.readString().toInt();

    if(estado==1)
    {
      b = v;
      estado = 2;
    }
    else if(estado==3)

{
  delay(3000);
  h = v;
  estado = 4;
}

}

  if(estado == 4)
{
  Serial.println("La area es igual: " + String(b*h));
  estado = 0;
}
}
```

Primero se declaran 2 variables de tipo int llamadas b y h, a continuación se declara otra variable tipo int llamada estado y se le asigna el valor de 0, se declara otra variable tipo int llamada v y en el método loop se crea un if estado es igual a 0 lo cual es cierto entonces, se imprime que ingresa la base en pantalla y a estado se le asigna el valor de 1, después de crea un else if estado es igual a 2 se imprime ingresa la altura en pantalla y estado se le asigna 3, posteriormente se crea un if para saber si el usuario ha mandado información y posteriormente se le asigna dicha entrada la variable v, previamente leída con la función Serial.readString y convertida a int por la función toInt.

Se crea un if si estado es igual a 1 a b se le asigna el valor de v, y a estado se le asigna el valor de 2, posteriormente se crea un if else si estado es igual a 3, con la funcion delay espera 3 segundos para que el usuario ingrese la información, a h se le asigna el valor de v, y a estado se le asigna el valor de 4.

Saliendo del primer if se crea el ultimo if estado igual a 2, si se cumple la condición de imprime en pantalla El área es igual: , y el resultado de la multiplicación de  $b * h$  convertido a String, y después a estado se le asigna el valor de 0, para que vuelva a realizar el ingreso de valores y la multiplicación.

## Resultados:

COM4

```
Ingrese la base
Ingrese la altura
La area es igual: 20
Ingrese la base
```

## **Ejercicio 8. Promedio**

**Descripción:** Realizar un programa en Arduino que sea capaz de calcular el promedio de n números ingresados con el usuario, y que el programa no continúe hasta que el usuario ingrese un dato.

### **Introducción:**

Se requiere conocimiento de las siguientes instrucciones de Arduino:

`Serial.begin()`: con valor 9600 permite inicializar los pines RX y Tx para que puedan ser usados como puerto serial. Además, configura el puerto con una velocidad estándar de 9600 Baudios por segundo.

`Serial.available()`: para determinar la cantidad de bytes que no se han leído del puerto Serial. Si existe más de 1 byte sin leer, entonces la función utiliza a la variable Dato para, mediante la función Serial.

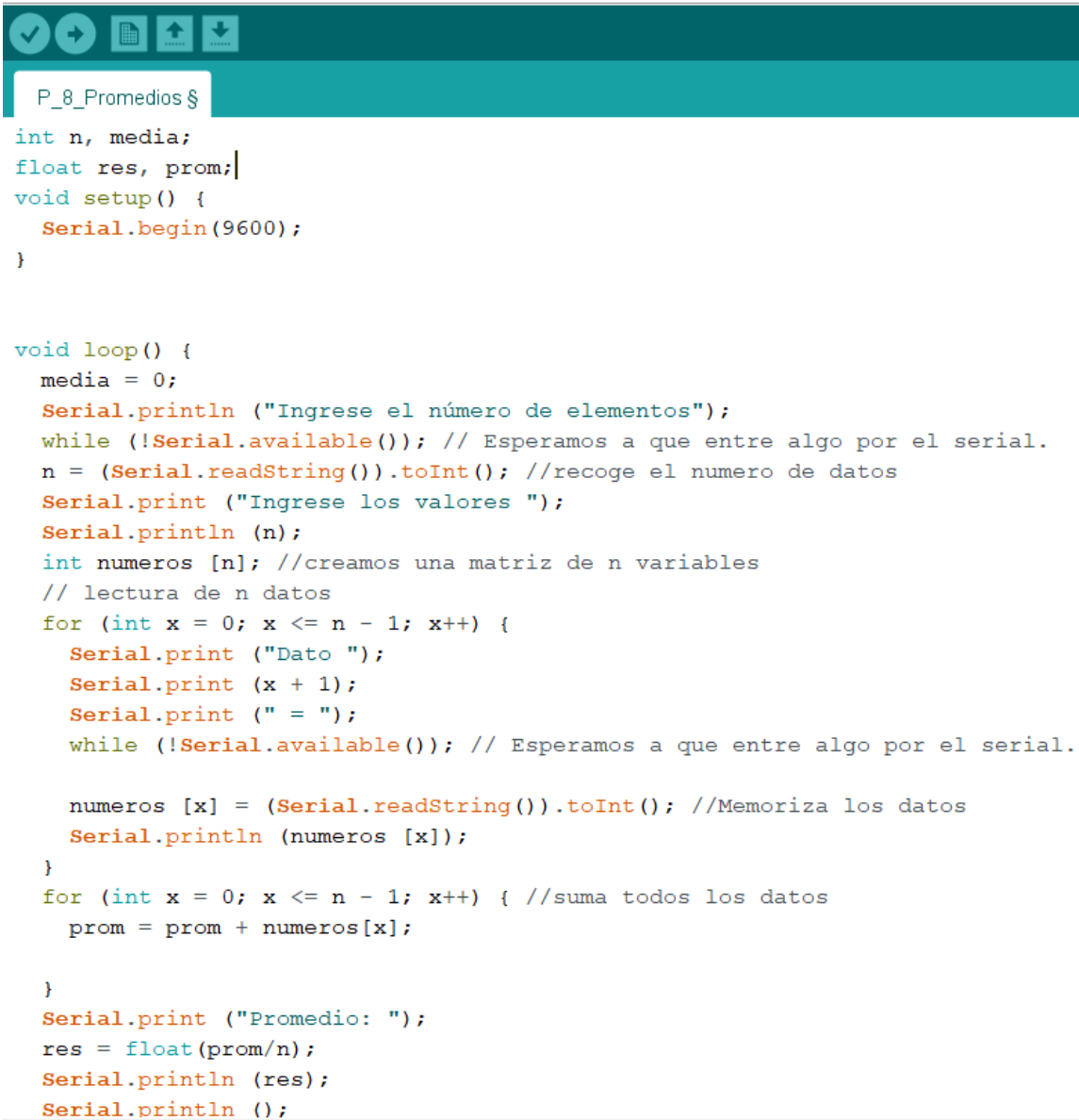
`Serial.setTimeout()`: se usa para establecer el número de milisegundos que se debe esperar para que haya datos disponibles en el puerto serie (por defecto 1000 milisegundos = 1 segundo).

`Serial.readString()`: lee caracteres de un stream y los escribe en un string. La función termina si el tiempo máximo de lectura ha expirado.

`Serial.println()`: escribe datos en el puerto serial. Es usualmente útil para observar los datos producidos en el programa o para imprimir datos en otros dispositivos conectados al puerto serial.

`delay`: Sirve para que el programa se tome cierta cantidad de tiempo (en milisegundos) especificada en su sintaxis para poder ejecutar la siguiente instrucción.  
`delay(1000)` //para esperar un segundo.

### **Desarrollo:**



```

P_8_Promedios $
int n, media;
float res, prom;
void setup() {
    Serial.begin(9600);
}

void loop() {
    media = 0;
    Serial.println("Ingrese el número de elementos");
    while (!Serial.available()); // Esperamos a que entre algo por el serial.
    n = (Serial.readString()).toInt(); //recoge el numero de datos
    Serial.print("Ingrese los valores ");
    Serial.println(n);
    int numeros [n]; //creamos una matriz de n variables
    // lectura de n datos
    for (int x = 0; x <= n - 1; x++) {
        Serial.print("Dato ");
        Serial.print(x + 1);
        Serial.print(" = ");
        while (!Serial.available()); // Esperamos a que entre algo por el serial.

        numeros [x] = (Serial.readString()).toInt(); //Memoriza los datos
        Serial.println(numeros [x]);
    }
    for (int x = 0; x <= n - 1; x++) { //suma todos los datos
        prom = prom + numeros[x];
    }
    Serial.print("Promedio: ");
    res = float(prom/n);
    Serial.println(res);
    Serial.println();
}

```

Primero se declaran 2 variables de tipo int llamadas n y media, y dos variables de tipo float llamadas prom y res, posteriormente en el loop, se le asigna el valor de 0 a la variable media, después con un serial.println se pide ingresar el número de elementos a promediar, posteriormente se crea un while con la condición de ¡Serial.avaible.() con esto no hará nada hasta que se ingrese un valor, después a n se le asigna el valor que el usuario ingreso se lee y se convierte a String y después se convierte en int, después se imprime en pantalla ingrese los valores y se imprime en pantalla a el número de valores que ingresó el usuario, después se crea un vector de tipo int con el tamaño de la variable n, se crea un for inicializando y creando la variable x con valor de 0, con la condición si x es menor o igual a n-1 e incrementando x en 1, en las instrucciones dentro del for se imprime Dato, después se imprime el valor de x + 1 y se imprime =.



Posteriormente se crea un while con la condición de `Serial.available()` con esto no hará nada hasta que se ingrese un valor, después a el arreglo números en la posición `i` se le asigna el valor que el usuario ingreso se lee y se convierte a String y después se convierte en int, y se imprime en pantalla el valor asignado.

Después se crea un for inicializando y creando la variable `x` con valor de 0, con la condición si `x` es menor o igual a `n-1` e incrementando `x` en 1, en la única instrucción del for es asignarle a la variable `prom`, su mismo valor + el dato que está en el vector números en la posición `x` (en la que va el ciclo), saliendo del for se imprime el texto Promedio:, después a la variable `res` se le asigna la división de `prom/n` casteando a float para que se guarde un número decimal, de lo contrario guarda el valor como entero, por último se imprime en pantalla el valor de `res`.

### Resultados:

```
COM4
Ingrese el número de elementos
Ingrese los valores 4
Dato 1 = 3
Dato 2 = 6
Dato 3 = 8
Dato 4 = 9
Promedio: 6.50

Ingrese el número de elementos
```