CS13001-CS13051 y CS13011-CS13061
Tecnologías de la información I

Estimado colega:

En la actualidad, las tecnologías de la información fungen una parte primordial en el desarrollo de todas las naciones, ya no simplemente de aquellas llamadas de primer mundo. Considero que es responsabilidad de cada uno de los profesionales de la tecnología y, sobre todo, de los educadores, prepararnos de la mejor manera para los vertiginosos cambios que suceden a nuestro alrededor.

Probablemente, al igual que yo, viviste el enorme crecimiento, tanto en computación como en telecomunicaciones, que se ha visto en los últimos años; pasamos de enormes ordenadores hace menos de cincuenta años, al uso generalizado del Smartphone en la actualidad. Por lo cual, es nuestra responsabilidad, como educadores en el ámbito de la informática, preparar a nuestros alumnos para que no solo sean espectadores en el cambiante panorama al que nos enfrentamos, sino que ellos mismos sean motores de cambio para el ámbito tecnológico.

El curso Tecnologías de la información I se propone generar el uso del pensamiento lógico en la programación de sistemas embebidos en el alumno; como bien sabemos, actualmente la capacidad de programar dejó de ser una herramienta específica, para pasar a ser prácticamente una habilidad necesaria para cualquier campo.

Por este motivo, la competencia del curso es la siguiente:

> Utiliza lenguajes de programación y uso de lógica computacional para la solución de problemas.

Esta competencia se centra en que el alumno desarrolle la lógica computacional y la aplique en el diseño de robots de competencia.

Es por eso que este curso, más que enfocarnos en el uso del IDE que utilizaremos, que se llama Arduino, nos enfocamos en que el alumno obtenga las herramientas cognitivas necesarias para que, si ese es su deseo, pueda continuar e implementar lo aquí aprendido, en cualquier otra plataforma que necesite.

Seleccionamos Arduino porque es una plataforma *open source*, de gran uso en la actualidad y escalable. Adicionalmente, estaremos utilizando Raptor, este software de diseño de diagramas de flujo permitirá al alumno desarrollar su pensamiento lógico, cabe mencionar que Raptor es gratuito.

La propuesta que se presenta a continuación busca generar, motivar y sostener, en los estudiantes, hambre de crecimiento en el campo informático, empezando desde el núcleo más básico: la programación de periféricos.

> La placa Arduino que vamos a usar es la Arduino Mega 2560 R3, puedes encontrarla con o sin su cable USB.

Destacamos que el formato elegido para la realización de este curso se focaliza en contenidos clave, en relación con herramientas actuales de programación y diseño de prototipos robóticos; estas mismas herramientas se ven actualizadas constantemente, por lo que sugerimos un constante análisis de las competencias y necesidades en boga.

En vez de contar con un libro específico, decidimos incluir recursos abiertos y gratuitos para el alumno, tales como tutoriales de Arduino y Raptor, que podrás encontrar en la sección de Información General, Bibliografía. A continuación, te presentamos estos tutoriales:

"Los siguientes recursos son materiales de apoyo, adicionales al contenido del curso; al entrar a cada sitio deberás considerar los términos y condiciones que rigen al mismo".

- Para conocer más sobre el uso y manejo del Arduino, te recomendamos leer: Arduino. (2018). *Education.* Recuperado de https://www.arduino.cc/en/Main/Education
- Para conocer más sobre Raptor para principiantes, te recomendamos leer: Programming Steps. (s.f.). *Raptor flowchart quick start for beginners*. Recuperado de https://programming

Este curso es totalmente práctico, usa las técnicas didácticas de aula invertida y aprender haciendo, diferenciadores de la Universidad Tecmilenio; además, lo complementa con la realización de retos, los cuales permitirán que el alumno incremente su sentimiento de Logro, elemento del Ecosistema de Bienestar y Felicidad.

> Adicionalmente, el curso no presenta explicación de los temas, solo recursos y la guía de lo que debe abordarse en cada uno, para solucionar el reto. Por ser aula invertida, invitamos a los profesores a crear sus propios recursos para compartir con los alumnos.

El contenido está constituido por 13 retos, ya que en los temas 14 y 15 el alumno se dedicará a realizar el reto final. La realización de este reto es obligatoria, ya que este curso no cuenta con evaluación final tradicional (examen escrito), sin embargo, la evaluación final se pondera con el reto final. Por tal motivo, es importante que expliques a tus alumnos que, si no presentan el reto final, por reglamento Académico están reprobados, ya que el reto se considera el examen.

Finalmente, la materia cuenta con una Feria Tecnológica, en la cual el alumno sorteará un laberinto (realizado por ti y los demás profesores de la materia), dicha feria permite poner en práctica todos los conocimientos adquiridos durante el curso y presentarlos ante toda la comunidad Tecmilenio.

Esperamos que la participación en esta experiencia y el intercambio con otros profesores, contribuyan a enriquecer el trabajo de enseñanza en su aula.

Episodio 1. Introducción a la programación de periféricos

En este episodio se muestran los principios básicos de la programación, empezando por la parte más importante, en este caso me refiero a los diagramas de flujo. Es importante que el alumno entienda los componentes y la función de cada uno de ellos, al momento de diseñar un algoritmo.

Adicionalmente, buscaremos que el alumno se familiarice con el uso del software Arduino, el cual será usado durante todo el curso, para programar cada uno de los retos que serán planteados.

En el tema 1 se explican las funciones más básicas del software, así como las funciones y pestañas que muestra la interfaz de usuario. En el tema 2 se explica la importancia del algoritmo y su relación con los diagramas de flujo y la programación de código a partir del mismo. El tema 3 explica los diferenciadores de la programación, enfocada a periféricos y comienzan a utilizarse las funciones de código básico de Arduino, para aplicaciones físicas reales.

En el tema 4 se exploran más a fondo las funciones básicas del mapeo de código y se continúa trabajando con aplicaciones reales, para usos en la vida diaria. Por último, en el tema 5 se aborda la introducción de datos por medio de lecturas externas en nuestra placa, lo cual nos servirá en el resto del curso.

Tema 1. Algoritmos y diagramas de flujo

En este tema abordaremos más a fondo el uso de los diagramas de flujo, por ello es necesario que el alumno comprenda cada uno de los elementos requeridos en un diagrama de flujo, para su correcto funcionamiento.

Asegúrate de enfocarte en el uso de algoritmos, para diseñar cada uno de los diagramas que requerirá a partir de aquí. Recomiendo empezar con algún algoritmo sencillo y cotidiano, quizás alguna receta de cocina o alguna dinámica, donde el alumno escriba los pasos a seguir para un proceso (atarse los zapatos, prepararse para clases, entre otros).

Una vez comprendido el significado y alcance los algoritmos, representar ese proceso con un diagrama de flujo en Raptor. Cuando haya comprendido el proceso a seguir, podremos

continuar con el diseño del diagrama de flujo que utilizará para resolver el segundo reto de este curso, que veremos a continuación.

Reto 1

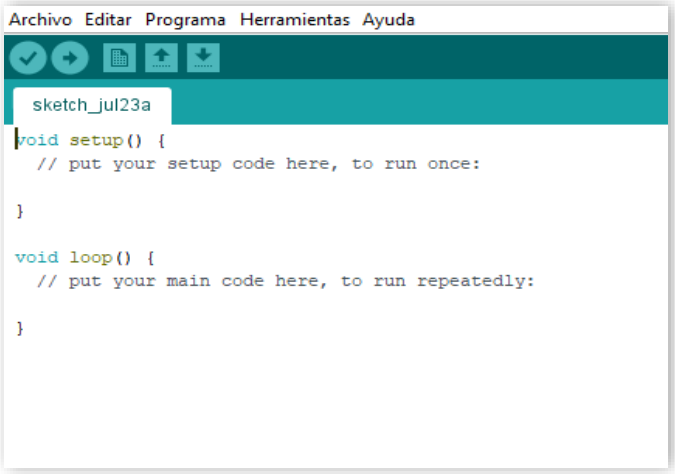
El alumno va a realizar la video lección de MIT Blossoms: "Diagramas de Flujo ¿Pensamiento lógico?", las notas para los ejercicios se encuentran disponibles dentro de la video lección y en las instrucciones del reto.

1. Ingresa a y realiza la video lección "Diagramas de Flujo ¿Pensamiento lógico?".
2. Documenta lo realizado en cada una de las secciones de la video lección. Tu profesor te guiará para hacer esta documentación.
3. Graba un video de lo realizado en la video lección.
4. Redacta unas conclusiones sobre lo aprendido en la video lección.
5. Si queda tiempo, tu profesor realizará una explicación del tema.

Tema 2. Primeros pasos con Arduino

El enfoque principal de este tema es que el alumno se familiarice con la interfaz de usuario de Arduino y, por supuesto, con la placa que estarás utilizando durante todo el curso. A continuación, te presentaré la primera pantalla que verás al abrir el software.

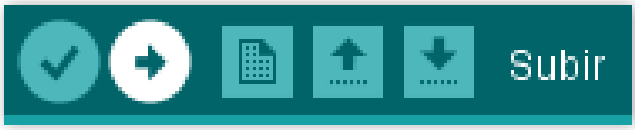
"Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos".



Notarás una pantalla bastante simple en realidad, al ser Arduino una plataforma de código escrito, la interfaz es bastante minimalista, ya que enfoca el uso de los recursos computacionales en el procesamiento, en lugar de aprovecharlos visualmente.

En algunas computadoras el software se descarga con idioma español, recomiendo dejarlo en inglés desde el principio, ya que la programación se enfoca en ese idioma y el alumno podría perder un poco el hilo por culpa de la interfaz.

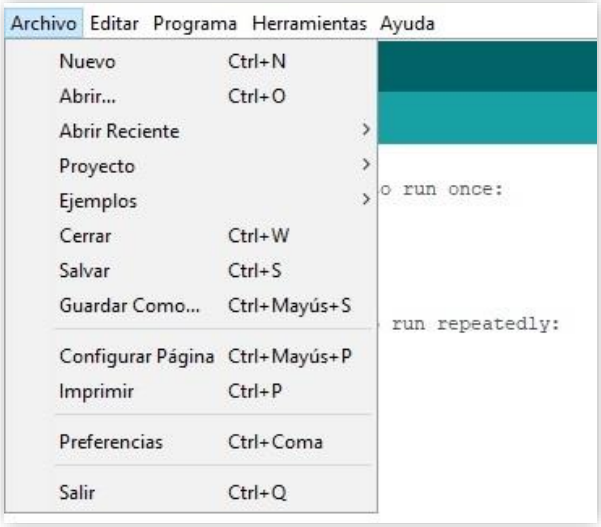
En la siguiente imagen puedes ver los botones principales de la plataforma, de izquierda a derecha son llamados: verificar, subir, nuevo, abrir y salvar.



A continuación, te menciono para qué se utiliza cada uno de ellos.

- Verificar: Se usa para compilar tu código, con este botón nos aseguramos de que no haya errores en el código que acabamos de escribir.
- Subir: Una vez que hayamos verificado que todo esté correcto, usaremos este botón para cargar nuestro programa en la placa Arduino.
- Nuevo: Sirve para empezar un nuevo programa.
- Abrir: Abre algún programa que hayamos escrito con anterioridad.
- Guardar: Este botón nos servirá para que no perdamos el código que escribimos; recuerda que no basta con cargarlo en la placa, ya que necesitamos tenerlo a la mano para alguna revisión futura.

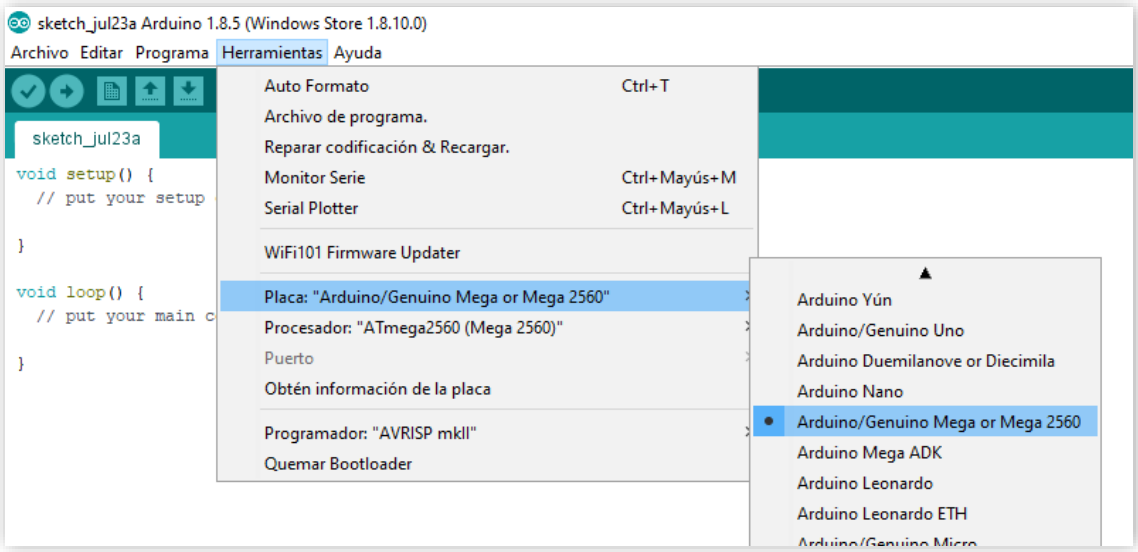
Como notarás, la interfaz de usuario es bastante amigable, por ejemplo, en la pestaña **Archivo** encontrarás las opciones referentes a tu proyecto y también algunas referentes al formato en que se ve el software.



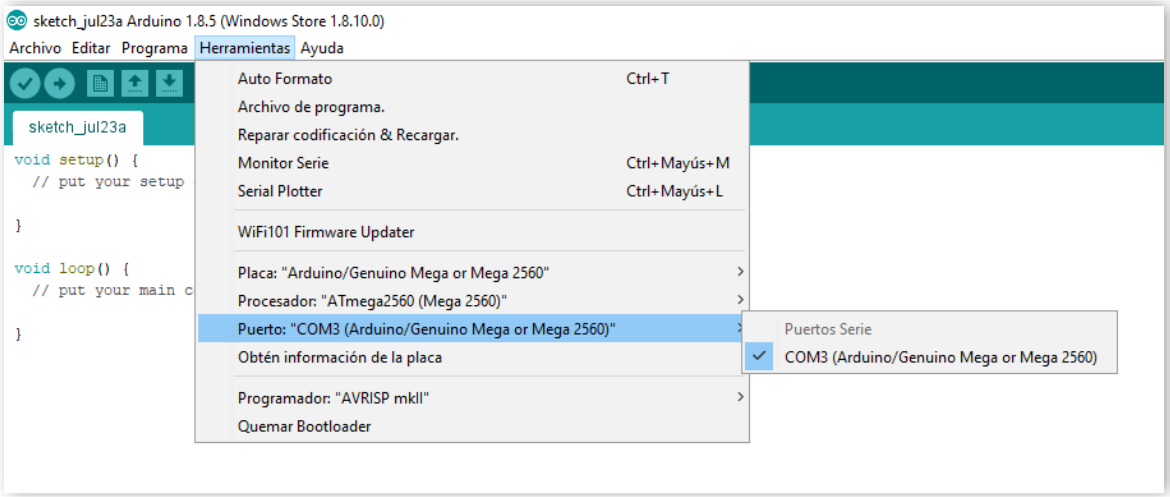
Conectando Arduino a la PC

Lo primero que tenemos que hacer es conectar nuestro Arduino a la PC, para ello utilizaremos un cable USB AB, algunas veces conocido como cable de impresora; una vez que lo tengamos conectado, configuramos el puerto.

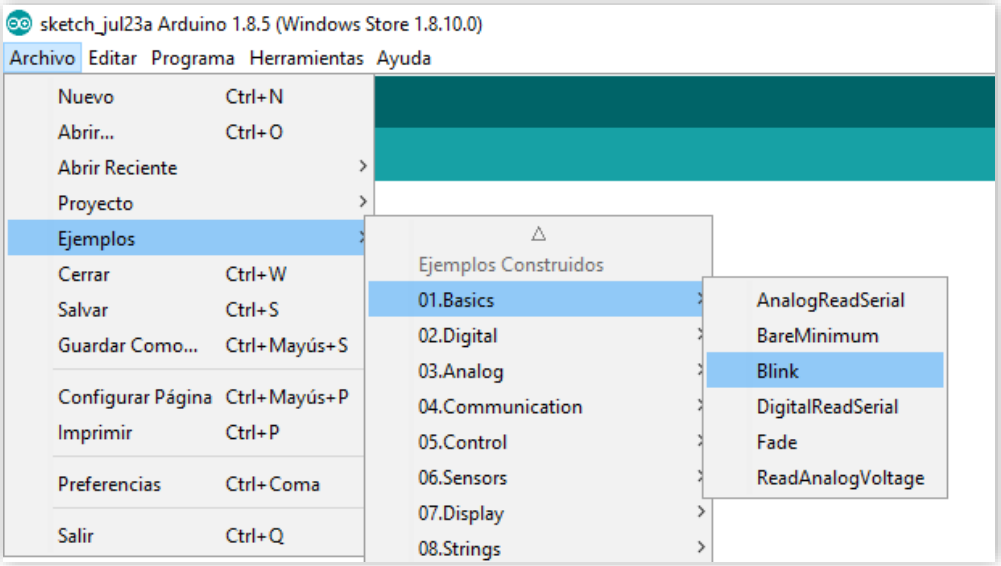
Después, iremos a la pestaña **Herramientas**, elegimos la opción **Placa** y después elegiremos la placa que usaremos, en este caso será **Arduino/genuino Mega or Mega 2560**. Con esto, nuestro IDE sabrá qué versión utilizas; ahora, configuramos el puerto de salida.



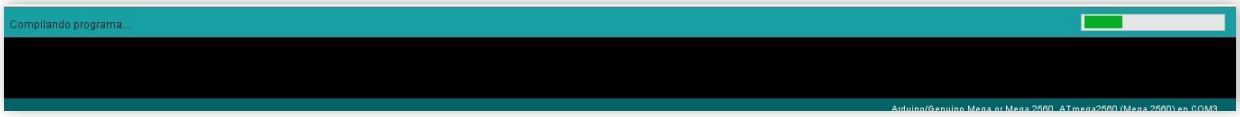
Ya que conectamos el Arduino, iremos a la sección **Puerto** en la misma pestaña y elegiremos el puerto al que conectamos la placa, recuerda que el número de puerto cambiará para cada alumno.



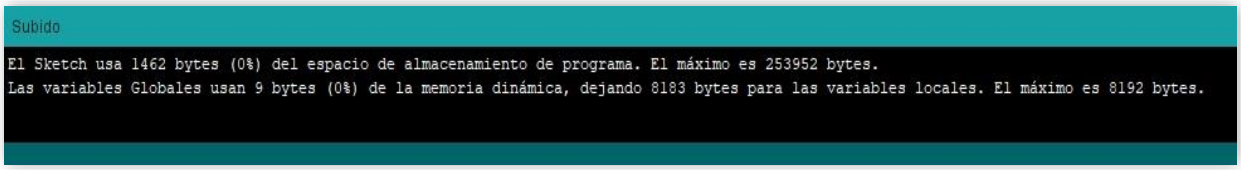
Como primer acercamiento, recomiendo que el alumno tenga la posibilidad de trabajar abriendo alguno de los códigos de ejemplo que vienen en el software. Para ello, iremos a la pestaña **Archivo**, después a la sección **Ejemplos**, seguidamente a la opción **Basics** y, por último, haremos clic en el código llamado **Blink;** en la siguiente imagen se muestra el proceso a seguir.



Este proceso abrirá el primer código que cargaremos a nuestra placa, para probarla. En la nueva pantalla que se abra, hacemos clic en **subir,** con el Arduino conectado y, después de unos segundos, nuestro código se cargará.



Primero notarás que la parte de abajo cambia, con el mensaje “Compilando programa” y una pequeña barra llenándose, la cual, después de unos segundos, cambiará a la siguiente:



El mensaje cambiará a “Subido” y la barra desaparecerá, adicionalmente, en nuestra placa empezará a parpadear un LED de color naranja, que tiene conectado al pin 13. De esta forma, nos daremos cuenta de que nuestra placa está configurada de manera correcta y funciona adecuadamente.

Reto 2

En este reto realizarás la video lección MIT Blossoms *Diagramas de Flujo ¿Pensamiento lógico?*, a continuación, te comparto el enlace:

"Los siguientes recursos son materiales de apoyo, adicionales al contenido del curso; al entrar a cada sitio deberás considerar los términos y condiciones que rigen al mismo".

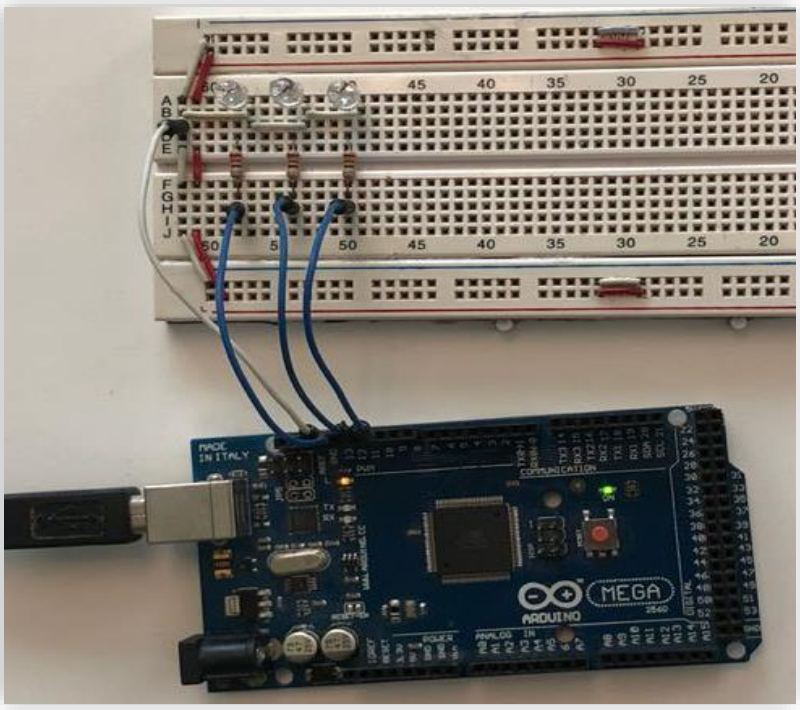
MIT Blossoms. (2018). *Diagramas de Flujo ¿Pensamiento lógico*? Recuperado de https://blossoms.mit.edu/videos/lessons/flow_charts_logical_thinking

Lo importante de este reto es que introduzcas a los alumnos al diseño de diagramas de flujo, lo cual nos servirá para la programación. Si terminas la videolección en una hora, te sugerimos que inicies con el tema y, posteriormente, si te queda tiempo y cuentan con el material, pueden realizar el siguiente reto (opcional), en el que realizarás la programación del encendido y apagado de tres ledes, de manera consecutiva.

Como puedes notar, el reto 1 es bastante sencillo y únicamente habría que modificar el código que cargamos antes en nuestra placa. A continuación, te comparto las instrucciones, el circuito y el código con el que vas a trabajar en el reto.

Instrucciones para el reto sugerido del tema 2

1. En este reto realizarás la programación del encendido y apagado de tres ledes, de manera consecutiva.
    a. En la protoboard ensambla los tres ledes, junto con su resistencia. Utiliza la siguiente imagen para guiarte a la hora de ensamblar tu circuito.
    Al momento de ensamblar, recuerda que la patita más larga de un led siempre es el positivo; en el caso de que tus ledes tengan ambas patitas del mismo tamaño, puedes revisar la polaridad mirando el encapsulado, la que es más pequeña es el lado positivo y se conectará a la salida de nuestro Arduino.



b. Escribe un código en el IDE de Arduino, el cual realice el encendido de los ledes de manera consecutiva. Una vez que se encuentren todos encendidos, deberán apagarse al mismo tiempo y la secuencia volverá a iniciarse de manera infinita. A continuación, encontrarás, como ejemplo, unas cuantas líneas de código utilizadas para encender y apagar un led.
c. La siguiente imagen muestra un ejemplo de código, donde se lleva a cabo el encendido y apagado de un led a razón de un segundo. Puedes usarlo como base para escribir el tuyo.

"Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos".

```
1  void setup() {
2    int LED=13;
3    pinMode(13, OUTPUT);
4  }
5
6  void loop() {
7    digitalWrite(13, HIGH);
8    delay(1000);
9    digitalWrite(13, LOW);
10   delay(1000);
11 }
12
```

2. Asegúrate de comentar tu programa, para evitar código basura a la hora de escribirlo.
3. Documenta, en un video, cada proceso de lo realizado en tu reto, ya que éste será parte del entregable.

Criterios de evaluación de la actividad:

1. Funcionamiento del código.
2. Presentación del código y el circuito armado.
3. Presenta un video que contiene las siguientes secciones:
    a. Explica y demuestra el funcionamiento del código.
    b. Explica, arma y comprueba la correcta configuración de los tres ledes.

Tema 3. Programación de periféricos

Hasta el momento del tema anterior, se han visto comandos básicos y secuencias de código sencillas y sin propósitos claros. A partir de aquí, se empezarán a ver estructuras un poco más complejas y es importante que el alumno tenga entendida lo más posible la interfaz de usuario que maneja Arduino, además de las instrucciones básicas como el void setup y el void loop, ya que éstas dos deberán usarse, sin importar el código que escribamos.

Cuando empezamos con periféricos, como el nombre lo indica, es cualquier dispositivo externo al Arduino y es el enfoque que debemos dar a partir de aquí. Se debe entender la estructura de nuestros proyectos como modular, teniendo en primera instancia a nuestra placa Arduino y cada uno de los elementos que añadimos (ledes, potenciómetros, motores, pantallas, botones) como complementos.

Es importante que, como guía que seremos de nuestros alumnos, tengamos un amplio entendimiento de los dispositivos de entrada y salida, como elementos esenciales de cualquier sistema de control. Podemos utilizar los dispositivos de entrada y salida como analogía para cada uno de los elementos utilizados en los retos que realizaremos.

Recomiendo dar un pequeño recordatorio de las funciones y los tipos de dispositivos de entrada y salida como introducción, para el correcto entendimiento de los procesos a seguir. Adicionalmente, debemos guiar a nuestros alumnos para que enfoquen los bloques de su programación a dichos dispositivos y puedan lograr un código más fluido, en cuanto a funciones y comandos.

Reto 3

En este reto realizarás la programación de un semáforo, justo cómo funcionan los semáforos en las calles.

Al igual que los retos anteriores, en éste no haremos más que secuencias de encendido, solo que le daremos un propósito a esta secuencia, su uso en un semáforo.

A continuación, se mostrará parte del código ideal que debería generarse para su revisión.

```
*************/
/*  Semáforo */
/*************/

//Inicia la secuencia void en la que des de alta los ledes.
void setup() {
  pinMode(13,OUTPUT); //Led verde 1
  pinMode(14,OUTPUT); //Led amarillo 1
  pinMode(15,OUTPUT); //Led rojo 1
  pinMode(16,OUTPUT); //Led verde 2
  pinMode(17,OUTPUT); //Led amarillo 2
  pinMode(18,OUTPUT); //Led rojo 2
}
```

```
void loop() {
 digitalWrite(13,HIGH); //Encendemos el led verde 1 y el rojo 2.
 digitalWrite(18,HIGH);
 delay(5000);          //Esperamos cinco segundos.
 digitalWrite(13,LOW);  //Aquí empieza la secuencia de parpadeo.
 delay(200);           //Esperamos .2 segundos.
 digitalWrite(13,HIGH);  //Apagamos el led verde 1.
 delay(200);
 //Replica esta secuencia para el resto de los ledes.
}
```

Como podrás notar, aunque la secuencia en este código es más compleja que la del reto anterior, el código es más corto. Eso es porque las instrucciones de apagado y encendido son menores a las utilizadas en el anterior. El circuito del reto anterior servirá también para éste, solo hay que reemplazar los ledes por los de colores verde, amarillo y rojo, además de ordenarlos alternadamente, respetando la configuración del semáforo.

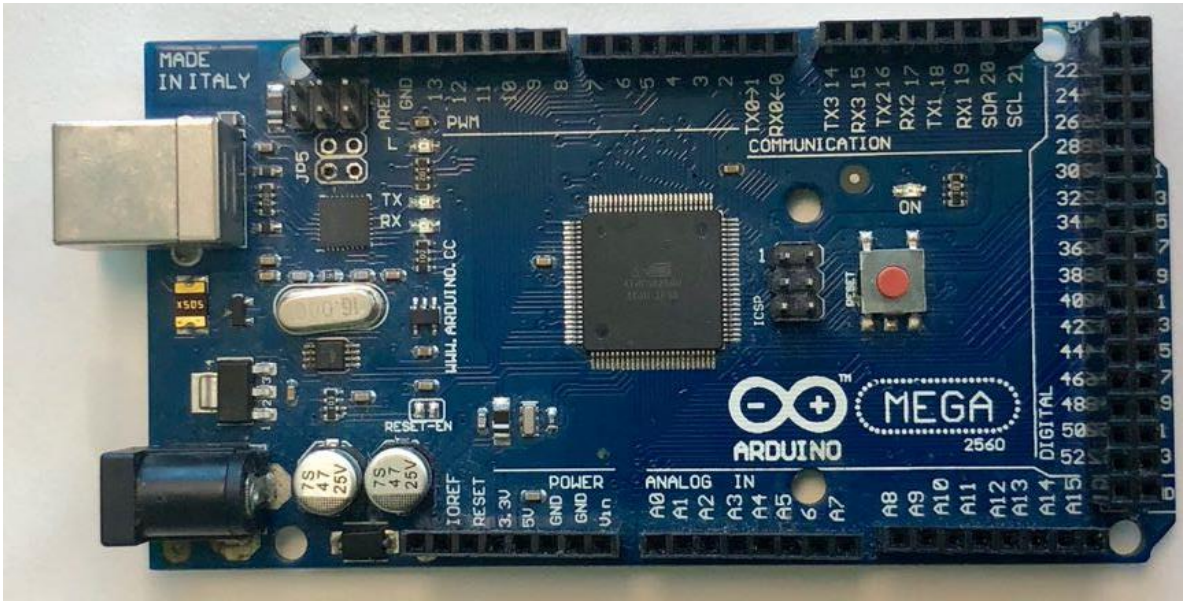
Tema 4. Manejo de datos: escritura de puertos

Hasta ahora, hemos utilizado puertos de la placa Arduino como salidas y, sin darnos cuenta, hemos escrito en esos puertos; esa es la idea principal del curso, que el alumno comprenda de manera sencilla, empírica e intuitiva las funciones de nuestra placa. En este tema se debe comprender el concepto de puerto y aprender a usarlos para escribir datos en ellos; que, como ya dije anteriormente, lo hemos hecho desde el momento en que empezamos a utilizar la placa.

En la placa Arduino, cada uno de los pines marcados con un solo número es un puerto, estos nos sirven tanto para introducir datos, como para extraerlos; cuando el Arduino es quien envía los datos, decimos que estamos escribiendo uno de los puertos. Para eso, utilizamos el comando OUTPUT, así es, el comando que hemos utilizado todo este tiempo.

Ahora, debemos guiar al alumno para que no se deje llevar por el hecho de que ya utilizó este comando antes, la escritura de puertos depende mucho del dato que queramos escribir.

En el reto específico de este tema se usarán datos binarios, para convertirlos en datos visuales que se interpretan como en un reloj.



En nuestra placa, como podemos ver, tenemos pines de alimentación (Vin, GND, 5V, 3.3V), de control (reset), de comunicación (TX, RX, SDA, SCL), digitales (los que solo tienen números), analógicos (los que tienen el formato A#).

Son los pines digitales y analógicos los que sirven como puertos de entrada y salida de datos o, mejor dicho, para lectura y escritura de datos. En este tema haremos especial énfasis en los digitales, aunque la forma de programarse es igual para ambos.

Debemos hacer especial énfasis en el comando utilizado y también en el hecho de que un puerto declarado como salida solo puede usarse para eso, a menos que en el mismo código cambiemos su configuración a entrada.

Reto 4

En este reto realizarás la programación de un contador, utilizando un display de ledes de siete segmentos.

Primeramente, para resolver este reto, debemos comprender la configuración del display de siete segmentos, el cual no es otra cosa que ocho ledes conectados, de manera que se pueda representar, con ellos, los números del 0 al 9.

A continuación, se muestra una tabla con los pulsos binarios requeridos para hacer funcionar el display.

Cátodo común

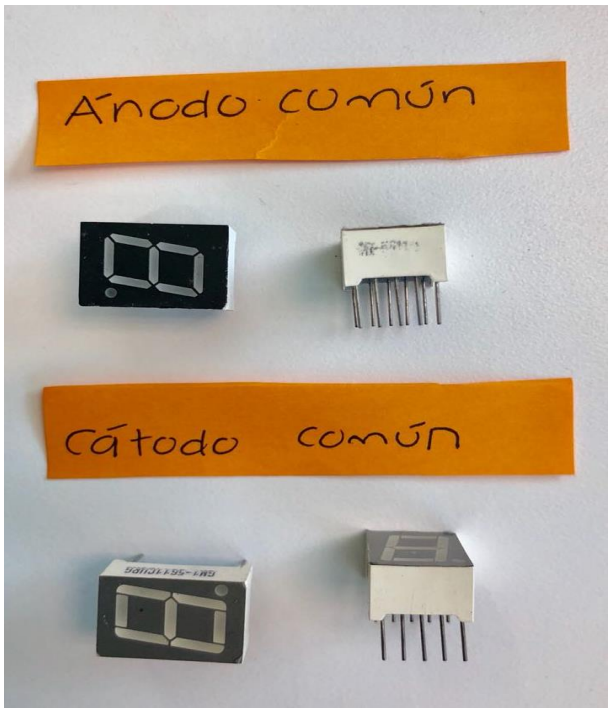
Número	A	B	C	D	E	F	G
0	1	1	1	1	1	1	0

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

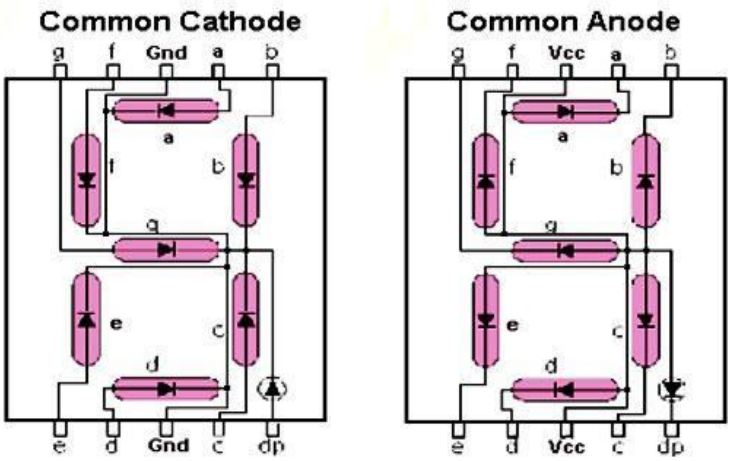
Ánodo común

Número	A	B	C	D	E	F	G
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0

La siguiente imagen muestra la forma física del display de siete segmentos.



La siguiente imagen muestra la configuración electrónica del display de siete segmentos.



Tema 5. Manejo de datos: lectura de puertos

Este tema y el anterior van muy de la mano, ambos se enfocan en el manejo de datos, pero, como podrás darte cuenta, el anterior lo hace en la escritura de puertos y éste en la lectura.

Leer un puerto básicamente es obtener un dato del exterior, por medio de un dispositivo externo; este dispositivo puede ser un sensor, un botón, entre otros.

Lo más importante en este tema es que el alumno comprenda la diferencia entre un puerto declarado como entrada y otro declarado como salida; especialmente, se puede notar a la hora de programar.

En las salidas usamos el OUTPUT y en las entradas el INPUT, éste es un nuevo comando que el alumno deberá comprender a la perfección, antes de que sea posible avanzar. Además del reto, recomiendo pequeñas actividades, como lecturas de algunos botones o incluso ver únicamente la lectura del potenciómetro, antes de escribirla en el led; para ello podemos utilizar el monitor serial, que nos mostrará todos los datos que necesitemos.

Reto 5

En este reto realizarás la programación de la lectura de un puerto analógico del Arduino, junto con la escritura de un puerto digital.

La escritura en el led ya la sabemos utilizar, así que solo es necesario agregar la lectura de un puerto analógico; recuerda que estos son los que tienen la A mayúscula antes del número.

A continuación, te muestro parte del código que utilizarás para este reto:

```
{
// Utilizaremos este nombre para declarar la lectura del potenciómetro, conectado al puerto A0.

const int pot =0;
int brillo;  //Variable para el brillo.

void setup ()

{
 /* Los pines analógicos se declaran como entrada automáticamente. */
 // Declaramos el led como salida.

pinMode (13, OUTPUT);
}

void loop (){
 /*Lee el valor del potenciómetro dividido entre cuatro, ya que
  solo se pueden usar valores entre 0 y 255 en analogWrite.  */

 /*Incluye un analogWrite que reciba dos valores, el pin a usar y la intensidad del voltaje;
  los valores de voltaje van de 0 a 255.*/

}
```

Episodio 2. Manejo de datos e introducción a sensores

En el episodio anterior aprendimos la configuración de nuestro IDE, además de algunos comandos básicos utilizados para escribir y leer distintos puertos de nuestra placa Arduino, aprendimos también a utilizar distintos periféricos básicos.

A partir de este tema es importante que nos aseguremos de que nuestro alumno ya tenga un entendimiento sólido de la interfaz de usuario de Arduino, pues a partir de aquí nos concentraremos en el uso de comandos más avanzados y que podrían confundir al usuario.

También es necesario que los pines o puertos del Arduino estén perfectamente identificados, ya que habrá algunas ocasiones en las que se correrá el riesgo de dañarlos si no se usan correctamente algunos de los dispositivos, así como los voltajes y corrientes de alimentación que puede soportar nuestra placa.

A continuación, anexo las especificaciones técnicas del equipo; para una revisión más extensa, recomiendo revisar la información completa en la página web del desarrollador.

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Este es el momento adecuado para separar nuestra placa, de la computadora; el alumno puede usar una batería de 9 volts para alimentar su placa, lo puede hacer desde el jack de alimentación o desde el pin Vin.

Adicionalmente, puede utilizar un cargador normal de celular y conectar ahí el cable USB AB que utiliza para la programación. La idea principal es que se haga a la idea de que no es necesario mantener conectado el Arduino a la PC, mas que cuando se está cargando el programa.

Este episodio está enfocado en el uso de sensores, tanto analógicos como digitales. Por lo tanto, siempre se usará el comando INPUT, no debemos olvidar que es el usado para obtener lecturas de datos externos.

En el tema 6 haremos la introducción a sensores, el alumno debe comprender la manera en que trabaja cada uno de ellos, para mayor entendimiento se puede utilizar la analogía de los cinco sentidos humanos; de esta manera, se relaciona el dispositivo con algo completamente conocido para él.

En el tema 7 veremos los distintos tipos de retrasos que podemos utilizar en Arduino y cómo cada uno de ellos afecta, de manera distinta, a nuestro código. Durante el tema 8 haremos la diferenciación entre los sensores analógicos y digitales, de manera que el alumno sea capaz de utilizar cualquiera de los dos, dependiendo de las necesidades de su prototipo.

A lo largo del tema 9, nuestros alumnos combinarán los conocimientos adquiridos en el episodio 1, con el avance de este episodio, para hacer un acercamiento a las interfaces humano-máquina y la lógica de control. Por último, en el tema 10 empezará a trabajar con estructuras de control perfectamente definidas, de manera que los códigos que se escriban comenzarán a ganar en simplicidad, ahorrando, de esta manera, los recursos a nuestra disposición.

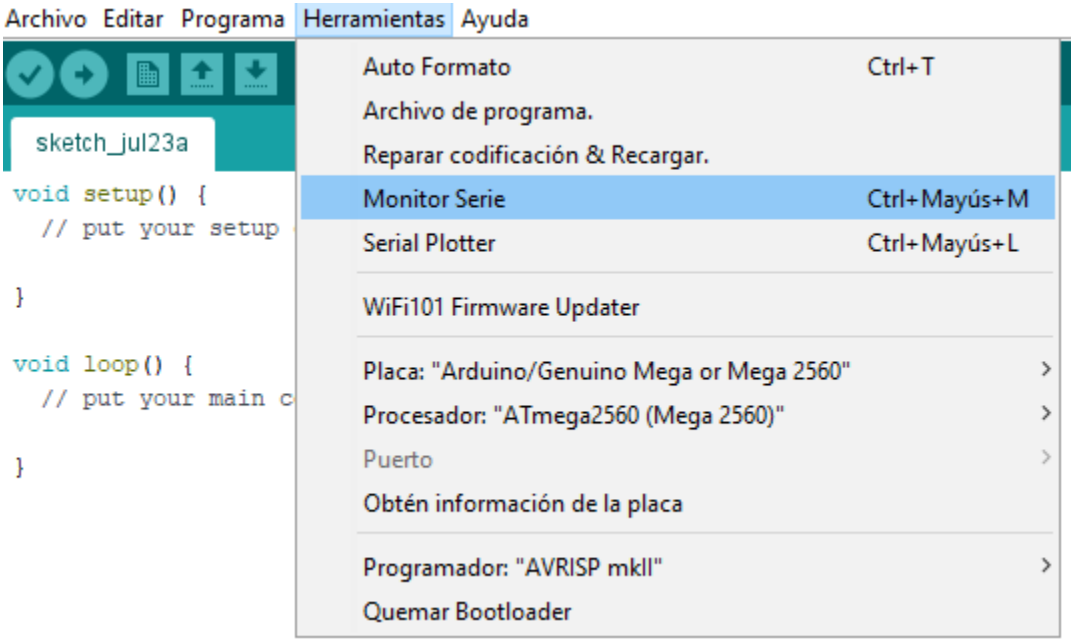
Tema 6. Manejo de sensores

A partir de aquí, nuestros conceptos y comandos serán algo más complejos para nuestros alumnos, por lo que debemos hacer hincapié en mantener la atención de nuestro grupo central y evitar las distracciones externas lo más posible.

Recomiendo, como principal enfoque, el uso de analogías, en las que el alumno pueda relacionar conceptos conocidos por él, con los conceptos nuevos. Un ejemplo podría ser utilizar los sentidos humanos como comparación de los sensores, aquellos dispositivos que nos otorgan datos externos que, como seres vivos, utilizamos para tomar decisiones.

Recuerda utilizar cada una de las funciones del IDE de Arduino, especialmente el puerto serial en este caso, ya que será nuestra primera opción para ver los datos que estaremos leyendo desde los pines de nuestro Arduino. A continuación, te muestro los pasos para tener acceso a esta herramienta.

“Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos”.



En la pestaña de **Herramientas**, seleccionamos la opción **Monitor Serie** o podemos usar directamente el atajo del teclado Ctrl+Mayús+M, como aconseja la misma opción. De esta forma, se abrirá una pantalla adicional, donde podremos observar todos los datos que recabemos desde los puertos del Arduino.

Reto 6

En este reto realizarás la lectura de un sensor, por medio de uno de los puertos del Arduino y lo almacenará, utilizando el puerto serial del Arduino.

En este reto la codificación es bastante sencilla y nuestro código será corto. A continuación, muestro parte del código que puedes utilizar en el caso del sensor de temperatura LM35. Aquí comenzarán a declarar variables de distintos tipos, es necesario que el alumno comprenda los tipos a su disposición y, sobre todo, el momento en que debe usar cada uno de ellas.

```
// Para este reto debemos declarar algunas variables; en este caso, de tipo flotante, ya que son las que requerimos en el caso del sensor de temperatura LM35. Si el alumno decide utilizar algún otro sensor, el tipo de variable a declarar cambiará. //

float tempC; // Variable para almacenar el valor obtenido del sensor (0 a 1023).
int pinLM35 = 0; // Variable del pin de entrada del sensor (A0).

void setup()
 {
  // Configuramos el puerto serial a 9600 bps. Con esto,
 //hacemos posible el uso del monitor serial para leer los datos.
Serial.begin(9600);

}
```

Tema 7. Lectura de puertos con retraso

Cuando utilizamos algún puerto como entrada de datos, nuestro procesador se mantiene leyendo dicho puerto durante todo el tiempo que nuestro código esté funcionando, es por ello que los recursos de memoria que consume aumentan, conforme incrementamos la cantidad de puertos utilizados como entradas de datos.

Una forma de eliminar este inconveniente es con el uso de retrasos, el más utilizados es el delay( ), aunque este retraso nos sirve para reducir el uso de la memoria, en realidad la reducción que obtenemos es mínima, por lo que siempre buscamos alguna opción más eficiente. Dado el caso, podemos utilizar la función millis( ), que básicamente es el mismo retraso, pero, a diferencia del delay( ), no hace gasto de memoria y además es posible interrumpirlo con interrupciones externas, algo que es imposible cuando se usar el delay( ).

Reto 7

En esta primera parte se muestra el encendido y apagado de un led, por medio de la función delay( ); recuerda que si usamos esta función y queremos, por ejemplo, detener nuestro código por medio de un botón, eso no pasará, ya que el Arduino primero realizará la espera, antes de tomar en cuenta lecturas externas.

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);   // Set the LED on.
  delay(1000);              // Wait for a second.
  digitalWrite(13, LOW);    // Set the LED off.
  delay(1000);              // Wait for a second.
}
```

Analicemos el código, la primera parte enciende un led; pasamos a la siguiente línea y nuestra placa no hará más que esperar, a menos que lo reiniciemos o cortemos la alimentación. En este caso no es tanto problema, ya que solo es un segundo, pero, ¿qué pasa cuando son 20 minutos? ¿Una hora? ¿Dos meses? No podríamos detenerlo más que desconectándolo, además del enorme gasto de memoria que llevamos encima.

Veamos una forma distinta, aunque más complicada, de utilizar el delay( ), sin perder el acceso las interrupciones externas.

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);   // Set the LED on.
```

```
  for (int x=0; x < 1000; x++) {  // Wait for one second.
    delay(1);
  }
  digitalWrite(13, LOW);   // set the LED on
  for (int x=0; x < 1000; x++) {  // Wait for one second.
    delay(1);
  }
}
```

Como puedes ver, el código es bastante parecido al anterior, solo que en este caso se utiliza un ciclo de repetición For para hacer de retardo; el delay es únicamente de un milisegundo, pero gracias al ciclo For, se repite mil veces, con lo que tenemos un retraso de un segundo y además obtenemos la posibilidad de utilizar interrupciones externas sin problemas.

Esta parece una buena solución, pero imagina la cantidad de memoria RAM que consume el ciclo de repetición que estamos utilizando.

Ahora, veamos el uso de la función millis( ).

```
int ledState = LOW;          // Variable utilizada para almacenar el estado del led.

// Normalmente utilizamos unsigned long para variables que almacenan tiempo.
// Esto pasa porque el valor se volverá enorme muy rápido, para guardarlo en un int.
unsigned long previousMillis = 0;     // Variable que usaremos para almacenar el tiempo.

const long interval = 1000;         // Intervalo con el que parpadeara nuestro led.

void setup() {
 // Set the digital pin as output:
 pinMode(13, OUTPUT);
}

void loop() {
//Creamos otra variable para usar la función.
//Ésta nos servirá para comparar cuando sea tiempo de encender el led.
 unsigned long currentMillis = millis();

 if (currentMillis - previousMillis >= interval) {
  // Almacenamos la última vez que parpadeó el led.
  previousMillis = currentMillis;

  // Si está apagado, lo encendemos y viceversa.
  if (ledState == LOW)
{
    ledState = HIGH;
  } else {
    ledState = LOW;
  }
```

```
  // Set the LED with the ledState of the variable:
  digitalWrite(13, ledState);
 }
}
```

Aunque el código es casi del mismo tamaño que los anteriores, éste es un poco más complejo; lo principal a resaltar es el uso de las variables específicas, normalmente usamos int o float para guardar datos, en este caso, el valor de nuestra variable crece demasiado rápido, por lo que ambas variables se vuelven obsoletas y tenemos que cambian a una de tipo long.

Tenemos que asegurarnos de que el alumno comprenda las diferencias entre cada una de las formas de generar retrasos, ya que aunque el reto se puede llevar a cabo con cualquiera de ellas, siempre es mejor usar la función millis( ).

Tema 8. Sensores analógicos y digitales

En este tema no hay mucho que resaltar, ya se han manejado sensores en los temas anteriores, así que solamente es necesario recalcar las diferencias entre cada uno de ellos, recalcar las diferencias entre los dos tipos: analógico y digital y asegurarse de que el alumno comprenda la diferencia entre ellos.

Como recomendación, podemos hacer una actividad en la que usemos algunos finales de carrera para comprender el uso de sensores digitales a fondo, ya que los demás temas únicamente han usado sensores analógicos, de esta manera podemos fortalecer el entendimiento de los mismos.

Cabe recordar que debemos utilizar los puertos digitales (los que son solamente enumerados) para los sensores digitales y los puertos analógicos (aquellos que tienen una A antes de la numeración) para los sensores analógicos.

Reto 8

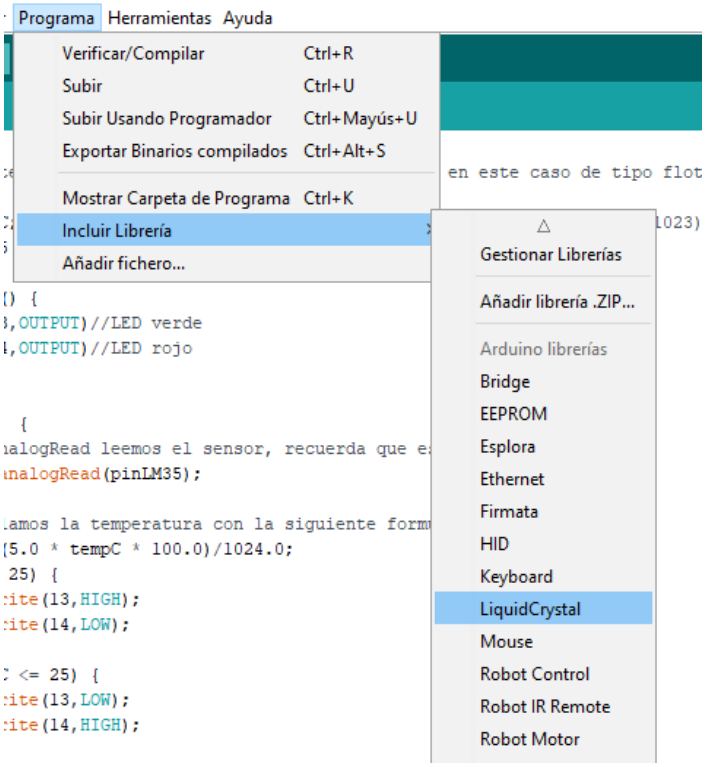
En este reto realizarás la lectura de varios sensores, por medio de los puertos del Arduino y utilizarás una pantalla LCD para mostrar los datos obtenidos, por medio de cada uno de ellos.

Como principal cambio, para este reto tenemos la añadidura de la pantalla LCD y, con ello, la introducción de las librerías para Arduino. Al ser un elemento algo complicado de manejar, preferimos hacerlo de esta forma, ya que el control de la pantalla, por medio de lenguaje máquina, se vuelve bastante complejo.

La mayoría de las librerías ya vienen precargadas en el IDE, en el caso de que no las encuentres ahí, puedes descargarlas desde el sitio oficial de la plataforma. Para añadirlas a tu código solo tienes que seguir los siguientes pasos: Ir a la pestaña de **Programa**, dar clic en la opción de **Incluir Librería** y, después, seleccionar la librería que desees añadir a tu código. En este caso debemos utilizar la librería **Liquid Crystal**, que es la que nos permitirá utilizar la pantalla LCD.

A continuación, encontrarás una imagen de referencia sobre el proceso a seguir.

"Esta pantalla se obtuvo directamente del software que se está explicando en la computadora, para fines educativos".



Tema 9. Lectura de variables y estructuras de control

En este tema utilizaremos lo que conocemos acerca de las librerías en Arduino, junto con escritura y lectura de puertos. Es importante que, para este punto, nuestros alumnos conozcan el uso correcto de los puertos de entrada y salida de Arduino. Además, es necesario que comprendan, al menos de manera básica, la función de las librerías y el manejo de sensores, de acuerdo a su tipo.

Recomiendo relajarlos un poco con un código sencillo, antes de comenzar con el reto, quizás algún juego de luces y botones o incluso un dado electrónico, algo sencillo, que les lleve poco tiempo y no los haga batallar mucho.

Reto 9

En este reto veremos las estructuras de control que, aunque ya utilizamos antes en algún código, no hemos ahondado en los diferentes tipos a los que tenemos acceso en Arduino; a continuación, te mostraré algunos códigos sencillos, donde se usan estas estructuras de control. También necesitamos comprender cada uno de los operadores que podemos utilizar para esas estructuras.

x == y (x es igual a y)
x != y (x es diferente a y)
x < y (x es menor a y)
x > y (x es mayor a y)
x <= y (x es menor o igual a y)
x >= y (x es mayor o igual a y)

Empezamos con la estructura if; esta estructura es sencilla, se ejecutará si y solo si la condición se cumple. Una vez ejecutada, no se repetirá hasta que la condición vuelva a ser verdadera.

```
{
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120){ digitalWrite(LEDpin, HIGH); }

if (x > 120){
 digitalWrite(LEDpin1, HIGH);
 digitalWrite(LEDpin2, HIGH);
}
```

En el código anterior podemos ver el uso de la estructura if, los tres ejemplos hacen lo mismo y las tres son correctas; la sintaxis es la siguiente:

```
if (condición)
{
 Proceso a realizar.
}
```

Esta estructura puede crecer un poco, de acuerdo a lo que necesitamos, añadiendo el Else, que no es otra cosa que el proceso a realizar cuando la condición no se cumple; a continuación, te mostraré un ejemplo de la sintaxis que usamos.

```
if (condición 1)
{
 // Proceso A
}
else if (condición 2)
{
 // Proceso B
}
else
{
//Proceso
}
```

La siguiente es el while:

```
var = 0;
while(var < 200)
{
 // Do something repetitive 200 times.
 var++;
}
```

Esta estructura se mantiene funcionando, siempre y cuando la condición se cumpla; mientras se siga cumpliendo, el proceso se ejecutará en un bucle, sin salir de la estructura. No hay mucho que decir de esta estructura, solo que es usada en ocasiones, cuando queremos que un proceso se lleve a cabo a partir de una variable que no cambie demasiado rápido.

A continuación, veremos el for, esta estructura se repetirá un número finito de veces, dependiendo de cómo lo declaremos nosotros mismo; la sintaxis es bastante sencilla y se muestra a continuación.

```
for (Valor en el que se inicia; condición; incremento).
{
        //Proceso o procesos.
}
```

La siguiente estructura de control es una de las más complejas y, por ende, debe tratarse con cuidado, pues, aunque parece sencilla, su sintaxis podría confundir a nuestros alumnos. Esta estructura es conocida como *switch case* y, a continuación, te mostraré un ejemplo de su funcionamiento.

```
switch (var)
{
 case label1:
  // statements
  break;
 case label2:
  // statements
  break;
 default:
  // statements
}
```

Tema 10. Interfaz humano-máquina: muestreo de datos

Durante este tema las cosas comienzan a ponerse aún más interesantes, empezamos a aprovechar al máximo el número de pines de nuestra placa Arduino.

Lo primero que debemos hacer es lograr que nuestros alumnos entiendan a qué nos referimos cuando hablamos de una interfaz humano-máquina; básicamente es un panel de control, normalmente tienen una pantalla (que en este reto compensaremos con la LCD de 16x2), además tienen una botonera, desde la cual introducimos datos para controlar lo que se muestra en esa pantalla.

Normalmente, es cuando mostramos los datos que estamos obteniendo desde los sensores externos. Como en los temas anteriores, no es necesario que nos detengamos mucho en los conceptos ya vistos, necesitamos enfocar nuestros esfuerzos en la planeación y, sobre todo, el diseño de nuestro diagrama de flujo, ya que en este código es de suma importancia saber lo que hará, para saber cómo programarlo.

Reto 10

En este reto realizarás el prototipo de una interfaz humano-máquina, por medio del Arduino, utilizando diversos sensores, una LCD y algunos botones para control.

Primero, necesitamos declarar todos los botones que utilizaremos como controles de nuestra interfaz. A continuación, te mostraré parte del código que vas utilizar, recuerda incluir la librería LiquidCrystal.h, que nos permite controlar los diplays LCD:

```
#include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

float tempC;
int pinLM35 = 0;
long Distancia;
long tiempo;
int control=0
void setup() {
 lcd.begin(16, 2);
 pinMode(4, OUTPUT);
 pinMode(5, INPUT);
 pinMode(13,INPUT);
 pinMode(14,INPUT);
 pinMode(14,INPUT);
 pinMode(15,INPUT);
 pinMode(16,INPUT);
}
```

Episodio 3. Robótica

Como es bien sabido, el mundo de la robótica avanza cada vez más rápido. El propósito de este episodio es acercar al alumno, de manera práctica, al mundo de la robótica.

Es importante que el alumno, llegado a este punto, sea capaz de generar código sin ayuda y a generarlo a partir de ingeniería inversa, utilizando tanto el código que ha generado antes, como códigos de uso libre en la web.

Durante el tema 11 se mostrará la forma más eficiente de operar motores, utilizando nuestra placa Arduino. En el tema 12 utilizaremos esos mismos motores, pero ahora los controlaremos mediante las lecturas que hagamos con nuestros sensores.

En el tema 13 explicaremos los principios básicos de la inteligencia artificial y la toma de decisiones en nuestros programas; continuaremos con la automatización de procesos en la robótica. Por último, utilizaremos los conocimientos adquiridos para desarrollar procesos conjuntos, para aprovechar al máximo la simulación de procesamiento paralelo.

Tema 11. Manejo de motores

Este tema puede ser algo complicado para ti, por el hecho de que los motores son algo difíciles de controlar, afortunadamente, el uso del driver L298N nos servirá para controlar el motor, sin la necesidad de fuentes externas.

Nuestro driver se controla prácticamente de la misma manera que controlamos cualquier periférico de salida en nuestro Arduino (led, buzzer, entre otros). Es necesario que nuestros alumnos hayan entendido completamente el uso de la declaración de entradas y salidas. A este punto, todo alumno debe tener entendidos estos conceptos o, de lo contrario, le será imposible realizar los retos a partir del número 11.

Reto 11

En este reto realizarás el control de un par de motores conectados a los puertos digitales del Arduino.

A continuación, verás el código de ejemplo como debería quedar, lo único que estamos haciendo es una secuencia de encendido y apagado de pines, justo como en los primeros retos. Para que el código funcione con tu Arduino, debes tener conectadas los componentes, como lo indica el código.

```
int IN3 = 5;
int IN4 = 4;
int IN5 = 6;
int IN6 = 7;
void setup()
{
 pinMode (IN4, OUTPUT);
 pinMode (IN3, OUTPUT);
```

```
  pinMode (IN5, OUTPUT);
  pinMode (IN6, OUTPUT);
}
void loop()
{
  // Motores giran en un sentido.
  digitalWrite (IN4, HIGH);
  digitalWrite (IN3, LOW);
  digitalWrite (IN6, HIGH);
  digitalWrite (IN5, LOW);
  delay(10000);
  digitalWrite (IN4, LOW);
  digitalWrite (IN6, LOW);
  // Uno de los motores gira en sentido inverso.
  digitalWrite (IN3, HIGH);
  digitalWrite (IN6, HIGH);
  delay(5000);
  // Motores no giran.
  digitalWrite (IN3, LOW);
  digitalWrite (IN6, LOW);
  delay(3000);
}
```

Tema 12. Control de motores de acuerdo a lectura de sensores

Acabamos de aprender cómo configurar motores para hacer que avancen y giren en diferentes sentidos.

En este tema combinaremos la lectura de sensores que podemos utilizar para robótica. Durante todo el episodio 2 estuvimos manejando sensores; así que, en este tema, solo debemos asegurarnos de que nuestros alumnos desempolven un poco los conocimientos que ya tienen a la mano, para desarrollar un excelente robot.

Recuerda que puedes utilizar alguna actividad sencilla para que el alumno retome los conocimientos que ya tenía y pueda seguir usándolos.

Reto 12

Para este reto se te pedirá que realices el ensamblado de un carro robótico, capaz de sortear un camino prediseñado por tu profesor.

En el reto anterior utilizamos motores, ahora solo es cuestión de utilizar esos movimientos para añadir los sensores al robot y programar, utilizando estructuras de control, de acuerdo con las lecturas que tengamos. A continuación, te muestro un ejemplo de cómo debería quedar el código, utilizando un sensor ultrasónico.

```
//Declaramos los puertos para los motores.
int IN1= 0;
int IN2= 1;
int IN3= 6;
int IN4= 5;
const int Trigger= 12;
//Declaramos los puertos para el sensor ultrasónico.
const int Echo= 13;

void setup() {
pinMode(IN1,OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);
pinMode(Trigger, OUTPUT);
pinMode(Echo,INPUT);
digitalWrite(Trigger, LOW);
}

void loop() {
long t1;
long d1;
digitalWrite(Trigger,HIGH);
delayMicroseconds(10);
```

```
digitalWrite(Trigger,LOW);

t1= pulseIn(Echo, HIGH);
d1= (t1/2)/29;

if (d1>25) {
digitalWrite(IN1,LOW);
digitalWrite(IN2, HIGH);
digitalWrite(IN3,HIGH);
digitalWrite(IN4, LOW);
}
else {
digitalWrite(IN1,LOW);
digitalWrite(IN2, LOW);
digitalWrite(IN3,HIGH);
digitalWrite(IN4, LOW);
}
}
```

Llegados a este punto y conociendo el uso de las librerías, podríamos reemplazar algunas líneas de código, utilizando las librerías, tanto del uso del driver, como la del sensor ultrasónico.

Tema 13. Toma de decisiones con base en lectura de datos externos

Como parte de la robótica, tenemos la inteligencia artificial, claro que en este caso hablamos de una inteligencia artificial bastante sencilla, lo más avanzado que podríamos hacer es leer datos desde un par de sensores y, a partir de ahí, hacer que nuestro robot decida su siguiente movimiento.

Reto 13

Tengo que hacer especial énfasis en el hecho de que, llegados a este momento, los alumnos deberían entender la manera en qué trabaja la plataforma Arduino, no hay manera de que los retos presentados aquí se puedan resolver sin tener el conocimiento adquirido en los temas anteriores.

Recomiendo reutilizar el código, obviamente sin sobrescribirlos, pues es necesario que tengamos a la mano los códigos utilizados durante cada uno de los retos.

Tema 14. Robótica autónoma

En el tema anterior se aprendió la toma de decisiones por medio de lecturas de sensores, en este tema aprenderemos a utilizar de manera independiente cada una de las lecturas, para llevar a cabo distintas funciones, conforme las lecturas que tengamos.

Hay que hacer especial énfasis en el uso de las estructuras de control aprendidas hasta ahora, ya que eso ayudará a nuestros alumnos en el desarrollo de su reto final; recuerda que ese reto es el que cuenta como examen final.

También debemos desempolvar los datos que conocemos sobre los retardos, recuerda que hay algunos que no nos permiten leer interrupciones externas, tales como el delay u otros que, aunque permiten interrupciones externas, consumen demasiada memoria de proceso, lo que podría alentar todo nuestro proceso.

Llegados a este punto, el alumno debe tener los conocimientos necesarios para llevar a cabo cualquier tipo de programa, incluso algunos que no han sido tratados en este curso.

Tema 15. Diseño de procesos conjuntos

En la industria, como en la vida, el tiempo es un factor sumamente importante a la hora de decidirnos por algún proceso o prototipo. En la parte de la programación de periféricos, ese tiempo se gana llevando a cabo la mayor cantidad de operaciones en el menor tiempo posible.

Aunque la placa Arduino no está preparada para llevar a cabo enormes cantidades de operaciones, como algún otro sistema embebido en el mercado, tenemos a oportunidad de

aprovechar distintas funciones que nos sirven para simular estos procesos de manera paralela, lo cual nos dará una ganancia bastante considerable en cuanto a tiempo y recursos de procesamiento.

En este punto, todo lo que debes hacer es recordar los elementos que utilizamos a lo largo del curso, especialmente aquellos que nos benefician en gasto de recursos.

Por ejemplo, las librerías nos ahorran infinidad de líneas de código, pero también hacen que nuestro procesador almacene, en su memoria, información que a veces necesitamos muy poco o que incluso no necesitamos, lo que provoca que perdamos calidad de almacenamiento.

Otro factor importante es el uso de los retardos, aunque el delay() es bastante sencillo, pero terminamos utilizando toda la memoria de proceso para realizar los nules, además, anulamos nuestras interrupciones externas.

Este tipo de situaciones son las que debemos tomar en cuenta al momento de programar, para evitar la pérdida de tiempo y aprovechar nuestros recursos al máximo.

Asegúrate de que tus alumnos recuerden esto, pues les beneficiará en el diseño de su reto final, por la gran cantidad de sensores que usarán.

Reto final

Escribe, en el IDE de Arduino, un código que realice la lectura de cada uno de los sensores que utilizaremos.

i. El robot se encargará de esquivar obstáculos a su alrededor, dejados aleatoriamente por tu profesor.
ii. El sensor seguidor de línea se encargará de hacer que el robot se mantenga por un camino, trazado con cinta aislante color negro.
iii. Los sensores de final de carrera evitarán que el robot se desvíe demasiado del camino trazado, cuando tenga que esquivar un obstáculo.
iv. Por último, el sensor LM35 se encargará de controlar el encendido del led; cuando la temperatura alrededor del robot llegue a un nivel determinado por tu profesor, el led rojo deberá encenderse, en ese momento tu robot deberá entrar en una rutina de **retorno a casa,** donde retrocederá todo lo que ha avanzado hasta ahora. Nuestro led debe mantenerse encendido, hasta que nuestro prototipo sea reiniciado.

Te deseo mucho éxito en este reto. Recuérdale a tus alumnos que el reto final es la evaluación final, en vez de tener un examen con preguntas y respuestas. Por lo tanto, si algún alumno no lo realiza, reprueba la materia por no presentar la evaluación final.

Feria Tecnológica

La explicación que te presento a continuación la podrás encontrar dentro de tu curso, en la Información General, Feria Tecnológica.

Para la feria, te sugiero que se concentren en el desarrollo o diseño del laberinto. No usen colores negros para el diseño, ya que los sensores son de baja potencia y los robots no podrán identificar este color.

Para la premiación, les sugiero dos categorías: al mejor diseño del robot o más novedoso y al robot que terminó el laberinto en el menor tiempo.

Adicionalmente, les sugiero crear un comité, ya que será más sencillo entre varios profesores presentar un evento de este tipo.

A continuación, se mencionan los elementos a evaluar en la competencia:

a. Presentación del robot.
b. Funcionalidad del código.
c. Desempeño en la prueba.
d. Diseño del robot.

La participación en la feria es obligatoria, ya que se evalúa y se contempla como parte de la calificación final de la materia.