REDUBOT
RED EDUCATIVA DE ROBÓTICA

GUÍA DE ESTUDIO

Curso en línea

Microcontroladores I

Autor: Kalid Gabriel Gómez Aceves

Noviembre 2021

AlfaDigital
Dirección de Alfabetización Digital

Recrea
Educación para refundar 2040

Educación

Jalisco
GOBIERNO DEL ESTADO

Educación

AlfaDigital

Recrea
Educación para refundar 2040

Guía de Estudio

Curso en línea

Microcontroladores 1

Autor: Kalid Gabriel Gómez Aceves

Asesores: Kalid Gabriel Gómez Aceves

Luis Alonso Castañeda Vázquez

Noviembre de 2021

Contenido

Microcontroladores 1

Bienvenidos al curso "Microcontroladores 1"

Estimados estudiantes reciban un cordial bienvenida a este curso en línea: Microcontroladores 1 ofrecido por la Secretaria de Educación del Estado de Jalisco a través de la Dirección de Alfabetización Digital.
En la plataforma Alfa Online tenemos el objetivo de brindar al docente cursos en línea que le permitan fortalecer sus competencias tecnológicas. Una de las posibilidades para desarrollarlas es el uso cotidiano de las TIC en el aula.
REDUBOT tiene el objetivo de brindar cursos especializados en diferentes áreas de la robótica y la programación para así fortalecer sus competencias tecnológicas, lógica y de resolución de problemas.
En este curso se aprenderán las bases para el desarrollo de proyectos basados en microcontroladores, promoviendo a lo largo del curso nociones de razonamiento lógico, electrónica y programación usando la tarjeta de desarrollo UNO.
Si tienes cualquier duda con este curso, siempre podrás ingresar al foro *Pregúntale al experto*, para que recibas respuesta a tus inquietudes.

¡Bienvenidos al curso!

Justificación

La educación es un derecho básico para el desarrollo de una sociedad sustentable, incluyente y justa. Para lograr los objetivos del Plan Estatal de Gobernanza y Desarrollo de Jalisco visión 2030 debemos incrementar la calidad y accesibilidad educativa en todos los niveles, modalidades y servicios de manera inclusiva y equitativa, enfocándonos a una formación integral centrada en el aprendizaje de las y los estudiantes, por lo cual implementamos procesos innovadores y acordes a este siglo.
No podemos dejar de lado el fortalecimiento del perfil profesional del personal, directivo, docente y de apoyo en las áreas académicas y tecnológicas; ya que las mismas contribuyen a la implementación de procesos de enseñanza-aprendizaje.
Como uno de los quehaceres de esta Dirección se encuentra la promoción del uso de la tecnología y el desarrollo de habilidades digitales en apoyo al fortalecimiento de la calidad educativa.
La educación en línea es una propuesta para llegar a los docentes y directivos de regiones aisladas.

Objetivo general del curso

Fortalecer las competencias tecnológicas, digitales y comunicativas del docente para dinamizar los procesos de enseñanza y aprendizaje de los estudiantes en la construcción del conocimiento utilizando las herramientas digitales y de Internet.

Objetivo particular del curso

Introducirse en el uso de tarjetas de desarrollo para aprender las bases de la programación de microcontroladores haciendo uso de nociones de razón y lógica, desarrollado especialmente para jóvenes mayores de 12 años.

Objetivo del módulo 1: Introducción a las tarjetas de desarrollo
Conocer qué es un microcontrolador, así como la manera de comunicarse con el mismo.

Objetivo del módulo 2: Bases de la programación e interacción con el microcontrolador
Conocimiento de la tarjeta, uso del IDE y asignación de variables.

Objetivo del módulo 3: Condicionales y ciclos
Uso de los recursos básicos de la programación

Objetivos del módulo 4: Protocolo de comunicación UART
Nociones de sistema binario, ASCII e información básica de protocolos de comunicación

Metodología

Aprender a emplear las bases de la programación y usarla para interactuar con el entorno haciendo uso de la tarjeta de desarrollo UNO, usando los conocimientos en la solución de problemas empleando soluciones lógicas.
Hacer uso de los puertos de salida de la tarjeta para el uso de indicadores y actuadores en robótica básica de baja potencia mientras se desarrollan proyectos simples.

Contenido

Módulo 1: Introducción a las tarjetas de desarrollo y microcontroladores

- Historia de los microcontroladores
- Tarjetas de desarrollo
- Pines de salida
- Entorno de Desarrollo Integrado (IDE)
- Compilar y correr programas
- Configuración de la terminal
- Actividad 1

Módulo 2: Bases de la programación e interacción con el microcontrolador

- Imprimir en la terminal
- Creación de variables
- Operaciones matemáticas
- LEDs
- Uso de *delays*
- Actividad 2

Módulo 3: Condicionales y ciclos

- Operadores de comparación
- Condicional "si" *(if)*
- Ciclo para *(for)*
- Ciclo mientras *(while)*
- Actividad 3

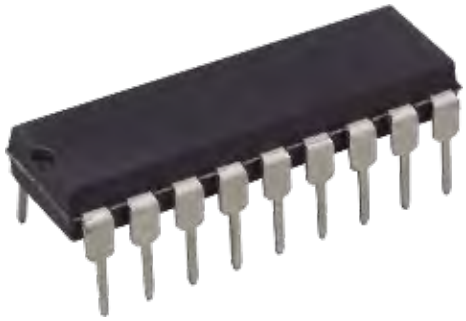
Módulo 4: Comunicación serial

- Protocolo de comunicación
- Bases del sistema binario
- ASCII
- UART
- Actividad 4
- Actividad integradora

Módulo 1: Introducción a las tarjetas de desarrollo y microcontroladores

Microcontroladores

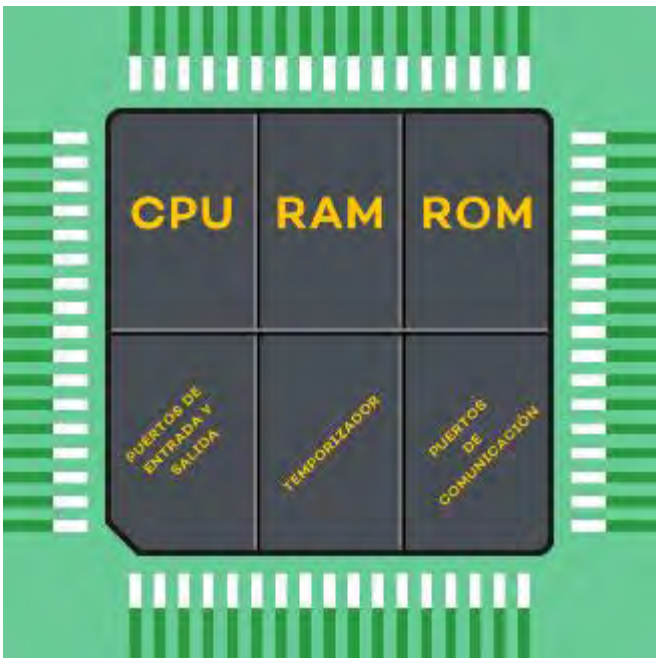
Cuando se inventó el transistor en 1948 por laboratorios Bell, abrió el camino para que máquinas grandes y costosas que funcionaban con tubos de vacío fueran compactas además de económicas. Este descubrimiento causó que en 1961 se desarrollara el primer circuito integrado comercial.



Los **circuitos integrados**, también conocidos como microchips, son encapsulados normalmente de plástico que dentro tienen desde unos cuantos hasta millones de transistores además de otros componentes electrónicos que cumplen un fin en específico. Esta tecnología se fue perfeccionando hasta que en 1971 surge el primer microprocesador comercial desarrollado por Intel.

Los **microprocesadores** son el equivalente del cerebro de una máquina, son los encargados de seguir instrucciones, realizar operaciones matemáticas básicas y mandar información a la memoria.

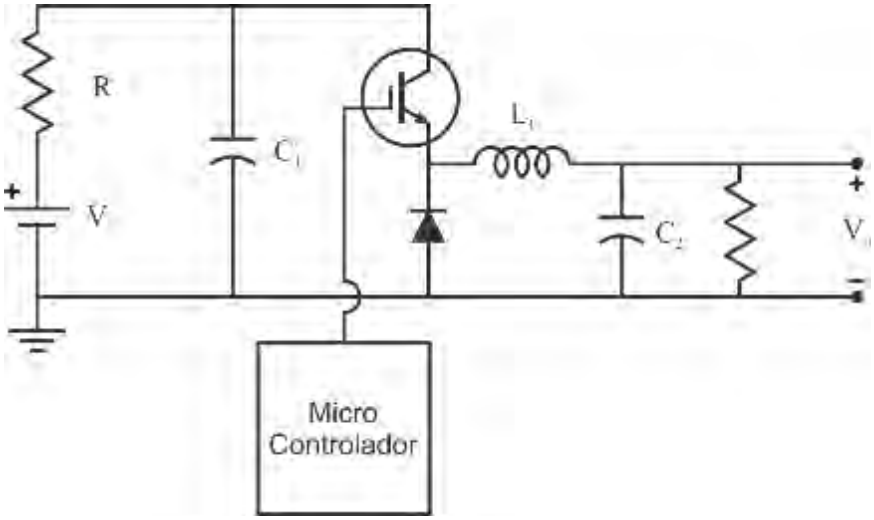
Pero un microprocesador al igual que un cerebro requiere de ciertas partes para funcionar correctamente, como lo son la **memoria RAM** que es donde se almacenan los datos de los programas que se estén corriendo y la **memoria ROM** que es donde se guardan los programas que deberá ejecutar el microprocesador.
Finalmente llegamos a la parte central de este curso: los **microcontroladores** los cuales integran el microprocesador, la RAM y la ROM entre otras cosas en un mismo circuito integrado.



Pero ¿Para qué sirven los microcontroladores? Estos se encuentran en todas las máquinas que nos rodean y diariamente los usamos, se utilizan desde en cohetes y satélites, hasta para mover el puntero de tu computadora, algunos ejemplos son:

Un carro utiliza un microcontrolador para las bolsas de aire, otro en el sistema de frenos, uno más en el tablero y cada uno de ellos desempeña su tarea con el objetivo de enviar después la información a la computadora central.

Los cargadores del celular también cuentan con su propio microcontrolador para regular el voltaje que entra al celular y de este modo no quemarlo.



El *mouse* de nuestra computadora también procesa los fenómenos físicos para finalmente enviar los movimientos que hacemos y los manda a nuestra computadora.

Tarjetas de desarrollo

Los microcontroladores necesitan de ciertos voltajes y tienen ciertos pines para ser programados y necesitan de un reloj externo en su funcionamiento entre otras cosas, afortunadamente, para que nosotros podamos trabajar con los microcontroladores existen las tarjetas de desarrollo.
Las tarjetas de desarrollo nos facilitan la configuración del microcontrolador y nos da acceso a sus pines de salida de tal forma que podremos integrar nuestro microcontrolador con nuestros circuitos de una forma más sencilla.
Por ejemplo, en este curso trabajaremos con un microcontrolador ATMEL en una de las tarjetas de desarrollo más famosas en la actualidad llamada UNO.

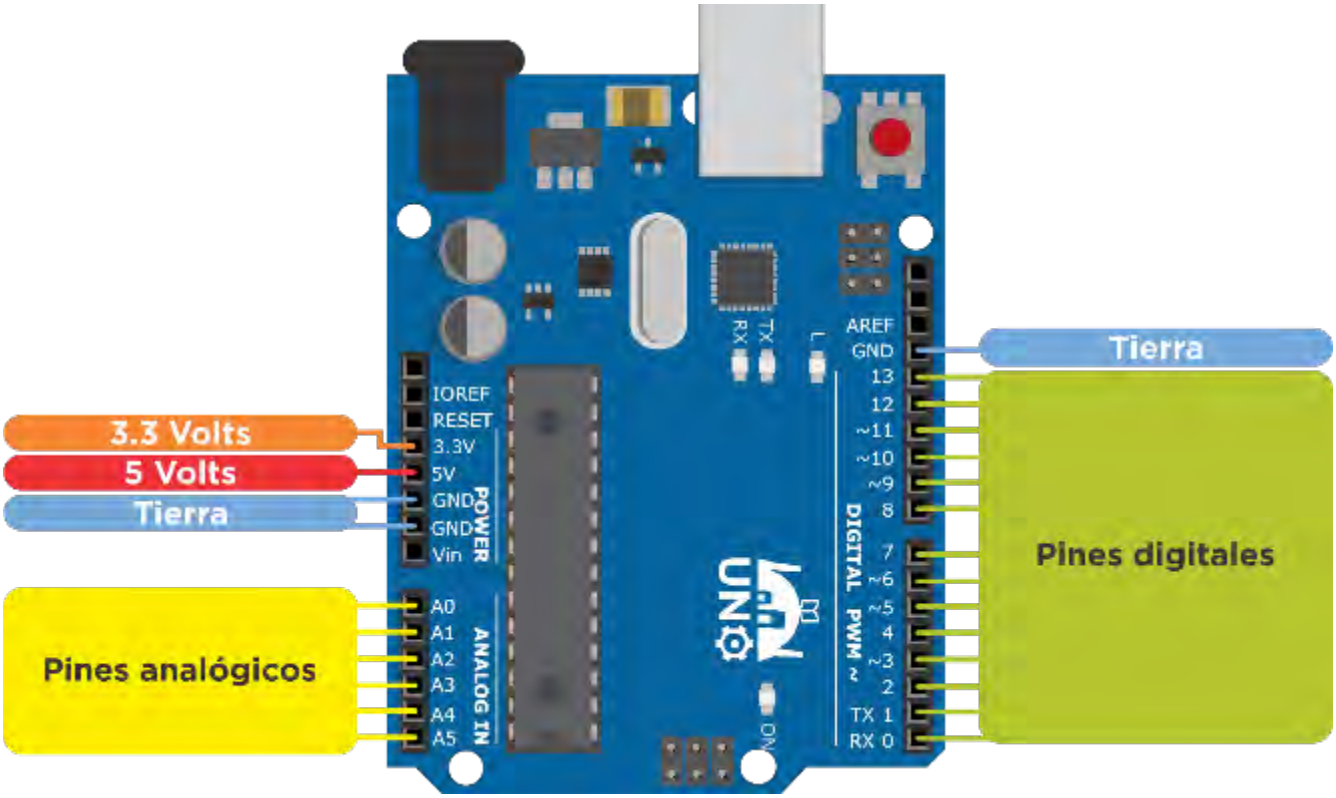
Microcontrolador usado en este curso: ATMEGA328P

Tarjeta de desarrollo usada en este curso: UNO

Pines de la tarjeta de desarrollo

Seguro a esta altura del curso te estarás preguntando ¿Cómo interactúan mis circuitos con mi microcontrolador? La tarjeta de desarrollo cuenta con unos pines de los cuales con cables podemos conectarlos a nuestra placa de pruebas o en inglés *protoboard.*

Nuestro microcontrolador tiene entradas y salidas tanto analógicas como digitales que se conectan a nuestra tarjeta de desarrollo para su uso, en este curso aprenderemos a usar únicamente las salidas digitales para nuestros proyectos (Las que están en verde)



Pinout: Pines de la tarjeta UNO

Nota: es importante tener siempre a la mano antes de conectar nuestros circuitos una imagen de los pines conocido en inglés como el *pinout,* para NO quemar nuestro dispositivo.

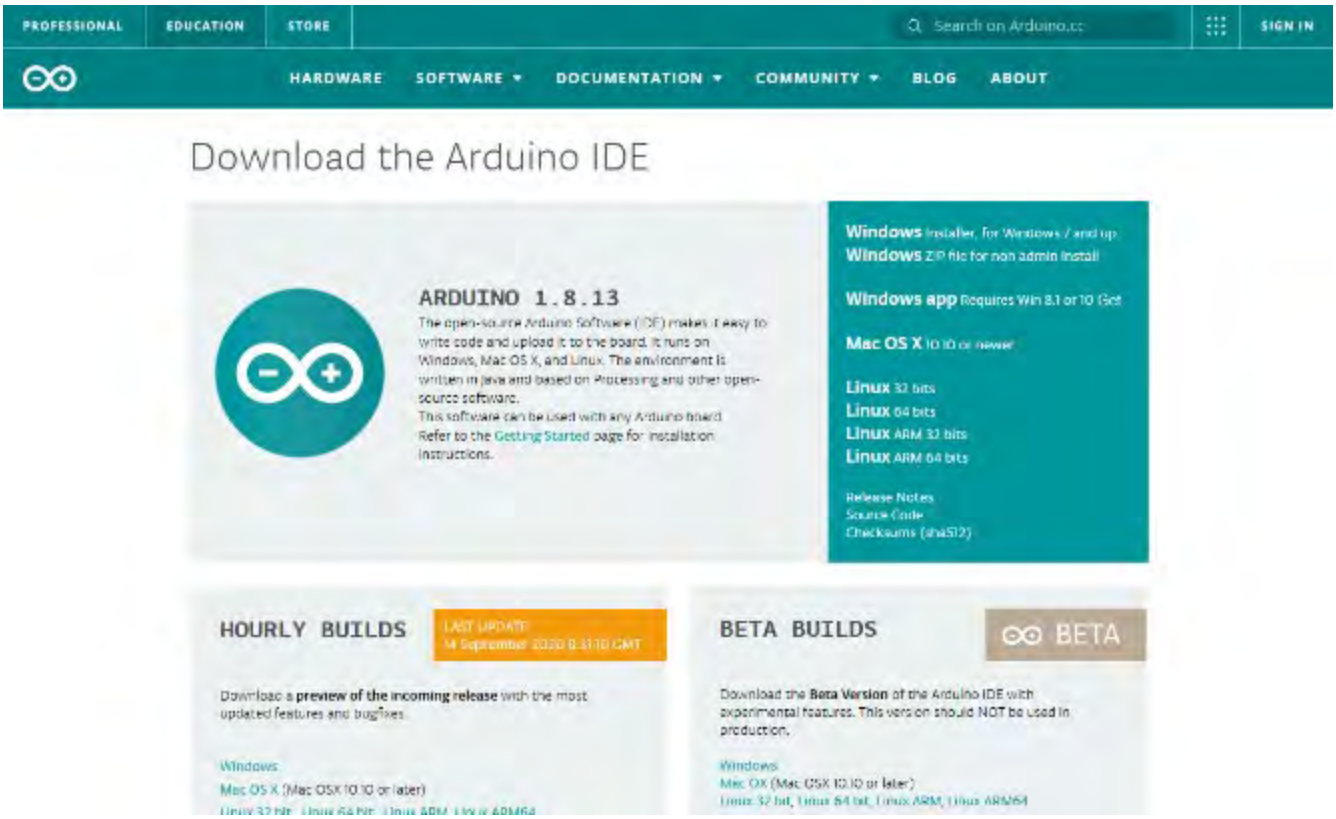
Entorno de desarrollo integrado

Finalmente, hay que poder comunicarnos y darle instrucciones al microcontrolador para que realice las tareas que nosotros indiquemos, por eso tendremos que aprender a programar en el lenguaje adecuado, que en nuestro caso es el lenguaje de programación *Arduino* el cual está basado en un lenguaje muy conocido en el ámbito de la programación llamado C++.

Si aún no sabes programar, ¡No te preocupes! A través del curso iremos aprendiendo lo necesario para programar nuestro microcontrolador.

Para poder empezar a programar, antes debemos instalar un entorno de desarrollo integrado por sus siglas en inglés *IDE*, el cual con el microcontrolador que usaremos en este curso es el que encontraremos en el siguiente enlace: www.arduino.cc/en/Main/Software

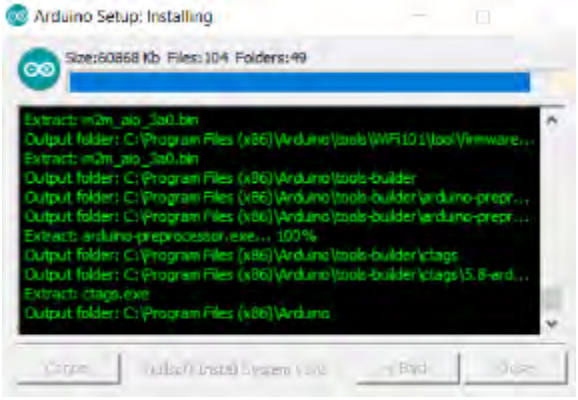
Para instalar el IDE, primero damos clic en el enlace anterior, ya que el sitio está en inglés, los llevaremos paso a paso en la instalación del programa.



Dentro seleccionaremos el sistema operativo que corra nuestra computadora (si tienes dudas sobre qué sistema operativo usas, pide ayuda a un adulto para averiguarlo).

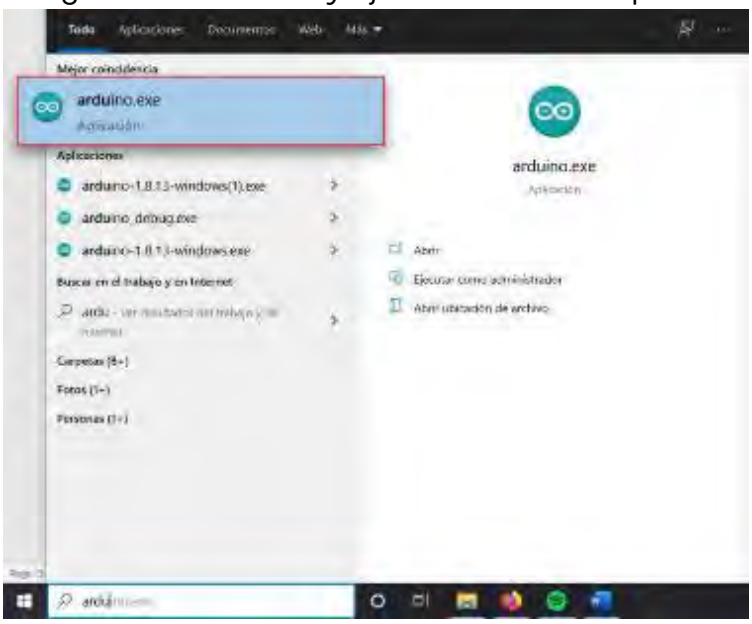
Ya que hayamos dado clic sobre el sistema operativo que usamos, damos clic en descargar o "*just download*" y guardamos el archivo. Una vez terminada la descarga, en caso de haber usado Windows, damos doble clic en el instalador.

Hacemos clic en *"I Agree"* e instalamos el programa dando clic en siguiente las veces que sea necesario.

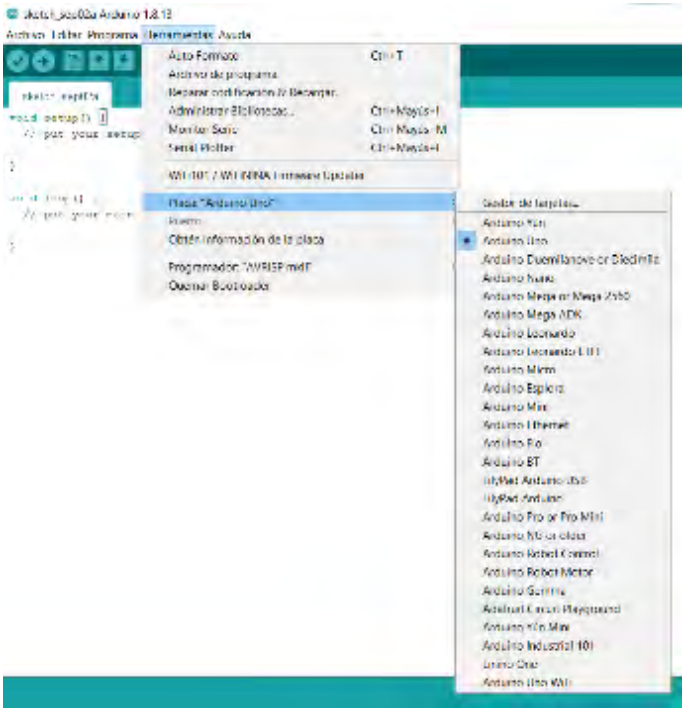


Configurar, compilar y correr un programa

Para empezar a programar, primero abriremos el IDE que instalamos anteriormente, en caso de tener Windows instalado pondremos en la parte inferior izquierda, en nuestra barra de navegación “Arduino” y ejecutaremos la aplicación.

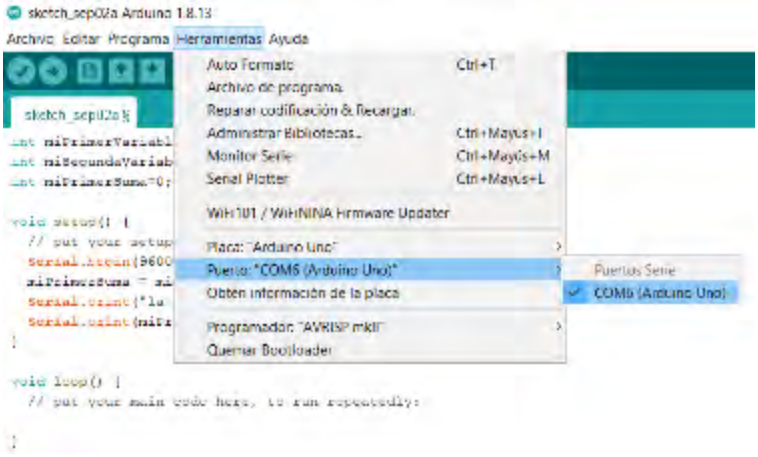


Para configurar por primera vez nuestro IDE, tenemos que indicarle el tipo de tarjeta que vamos a usar, entonces con el programa abierto damos clic en >> Herramientas/Placa/Arduino Uno.



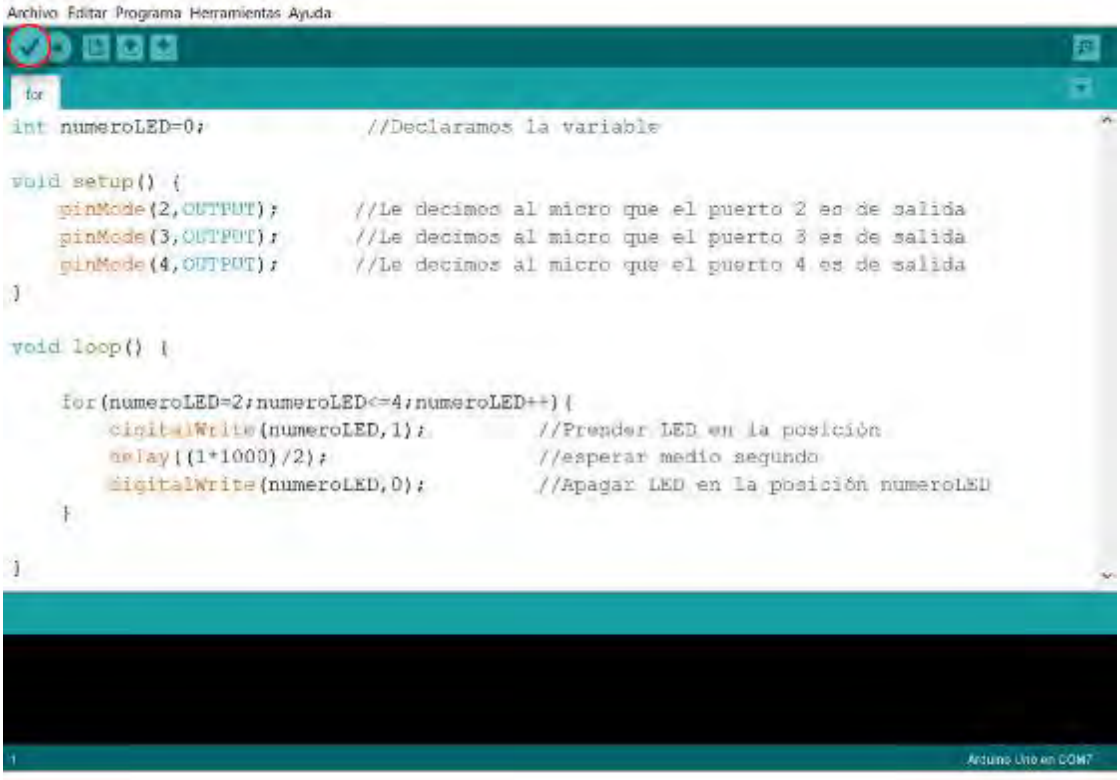
Una vez seleccionada nuestra tarjeta, debemos enseñarle a la computadora a identificar nuestra tarjeta de desarrollo, para hacer esto la tenemos que conectar con un cable A a USB.

Finalmente, damos clic en >> *Herramientas/Puerto* y seleccionamos nuestra tarjeta.



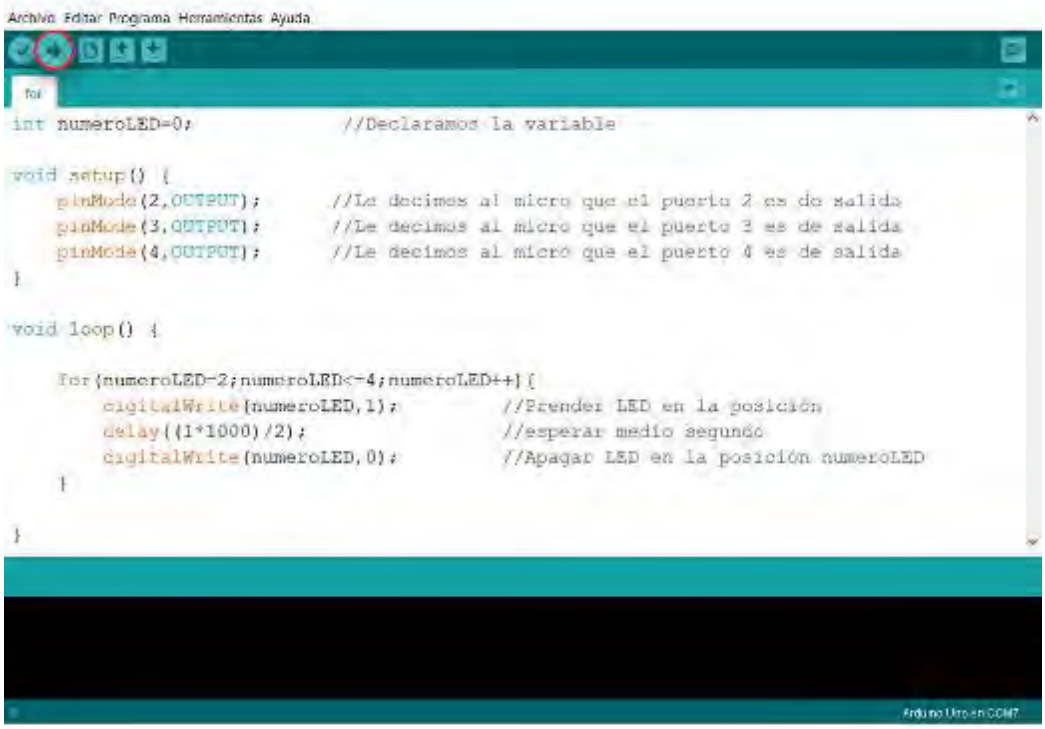
Compilar o verificar

Cuando estemos programando, el IDE tiene una función conocida en el mundo de la programación como *compilar* en el cual busca automáticamente errores en nuestro código y nos lo señala en caso de ser así. Para evitar que enviemos programas que no funcionan al microcontrolador, el IDE no nos permite cargar un programa a nuestra tarjeta sin antes compilar el código.
Ya que es algo que realizaremos muy seguido, es importante recordar que los códigos se compilan dando clic en el siguiente botón.



Correr un programa

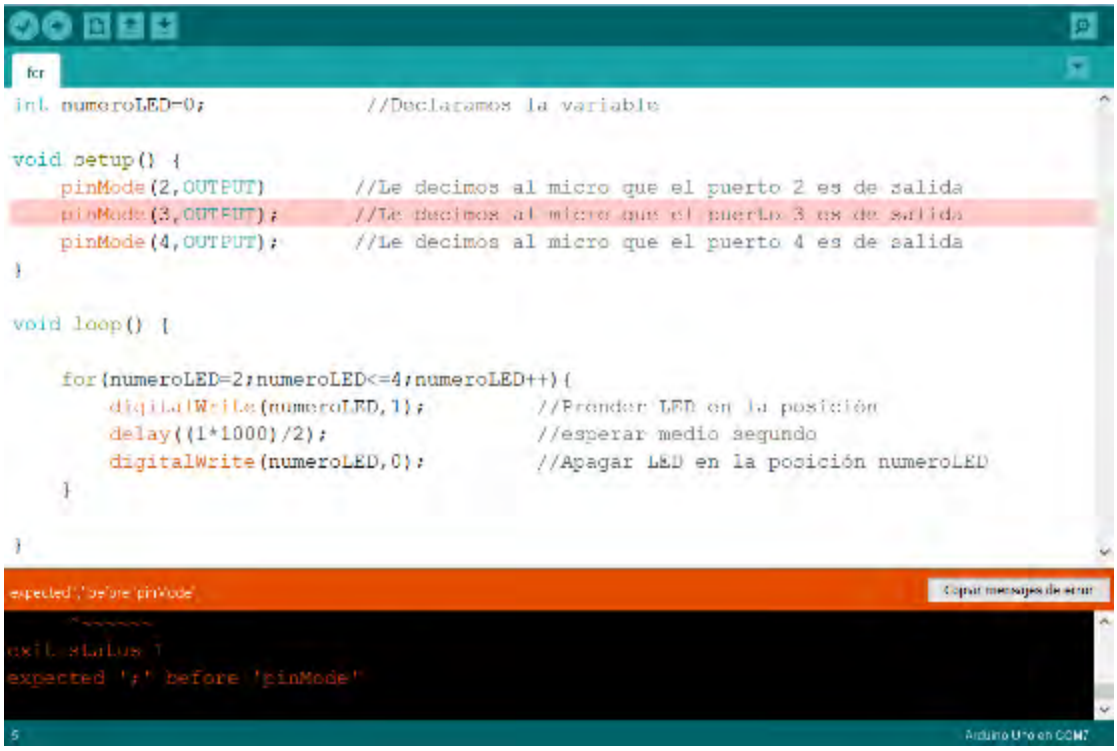
Una vez que hayamos compilado el programa, podemos mandar el programa a nuestro microcontrolador, para evitar errores al cargarlo, recomendamos que no haya nada conectado a los pines digitales 0 y 1 cuando mandemos al microcontrolador el programa por primera vez. También verifica que tus conexiones no tengan un corto circuito y todo esté perfectamente conectado.



Errores comunes de compilación

En el transcurso del curso será normal encontrarse con problemas a la hora de compilar, y estos errores vienen acompañados de una consola (La pantalla negra en la parte inferior del IDE) con letras rojas. Estos errores tendrán una explicación en inglés del problema además de la línea donde ocurre.

Ya que en el material de este curso los códigos fueron compilados y ejecutados, los errores más comunes que ocurrirán son de sintaxis.



Como puedes ver en la imagen anterior, después de compilar el IDE nos menciona que hace falta un punto y coma, resaltándonos la línea de código que no pudo ejecutar debido a este error.

Si nuestro error es de sintaxis, los errores más comunes son:

- Ausencia de punto y coma entre acciones
- Falta de algún paréntesis que abre y nunca cierra
- No usar adecuadamente las mayúsculas
- Escribir erróneamente las palabras

Otros errores que pueden ocasionar que nuestro código no compile es no haber declarado las variables o escribirlas de forma distinta o no usar una estructura válida para ciclos y condicionales.

En caso de encontrar un error en tu código y no encontrar el error en los ya mencionados, se recomienda buscar el error en internet y buscar ayuda en foros,

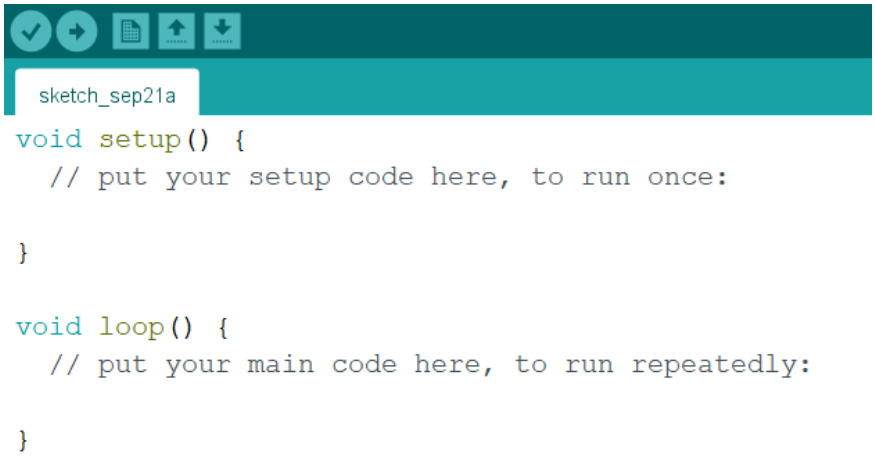
recuerda que en el mundo de la programación en general, el idioma oficial es el inglés, por lo que te será de mucha ayuda entenderlo.

Terminal, comentarios y funciones básicas

Para poder trabajar debemos familiarizarnos con algunos aspectos básicos del lenguaje de programación y de nuestra IDE.

Funciones en un programa nuevo

Cuando abrimos un programa nuevo, el IDE nos aparecerán dos estructuras que se conocen como funciones, las cuales deben tener al menos una para que nuestro código pueda funcionar.



Como podemos ver, la primera se llama configuración, o en inglés *setup*. En él, todo lo que escribamos dentro de los corchetes "{ }" se realizará una sola vez.
Por otra parte, lo que esté en la función bucle (o ciclo) loop en inglés, repetirá una y otra vez todo lo que esté dentro de sus corchetes "{ }" empezando de arriba abajo.

Comentarios

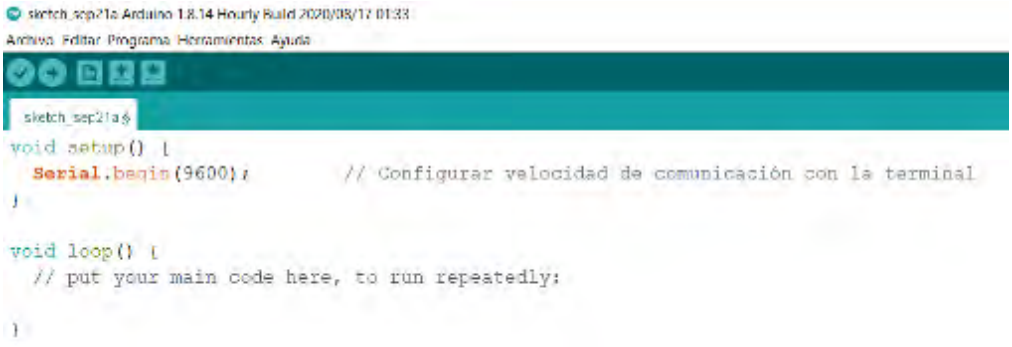
Nosotros para entender mejor nuestros códigos podemos agregar comentarios los cuales serán ignorados por nuestro microprocesador. También ayuda a que si leemos un código viejo podremos recordar que es lo que hacía.
Para hacer esto pondremos dos diagonales de la siguiente forma:

```
// Esto es un comentario
```

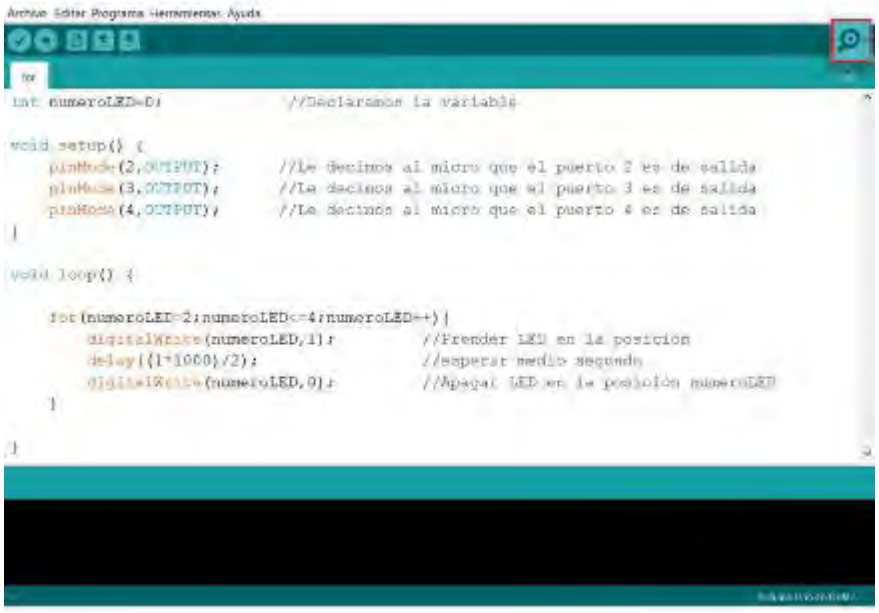
Recuerda que los comentarios no tienen un sentido funcional, únicamente son para nuestra mejor comprensión.

Terminal

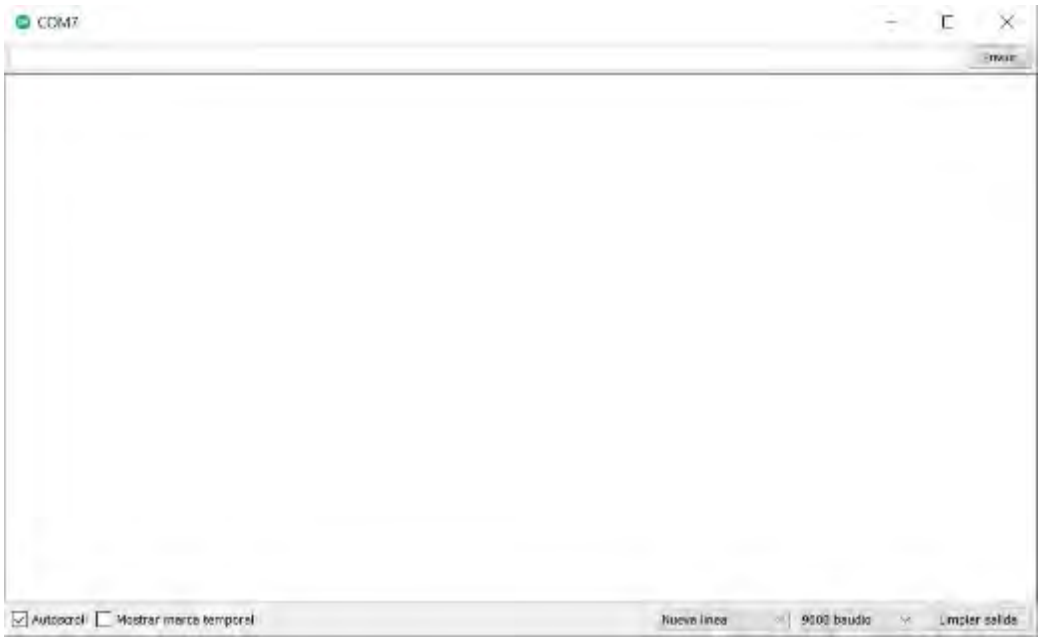
Una terminal es una ventana donde interactuaremos con nuestro microcontrolador cuando esté corriendo un programa, por eso debemos configurar el número de cambios de la señal por segundo conocido también como baudio. Es para que nuestra computadora mande y reciba información a la misma velocidad que lo hace el microcontrolador y ambos puedan entenderse.
Para habilitar la comunicación con la terminal, en la función *setup* hay que agregar la siguiente línea.



Y finalmente, con la tarjeta conectada, abriremos la terminal desde nuestro IDE con el siguiente botón.



Nota: ¡Recuerda que debes tener el mismo baudio en ambas partes! 9600 es un estándar.



Desde aquí más adelante enviaremos y recibiremos información de nuestra tarjeta de desarrollo

Actividad 1: Teoría

Seleccione la respuesta correcta:

¿Cuál es el microcontrolador que usaremos en este curso?

a. Arduino
b. ATMEGA328P
c. UNO
d. MSP430

¿Cuántos puertos de tierra tiene nuestra tarjeta de desarrollo?

a. 0
b. 1
c. 2
d. 3

¿Cuántos pines o "patitas" tiene el circuito integrado de nuestro microcontrolador?

a. 16
b. 24
c. 28
d. 32

¿Cuántos pines o "patitas" tiene el circuito integrado de nuestro microcontrolador?

a. 16
b. 24
c. 28
d. 32

En el IDE que descargamos, ¿Qué hace el botón de la siguiente figura?

a. Compilar-verificar
b. Correr programa
c. Abrir terminal
d. Guardar

En el IDE que descargamos, ¿Qué hace el botón de la siguiente figura?

a. Compilar-verificar
b. Correr programa
c. Abrir terminal
d. Guardar

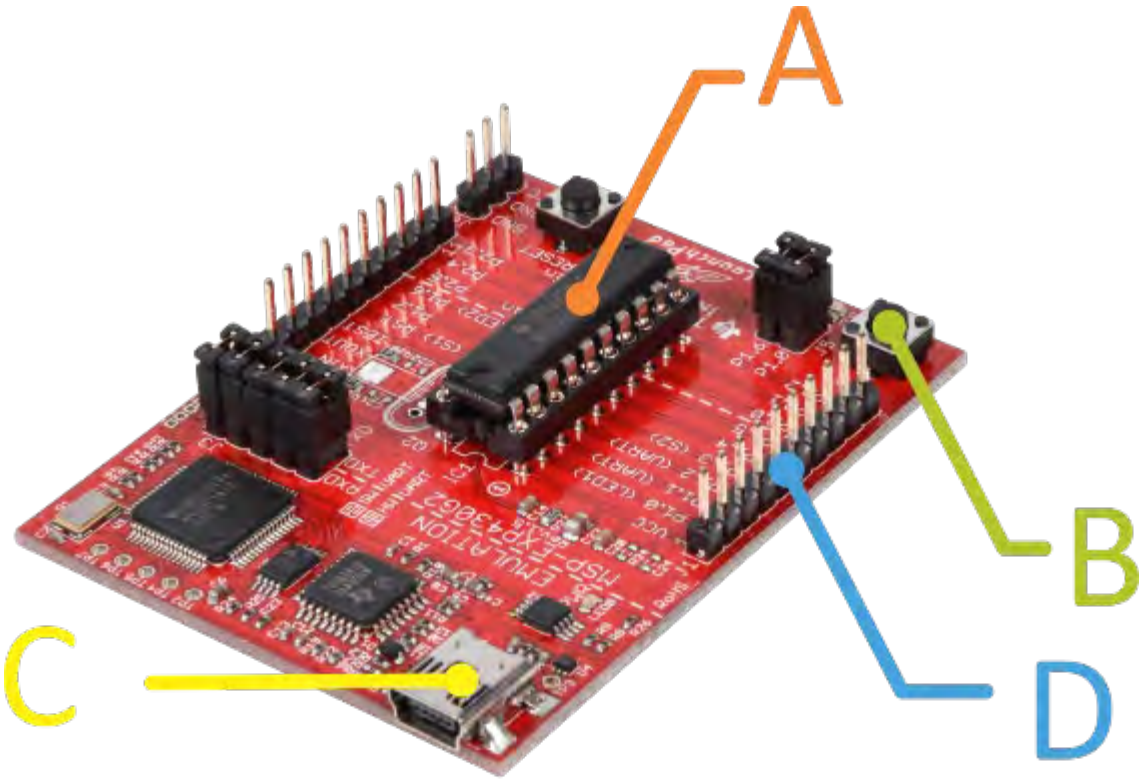
En el IDE que descargamos, ¿Qué hace el botón de la siguiente figura?

a. Compilar-verificar
b. Correr programa
c. Abrir terminal
d. Guardar

¿Para qué son los comentarios?

a. Para explicarle al microcontrolador que debe hacer
b. Para dar indicaciones sin usar código
c. Para decorar el código
d. Para que nosotros entendamos mejor que es lo que programamos

Esta es otra tarjeta de desarrollo que usa otro microcontrolador, seleccione la opción que relacione los incisos de manera correcta



a.

A - Puerto de alimentación
B - Botón
C - Microcontrolador
D - Pines

b.

A - Microcontrolador
B - Pines
C - Puerto de alimentación
D - Botón

c.

A - Pines
B - Botón
C - Puerto de alimentación
D - Microcontrolador

d.

A - Microcontrolador
B - Botón
C - Puerto de alimentación
D - Pines

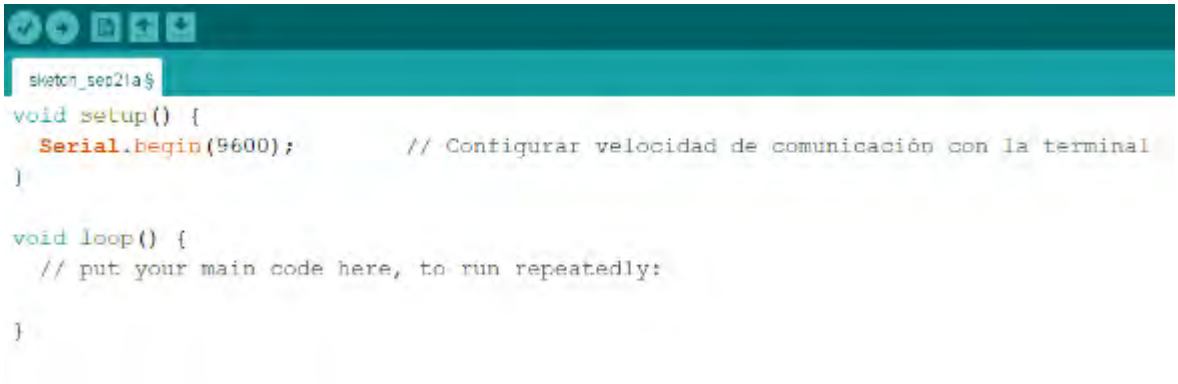
Módulo 2: Bases de la programación e interacción con el microcontrolador

Imprimir en la terminal

Para poder enviar mensajes desde el micro a nuestra terminal y de esta forma hacer pruebas, ver el estado del programa entre otras cosas, se habilita en primer lugar la comunicación con la computadora desde nuestro código, esto se hace escribiendo en la función *setup* la siguiente línea:

```
Serial.begin(9600);
```

Hay que tener en cuenta que, en el mundo de la programación, son muy importantes las distinciones entre mayúscula y minúscula, además de que en ciertos lenguajes de programación, después de hacer un cálculo, una declaración, llamar a una función entre otras cosas se debe poner un punto y coma " ; " cuando no lleva corchetes.
Una vez habilitado tendremos lo siguiente:



Como recordarás, la configuración correrá una sola vez ya que está en el bloque *setup*.

Ahora, para poder enviar mensajes desde nuestro microcontrolador a la computadora utilizaremos la siguiente función:

```
Serial.print("¡Hola mundo!");
```

Ya que no queremos que se esté escribiendo en pantalla indefinidamente, también lo escribiremos en la función *setup* de la siguiente manera:

```
void setup() {
  Serial.begin(9600);           // Configurar velocidad de comunicación con la terminal
  Serial.print("¡Hola mundo!"); //Línea para poder enviar a pantalla información
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Finalmente conectamos el microcontrolador, compilamos y si no escribimos algo mal, la consola se verá de la siguiente manera:

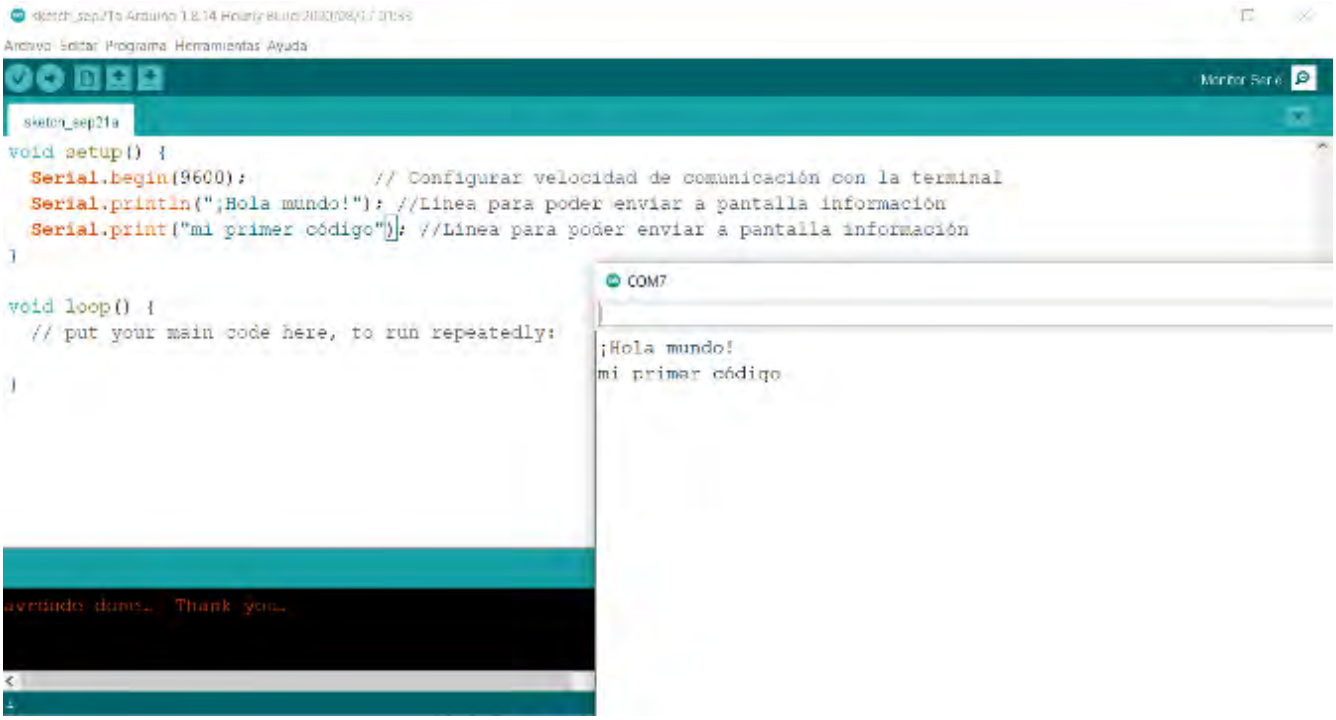
Finalmente abrimos una terminal y corremos el programa, deberías poder ver lo siguiente:

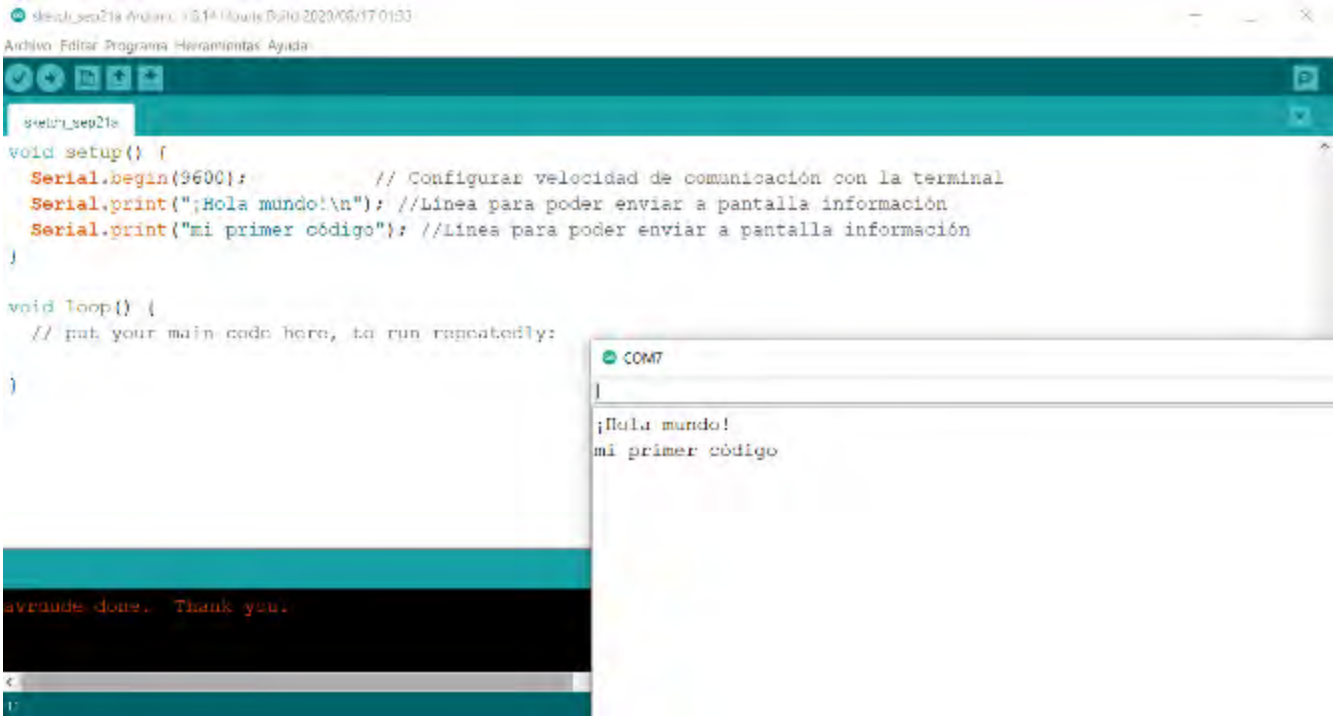
Si quieres que al terminar pase de línea como si presionaras “enter”, sustituye el comando por el siguiente:

```
Serial.println("¡Hola mundo!");
```

o puedes utilizar su equivalente

Serial.print("¡Hola mundo \n");





Creación de variables y tipos de datos

Primero vamos a definir lo que es una variable. Una variable es una forma de almacenar información dentro de nuestro microprocesador que de acuerdo con la instrucción que le demos cambiará si nosotros lo indicamos de esta manera. Las variables pueden ser números enteros, decimales, caracteres o letras, y cadenas de texto entre otros que son más específicos.

Para nosotros poder crear una variable primero hay que declararla, en este curso utilizaremos variables globales que se declaran en la parte superior del código y podrás cambiarlas en cualquier línea.

Si quisiéramos guardar una variable entera, ya sea positiva o negativa (-1000, -542, 0 245, 3120) usaremos una variable de tipo entero que se declara como int que viene de la palabra entero en inglés, acompañada por un nombre que queramos darle, finalmente la igualamos a un valor como a continuación.


```
int     miPrimerVariable = 123;

void setup() {

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Nota: las variables de tipo entero funcionan únicamente en un rango de -32,768 a 32,767

En programación es muy importante escribir con las mayúsculas y minúsculas cada que se usa una variable, ya que no es lo mismo usar: miPrimerVariable que miprimervariable

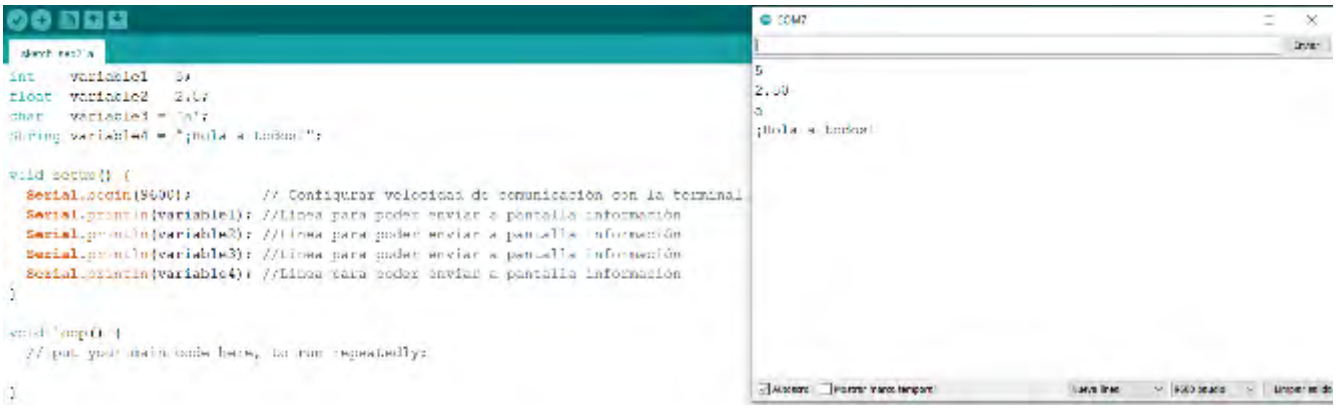
Cuando el microcontrolador lea esta línea de código, sabrá que hay una variable entera que se llama “miPrimerVariable” con un valor de 123.

Podemos escoger el nombre que queramos, este no puede empezar con un número, además hay que tener cuidado con que el nombre de la variable que hayamos escogido no tenga espacios o caracteres que no sean comunes en inglés como la “ñ”. No olvides que al finalizar la declaración se debe poner Símbolo un punto y coma, esto le indicará al microcontrolador que termina ahí la instrucción.

En el mundo de la programación hay otras variables además de los enteros. Podemos ver algunas de las variables que puedes utilizar en la siguiente tabla:

Tipo	Nombre	Ejemplo
Entero	*int (integer)*	int variable = 5;
Decimal	*float*	float variable = 5.32;
Carácter	*char (character)*	char variable = 'a';
Cadena	*String*	String variable = "Hola a todos ";

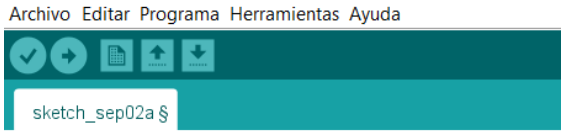
Como puedes ver, declaramos un símbolo utilizamos comillas simples mientras que en textos comillas dobles. Recuerda que las cadenas de texto deben llevar la palabra reservada *String* empezando con mayúsculas. Recuerda que, al finalizar la declaración, debes utilizar un punto y coma ";".
Para finalizar probemos imprimiendo variables con el siguiente código:



Operaciones matemáticas

Ahora aprenderemos a hacer operaciones con variables de tipo entero y decimales, para ello podemos utilizar variables, números y algunos símbolos. Al crear una variable primero hay que declararla o crearla, en este curso utilizaremos variables globales que se declaran en la parte superior del código y podrás cambiarlas en cualquier línea.

Por ejemplo, sumemos dos variables de tipo entero, para esto, declararemos 2 variables con los valores a sumar y una variable más donde guardaremos el resultado.




```
int miPrimerVariable= 123;
int miSegundaVariable= 42;
int miPrimerSuma=0;

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

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Luego indicaremos que realice la suma como si fuera una ecuación, escribiendo del lado izquierdo de la igualdad la variable que queremos que cambie de la siguiente forma.

```
int miPrimerVariable = 123;
int miSegundaVariable = 42;
int miPrimerSuma=0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);//Baudios o símbolos por segundo
  miPrimerSuma = miPrimerVariable + miSegundaVariable;
  Serial.print("la suma es: ");
  Serial.print(miPrimerSuma);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Ponemos la suma dentro de los paréntesis de la función *setup* porque queremos que realice la operación únicamente cuando inicia.

Finalmente conectamos la tarjeta y abrimos una terminal. Para recordar, es con el botón de la esquina superior derecha.

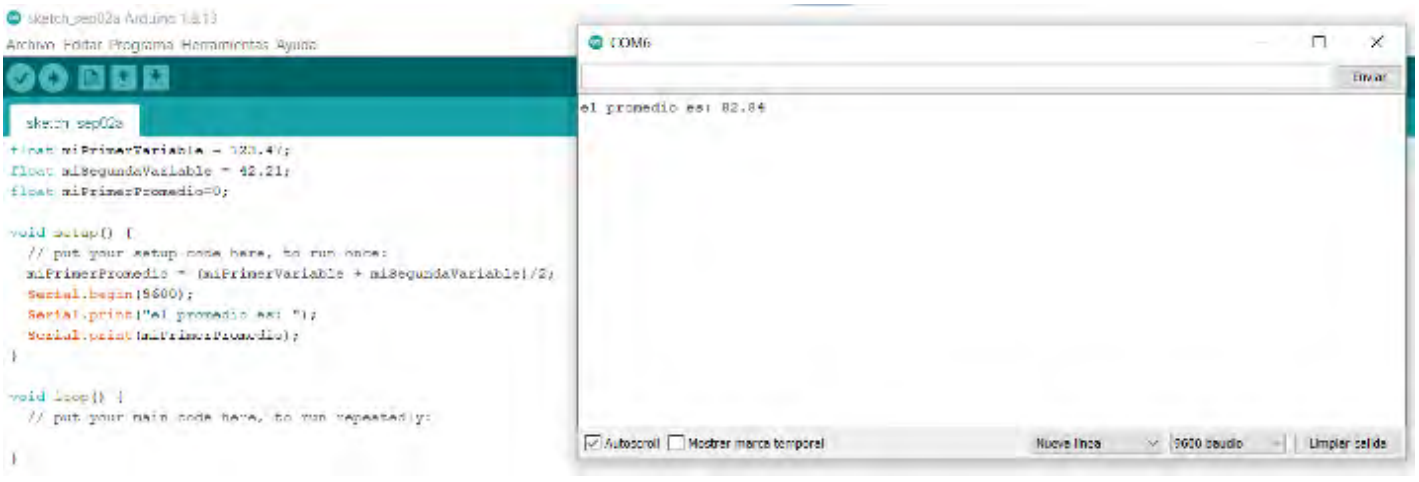
La terminal debe tener la misma velocidad en baudios que el programa (9600), finalmente compilamos del programa y lo corremos, nuestra pantalla debe mostrar la siguiente información.

también podemos utilizar otro tipo de operaciones como lo son la resta, multiplicación y división al igual que utilizar paréntesis.

- suma ----------> +
- resta----------> -
- multiplicación --> *
- división ----------> /
- raíz ----------> sqrt( )

(*sqrt* es el acrónimo de *square root* que en español sería raíz cuadrada)

Ahora, si quisiéramos hacer un promedio, al igual que en papel, podemos emplear el uso de paréntesis para que el orden de las operaciones sea el adecuado, esto se hace de la siguiente manera:

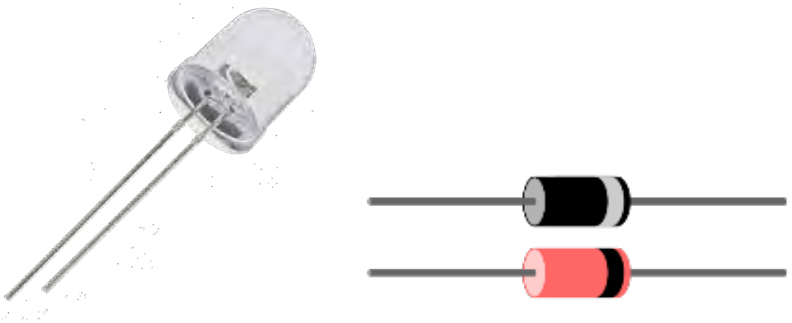


Como puedes ver, en esta ocasión utilizamos variables con decimales, esto lo hicimos ya que de otra forma solo quedaría guardado el promedio como 82.

LEDs y puertos de salida

En esta sección aprenderemos a utilizar un LED y los puertos de salida, pero, antes que nada, ¿Qué es un LED?

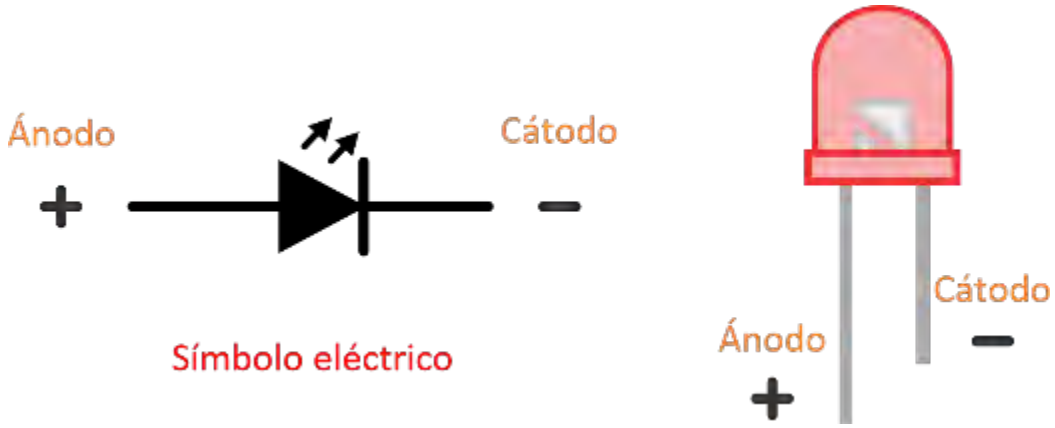
Un LED por sus siglas en inglés es Diodo Emisor de Luz *(Light Emitting Diode)* el cual es un componente electrónico de la familia de los semiconductores que cuenta con dos terminales, mientras que el uso de la mayoría de los diodos es bloquear la circulación de corriente en un sentido y permitirlo en el otro, los LEDs cuando están conectados a la corriente de manera correcta emitirán luz.



Led a la izquierda y diodo convencional a la derecha

Todos los diodos cuentan con dos terminales, el ánodo y el cátodo: El ánodo es la terminal que debe ir conectada a la fuente de voltaje (+) mientras que el cátodo se utiliza para cerrar el circuito (-).

En los LEDs podemos distinguir entre el ánodo y el cátodo tienen una de sus terminales más cortas que la otra, de esta manera podremos identificar y conectar correctamente un led a nuestros proyectos. Hay que tener en cuenta que el símbolo que veremos en los diagramas para este componente es una flecha que apunta a una línea perpendicular como los diodos, pero si este diodo es un LED de la flecha saldrán dos flechas más pequeñas que indican que emite luz.



Antes de prender nuestro primer led, hay que recordar la ley de Ohm, que dice que el voltaje es directamente proporcional a la corriente (intensidad eléctrica) por la resistencia (V=IR), debido a que un LED no cuenta con un valor considerable de resistencia, pero ya que necesita de corriente, los LEDs para evitar que se quemen deberán siempre ir acompañados de una resistencia, durante este curso utilizaremos resistencias de 330 Ohms para prenderlos.

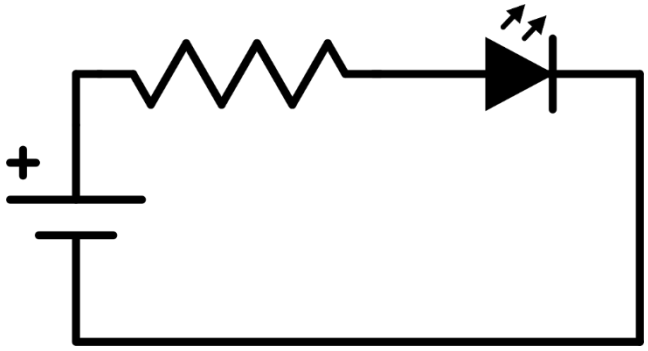
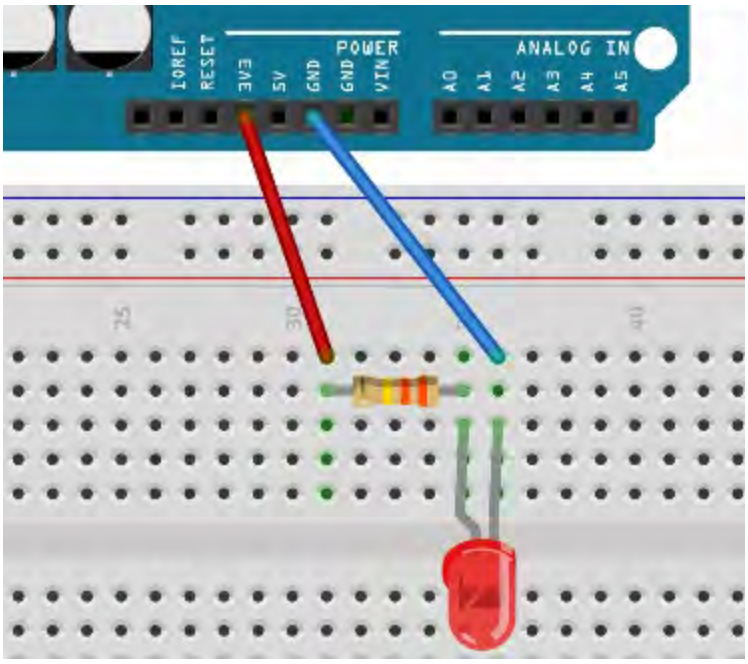
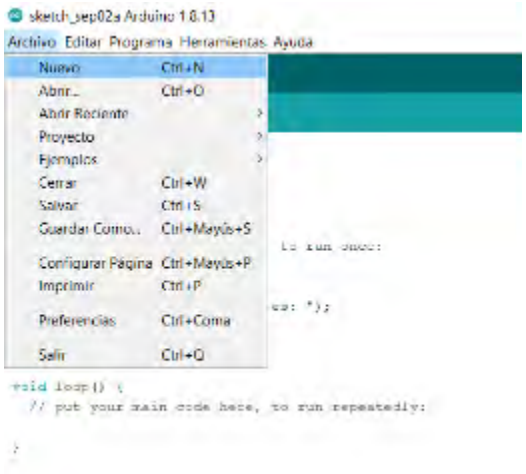


Diagrama para encender un LED

Vamos a probar el funcionamiento de este circuito utilizando nuestra tarjeta, esto lo haremos de la siguiente forma con el microcontrolador conectado a nuestro ordenador.



¿El LED prendió? de no ser así verifica las conexiones y verifica que el LED tenga su terminal más larga conectada a la resistencia.
Ahora veremos cómo prender el LED usando código, para esto abriremos un nuevo proyecto dando clic en archivo y nuevo.



Antes que nada, los pines que usaremos, debemos indicarle al microcontrolador que los usaremos como salidas. para esto en la función *setup* llamaremos a la función ***pinMode*** que como primer parámetro recibe el pin que usaremos y el segundo argumento es

input (puerto de entrada en inglés) u *output* (puerto de salida en inglés). para este proyecto escribiremos lo siguiente:

```
void setup() {
  // put your setup code here, to run once:
  pinMode(12 , OUTPUT);  //definir pin como salida
}
```

Ya que el objetivo de esta práctica es prender y apagar un led en intervalos de un segundo, esta vez trabajaremos dentro de la función *loop*.
Usaremos la función ***digitalWrite***, a la cual se le dan dos parámetros, el primer parámetro es el pin de salida que queremos que de voltaje (5 volts), mientras que el segundo estado es 0 para apagado y 1 para encendido, entonces escribimos lo siguiente.

```
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(12,1);   // poner el Pin en ENCENDIDO
```

separado por comas le decimos que el pin 12 deberá estar encendido es decir en 1.

Retrasos o "Delays"

La siguiente tarea es hacer que el microcontrolador espere 1 segundo para apagarse. Esto se puede lograr fácilmente usando una función conocida como *delay* (en español retraso) que tiene la siguiente estructura:

```
delay(milisegundos);
```

Un *delay* es comparable con un reloj de cocina, una vez que lo configuras, empezará a contar desde 0 hasta la cantidad que le hayamos indicado, y al finalizar en vez de hacer un sonido, nuestro programa continuará.

La función recibe como parámetro una cantidad en milisegundos, como nosotros solemos trabajar en segundos, hay que multiplicar la cantidad por mil de la siguiente manera.

```
delay(1*1000);
```

Otros ejemplos son los siguientes:

TIEMPO	CÓDIGO
1 segundo	delay(1*1000);
5 segundos	delay(5*1000);
1 minuto	delay(1*60*1000);
5 minutos	delay(5*60*1000);

Hay que recordar que los pines de salida entregan 5 volts y no hay que pedirles más de 0.5 Amperes

Continuando con el código anterior, ya que debemos mantener el LED encendido un segundo y apagado otro, ingresamos la siguiente línea de código:

```
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(12,1);    // poner el Pin en ENCENDIDO
  delay(1*1000);         // espera un segundo (1 segundo * 1000 = 1000 milisegundos)
```

Finalmente lo apagamos, repitiendo la primera línea de código y poniendo otro retraso de un segundo

Archivo Editar Programa Herramientas Ayuda

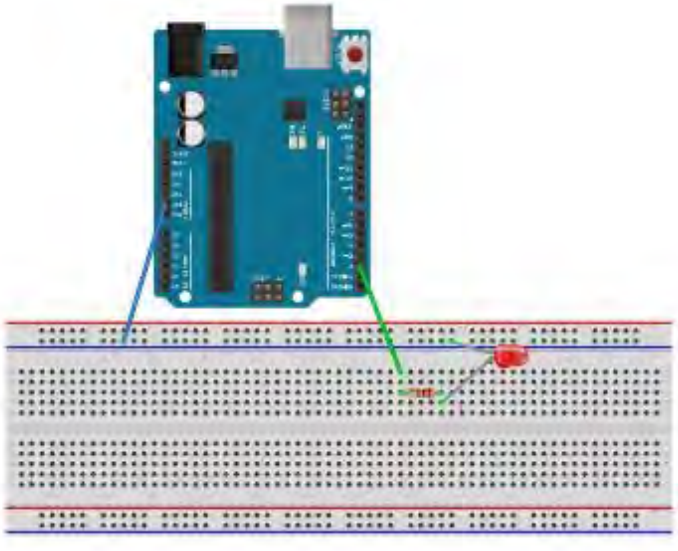
PrenderLed §

```
void setup() {
  // put your setup code here, to run once:
  pinMode(12 , OUTPUT);  //definir pin como salida
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(12,1);    // poner el Pin en ENCENDIDO
  delay(1*1000);         // espera un segundo (1 segundo * 1000 = 1000 milisegundos)
  digitalWrite(12,0);    // poner el Pin en APAGADO
  delay(1*1000);         // espera un segundo (1 segundo * 1000 = 1000 milisegundos)
}
```

Recuerda que al estar dentro de la función *loop,* una vez que termina la última línea del código vuelve a empezar desde el comienzo lo que está en esta función.

Después guardamos el programa dando clic en archivo, y en *guardar como*. le podemos dar el nombre que queramos.

Finalmente hay que conectar nuestro sistema de la siguiente forma.



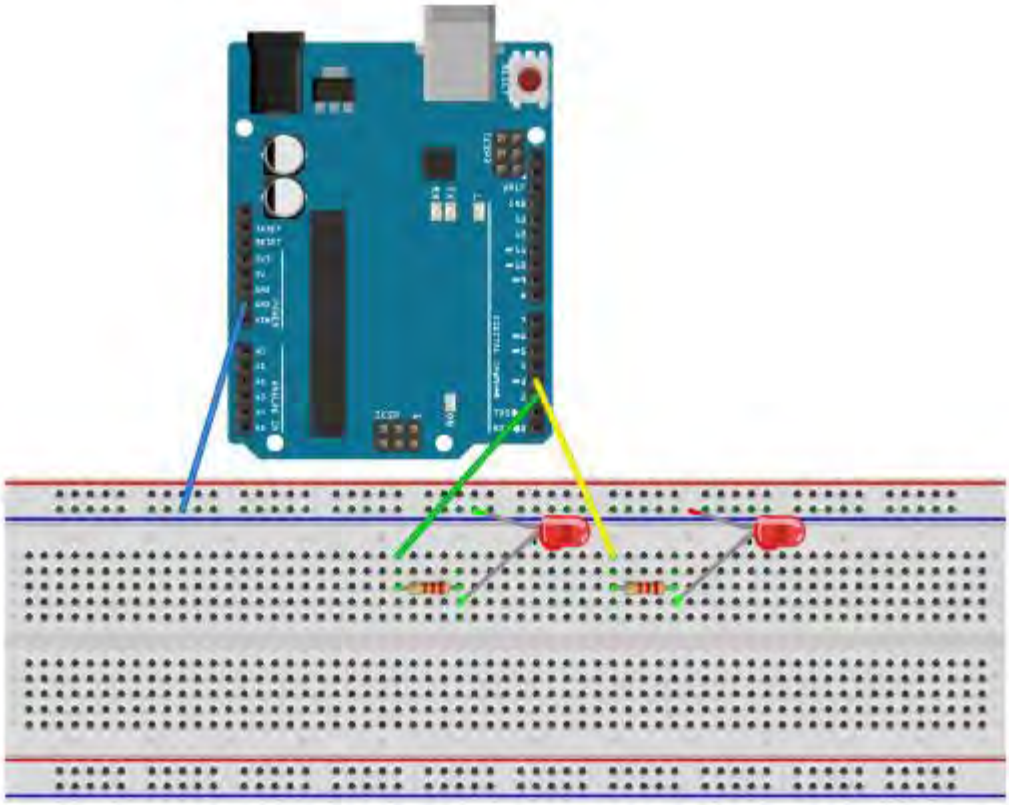
Pin 12 a resistencia, resistencia a led, led a tierra

Actividad 2: Camión de bomberos

Utilizando el siguiente diagrama, realice un código que:

- Prenda el LED izquierdo y por medio de la terminal diga: LED izquierdo encendido
- Después de un segundo este se apague
- Prenda el LED derecho y por medio de la terminal diga: LED derecho encendido.
- Después de un segundo este se apague
-

Introduzca su código en la función *loop* para que esto se repita de la misma forma que las luces un camión de bomberos.



Módulo 3: Condicionales y ciclos

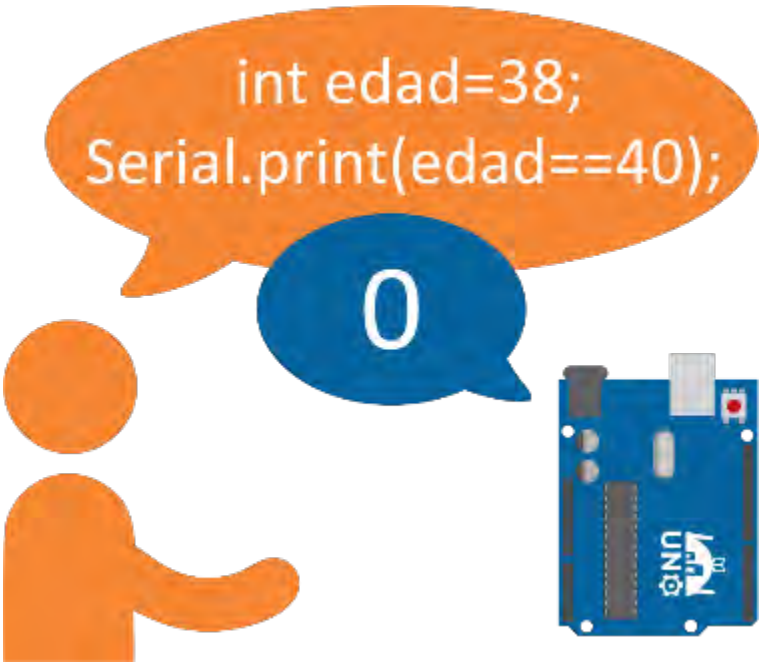
Operadores de comparación

En este módulo nos adentraremos en una de las partes más importantes de la programación, pero para ello hay que antes tener claro lo que son los operadores de comparación ya que para poder hacer condicionales y ciclos es importante tener en mente como se utilizan estos operadores.

Los operadores de comparación son comparaciones entre dos variables que se pueden responder con un verdadero o falso, sirven para poder identificar si un evento se está cumpliendo o no lo está haciendo, por ejemplo:

Si le preguntas a alguien si su edad es de 40 años te puede a responder con un "si" o un "no", en caso de que diga que no, podrías preguntarle también si su edad es mayor a los 40 y de la misma manera te puede responder con un "si" o "no".

Este tipo de preguntas también se las podemos hacer a nuestro microcontrolador, para formular esta pregunta en el lenguaje que en este curso utilizamos, sería de la siguiente forma:



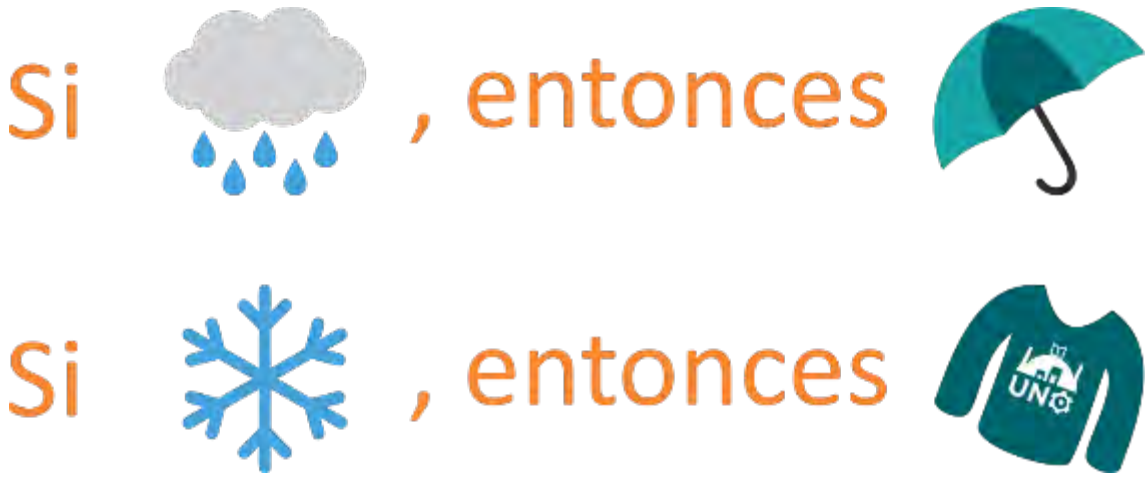
En general, todos los microprocesadores responden cuando se les hace una pregunta con un operador de comparación con un 1 o un 0, siendo 1 el equivalente a verdadero y 0 el corresponde a falso, a esto se le conoce como una respuesta booleana
En la siguiente tabla te mostramos algunos de los operadores de comparación que puedes llegar a necesitar en tus proyectos personales y en este curso ¿Puedes imaginarte el uso que les puedes dar?

EN ESPAÑOL	EN EL MICROCONTROLADOR
x igual que y	x == y
x diferente que y	x ! = y
x menor que y	x < y
x mayor que y	x > y
x menor o igual que y	x <= y
x mayor o igual que y	x >= y

Recuerda que “x” y “y” son variables que nosotros asignaremos en la programación

Condicionales "if"

En el día a día tomamos decisiones cuando ciertos eventos suceden y cambiamos nuestra forma de actuar dependiendo de ello: si hace frío **entonces** me pongo suéter, **si** está lloviendo **entonces** salgo con paraguas, si tengo sed **entonces** tomo agua.



Al igual que nosotros tomamos decisiones basadas en "si ocurre esto entonces hago esto", los procesadores son capaces de tomar decisiones basados en hechos y a este tipo de proceso se le conoce como **condicional.**

En la mayoría de los lenguajes de programación se utiliza la palabra ***if*** que en inglés es el condicional "si", y su estructura en el lenguaje de programación sería la siguiente.

```
if(operador de comparación){
        //Escribir tus acciones aquí
}
```

Como puedes ver, los condicionales llevan un operador de comparación dentro, y cuando es verdadero, lo que está dentro de los corchetes se ejecutará.

Este condicional puede ser complementado con otro condicional conocido como else, este condicional funcionará si no ha entrado en el *if* que previamente debimos haber definido, esto funciona de la siguiente manera:

```
if(operador de comparación){
        //Escribir tus acciones aquí

else{    // si el caso anterior no se cumplió
        //Realiza
}
```

Imaginemos ahora el caso de que queremos que nuestro microcontrolador funcione como un semáforo, queremos que primero prenda el color verde por 5 segundos, luego prenda el amarillo 2 segundos y finalmente prenda el rojo durante 5 segundos:

Pudiendo decir que, si el contador está en 0 empiece en verde, si está en 5 cambie a amarillo y que si está en 7 pase a rojo. Nosotros tendríamos que utilizar otro recurso de los condicionales conocida en inglés como ***else if*** que en español es "o si" cuya estructura es la siguiente:

```
if(operador de comparación){
        //Escribir tus acciones aquí
}
else if(otro operador de comparación distinto){
        //Realiza acciones distintas aquí
}
```

Puedes poner la cantidad de *else if* que necesites, después de un *if*, en caso de necesitar un *else* este funcionará si no entró en un *if* o *else if*, esto con la siguiente estructura:

```
if(operador de comparación){
        //Escribir tus acciones aquí
}
else if(operador de comparación 2){
        //Realiza acciones distintas aquí
}
else if(operador de comparación 3){
        //Realiza acciones distintas aquí
}

else{    // si ningún caso anterior se cumplió
        //Realiza
}
```

Ahora, regresando al ejercicio del semáforo, el código sería el siguiente:

Semaforo §

```
int contador=0;              //Declaramos la variable del contador

void setup() {
    pinMode(2,OUTPUT);       //Le decimos al micro que el puerto 2 es de salida
    pinMode(3,OUTPUT);       //Le decimos al micro que el puerto 3 es de salida
    pinMode(4,OUTPUT);       //Le decimos al micro que el puerto 4 es de salida
}

void loop() {

    if(contador==0){         //si nuestro contador vale 0
      digitalWrite(2,1);     //Prendemos el LED 2
      digitalWrite(3,0);     //Apagamos  el LED 3
      digitalWrite(4,0);     //Apagamos  el LED 4
    }
    else if(contador==5){    //si nuestro contador vale 5
      digitalWrite(2,0);     //Apagamos  el LED 2
      digitalWrite(3,1);     //Prendemos el LED 3
      digitalWrite(4,0);     //Apagamos  el LED 4
    }
    else if(contador==7){    //si nuestro contador vale 7
      digitalWrite(2,0);     //Apagamos  el LED 2
      digitalWrite(3,0);     //Prendemos el LED 3
      digitalWrite(4,1);     //Apagamos  el LED 4
    }

    delay(1*1000);         //Esperamos un segundo
    contador=contador+1; //Sumamos 1 al contador

    if(contador>=12){      //Si nuestro contador es mayor o igual que 12
      contador=0;          //Reiniciamos el contador
    }
}
```

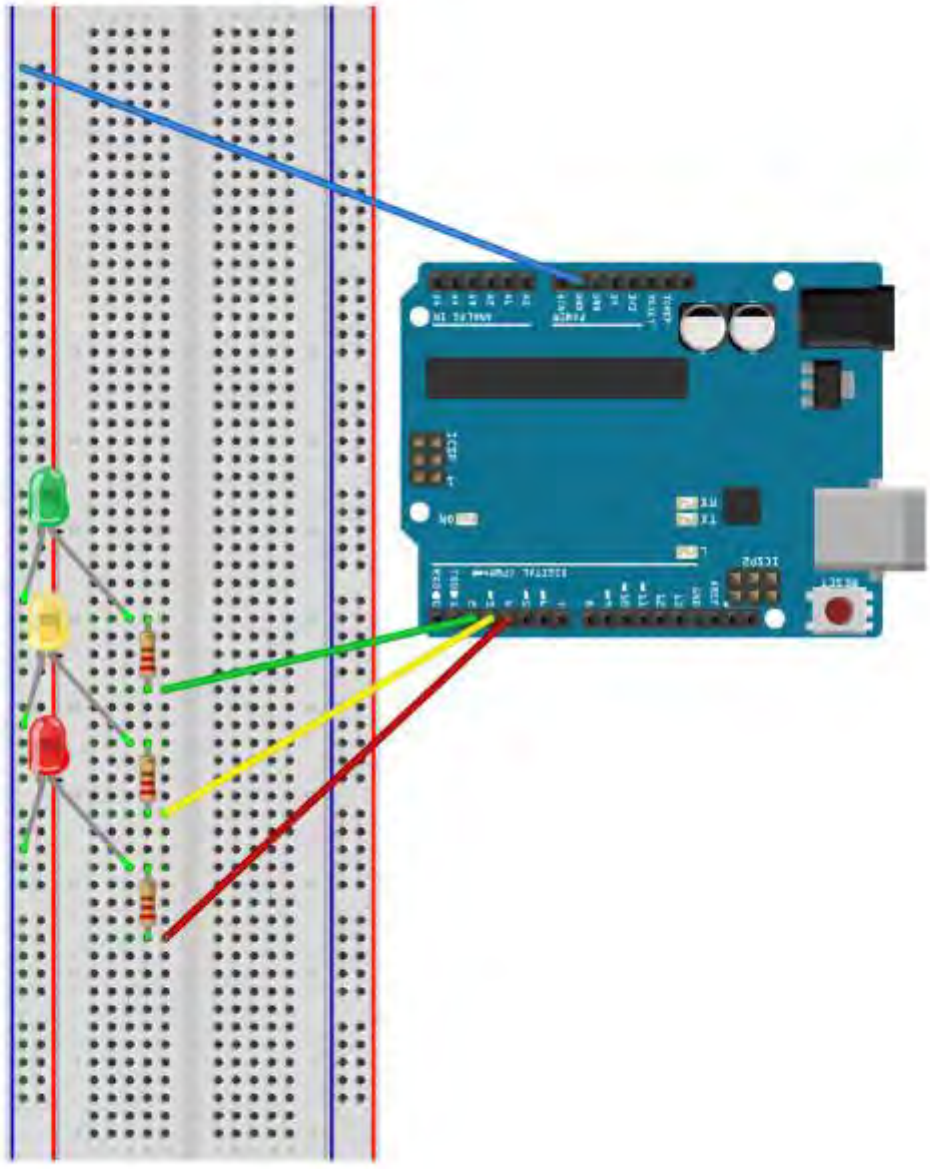


Diagrama a utilizar para el código del semáforo.

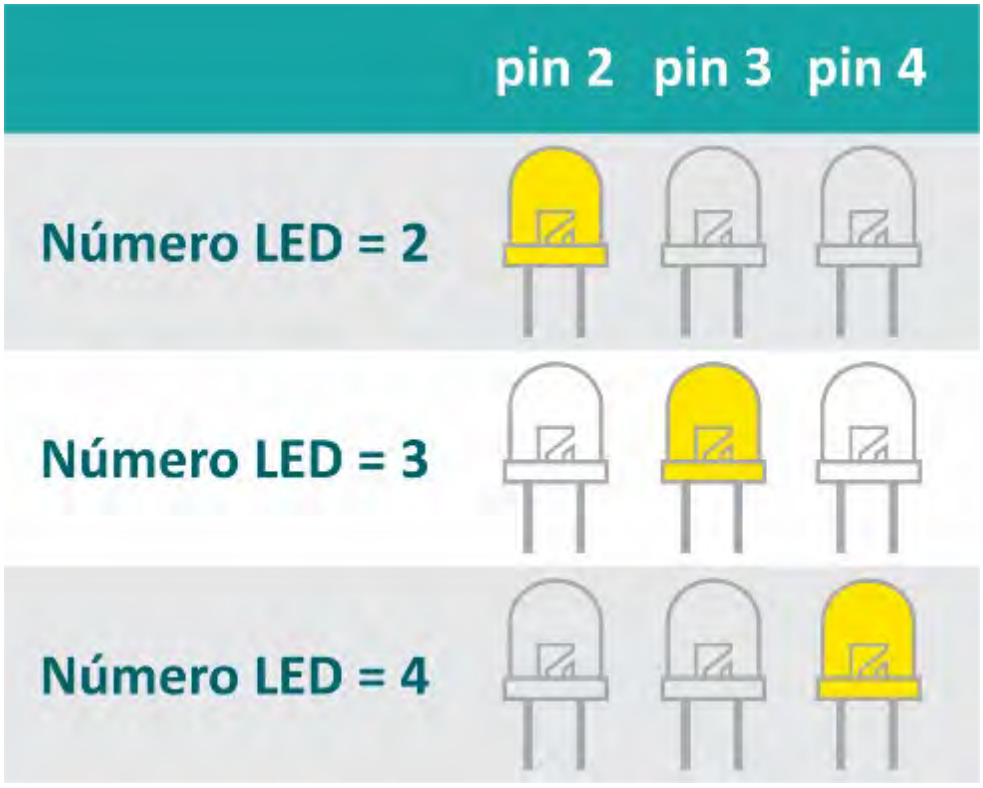
Ciclo "for"

Los ciclos son una herramienta de programación que ayuda a hacer tareas repetitivas cierta cantidad de veces, en este curso utilizaremos los dos principales tipos de ciclos. Los ciclos *for* ("para" en español) es un ciclo que se utiliza para realizar un número determinado de "vueltas". la estructura en español de esta herramienta es la siguiente:

Para la variable x=y, mientras sea verdadero operador condicional, cambiar x

por ejemplo, si nosotros quisiéramos que nuestros leds prendieran uno después del otro como si la luz fuera caminando:

Para la variable número LED=2, mientras sea verdadero número LED menor o igual a 5, sumar 1 a número LED



Que en lenguaje de programación se escribe de la siguiente manera:

```
//2 porque los LEDs están conectados desde el puerto 2

for(numeroLED=2;numeroLED <=4;numeroLED++){

        digitalWrite(numeroLED,1);       //Prender LED en la posición

        delay((1*1000)/2);               //medio segundo 500 milisegundos

        digitalWrite(numeroLED,0);       //Apagar LED en la posición numeroLED

}
```

Nota: numeroLED++ es una forma de abreviar numeroLED = numeroLED +1

El código completo sería el siguiente:

for §

```
int numeroLED=0;             //Declaramos la variable

void setup() {
    pinMode(2,OUTPUT)        //Le decimos al micro que el puerto 2 es de salida
    pinMode(3,OUTPUT);       //Le decimos al micro que el puerto 3 es de salida
    pinMode(4,OUTPUT);       //Le decimos al micro que el puerto 4 es de salida
}

void loop() {

    for(numeroLED=2;numeroLED<=4;numeroLED++){
        digitalWrite(numeroLED,1);          //Prender LED en la posición
        delay((1*1000)/2);                  //esperar medio segundo, puedes poner 500 milisegundos
        digitalWrite(numeroLED,0);          //Apagar LED en la posición numeroLED
    }

}
```

Recuerda que la conexión sería la misma que la del semáforo.

Ciclo "while"

Los ciclos ***while*** ("mientras" en español) son para hacer una acción mientras una condición no se cumpla. La estructura en español sería la siguiente:

```
While(condicional){

    //código a repetir aquí

}
```

Podemos usar el ciclo *while* como si fuera un ciclo *for* agregando a la parte final una suma a la variable del condicional, de esta manera quedaría de la siguiente manera:

```
While(x<5){

    //código a repetir aquí

    x++;

}
```

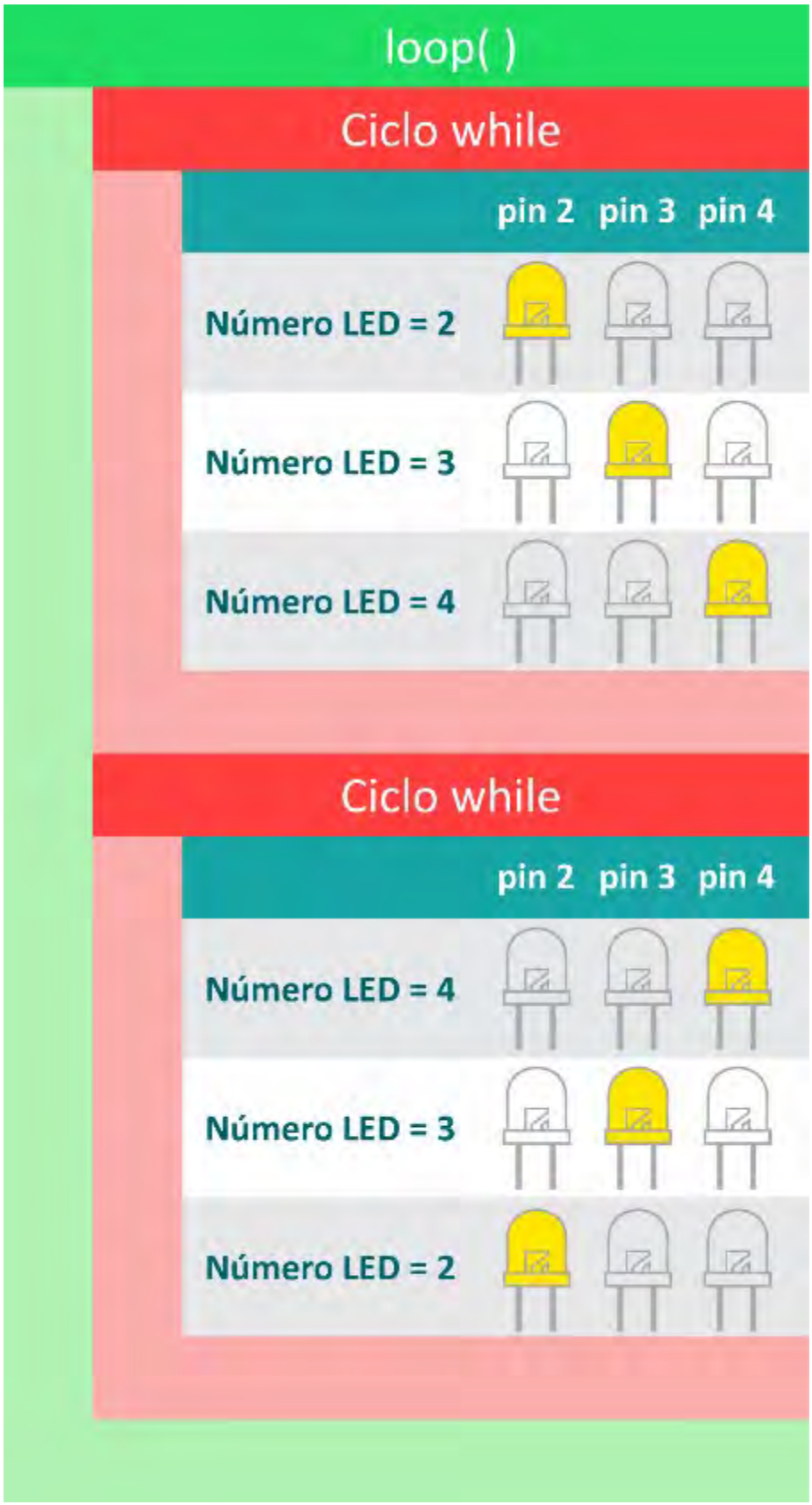
A diferencia del ciclo for, este se puede utilizar para situaciones donde no sabemos cuántas veces tenemos que correr la función para dejar de repetirlo y es muy útil cuando se trabaja con entradas externas.

Actividad 3.1: Luz que rebota

Con 3 LEDs conectados a los puertos digitales 2,3 y 4, realice lo siguiente:

- Configure los puertos 2,3,4 como salidas.
- Crea la variable contador
- Basándote en el ejercicio anterior, utiliza un ciclo while para que prendan los LEDs de manera ascendente.
- Una vez terminado el ciclo anterior, realice uno que prenda los LEDs de manera descendente y colóquelo debajo del anterior.
- Ingrese sus dos ciclos en la función *loop* para que se repita este ciclo indefinidamente.

Pista: Ten cuidado con el valor que tenga tu contador antes de empezar un ciclo y el valor que tendrá al terminarlo, si no está en el valor que deseas, puedes reinicializar su valor antes del siguiente ciclo.


loop( )
Ciclo while
pin 2 pin 3 pin 4
Número LED = 2
Número LED = 3
Número LED = 4
Ciclo while
pin 2 pin 3 pin 4
Número LED = 4
Número LED = 3
Número LED = 2

Actividad 3.2: Contador a 10

Haciendo uso de condicionales, y los puertos 2 y 3 realice lo siguiente:

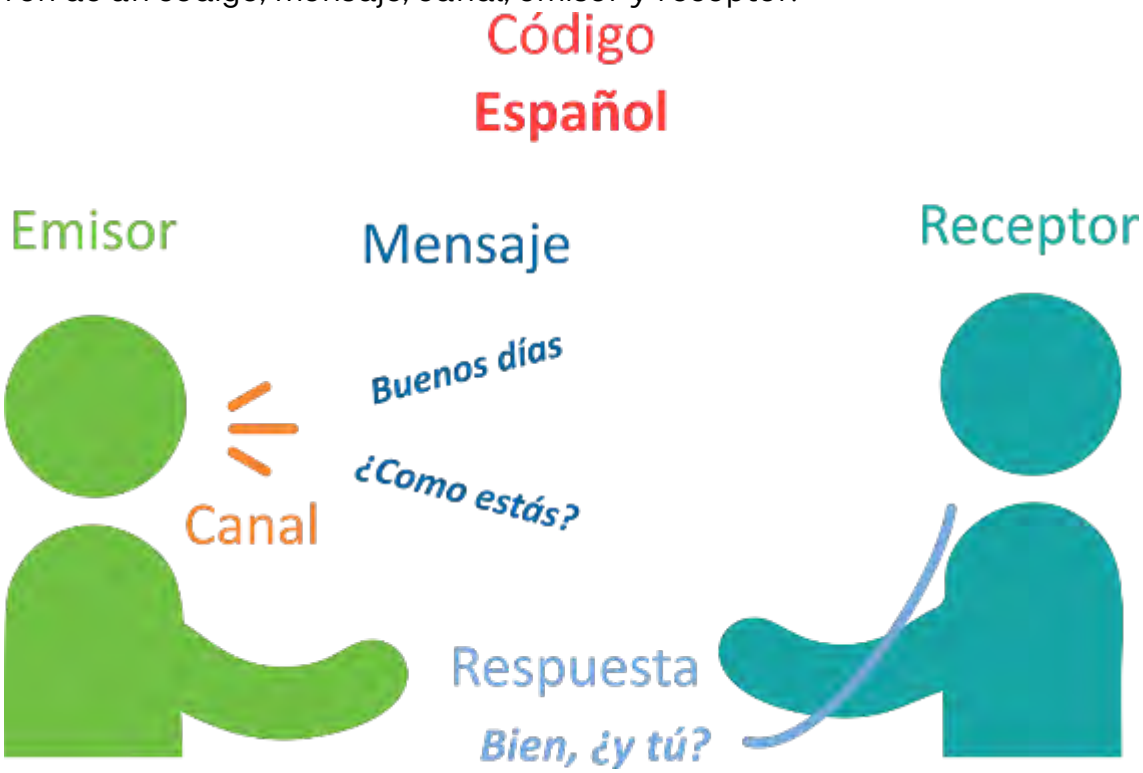
- Configure los puertos 2,3 como salidas.
- Crea la variable contador
- Dentro de la función *loop* haga que vaya cambiando de 0 a 10
- Haga que el LED 2 parpadee cada que cambie la variable contador
- Si la variable contador llega a 10 reiníciela a 0 y haga que parpadee el LED 3

Pista: Puede ayudarte mostrar en consola la variable contador para que sepas corregir errores

Módulo 4: Comunicación serial

Bases de la comunicación

Los microcontroladores necesitan de formas para comunicarse con otros microcontroladores o computadoras, pero al igual que las personas, ambos dispositivos deben estar comunicándose de la misma forma para poder intercambiar información. La comunicación entre microprocesadores al igual que las personas requieren de un código, mensaje, canal, emisor y receptor.



El código cuando hablamos de procesadores se le conoce como protocolo de comunicación. Existen infinidad de protocolos que se especializan en ciertas áreas, tales como en el área automotriz, en conexión con servidores (internet), dispositivos electrónicos entre muchos otros.

En este curso utilizaremos el protocolo que utilizaremos se conoce como **UART** o puerto serie que en español sería **recepción-transmisión asíncrona universal** que comúnmente se utiliza para transmitir bytes de memoria.

Pero primero, ¿Qué son los bytes?

Bits y bytes

Recuerdas cuando mencionamos al inicio del curso que los microprocesadores funcionan a partir de transistores, los cuales tienen dos estados principalmente. encendido y apagado.

Es por eso por lo que los procesadores a diferencia de los humanos trabajan usando dos dígitos (cero y uno) a esto se le conoce como **bit**, el cual es la cantidad más pequeña de información que procesan las computadoras, esto significa que los números los lee el microprocesador de la siguiente forma:

SISTEMA DECIMAL HUMANO	SISTEMA BINARIO (PROCESADORES)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Como puedes ver, al igual que con el sistema decimal, utilizamos todos los dígitos que tenemos, y al llegar al límite, empezamos de nuevo empezando de nuevo cambiando el valor de un dígito a la izquierda.

0,1,2,3,4,5,6,7,8,9,10

Entonces ya que definimos lo que es un bit, podemos definir lo que es un **byte**, un byte es el conjunto de 8 bits, es decir, desde:

0000 0000 hasta 1111 1111 ó desde 0 hasta 255

Esto significa que por UART mandaremos algún número entre 0 y 255, es decir un byte a la vez.

ASCII

Recordando lo que vimos en el primer módulo, existen variables que no son numéricas tales como los caracteres, los caracteres cuando se utiliza el formato ASCII (se pronuncia aski) tienen el tamaño de un byte, para hacer esto se asigna a cada valor numérico un valor entre 0 y 255. **ASCII** por sus siglas en inglés *American Standard Code for Information Interchange* o en español Código Estándar Americano para Intercambio de Información.

Caracteres ASCII de control			Caracteres ASCII imprimibles						ASCII extendido (Página de código 437)							
00	NULL	(carácter nulo)	32	espacio	64	@	96	`	128	Ç	160	á	192	└	224	Ó
01	SOH	(inicio encabezado)	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
02	STX	(inicio texto)	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
03	ETX	(fin de texto)	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
04	EOT	(fin transmisión)	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
05	ENQ	(consulta)	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
06	ACK	(reconocimiento)	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
07	BEL	(timbre)	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
08	BS	(retroceso)	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
09	HT	(tab horizontal)	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	LF	(nueva línea)	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	VT	(tab vertical)	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	FF	(nueva página)	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	CR	(retorno de carro)	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	SO	(desplaza afuera)	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	SI	(desplaza adentro)	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p	144	É	176	░	208	ð	240	≡
17	DC1	(control disp. 1)	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	DC2	(control disp. 2)	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	DC3	(control disp. 3)	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	DC4	(control disp. 4)	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	NAK	(conf. negativa)	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	SYN	(inactividad sínc)	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	ETB	(fin bloque trans)	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	CAN	(cancelar)	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	EM	(fin del medio)	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	SUB	(sustitución)	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	ESC	(escape)	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	FS	(sep. archivos)	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	GS	(sep. grupos)	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	RS	(sep. registros)	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	US	(sep. unidades)	63	?	95	_			159	ƒ	191	┐	223	▀	255	nbsp
127	DEL	(suprimir)														

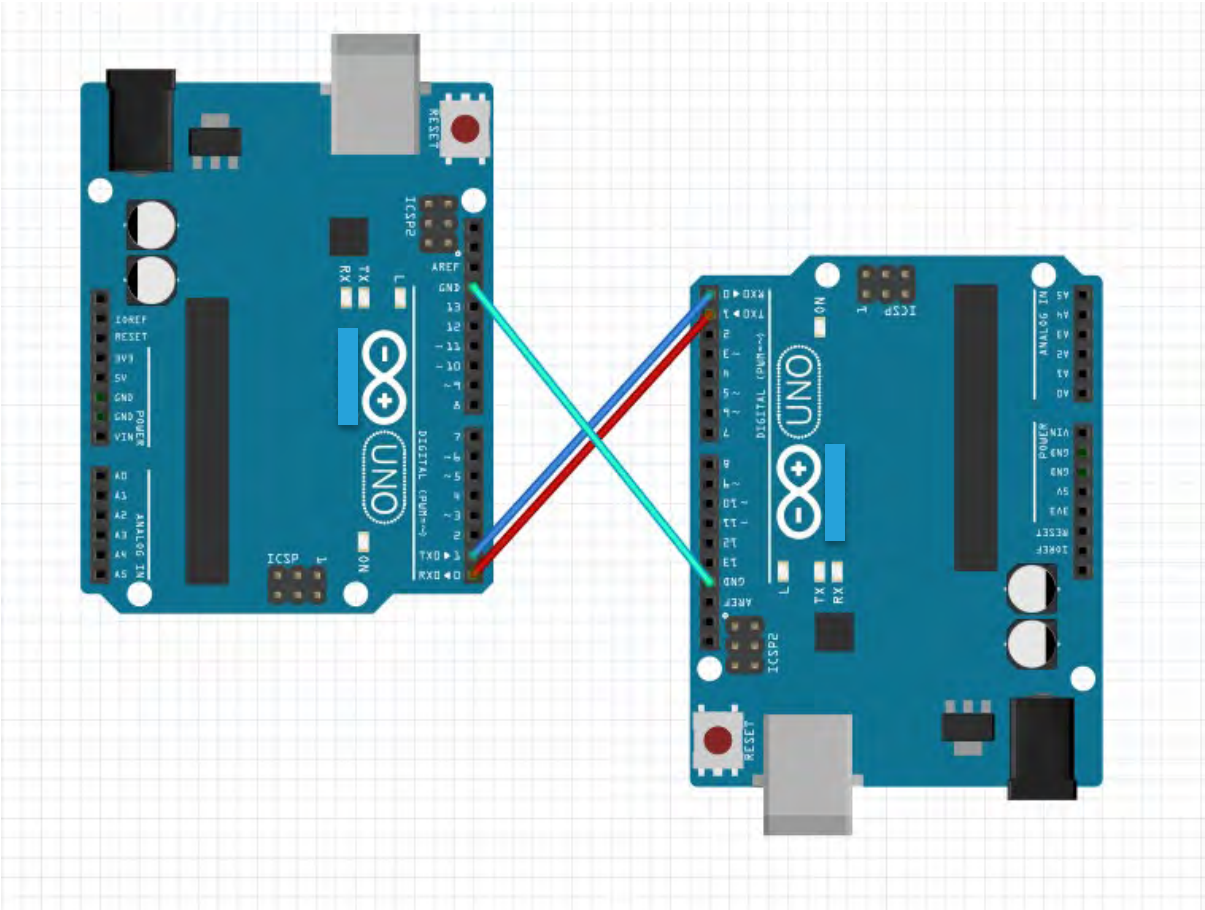
Nota cultural

La tecnología actualmente utiliza otro formato llamado UTF-8, esto debido a que el ASCII funciona bien mientras se trabaje en inglés, al necesitar más caracteres para otros idiomas e incluir otros símbolos, este sistema no alcanza, por lo que UTF-8 formato utiliza más bytes. Más del 95% del internet utiliza este formato.

Como puedes ver en la tabla anterior, el número 65 corresponde a la letra A mayúscula y la 90 corresponde a la Z mayúscula, mientras que la 97 corresponde a la "a" minúscula y la 122 a la "z" minúscula.

Bases de la UART

Con todo lo anterior en mente, es momento de enviar información usando la UART de nuestro microcontrolador, para ello es importante resaltar que necesitamos al menos 3 pines para que la comunicación funcione. Un cable emisor **(Tx)**, un cable receptor **(Rx)** y que tengan tierra conectada en común (La mayoría de los errores en este tipo de comunicación es porque la tierra de ambos dispositivos no es la misma)



Como puedes ver en el diagrama anterior, el puerto TX de un microcontrolador va conectado al puerto RX del otro microcontrolador, teniendo conectadas entre sí, las tierras de los microcontroladores.

En esta situación, nos comunicaremos usando el módulo bluetooth que se conoce como HC-06 para enviar texto desde un celular con Android como sistema operativo.

Configurar pines para UART (Software serial)

Nuestro Arduino puede enviar y recibir información por sus puertos digitales, debido a que los puertos 0 y 1 están directamente conectados a nuestro ordenador (La computadora envía por UART las instrucciones que programamos a nuestra tarjeta de desarrollo usando este protocolo), para evitar complicaciones a la hora de cargar el programa usaremos el puerto 2 y 3.

Para hacer esto usaremos código que alguien más ya programó, a esto se le conoce como importar una librería.

La librería o el código que tomaremos prestado se llama ***SoftwareSerial***, nosotros haremos uso de este código escribiendo la siguiente línea hasta arriba, en la parte superior de nuestro código.

```
#include <SoftwareSerial.h>
```

Esto se verá de la siguiente forma:

```
sketch_oct01a §
#include <SoftwareSerial.h>

void setup() {

}

void loop() {


}
```

Una vez hecho esto podremos configurar los pines que queramos como puertos UART mediante *software*

Nota: *Hardware* son todos los componentes que podemos tocar, como lo son los transistores, circuitos integrados, microcontroladores entre otras cosas, mientras que *software* es todo aquello intangible pero involucrado en computación y electrónica, como lo son los programas que nosotros hacemos.

Para empezar, tenemos que declarar antes de la misma manera que las variables una estructura con los puertos a usar, es por eso que antes de la función *setup* escribiremos la siguente línea

```
SoftwareSerial HC06(2, 3);    // declaramos los pines de recepción y emisión (Rx=2,Tx=3)
```

Finalmente, al igual que para la comunicación con la terminal, hay que definir el baudio que usaremos, en este caso esperamos recibir vía bluetooth y todo lo que recibamos lo enviaremos a la terminal.

```
void setup(){
    Serial.begin(9600);         // Configurar velocidad de comunicación con la terminal
    HC06.begin(9600);           // Configurar velocidad de comunicación con el módulo bluetooth
}
```

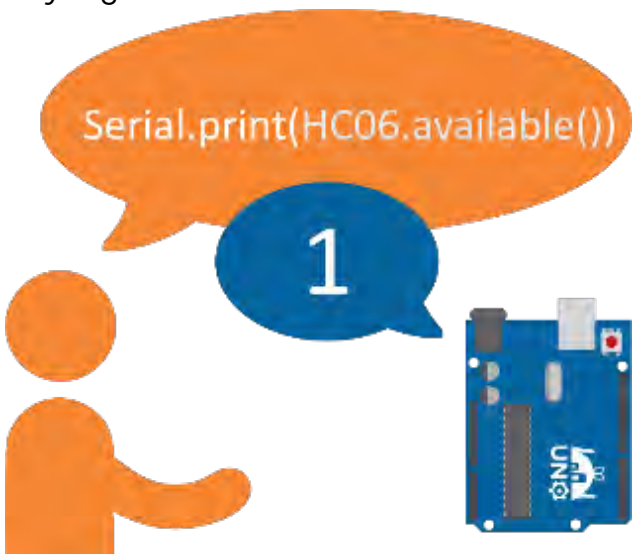
Con esto queda ya configurado el módulo bluetooth (HC-06).

Recibir información por UART

Una vez configurados los pines como en el capítulo anterior, nosotros podremos recibir información del módulo bluetooth el cual nos enviará por UART la información que recibe a nuestro microcontrolador.

El módulo bluetooth enviará todos los caracteres que enviemos a nuestro microcontrolador y este lo guardará para que nosotros podamos usarlo o manipularlo, a esta memoria donde se guarda lo que recibimos se le conoce como *buffer* o en español memoria intermedia.

Supongamos que nosotros ya recibimos la información y está almacenada en nuestro ***buffer***, ¿Cómo sabemos si hay información guardada? Esto lo haremos preguntándole al microcontrolador si hay algo almacenado.



Nota: Recuerda que las máquinas responden “si” con un 1 y “no” con un 0

Nosotros podemos preguntarle al microcontrolador si tiene guardado en el buffer designado al módulo bluetooth con la siguiente línea de código.

```
HC06.available()
```

Ya que la respuesta a esto será binaria, podemos usar esta función dentro de un condicional de la misma forma que hacíamos con los operadores de comparación vistos con anterioridad.

```
if(HC06.available()){
        //Escribir tus acciones aquí
}
```

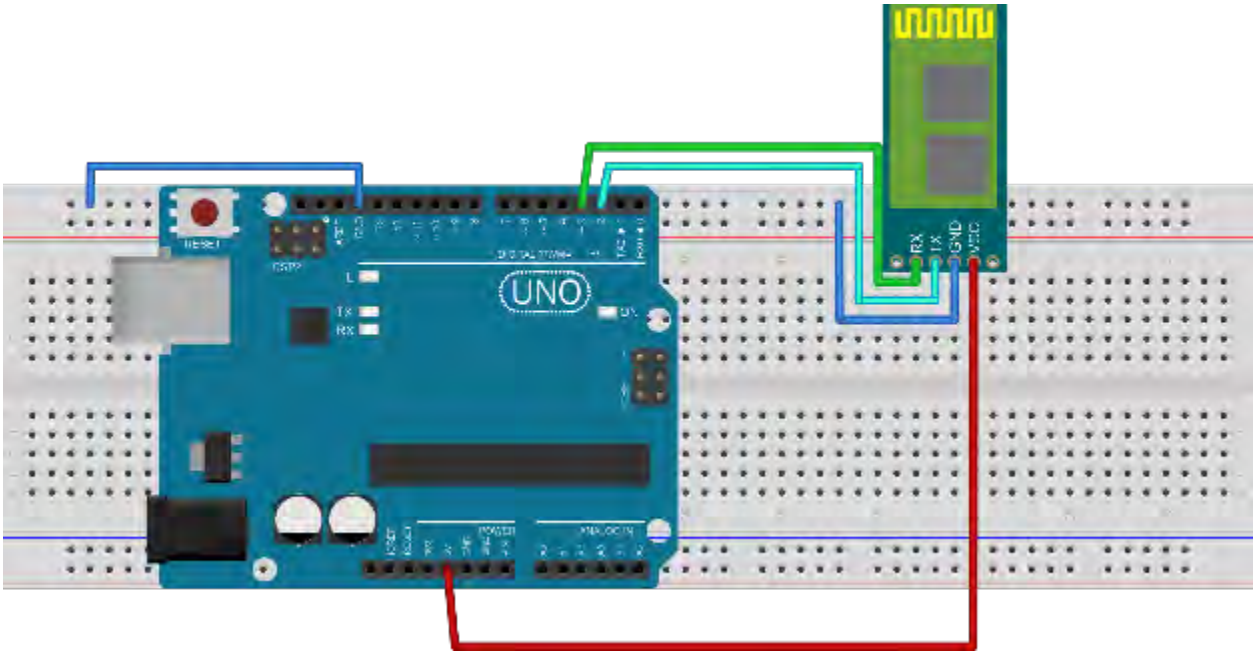
Y como ya sospecharás, la acción que queremos hacer, será leer el buffer y mandarlo a nuestra terminal, esto lo haremos con la siguiente línea:

```
HC06.read()
```

Nuestro código completo sería así:

UART §

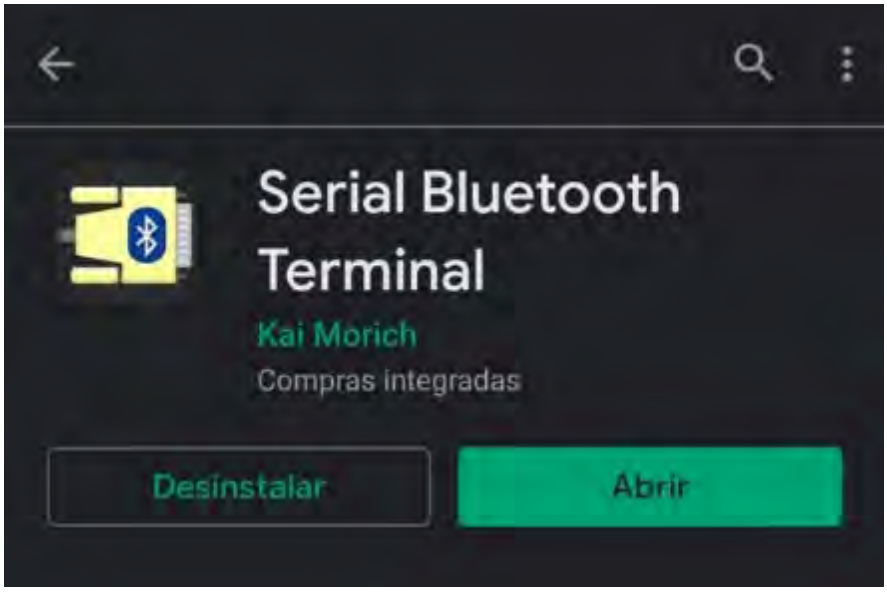
```
#include <SoftwareSerial.h>

SoftwareSerial HC06(2, 3);      // declaramos los pines de recepción y emisión (Rx=2,Tx=3)

void setup(){
    Serial.begin(9600);         // Configurar velocidad de comunicación con la terminal
    HC06.begin(9600);           // Configurar velocidad de comunicación con el módulo bluetooth
}

void loop(){
    if (HC06.available()){      //Si hay información en el buffer
        Serial.write(HC06.read()); //Mandar vía bluetooth lo enviado por la terminal
    }
}
```



Actividad 4: Enviar información por bluetooth

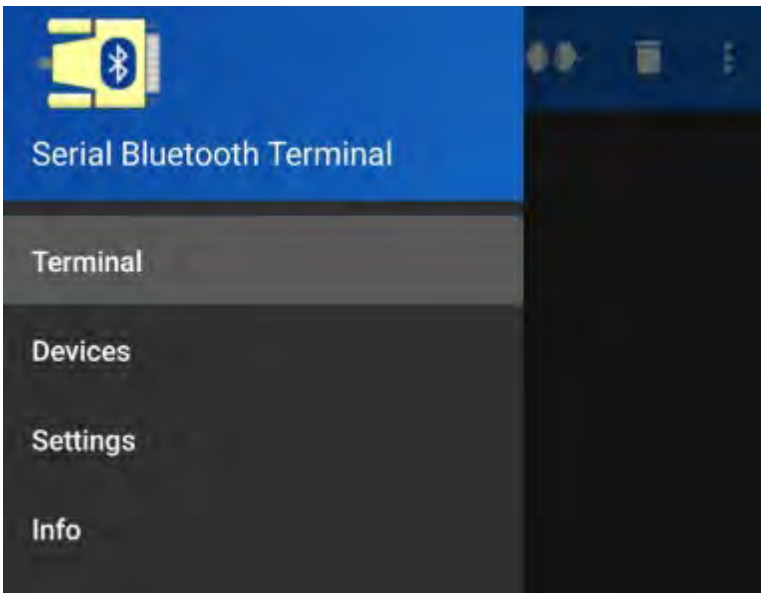
Usando el diagrama y código del capítulo anterior, haga lo siguiente:

1. Desde un teléfono inteligente con play store, descargue la aplicación Serial Bluetooth Terminal



2. Una vez instalada, compilamos y ejecutamos nuestro código
3. Abrimos los ajustes bluetooth de nuestro teléfono inteligente y buscamos nuestro dispositivo bluetooth

4. Nos conectamos a el dando clic, la contraseña por default de estos módulos es 1234 ó 0000, recuerda ingresar correctamente la contraseña.
5. Una vez vinculado, abrimos la aplicación y en el menú damos clic en *devices*, y escogemos nuestro dispositivo.



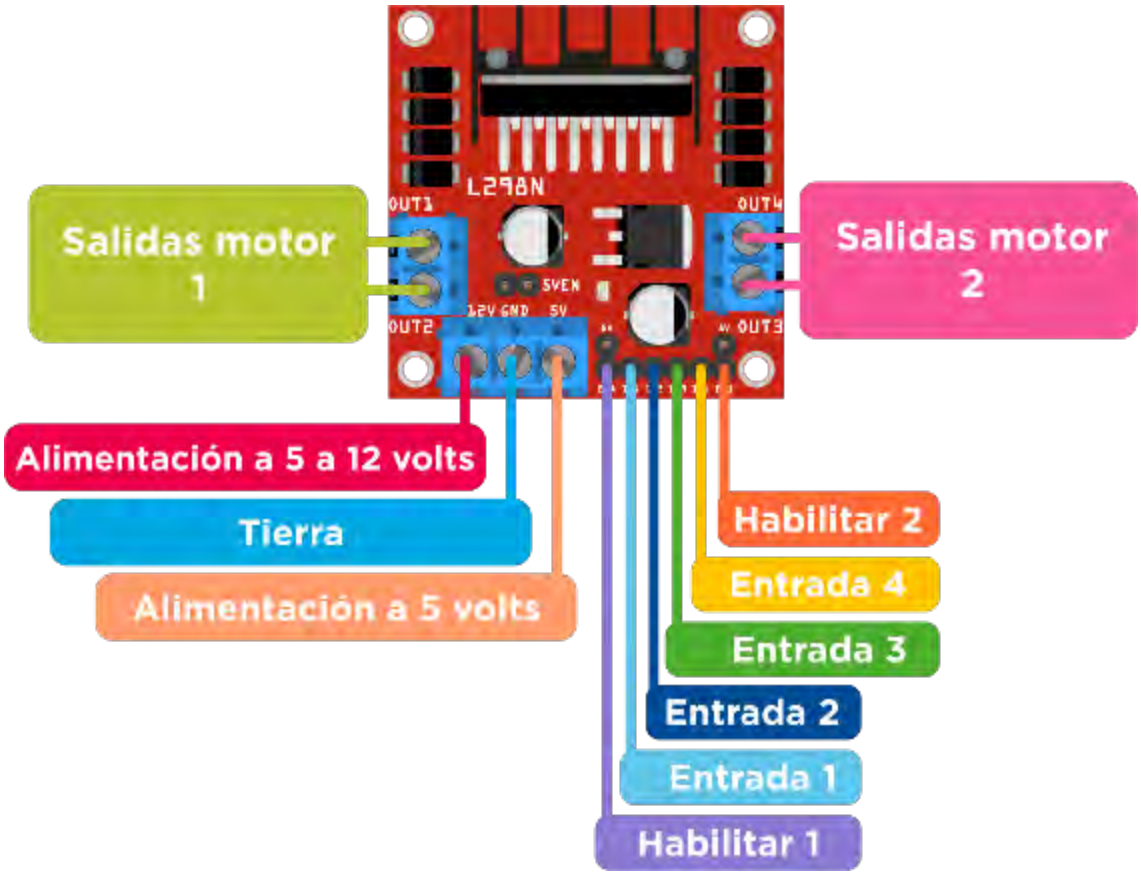
6. Finalmente escribe algo en el teclado de la aplicación, y mira la terminal de tu computadora ¿Qué sucedió?

Proyecto final

Para el proyecto final usaremos todo lo aprendido en este curso, este proyecto consiste de un robot simple que podrá ir hacia adelante, hacia atrás o girar sobre su propio eje comunicándose vía bluetooth haciendo uso de la UART.
Para hacer esto, haremos uso de un componente que no hemos aprendido a usar, conocido como puente H.

Controlador de motores dual (L298N)

Debido a que de los pines de la tarjeta de desarrollo no podemos mandar voltajes altos y corrientes de gran magnitud *(Hay que recordar que solo podemos dar 5V y 0.5A por los puertos digitales)* Necesitamos una etapa de transistores que, con los pines de nuestra tarjeta. Pero debido a que queremos controlar la polaridad (sentido) de nuestros motores también, usaremos un circuito conocido como **L298N.**



El comportamiento de este componente está descrito por la siguiente tabla.

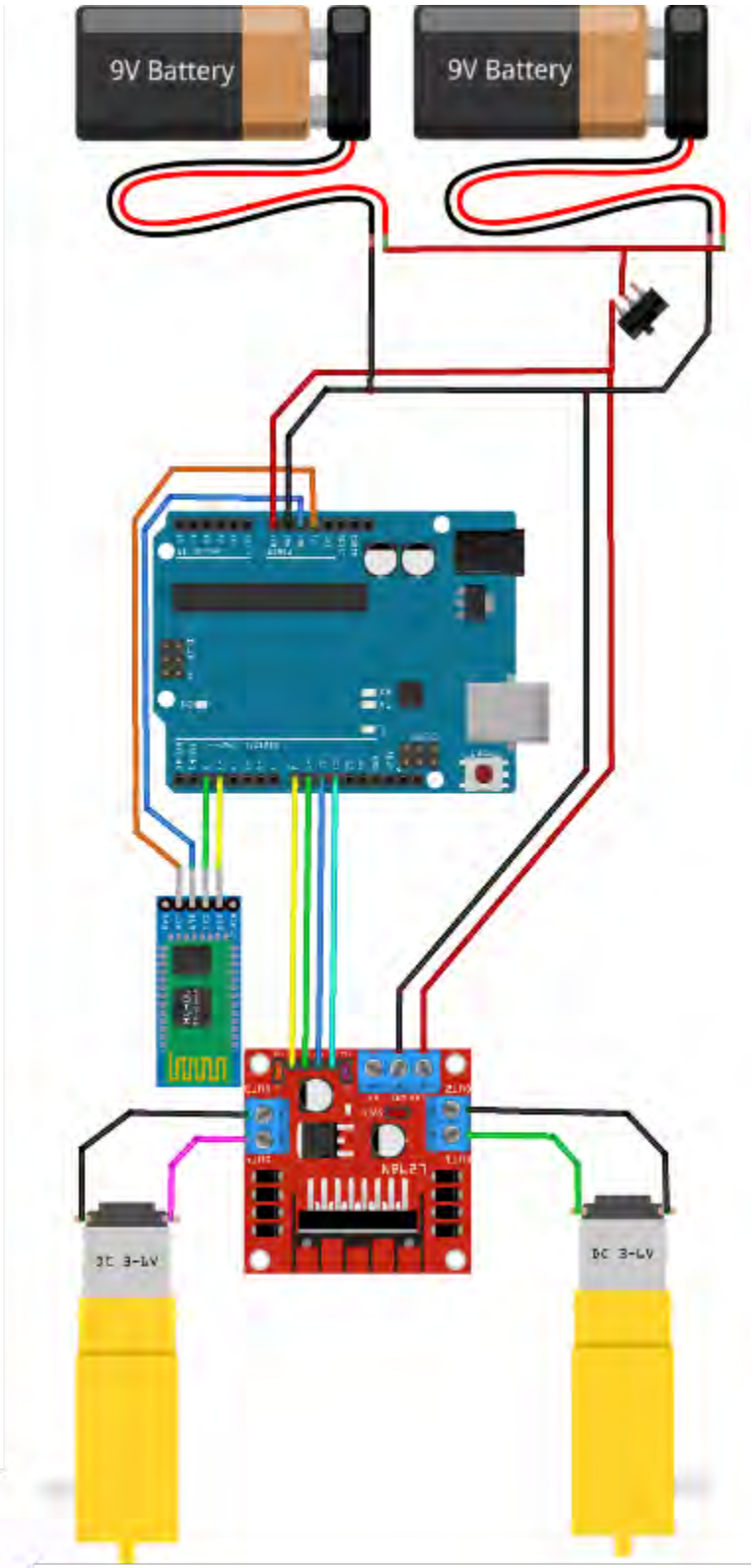
Controlador dual de motores L298N			
Habilitar 1	En. 1	En. 2	Salidas del motor 1
0 V	0 ó 5 V	0 ó 5 V	Motor apagado
5 V	5 V	0 V	Motor gira hacia adelante
5 V	0 V	5 V	Motor gira hacia atrás
5 V	0 V	0 V	Motor frena
5 V	5 V	5 V	Motor frena
Habilitar 2	En. 3	En. 4	Salidas del motor 2
0 V	0 ó 5 V	0 ó 5 V	Motor apagado
5 V	5 V	0 V	Motor gira hacia adelante
5 V	0 V	5 V	Motor gira hacia atrás
5 V	0 V	0 V	Motor frena
5 V	5 V	5 V	Motor frena

Cargamos en nuestro microcontrolador el siguiente código, te invitamos a analizarlo con lo visto durante este curso:

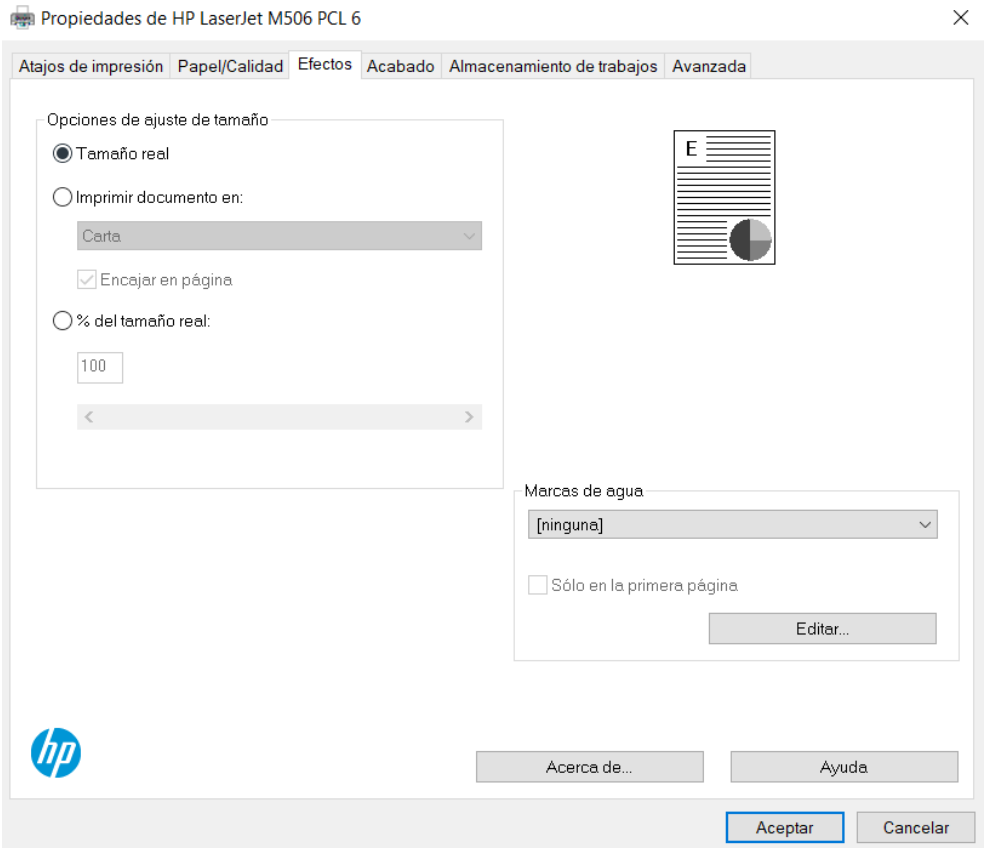
```
#include <SoftwareSerial.h>

SoftwareSerial HC06(2, 3);    // declaramos los pines de recepción y emisión (Rx=2,Tx=3)

char recibido;
int contador;

void setup(){
  HC06.begin(9600);        // Configurar velocidad de comunicación con el módulo bluetooth.
  pinMode(11,OUTPUT);      //motor izquierdo adelante.
  pinMode(10,OUTPUT);      //motor izquierdo atrás.
  pinMode(9,OUTPUT);       //motor derecho adelante.
  pinMode(8,OUTPUT);       //motor derecho atrás.
}

void loop(){

  //Leer bluetooth////////////////////////////////////////////////////////////////////////////
  if (HC06.available()){     //Si hay información en el buffer.
    recibido=HC06.read();   //Leer el char.

    //Mover carro/////////////////////////////////////////////////////////////////////////////

    if(recibido=='w'){        //mover motor izquierdo y derecho hacia adelante
     digitalWrite(8,0);
     digitalWrite(10,0);
     digitalWrite(9,1);
     digitalWrite(11,1);
    }

    else if(recibido=='s'){    //mover motor izquierdo y derecho hacia atrás
     digitalWrite(9,0);
     digitalWrite(11,0);
     digitalWrite(8,1);
     digitalWrite(10,1);
    }

    else if(recibido=='d'){    //mover motor izquierdo hacia adelante y derecho hacia atrás
     digitalWrite(8,0);
     digitalWrite(11,0);
     digitalWrite(9,1);
     digitalWrite(10,1);
    }
```

```
      else if(recibido=='a'){    //mover motor izquierdo hacia atrás y derecho hacia adelante
        digitalWrite(9,0);
        digitalWrite(10,0);
        digitalWrite(8,1);
        digitalWrite(11,1);
       }
      contador=0;
  }
  else{
   if (contador==5){
      digitalWrite(8,0);
      digitalWrite(9,0);
      digitalWrite(10,0);
      digitalWrite(11,0);
      contador=0;
   }
   else{
     contador++;
     delay(10);
   }
  }
}
```

Entonces armamos el siguiente circuito para nuestro vehículo.



Y después, imprimimos el dibuje del siguiente link: (https://drive.google.com/file/d/1CoEpG1UVfeXizLJOMXcIeMjvzv_bBRlZ/view?usp=sharing) Configurando nuestra impresora para que imprima a escala 1:1, para hacer esto, al presionar imprimir, en el menú de configuración vendrá esta opción similar a esta:



AlfaDigital
Dirección de Alfabetización Digital

Recrea
Educación para refundar 2040

Educación

Jalisco
GOBIERNO DEL ESTADO

AlfaOnline