

Meta_Javascript Alpha 1.2

Programación para principiantes

Jose David Cuartas Correa



LOS LIBERTADORES
FUNDACIÓN UNIVERSITARIA

Meta_Javascript Alpha 1.2: Programación para principiantes.

EDICIÓN DE PRUEBA 1.0: JULIO, 2020

© Fundación Universitaria Los Libertadores

© Jose David Cuartas Correa

LOS LIBERTADORES, FUNDACIÓN UNIVERSITARIA

Bogotá D. C., Colombia

Cra 16 No. 63 A - 68 / Tel. 2 54 47 50

www.ulibertadores.edu.co

Juan Manuel Linares Venegas

Presidente del Claustro

Patricia Martínez Barrios

Rectora

Ángela María Merchán Basabe

Vicerrectora Académica

Jose David Cuartas Correa

Diagramación y diseño portada

Licencia Creative Commons by-sa 4.0
<http://creativecommons.org/licenses/by-sa/4.0/deed.es>

INTRODUCCIÓN

Con este texto busco introducir al lector en los aspectos básicos de la primera versión de Meta_Javascript, un lenguaje de meta-programación que desarrollé para el principiante de la programación de aplicaciones web. Está basado en Meta_Processing y fue diseñado para ayudarte a crear código Javascript interactivo. Es una iniciativa personal que es influenciada por mi trabajo como director del Laboratorio Hipermedia¹ (Hitec Lab) en la Fundación Universitaria Los Libertadores (Bogotá, Colombia).

Lo que se busca con esta derivación de Meta_Processing es ir creando un ecosistema de aplicaciones que le permitan al usuario programar diferentes plataformas haciendo uso de un mismo lenguaje de programación. En la actualidad es casi indispensable aprender un lenguaje diferente para programar cada plataforma.

Este entorno de programación se fundamenta en las necesidades que identifiqué durante los años que oriente un curso de programación básica para diseñadores gráficos en la Fundación Universitaria Los Libertadores. También responde a parte de mis

¹Laboratorio Hipermedia <http://hiteclab.libertadores.edu.co/>

intereses investigativos desde cuando estaba realizando mi doctorado en Diseño y Creación, el cual finalicé con la tesis: "Programar el mundo en el contexto de las tecnologías libres y las culturas Hacker-Maker. Caso de estudio: Hitec Lab" (cuartas, 2017).

Meta_Javascript es una de varias iniciativas que he estado liderando en búsqueda de desmitificar las tecnologías y de contribuir en que se pueda cumplir una de las promesas incumplidas del software libre: Permitir que cualquier persona puedan modificar y adaptar el software que usa, para que pueda ajustarlo a sus necesidades particulares y gustos personales. Por ello hacer que el código de programación sea más fácil de escribir y de leer es uno de los propósitos de Meta_Processing y Meta_Javascript.

Introducción Alpha 1.2.

En esta nueva versión de Meta_Javascript se incorporan las siguientes mejoras: Se agrega la pestaña Configuración, permite desplazar las líneas de código con el mouse y se adicionan los idiomas Punjabi, Kannada, Bengali, Tamil, Koreano, Ruso y Aleman. Ahora se puede ejecutar el proyecto en dispositivos móviles (conectados a la misma red) escaneando un código QR. Se agrega soporte para comunicarse con tarjetas Arduino usando Firmata y tarjetas ESP usando la librería IoTControllerAP². No se necesita instalar Node.js ni

² <https://github.com/hiteclab/IoTControllerAP>

Johnny-Five, solo tener abierto Meta_Javascript. Ya se puede usar la instrucción *texto* para mostrar variables. Se agregan las instrucciones *multiplicar*, *dividir* y *fórmula* en la categoría Matemáticas, y la instrucción *código nativo* en la nueva categoría Avanzadas. Se agregan las instrucciones *hipervínculo* y *leerparámetro* en la categoría HTML y la instrucción *para* dentro de la categoría Estructuras. Adicionalmente cada vez que se hace clic en el botón ejecutar, se exporta el código meta del proyecto en tres archivos de texto, los cuales se guardan dentro de la carpeta **meta** del proyecto. El código contenido en cada pestaña se guarda en un archivo de texto con el mismo nombre de la pestaña y con extensión **.meta**. Por último se agrega soporte para los atajos de teclado: **ctrl+c**, **ctrl+x**, **ctrl+v**, **ctrl+z**.

Jose David Cuartas Correa
Bogotá, Colombia
2020

Agradecimientos:

Al apoyo incondicional recibido por parte de la Fundación Universitaria Los Libertadores quienes han creído en cada proyecto que desarrollamos desde el laboratorio Hitec.

Dedicatoria:

A mi esposa Shahzadi y a mis hijas Helen y Megan, quienes llenan de amor y alegría mi existencia.

CONTENIDO

1. META_JAVASCRIPT PRIMEROS PASOS	13
1.1. ¿Cómo abrir Meta_Javascript?	13
1.2. Ventanas que se abren al iniciar Meta_Prosessing	16
1.3. Elementos básicos de la interface	17
1.4. ¿Cómo seleccionar Idiomas?	18
1.5. ¿Cómo abrir un proyecto?	18
1.6. ¿Cómo ejecutar el código?	19
1.7. ¿Cómo generar el código QR del proyecto?	20
1.8. ¿Cómo actualizar el proyecto sin ejecutarlo localmente?	21
1.9. ¿Cómo agregar una línea de código?	21
1.10. ¿Cómo eliminar una línea de código?	22
1.11. ¿Cómo agregar instrucciones?	22
1.12. ¿Cuál es la estructura de archivos y carpetas en Meta_Javascript?	24
1.13. ¿Cómo abrir la carpeta data del proyecto actual?	26
1.14. ¿Cómo crear un nuevo proyecto?	27
1.15. ¿Cómo guardar el proyecto actual?	27
1.16. ¿Cómo exportar el proyecto actual como aplicación?	27
1.17. Ícono de configuración	28
1.18. Funciones: principal, teclado y ratón	30
1.18.1. Principal	30
1.18.2. Ratón	31
1.18.3. Teclado	31
1.18.4. Configuración	32
1.19. Atajos de teclado: ctrl+c, ctrl+x, ctrl+v, ctrl+z	33
2. INSTRUCCIONES BÁSICAS	35
2.1. Documentar el código	36
2.2. Coordenadas de pantalla	37

2.3.	Instrucciones para gráficos en pantalla.....	39
2.3.1.	fondo	39
2.3.2.	línea	40
2.3.3.	rectángulo.....	40
2.3.4.	elipse	41
2.3.5.	triángulo	42
2.3.6.	tamtexto.....	42
2.3.7.	texto	43
2.3.8.	imagen	44
2.3.9.	colorlinea.....	44
2.3.10.	relleno.....	45
2.3.11.	sinrelleno	46
2.3.12.	sinlinea.....	47
2.4.	Instrucciones multimedia	47
2.4.1.	tocanota	47
2.4.2.	sonido	48
2.5.	Instrucciones HTML.....	48
2.5.1.	hipervínculo.....	49
2.5.2.	leerparámetro	49
3.	VARIABLES, CONDICIONES Y CICLOS	52
3.1.	Variables.....	52
3.1.1.	¿Cómo se mira el listado de variables?	53
3.1.2.	¿Cómo se crea una variable?	53
3.1.3.	¿Cómo se elimina una variable?	54
3.1.4.	¿Cómo se inicializa una variable?	55
3.1.5.	¿Cómo asignarle un nuevo valor a una variable?	56
3.1.6.	¿Cómo se le suma un valor a una variable?	57
3.1.7.	¿Cómo se le resta un valor a una variable?	58

3.1.8. ¿Cómo se le asigna un valor aleatorio a una variable?	59
3.2. Condiciones	60
3.1. Ciclos	64
4. USANDO ARDUINO CON META_PROCESSING	67
4.1. Instalación de la librería Firmata en una tarjeta Arduino	67
4.2. Instrucciones Arduino	72
4.2.1. salidadigital.....	72
4.2.2. entradadigital	74
4.2.3. entradaanalógica	77
4.2.4. servo	79
5. USANDO ESP CON META_PROCESING	81
5.1. Instalación de la librería IoTControllerAP en una tarjeta ESP	82
5.2. Instrucciones IoTControllerAP	87
5.2.1. salidadigital.....	87
5.2.2. entradadigital	89
5.2.3. entradaanalógica	90
5.2.4. servo	92
6. EJEMPLOS DE CÓDIGO CON META_JAVASCRIPT	94
6.1. Dibujos abstractos con líneas	94
6.2. Ejemplo básico con el Teclado	95
6.3. Ejemplo básico con el Ratón	96
6.4. Animación	97
6.5. Piano	98
6.5.1. Piano simple.....	99
6.5.2. Piano pantalla de colores	99
6.5.3. Piano colores y notas en pantalla	100
6.6. Mini Juego	103
Fuentes de referencia	105

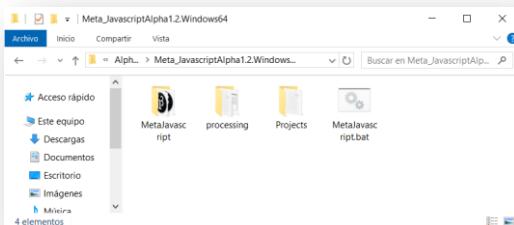
1. META_JAVASCRIPT PRIMEROS PASOS

1.1. ¿Cómo abrir Meta_Javascript?

Meta_Javascript fue desarrollado para que funcionara en los sistemas operativos: Windows, Mac y GNU/Linux. Los pasos para abrir Meta_Javascript varían un poco según el sistema operativo que se use, a continuación se describen los paso para cada sistema:

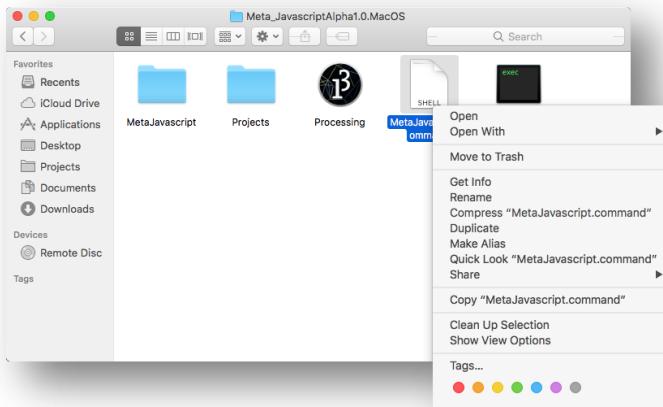
Windows

En Microsoft Windows se debe hacer doble clic en el archivo **MetaJavascript.bat**

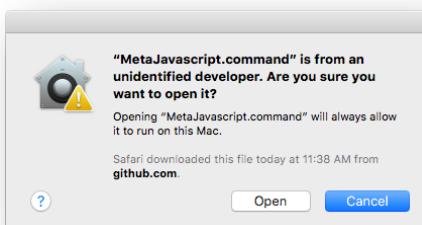


Mac OS

En Mac Os se debe hacer clic con el botón derecho del ratón sobre el archivo **MetaJavascript.command** y seleccionar la opción: **Open** o **Abrir**

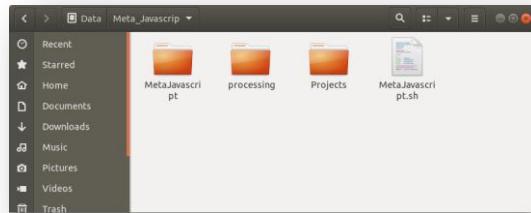


En la ventana que se abre se debe seleccionar la opción: **Open** o **Abrir**

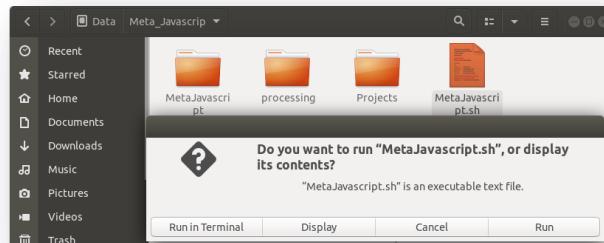


GNU/Linux

En GNU/Linux se debe hacer doble clic en el archivo **MetaJavascript.sh**



En la ventana que se abre se debe seleccionar la opción: **Run** o **Ejecutar** (también puede usar Ejecutar en Terminal si desea ver la ventana terminal de Meta_Javascript)

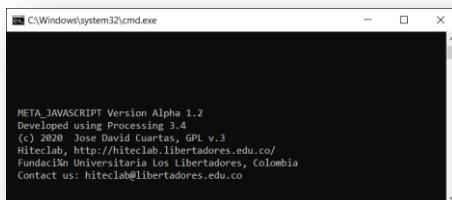


Si el script no abre al hacer doble clic sobre este, se puede ejecutar el siguiente comando en el terminal de Linux, para activar el anterior cuadro de dialogo.

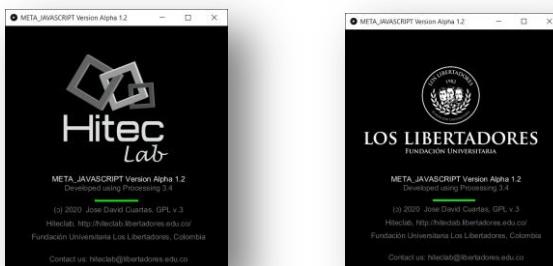
```
gsettings set org.gnome.nautilus.preferences executable-text-activation ask
```

1.2. Ventanas que se abren al iniciar Meta_Prosessing

Una vez se hace ejecuta el archivo Meta_Prosessing sin importar el sistema operativo en el que se esté trabajando primero se abre la ventana terminal donde se pueden ver mensajes que provienen de la ventana principal de Meta_Javascript.

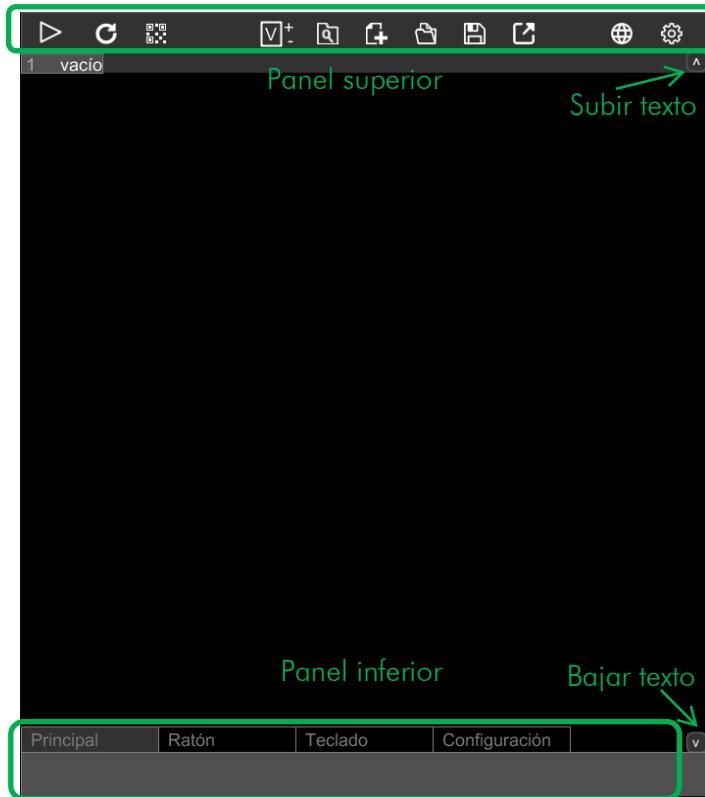


Luego se abre una segunda ventana animada de bienvenida a Meta_Javascript. Al hacer clic sobre esta ventana o al cerrarla se abre la ventana principal de Meta_Javascript.



1.3. Elementos básicos de la interface

En el panel superior están los botones: Ejecutar, Actualizar, Generar QR, Variables, Data, Nuevo, Abrir, Guardar, Exportar, Idiomas y Configuración.



En el panel inferior se encuentran las pestañas: Principal, Ratón, Teclado y Configuración. Y se encuentra la barra de descripción: en donde se muestra los nombres de los botones y el prototipo de las instrucciones.

1.4. ¿Cómo seleccionar Idiomas?

Para cambiar el idioma de **Meta_Javascript** se debe hacer clic en el ícono **idiomas** en la barra superior.



Luego en la ventana que se abre se debe hacer clic en el idioma deseado.

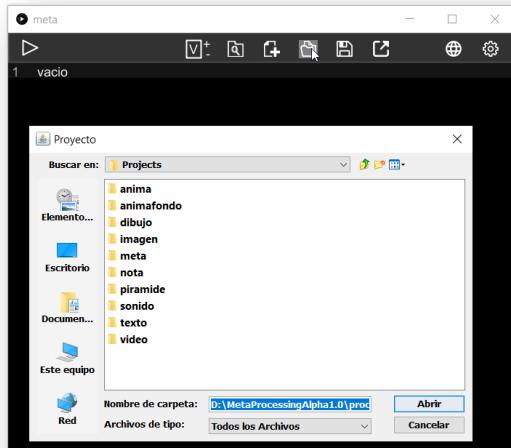


1.5. ¿Cómo abrir un proyecto?

Para abrir un proyecto se debe hacer clic en el ícono **abrir** en la barra superior.



A continuación se abre una nueva ventana en la que se puede seleccionar la carpeta del proyecto que se quiere abrir y se hace clic en botón abrir.



1.6. ¿Cómo ejecutar el código?

Para ejecutar el código se debe hacer clic en el ícono **ejecutar** en la barra superior.



Se espera uno segundos y debe aparecer una nueva ventana en la que se ejecutará el código creado.

1.7. ¿Cómo generar el código QR del proyecto?

Para ejecutar el proyecto en un dispositivo móvil conectado a la misma red, se puede generar un código QR que al escanearse permite direccionar a la URL del servidor local donde se puede ejecutar el proyecto en el navegador web del dispositivo móvil. Para esto se debe hacer clic en el ícono **Generar QR** en la barra superior.



Se espera uno segundos y debe aparecer una nueva ventana en la que se podrá ver el código QR que dirige al proyecto.



Si el equipo no cuenta con un software para escanear códigos QR también se puede usar la URL que aparece debajo del código QR y la cual se deberá ingresar en la barra de direcciones del navegador web del dispositivo móvil.

1.8. ¿Cómo actualizar el proyecto sin ejecutarlo localmente?

Para cuando se haga un cambio en el código y se quiera ver estos cambios en el dispositivo móvil, sin necesidad de ejecutarlo localmente o de generar nuevamente el código QR, se puede actualizar el servidor local del proyecto y luego actualizar el navegador web en el dispositivo móvil. Esto permite que en el dispositivo móvil que ya tiene la URL del proyecto en la barra de direcciones del navegador Web, se pueda ver los cambios efectuados en el código. Para esto solo se necesita hacer clic en el ícono **Actualizar** en la barra superior.



1.9. ¿Cómo agregar una línea de código?

Para agregar una línea de código se debe mover el cursor del ratón hasta que aparezca un círculo verde con el carácter más (+) en su interior.



Aparecerá una nueva línea de código una vez se haga clic en el círculo verde.

```
1 fondo ( px )  
2 sumar ( px , 0.5 )  
3 vacio
```

1.10. ¿Cómo eliminar una línea de código?

Para eliminar una línea de código se debe mover el cursor del ratón hasta que aparezca un círculo rojo con el carácter menos (-) en su interior, y se vea una línea gris que tacha toda la instrucción.

```
1 fondo ( px )  
2 sumar ( px , 0.5 )
```



La línea desaparece una vez se hace clic.

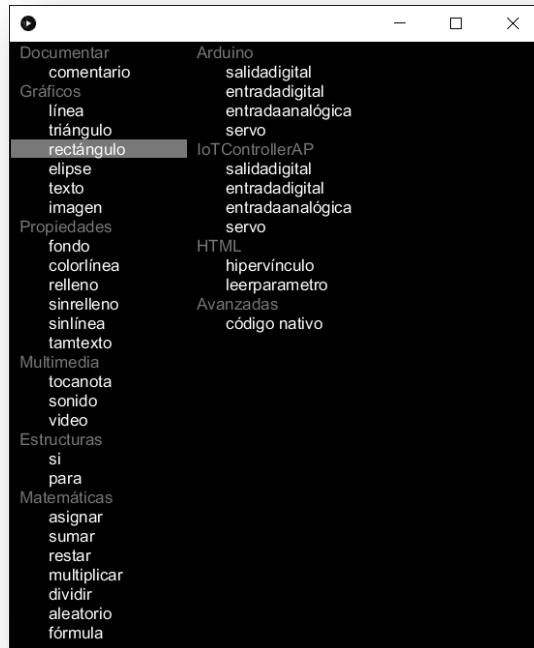
```
1 fondo ( px )
```

1.11. ¿Cómo agregar instrucciones?

Para agregar una instrucción se debe hacer clic en la palabra que dice **vacío**.



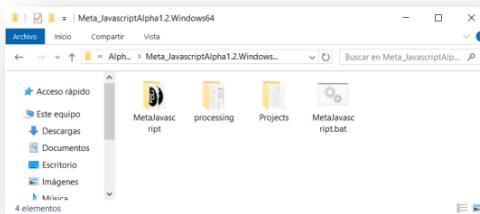
Una vez se hace esto, se abrirá una nueva ventana en donde aparecerán todas las instrucciones disponibles en Meta_Javascript organizadas por categorías así:



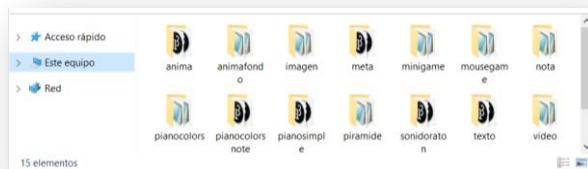
Cuando se ubica el cursor del ratón sobre alguna de estas instrucciones se resalta la instrucción, es este ejemplo se puede ver cómo se resalta la instrucción **rectángulo**. Al hacer clic se abrirá una nueva ventana en la que se podrán ingresar las propiedades de cada instrucción. La descripción de cada una de estas instrucciones se hará en el capítulo 2.

1.12. ¿Cuál es la estructura de archivos y carpetas en Meta_Javascript?

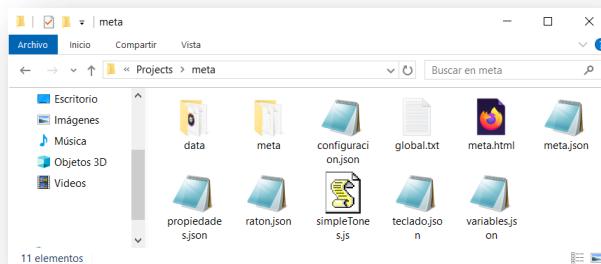
Dentro de la carpeta de **Meta_Javascript** se encuentra el archivo que ejecuta **Meta_Javascript** y tres subcarpetas. La carpeta **MetaJavascript** contiene los archivos que le permiten funcionar. En la carpeta **processing** hay una distribución de este lenguaje que se usa para ejecutar **Meta_Javascript**.



La carpeta **Projects** contiene las carpetas de cada uno de los proyectos de programas escritos usando el entorno de programación **Meta_Javascript**.



En la carpeta de cada proyecto se encuentran algunos archivos **.json**, un archivo **.js** y un archivo **.html**. Los archivos **.json** contienen las instrucciones en lenguaje Meta_Javascript, el archivo **.js** contiene la librería necesaria para reproducir notas usando el navegador de Internet y el archivo **.html** contienen el código JavaScript del proyecto, y es generado cada vez se hace clic en el ícono **ejecutar**.



A partir de esta versión se agrega el archivo **global.txt** en el cual se pueden agregar instrucciones usando el block de notas. Es útil para aquellos proyectos en los que se necesite agregar instrucciones o funciones globales por fuera de las funciones Principal, Ratón, Teclado y Configuración. Es una opción para usuarios avanzados.

Y por último en la carpeta **data** de cada proyecto se guardan los archivos que se usaran en la ejecución del programa como pueden ser imágenes, sonidos y videos.



1.13. ¿Cómo abrir la carpeta data del proyecto actual?

Para abrir la carpeta data del proyecto actual, se debe hacer clic en el ícono **data** en la barra superior.



Una vez se hace clic se abre en otra ventana la carpeta data del proyecto actual



1.14. ¿Cómo crear un nuevo proyecto?

Para crear un nuevo proyecto se debe hacer clic en el ícono **nuevo** en la barra superior.



En la ventana que se abre se debe escribir el nombre que se le quiere dar al nuevo proyecto y hacer clic en el botón aplicar.



1.15. ¿Cómo guardar el proyecto actual?

Para guardar el proyecto actual se debe hacer clic en el ícono **guardar** en la barra superior.



1.16. ¿Cómo exportar el proyecto actual como aplicación?

Para exportar el proyecto actual como aplicación se debe hacer clic en el ícono **exportar** en la barra superior.



La aplicación se guarda en la subcarpeta llamada **web** dentro de la carpeta del proyecto actual.

1.17. Ícono de configuración

Para cambiar las acceder a la opción de configuración se debe hacer clic en el ícono **configuración** en la barra superior.



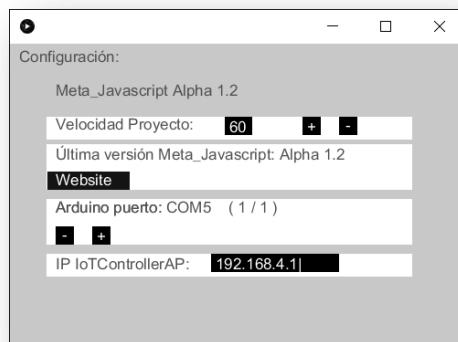
A continuación se abre una nueva ventana en la que se ofrecen cuatro opciones: cambiar la velocidad del proyecto, revisar cuál es última versión de Meta_Processing, seleccionar puerto Arduino y definir la dirección IP para el IoTControllerAP.



La opción **Velocidad** permite cambiar la velocidad del proyecto (una medida aproximada de cuadros por segundo, pero todavía no es del todo exacta).

La opción **Nuevo Meta_Javascript** permite revisar última versión de Meta_Javascript publicada en internet. La información que aparece después de los dos puntos es la versión disponible para descargar. Si se desea descargar esa nueva versión se puede hacer clic en el botón **Website** que apunta a al sitio web oficial de descarga de Meta_Javascript.

La opción **Arduino Puerto**, muestra los puertos de los dispositivos conectados a la computadora. En caso de conectarse una tarjeta Arduino por el puerto USB, se podrá seleccionar el puerto de conexión haciendo clic en los botones – o +. (Para más información ver Cap. 4)



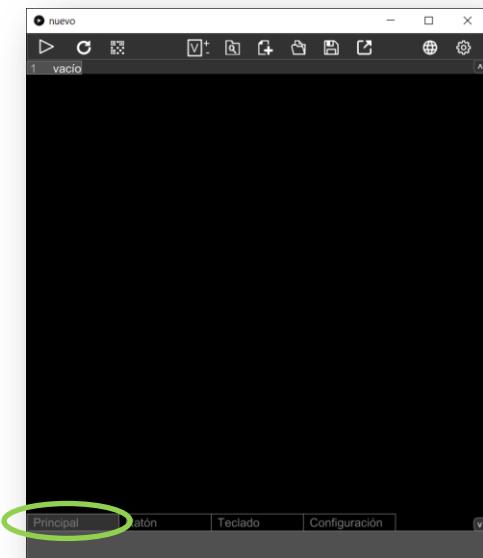
La opción **IP IoTControllerAP** permite cambiar la dirección IP de la tarjeta ESP corriendo IoTControllerAP.

1.18. **Funciones: principal, teclado y ratón**

Para escribir el código en Meta_Javascript, se pueden usar tres funciones: Principal, Ratón y Teclado. Cada una de estas funciones se selecciona haciendo clic en su pestaña correspondiente.

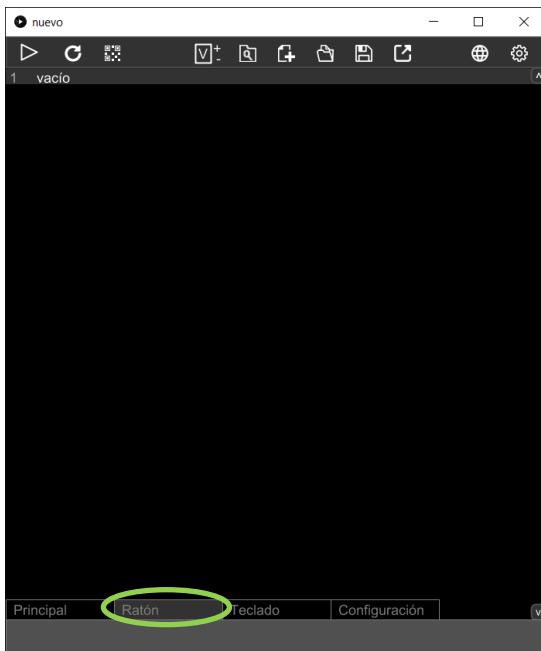
1.18.1. **Principal**

El código que se escribe en la pestaña **Principal** se ejecuta en un ciclo infinito, hasta que se cierre la ventana de la aplicación.



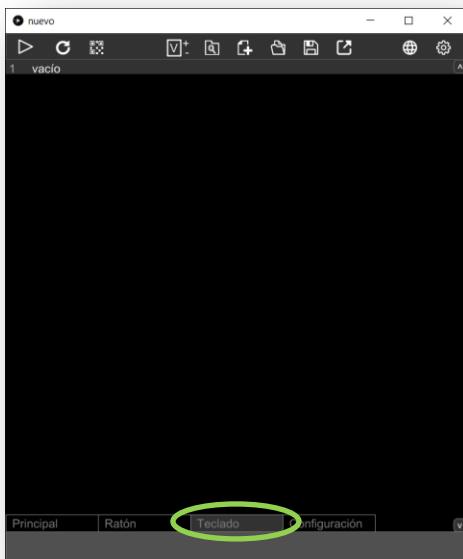
1.18.2. Ratón

El código que se escribe en la pestaña **Ratón** se ejecuta en el momento que se presiona cualquier botón del ratón.



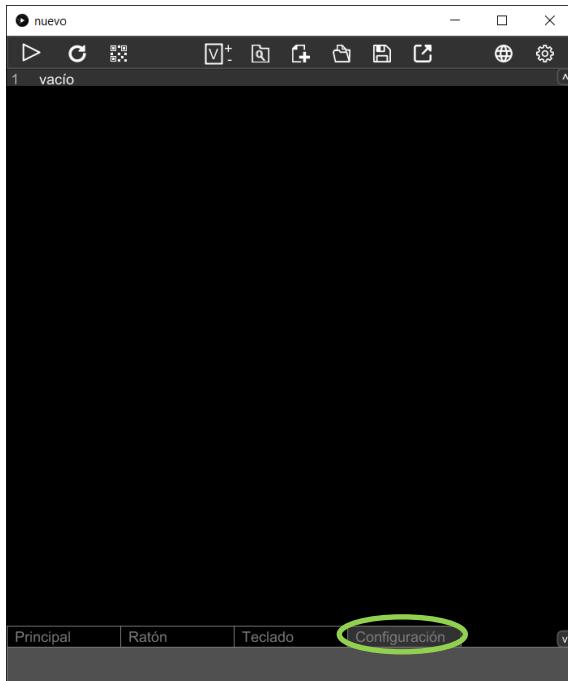
1.18.3. Teclado

El código que se escribe en la pestaña **Teclado** se ejecuta en el momento que se presiona cualquier tecla.



1.18.4. **Configuración**

El código que se escribe en la pestaña **Configuración** se ejecuta una sola vez justo en el momento que se abre la aplicación, es la primera función que se ejecuta, antes que Principal, Ratón o Teclado. Se usa para establecer la configuración inicial de la aplicación (como por ejemplo inicializar variables).



1.19. Atajos de teclado: **ctrl+c, ctrl+x, ctrl+v, ctrl+z**

Para esta versión Meta_Javascript Alpha 1.2. se agrega soporte para los atajos de teclado: **ctrl+c, ctrl+x, ctrl+v, ctrl+z**.

El atajo **ctrl+c** permite copiar la línea de código que este siendo señalada por el cursor del ratón.

El atajo **ctrl+x** permite cortar la línea de código que este siendo señalada por el cursor del ratón.

El atajo **ctrl+v** permite pegar la línea que se haya copiado o cortado en la posición de la línea de código que este siendo señalada por el cursor del ratón.

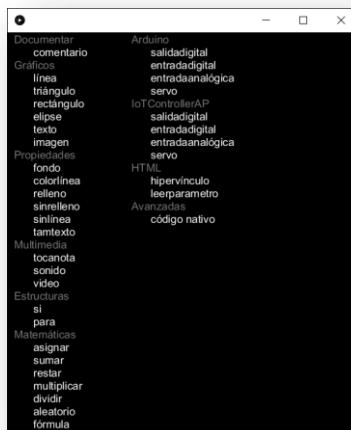
El atajo **ctrl+z** permite deshacer eliminar línea. Si se elimina una línea por equivocación, se puede recuperar la línea borrada presionando **ctrl+z**. Solo aplica para la última línea borrada.

2. INSTRUCCIONES BÁSICAS

En esta sección se abordarán las instrucciones básicas para programar con Meta_Javascript. Como se explicó en el punto 1.8. para agregar una instrucción se debe hacer clic en la palabra que dice **vacio**.



Luego, se abrirá una nueva ventana en donde se muestran todas las instrucciones disponibles en Meta_Javascript organizadas por categorías así:



2.1. Documentar el código

Documentar el código de programación es una de las primeras cosas que debe aprender cualquier persona que quiera aprender a programar. Para este propósito todos los lenguajes de programación permiten agregar líneas de **comentario**. La principal característica de esta línea de código es que no se ejecuta, está allí solo para darle información al programador sobre cómo funciona esa parte del código. Para agregar un comentario se debe hacer clic en la opción **comentario** dentro de la categoría de **Documentar**.



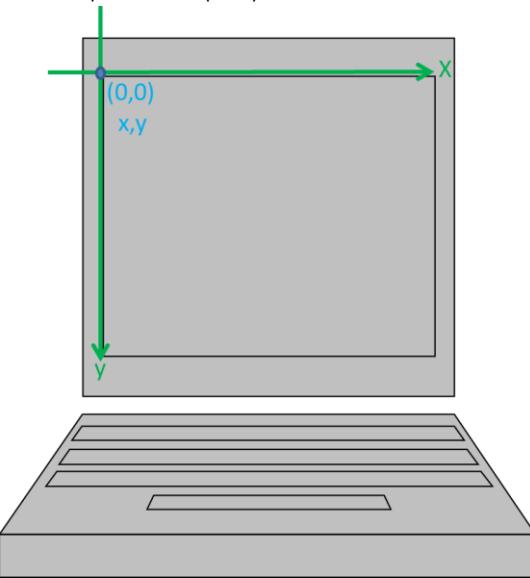
Entonces se abrirá una nueva ventana en la cual se escribe el comentario y se hace clic en **aplicar**.



2.2. Coordenadas de pantalla

Para poder hacer gráficos en pantalla es necesario primero conocer cómo funcionan las coordenadas de pantalla en Meta_Javascript. La posiciones en pantalla se miden en pixeles y cada pantalla tiene un cierto número de pixeles en el eje X y en el eje Y.

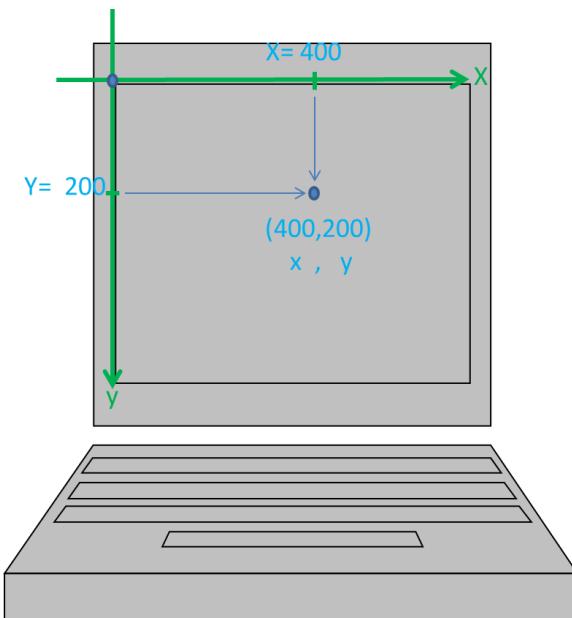
Como se puede observar en la gráfica 1, la esquina superior izquierda de la pantalla es el punto de origen del sistema de coordenadas. Este punto es la posición 0 en el eje X y la posición 0 en el eje Y. Las coordenadas siempre se organizan primero el valor en el eje X, luego una coma y después el valor en el eje Y, así pues este punto es (0,0).



Gráfica 1 Punto de origen en el sistema de coordenadas

Las posiciones en el eje X aumentan de izquierda a derecha y las posiciones en el eje Y aumentan de arriba hacia abajo. En la gráfica 2 se muestra un ejemplo para ilustrar un poco mejor este concepto.

Si se quiera ubicar el punto (400,200) en pantalla, lo que se hace es contar 400 pixels hacia la derecha desde el punto (0,0) de la pantalla y contar 200 pixels de hacia abajo desde el punto (0,0) de la pantalla. Así se ubica el punto (400,200).



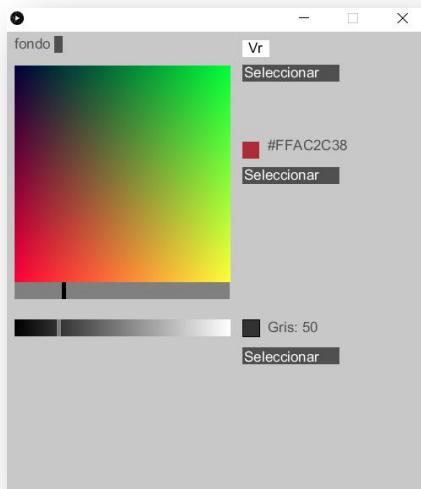
Gráfica 2 Punto en la posición 400 en X y 200 en Y de la pantalla

2.3. Instrucciones para gráficos en pantalla

Entre algunas de las instrucciones para mostrar en pantalla se encuentra: línea, **triángulo**, rectángulo, elipse, texto, imagen. A continuación se explicará cómo usar cada una de ellas.

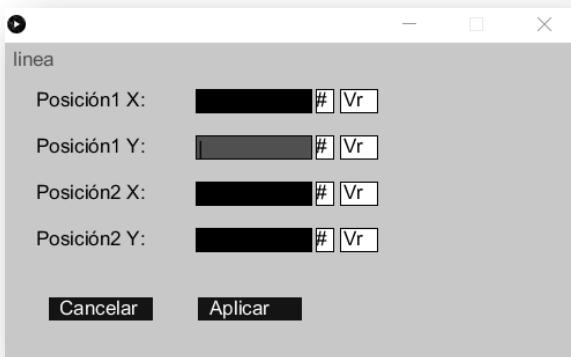
2.3.1. fondo

La instrucción **fondo** sirve para definir el color de fondo de toda la ventana de la aplicación. Esta instrucción borra todo que se esté mostrando en pantalla y deja toda la pantalla del color seleccionado.



2.3.2. Línea

La instrucción **Línea** sirve para dibujar una línea en pantalla. Para traza una línea se requiere definir la posición x,y del punto donde inicia la línea y la posición x,y del punto donde termina la línea.



2.3.3. rectángulo

La instrucción **rectángulo** sirve para dibujar un rectángulo en pantalla. Para usar esta instrucción se debe definir en las dos primeras casillas la posición x,y de la esquina superior izquierda desde donde se dibujará el cuadrado, y en las dos casillas siguientes definir su ancho y alto.



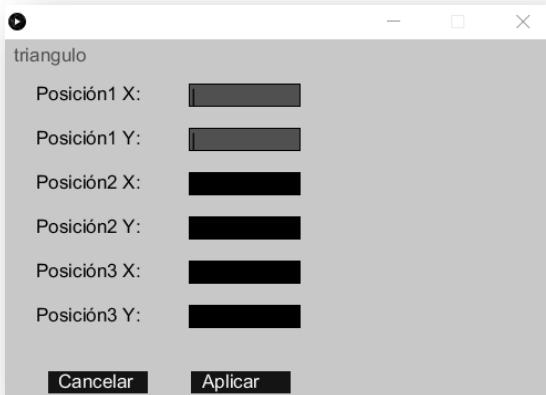
2.3.4. elipse

La instrucción **elipse** sirve para dibujar una elipse en la pantalla. Para usar esta instrucción se debe definir en las dos primeras casillas el punto central x,y desde donde se dibujará la elipse y en las dos casillas siguientes definir su ancho y alto.



2.3.5. triángulo

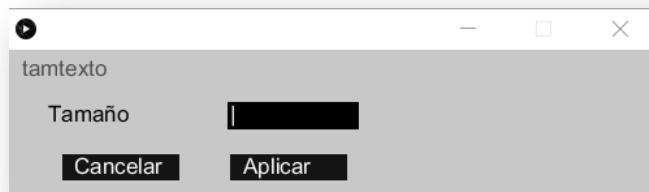
La instrucción **triángulo** sirve para dibujar un triángulo en pantalla. Para dibujar cualquier triángulo se requiere definir los tres puntos correspondientes a cada una de sus esquinas. Para usar esta instrucción se debe definir en las dos primeras casillas la posición del primer punto, en las dos siguientes casillas definir la posición del segundo punto y en las últimas dos casillas de finir la posición del tercer punto.



2.3.6. tamtexto

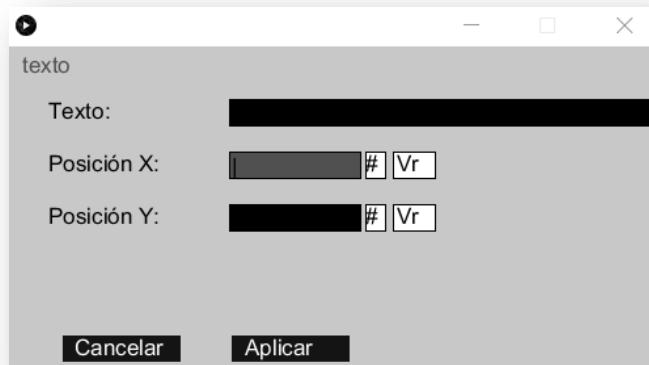
La instrucción **tamtexto** sirve para definir el tamaño de la fuente a la hora de mostrar un texto en pantalla. Para usar esta instrucción se debe llenar la casilla de tamaño con el número correspondiente al tamaño de fuente que se desea aplicar.

Para que esta instrucción tenga efecto se debe agregar antes de la instrucción **texto**.



2.3.7. **texto**

La instrucción texto sirve para mostrar texto en pantalla. Para usar esta instrucción se debe escribir en la primera casilla el texto que se desea mostrar, y definir en las dos siguientes casillas la posición x,y de la esquina inferior izquierda desde donde se comenzará a mostrar el texto en pantalla.



2.3.8. **imagen**

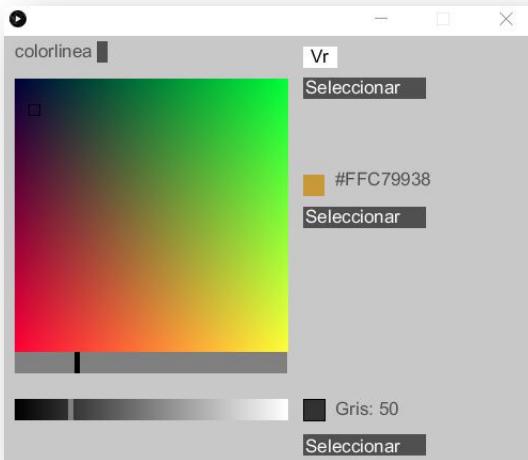
La instrucción **imagen** sirve para mostrar una imagen en pantalla. Para usar esta instrucción primero se debe guardar dentro de la carpeta **data** del proyecto la imagen que se quiere usar, luego se debe seleccionar la imagen que fue guardada en la carpeta data y por último definir en las dos siguientes casillas la posición x,y de la esquina superior izquierda desde donde se comenzará a mostrar la imagen en pantalla.



2.3.9. **colorlinea**

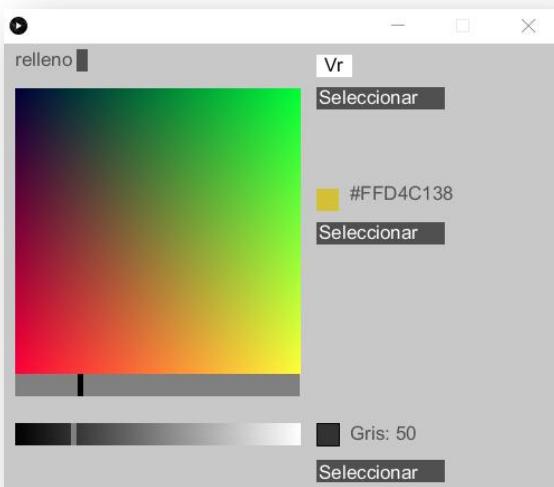
La instrucción **colorlinea** sirve para definir el color de las líneas y color del borde de las figuraras rectángulo, elipse y triángulo. Para asignar el color de línea usando esta instrucción se puede, utilizar el selector de color, o el selector de escala de grises y hacer clic en el botón **Aplicar**. También se puede usar una variable para cambiar de forma dinámica el color. Para que

esta instrucción tenga efecto se debe agregar antes de las instrucciones con las que se dibuja líneas, o figuras en pantalla.



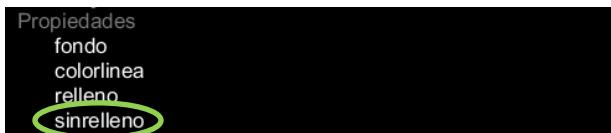
2.3.10. **relleno**

La instrucción **relleno** sirve para definir el color de relleno de las figuraras rectángulo, elipse y triángulo. Para asignar el color de relleno usando esta instrucción se puede, utilizar el selector de color, o el selector de escala de grises y hacer clic en el botón **Aplicar**. También se puede usar una variable para cambiar de forma dinámica el color. Para que esta instrucción tenga efecto se debe agregar antes de las instrucciones con las que se dibuja figuras en pantalla.



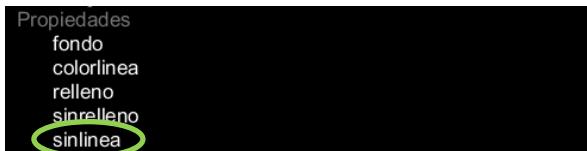
2.3.11. **sinrelleno**

La instrucción **sinrelleno** sirve para quitar el relleno de las figuras rectángulo, elipse y triángulo. Esta se verán transparentes y solo se verá su borde. Para usar esta instrucción solo se necesita que en la ventana de agregar instrucción, se haga clic en sobre la instrucción **sinrelleno** dentro de la categoría **Propiedades** y automáticamente se agregará esta línea al código del proyecto.



2.3.12. **sinlinea**

La instrucción **sinlinea** sirve para quitar el borde de las figuras rectángulo, elipse y triángulo. Si se agrega esta instrucción antes de la instrucción **línea** entonces no se mostrará en pantalla la línea. Para usar esta instrucción solo se necesita que en la ventana de agregar instrucción, se haga clic en sobre la instrucción **sinlinea** dentro de la categoría Propiedades y automáticamente se agregará esta línea al código del proyecto.

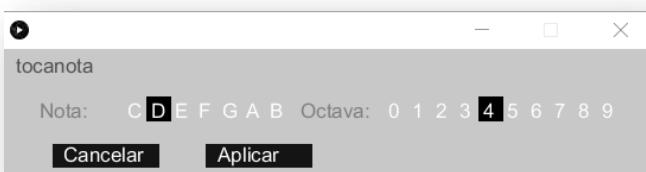


2.4. Instrucciones multimedia

Entre algunas de las instrucciones multimedia que se puede usar con Meta_Javascript se encuentra: **tocanota**, **sonido** y **video**. A continuación se explicará cómo usar cada una de ellas.

2.4.1. **tocanota**

La instrucción **tocanota** sirve para reproducir el sonido de una nota de la escala musical. Para usar esta instrucción se debe seleccionar la nota que se quiere tocar y luego seleccionar la octava que se quiere que suene la nota. Por último se debe hacer clic en el botón **Aplicar**.



2.4.2. sonido

La instrucción **sonido** sirve para reproducir un archivo de sonido en formato wav o mp3. Para usar esta instrucción primero se debe guardar dentro de la carpeta **data** del proyecto, el archivo de sonido que se quiere usar, luego se debe hacer clic en el botón **Seleccionar** para elegir el archivo que fue guardado previamente en la carpeta data y por último se debe hacer clic en el botón **Aplicar**.



2.5. Instrucciones HTML

Las instrucciones HTML que se puede usar ahora con Meta_Javascript son: **hipervínculo** y **leerparámetro**. A continuación se explicará cómo usar cada una de ellas.

2.5.1. hipervínculo

La instrucción **hipervínculo** sirve para saltar a otra página web ya sea local o en Internet. La ruta se abre en la misma pestaña del navegador, lo que significa que se interrumpe la ejecución del código Meta_Javascript para abrir la página web definida en el hipervínculo.



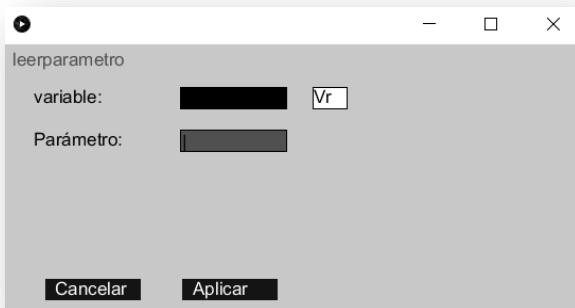
Un ejemplo podría ser un proyecto que abra la página web de **google** cuando se haga clic con cualquier botón del ratón. Para esto se debe agregar el siguiente código en la pestaña **Ratón**.



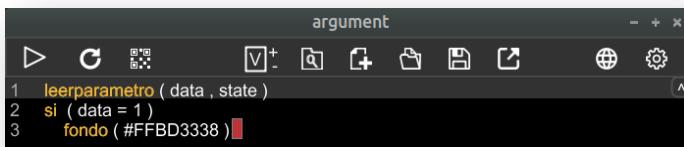
2.5.2. leerparámetro

La instrucción **leerparámetro** sirve para leer el parámetro que se puede enviar al archivo .html del proyecto en el momento que se le haga un llamado

desde la barra de dirección web o al hacer clic en un hipervínculo dentro del mismo o desde otro proyecto. Para usar esta instrucción en la primera casilla se debe seleccionar la variable en la cual se va a almacenar el valor que a leer y en la segunda casilla se debe escribir el nombre del parámetro que se quiere leer.



Un ejemplo podría ser crear un proyecto que permita leer el parámetro **state** y almacenarlo en la variable **data**. Para esto sería necesario agregar en la pestaña **principal** el siguiente código:



Si se quisiera que al hacer clic con cualquier botón de ratón se enviara un parámetro mediante parámetros en

URL, se podría usar la instrucción **hipervínculo** de la siguiente manera:

```
1 hipervínculo ( localhost:8000/argument.html?state=1 ) ^
```

En este caso se enviaría el valor **1** al el parámetro **state** a la página **argument.html**.

3. VARIABLES, CONDICIONES Y CICLOS

En esta sección se abordará los conceptos de variables y condiciones, los cuales son fundamentales a la hora de aprender a programar.

3.1. Variables

Una variable es un espacio de memoria reservado para almacenar un valor que va cambiando durante el trascurso de la ejecución del programa. En Meta_Javascript se hay dos tipos de variables: Las variables del sistema y las variables creadas por el usuario.

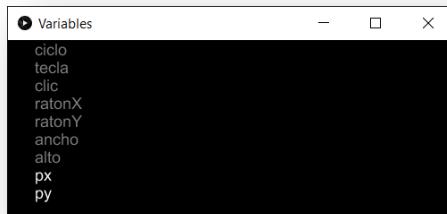
Las variables del sistema son: **ciclo**, **tecla**, **click**, **ratonX**, **ratonY**, **ancho** y **alto**. La variable **ciclo** se usa para contar el número de ciclos dentro de la estructura repetitiva `para`. La variable **tecla** almacena el valor de la última tecla presionada en el teclado. La variable **click** amacena el valor del último botón presionado en el ratón. La variable **ratonX** almacena la posición actual del ratón en el eje X. La variable **ratonY** almacena la posición actual del ratón en el eje Y. La variable **ancho** almacena el valor del ancho de la pantalla en la que ese está ejecutando el código. Y la variable **alto** almacena el valor del alto de la pantalla en la que ese está ejecutando el código.

3.1.1. ¿Cómo se mira el listado de variables?

Para mirar el listado de variables del proyecto se debe hacer clic en el ícono **variable**.



En la ventana que se abre se verán las variables que se están usando. Las que se muestran en color gris son las variables del sistema y las que se muestran de color blanco son las variables creadas por el usuario.



3.1.2. ¿Cómo se crea una variable?

Para crear una variable se debe hacer clic en el ícono más (+) que está a un lado del ícono **variable**.



En la ventana que se abre se debe escribir el nombre que se le quiere dar de la variable que se va a crear, en este ejemplo se le da el nombre **px**.

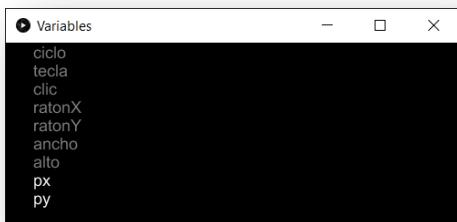


3.1.3. ¿Cómo se elimina una variable?

Para eliminar una variable se debe hacer clic en el ícono menos (-) que está a un lado del ícono **variable**.



En la ventana que se abre se debe hacer clic sobre el nombre de la variable que se quiere eliminar. Solo se pueden eliminar las variables que han sido creadas por el usuario, es decir, solo le pueden eliminar las variables que su texto es color blanco. Las variables en color gris son las variables del sistema y no se pueden eliminar.

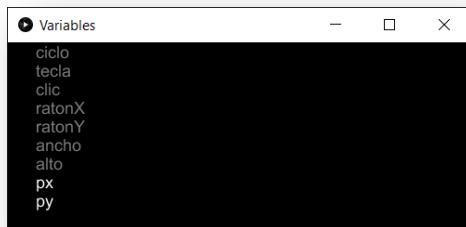


3.1.4. ¿Cómo se inicializa una variable?

Para inicializar una variable creada por el usuario se debe hacer clic en el ícono **variable**.



En la ventana que se abre se hace clic sobre la variable que se quiere inicializar.



Hecho esto aparecerá otra ventana en la que se debe escribir el valor con el que se quiere inicializar la variable. En este ejemplo se inicializa la variable con el valor 200.

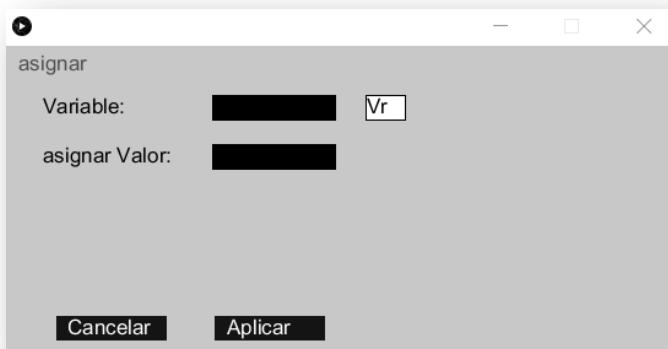


3.1.5. ¿Cómo asignarle un nuevo valor a una variable?

Dentro del código se le puede asignar un nuevo valor a una variable, para ello se debe agregar la instrucción **asignar** que se encuentra dentro de la categoría **Matemáticas** en la ventana de agregar instrucción.



Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere asignar un nuevo valor y en la segunda casilla se escribe el valor que se le va a asignar.

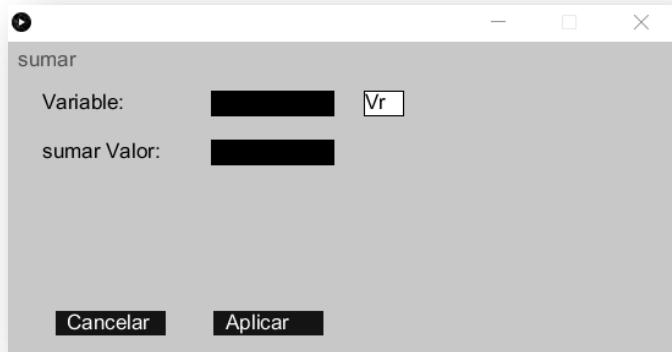


3.1.6. ¿Cómo se le suma un valor a una variable?

Dentro del código se le puede sumar un valor a una variable, para ello se debe agregar la instrucción **sumar** que se encuentra dentro de la categoría **Matemáticas** en la ventana de agregar instrucción.



Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere sumar el valor y en la segunda casilla se escribe el valor que se le va a sumar.

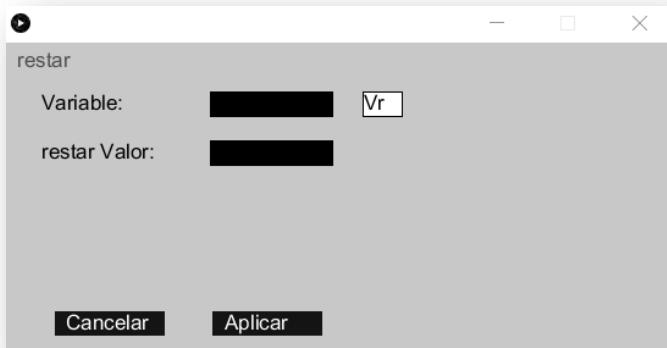


3.1.7. ¿Cómo se le resta un valor a una variable?

Dentro del código se le puede restar un valor a una variable, para ello se debe agregar la instrucción **restar** que se encuentra dentro de la categoría **Matemáticas** en la ventana de agregar instrucción.



Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere restar el valor y en la segunda casilla se escribe el valor que se le va a restar.

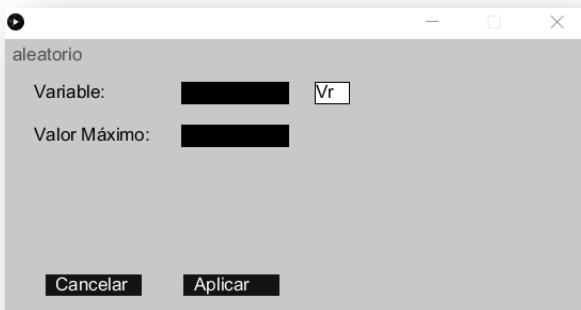


3.1.8. ¿Cómo se le asigna un valor aleatorio a una variable?

Dentro del código se le puede asignar un valor aleatorio a una variable, para ello se debe agregar la instrucción **aleatorio** que se encuentra dentro de la categoría **Matemáticas** en la ventana de agregar instrucción.



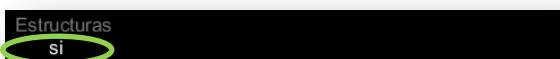
Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere asignar el valor aleatorio y en la segunda casilla se escribe el valor máximo que se podría generar aleatoriamente. De manera que se generaría un valor aleatorio entre 0 y el valor máximo que se le asigne a la instrucción.



3.2. Condiciones

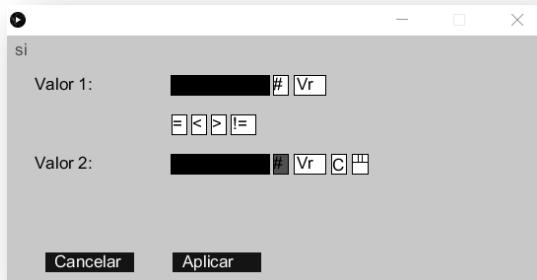
Las condiciones son un tipo de estructuras algorítmicas que le permiten al programa tomar decisiones según se cumpla una condición.

Para agregar una condición se debe agregar la instrucción **si** que se encuentra dentro de la categoría Estructuras en la ventana de agregar instrucción.

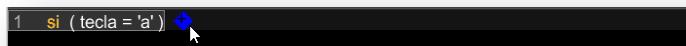


Luego en la ventana que se abre en la primera casilla se puede escribir un valor o seleccionar una variable. Luego se debe seleccionar un operador, el cual puede ser igual (=), menor que (<), mayor que (>) o

diferente (*i*=). Y en la segunda casilla se puede escribir un valor, seleccionar una variable, elegir una tecla o elegir uno de los botones del ratón.



Una vez se hace clic en el botón **Aplicar** se podrá ver la condición en la ventana de código de Meta_Javascript. Para agregar una instrucción dentro de la condición, se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.



Una vez se hace clic aparecerá la nueva línea de código vacía y se hace clic para asignarle la instrucción deseada. Si se quiere crea una instrucción más dentro de la condición entonces, se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.

```
1 si ( tecla = 'a')
2 fondo ( #FF1ED538 )
```

Si una vez se termina de agregar las instrucciones dentro de la condición, lo que se quiere es agregar otra línea de código pero por fuera de la condición, entonces se debe mover el cursor del ratón hasta que aparezca un circulo verde con el carácter más (+) en su interior.

```
1 si ( tecla = 'a')
2 fondo ( #FF1ED538 )
```

Si por el contrario lo que se quiere es agregar una línea para el caso cuando no se cumpla dicha condición entonces se podrán agregar líneas dentro del sino. Para esto se debe mover el cursor del ratón hasta que aparezca un rombo amarillo con el carácter más (+) en su interior.

```
1 si ( tecla = 'a')
2 fondo ( #FF1ED538 )
```

Una vez se seleccione la instrucción que se quiere usar se vería que adelante de la instrucción aparece la palabra **Sino**. Para cuando se quiera agregar más instrucciones dentro del **Sino** se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.



```
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
```

A screenshot of the Scratch script editor. A conditional block is selected, starting with 'si (tecla = 'a')'. The 'fondo' block is nested under it. Below the conditional, there is a 'Sino' block followed by another 'fondo' block. The 'fondo' block for the 'Sino' condition has a blue diamond cursor icon with a '+' sign inside it, indicating it's ready to be expanded.

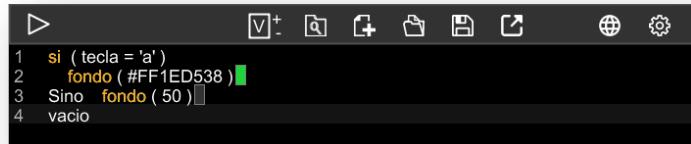
Si por el contrario se quiere es agregar otra línea de código pero por fuera del **Sino**, entonces se debe mover el cursor del ratón hasta que aparezca un circulo verde con el carácter más (+) en su interior.



```
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
```

A screenshot of the Scratch script editor. The same conditional block is shown, but now there is an empty line of code directly below the 'Sino' block. A green circle cursor icon with a '+' sign is positioned at the end of the line, indicating where new code can be added.

Al hacer esto entonces aparecerá una nueva línea vacía por fuera de las instrucciones de la condición.



```
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
4 vacio
```

A screenshot of the Scratch script editor showing a completed script. It consists of four blocks: a conditional 'si (tecla = 'a')', a 'fondo' block nested inside it, a 'Sino' block, and a final 'vacio' block. The interface includes various tool icons at the top.

3.1. Ciclos

Los ciclos en programación son un tipo de estructuras algorítmicas que ejecutan un conjunto de instrucciones múltiples veces mientras se cumpla una condición. Los ciclos son muy útiles para realizar tareas repetitivas, un ejemplo de uso puede ser cuando se quiere dibujar una cuadricula, se podría hacer escribiendo el código para dibujar línea por línea, o se podría hacer creando un ciclo para dibujar todas las líneas horizontales y creando otro ciclo para dibujar todas las líneas verticales. Si la cuadricula es de 3 líneas x 3 líneas no tiene mayor dificultad dibujar línea por línea, pero si la cuadricula es de 1000 x 1000 entonces con dos ciclos se puede realizar la tarea lo cual hace que el código sea optimizado y simplificado.

Para agregar un ciclo se debe seleccionar la instrucción **para** que se encuentra dentro de la categoría Estructuras en la ventana de agregar instrucción.



Luego en la ventana que se abre en la primera casilla se puede escribir el valor inicial de la variable del sistema llamada **ciclo**. Debajo de este casilla se puede seleccionar el operador para comparar el valor que contiene la variable ciclo con un valor que se debe

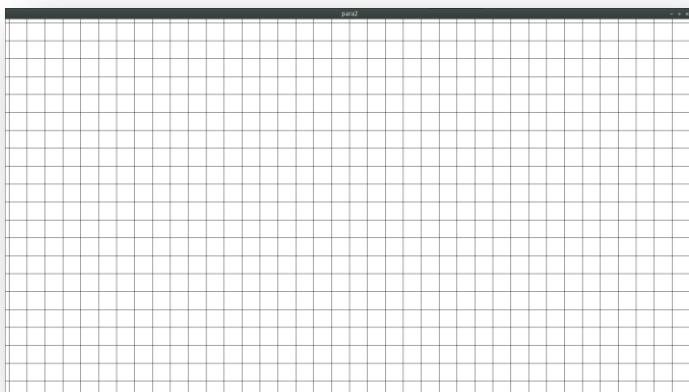
agregar en la segunda casilla. Por ultimo en la tercera casilla se define en cuanto se va a incrementar la variable ciclo una vez se ejecutan las instrucciones dentro de la estructura y se reinicia el ciclo para.



Un ejemplo del uso de la estructura para puede ser dibujar una cuadricula. El siguiente código pude escribirse en la pestaña ratón, de manera que cuando se presione alguno de su botones de dibuje la cuadrícula.

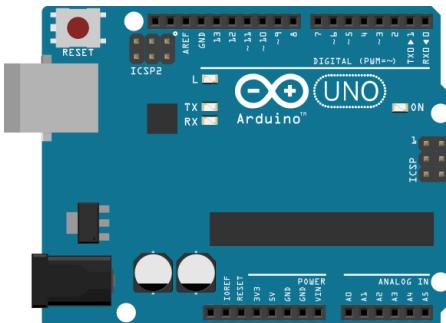
```
1 fondo( 255 )
2 para( desde 10 ; mientras ciclo < 1920 ; Incremento 50 )
3   línea( ciclo , 0 , ciclo , alto )
4 para( desde 10 ; mientras ciclo < 1080 ; Incremento 50 )
5   línea( 0 , ciclo , ancho , ciclo )
```

El resultado que se obtiene al ejecutar este código de 5 líneas es el siguiente:



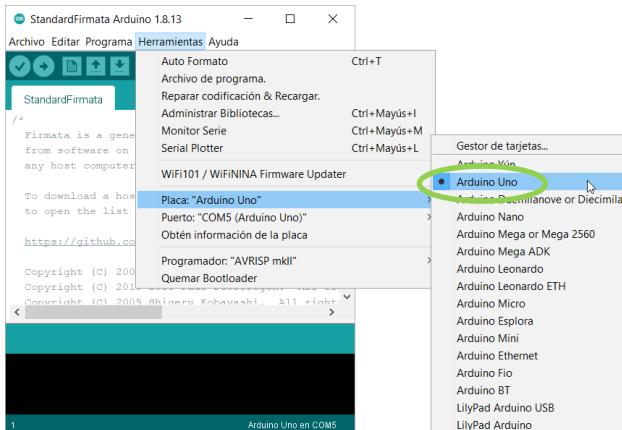
4. USANDO ARDUINO CON META_PROCESING

Para usar una tarjeta Arduino con Meta_Javascript se requiere que al Arduinio se le cargue la librería Firmata. De esta forma se puede establecer una comunicación por el puerto USB con la tarjeta y controlar varias de sus principales funciones: prender y apagar pines, hacer lectura digital de un pin, hacer lectura analógica de un pin y controlar servos. No se necesita instalar Node.js ni Johnny-Five, solo tener abierto Meta_Javascript.

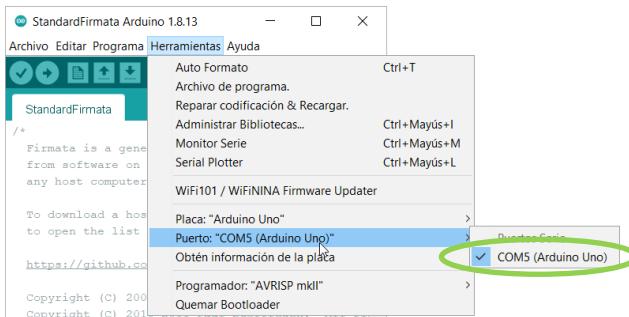


4.1. Instalación de la librería Firmata en una tarjeta Arduino

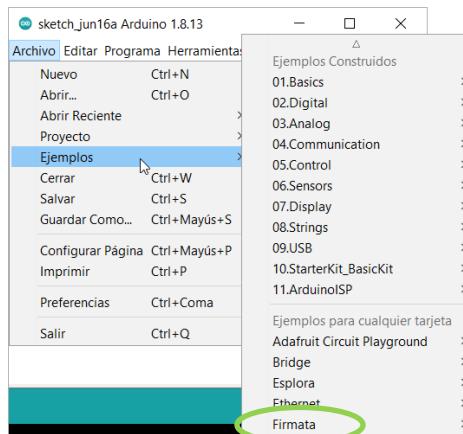
Para cargar la librería Firmata en la tarjeta el primer paso es abrir el IDE de Arduino y elegir la tarjeta Arduino a la que se va a cargar la librería Firmata.



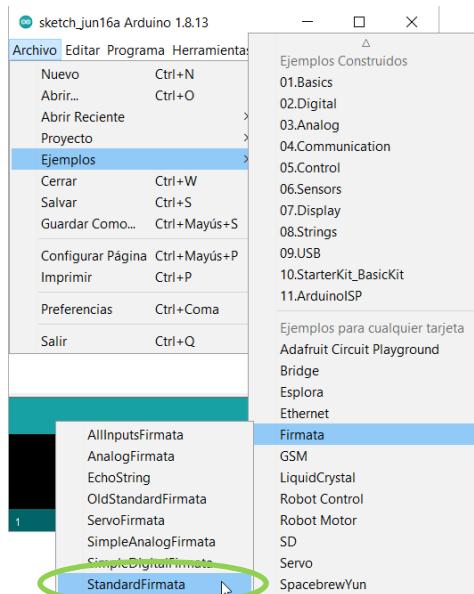
Luego se debe elegir el puerto al que está conectada la tarjeta Arduino.



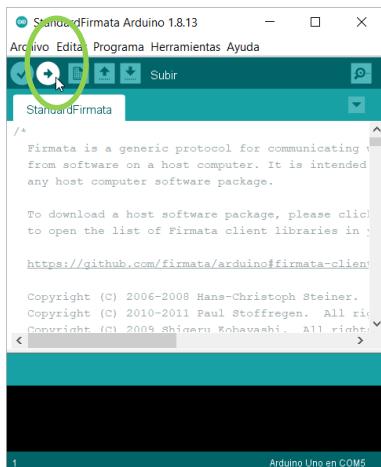
Luego se debe abrir los **ejemplos** de la librería Firmata, los cuales se acceden desde del menú **archivo**.



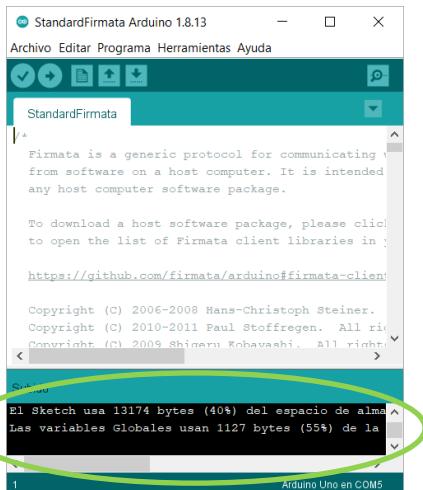
Después se debe hacer clic en la opción **Firmata** para ver todos los archivos de ejemplo de esta librería, y abrir el ejemplo: **StandardFirmata**



Con el ejemplo abierto, hacer clic en el ícono **subir**.



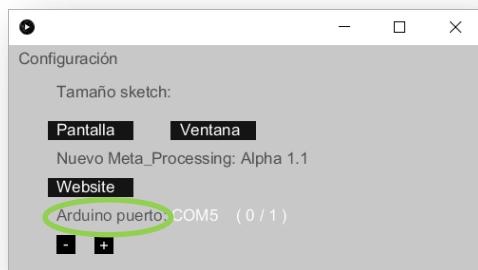
Cuando se termina de subir la librería a la tarjeta Arduino se debe ver algo así:



El siguiente paso es abrir Meta_Javascript y seleccionar el puerto en el cual se encuentra conectada la tarjeta Arduino. Para esto se debe hacer clic en el ícono **configuración** para abrir la ventana de configuración.



En la ventana que se abre se puede ver que la tercera opción es: **Arduino puerto**.



Para seleccionar el puerto de conexión con la tarjeta Arduino se debe hacer clic en los botones – o +. Cuando está conectado el Arduino al puerto USB el primer valor que se muestra es el nombre del puerto, y justo al lado se muestra un paréntesis con dos valores. El primer valor dentro del paréntesis indica el número del puerto actual y el valor seguido del / es el total de puertos. En caso de no haber ningún dispositivo conectado no se muestra el nombre del puerto, y en los números de puerto se muestra: (0/0).

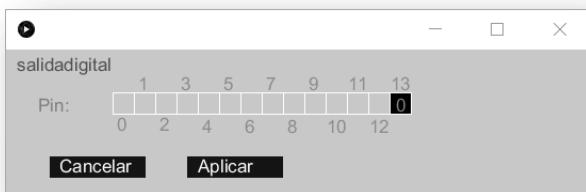
4.2. Instrucciones Arduino

Las instrucciones de Arduino que se puede usar con Meta_Javascript Alpha 1.2 son: **salidadigital**, **entrada-digital**, **entradaanalógica** y **servo**. A continuación se explicará cómo usar cada una de ellas.

4.2.1. salidadigital

La instrucción **salidadigital** sirve para prender o apagar un pin determinado de la tarjeta Arduino. Si se enciende un pin significa que este pin suministrará un voltaje superior a 3 voltios al dispositivo que esté conectado a dicho pin. Por el contrario si se apaga un pin, significa que este pin suministrará un voltaje inferior a 1.5 voltios al dispositivo que esté conectado a dicho pin.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin del Arduino que se quiere prender o apagar. Cuando el fondo de la casilla es negra entonces se apaga ese pin, si el color es blanco entonces se prende ese pin. Ejemplo pin 13 apagado:



Ejemplo pin 13 prendido:



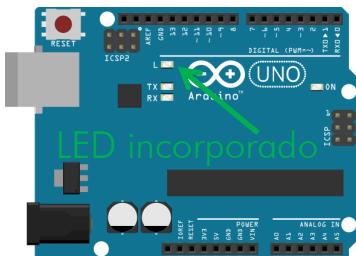
Si se quisiera crear un proyecto que prenda el pin 13 cuando se haga clic con el ratón y que se apague cuando se oprime cualquier tecla, en la pestaña **ratón** se pondría el siguiente código:

```
1 salidadigital ( 13, 1 )
```

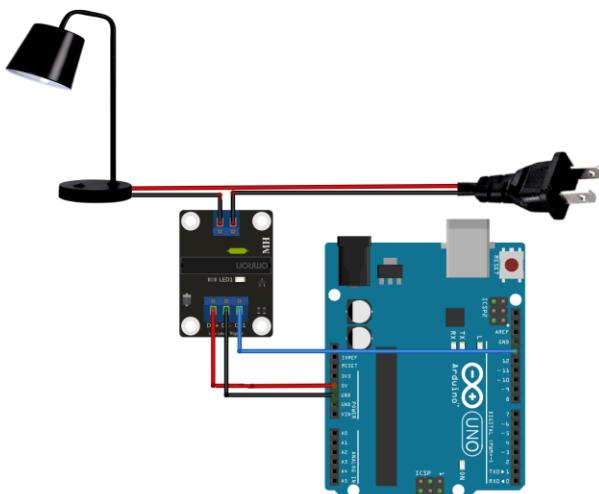
Y en la pestaña **teclado** se pondría el siguiente código:

```
1 salidadigital ( 13, 0 )
```

Para usar el anterior código no sería necesario construir ningún circuito con el Arduino, ya que se usaría el Led que está incorporado en todas las tarjetas Arduino Uno y que por defecto está conectado al pin 13.



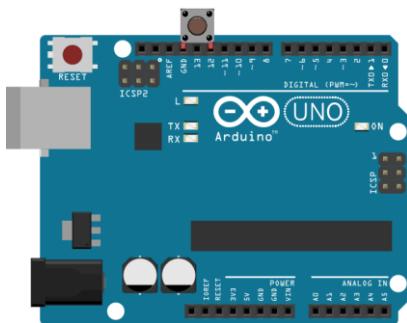
A continuación se muestra un circuito que se podría construir para prender y apagar una lámpara conectada a los 110 voltios, haciendo uso del mismo código. Para esto sería necesario un módulo relé de estado sólido para Arduino.



4.2.2. entradadigital

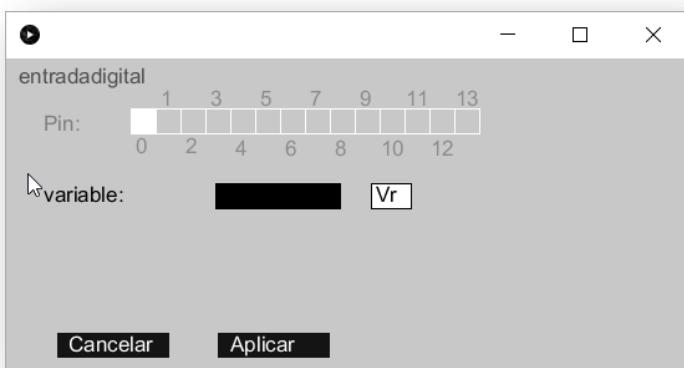
La instrucción **entradadigital** sirve para leer el estado de un pin digital de la tarjeta Arduino. Esta instrucción fue

creada para poder conectar un pulsador directamente a cualquier pin digital, sin necesitar construir un circuito adicional (Pull Up). Uno de los terminales de pulsador deberá estar conectado al pin GND y el otro al pin digital que se desee usar. En este ejemplo pin 12:



Esta instrucción almacena en una variable el estado del pin digital, si es 1 significa que el pulsador está presionado si por el contrario es 0 significa que el pulsador no está siendo presionado.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin del Arduino que se quiere leer y en la segunda casilla se debe seleccionar la variable que va a almacenar el estado del pin.



Si se quisiera crear un proyecto con un pulsador que cada vez que se presione el color de fondo cambie a verde, y que cuando no esté presionado se ponga el fondo de color rojo, en la pestaña **principal** se pondría el siguiente código:

```
1 entradasdigital ( 12, x )
2 si ( x = 1 )
3   fondo ( #FF15ED38 ) █
4 Sino fondo ( #FFDC3338 ) █
```

Para usar este código de ejemplo se debe tener conectado un terminal del pulsador al pin GND del Arduino y el otro terminal al pin 12 del Arduino.

4.2.3. entradaanalógica

La instrucción **entradaanalógica** sirve para leer el estado de un pin analógico de la tarjeta Arduino. El rango de valores que se puede obtener de esta lectura está entre 0 y 1023.

Esta instrucción almacena en una variable el estado del pin analógico, donde 0 equivaldría a una lectura de un voltaje inferior a 0.0049 voltios y 1023 a una lectura de un voltaje de 5 voltios.

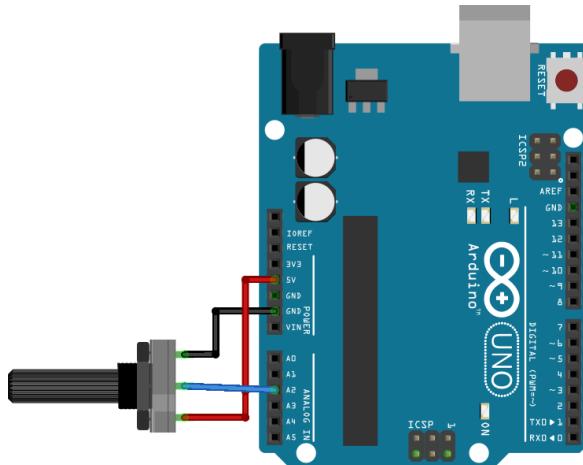
Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin analógico del Arduino que se quiere leer y en la segunda casilla se debe seleccionar la variable que va a almacenar el estado del pin.



Si se quisiera crear un proyecto con un potenciómetro conectado al pin analógico 2 y que al girarse cambie el color de fondo, en la pestaña **principal** se pondría el siguiente código:

```
1 entradaanalógica ( 2, giro )
2 si ( giro > 100 )
3   fondo ( 218 )
4 si ( giro > 400 )
5   fondo ( 158 )
6 si ( giro > 600 )
7   fondo ( 87 )
```

El circuito necesario para usar el anterior código de ejemplo seria el siguiente:



4.2.4. servo

La instrucción **servo** sirve para controlar el ángulo al cual se desea girar un servomotor conectado a alguno de los pines digitales de la tarjeta Arduino.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin del Arduino al cual está conectado el servomotor, y en la segunda casilla se debe seleccionar el valor del ángulo al que se quiere girar el servomotor.



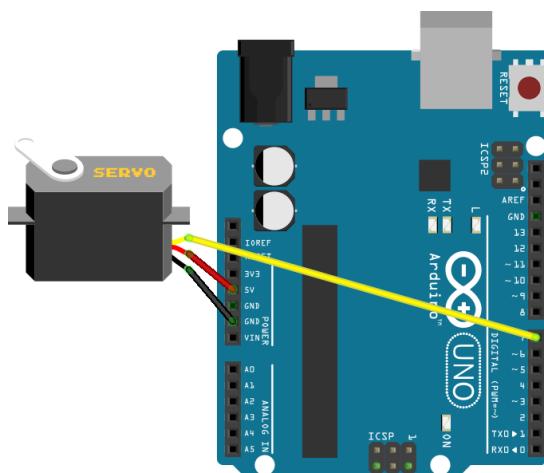
Si se quisiera crear un proyecto con un servomotor conectado al pin digital 7 para que gire en un sentido cuando se haga clic con el ratón y que gire en el sentido contrario cuando se oprima cualquier tecla, en la pestaña **ratón** se pondría el siguiente código:

```
1 servo(7,0)
```

Y en la pestaña **teclado** se pondría el siguiente código:

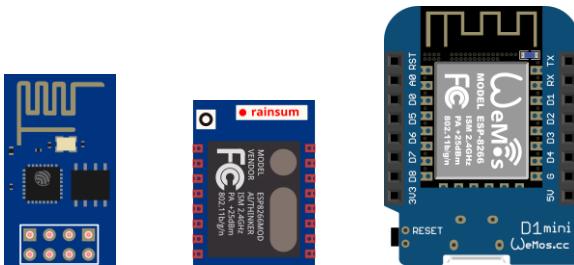
```
1 servo(7,180)
```

El circuito necesario para usar el anterior código de ejemplo sería el siguiente:



5. USANDO ESP CON META_PROCESING

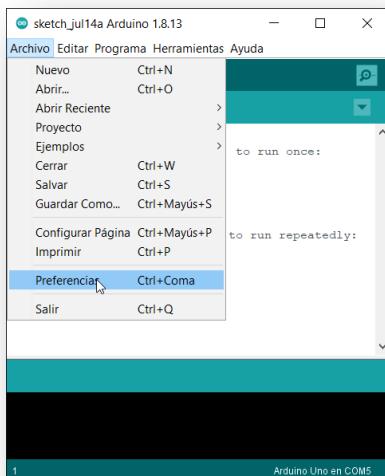
Para usar una tarjeta ESP con Meta_Javascript se requiere que a la ESP se le cargue la librería IoTControllerAP³. Esta librería permite que la tarjeta ESP funcione como un Access Point con nombre *IoTController* (no requiere contraseña). De esta forma se puede establecer una comunicación WiFi directa con la tarjeta y controlar varias de sus principales funciones: prender y apagar pines, hacer lectura digital de un pin, hacer lectura analógica de un pin y controlar servos. No se necesita instalar Node.js ni Johnny-Five, solo tener abierto Meta_Javascript.



³ <https://github.com/hiteclab/IoTControllerAP>

5.1. Instalación de la librería IoTControllerAP en una tarjeta ESP

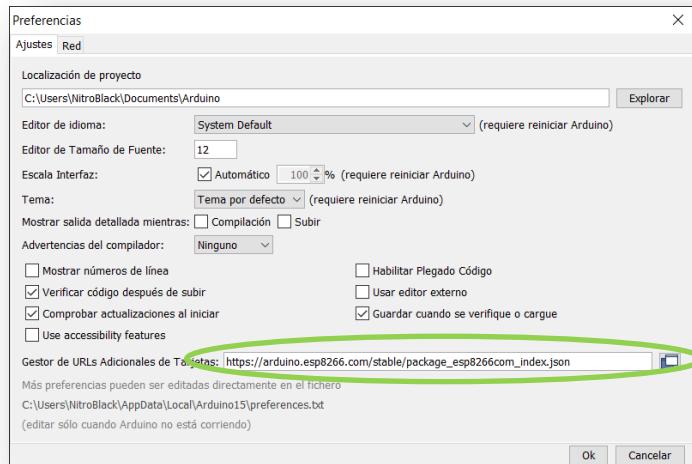
Para cargar la librería IoTControllerAP en la tarjeta ESP el primer paso es abrir el IDE de Arduino e instalar el controlador de estas tarjetas. Para esto es necesario hacer clic en la opción **Preferencias** dentro del menú principal **Archivo**.



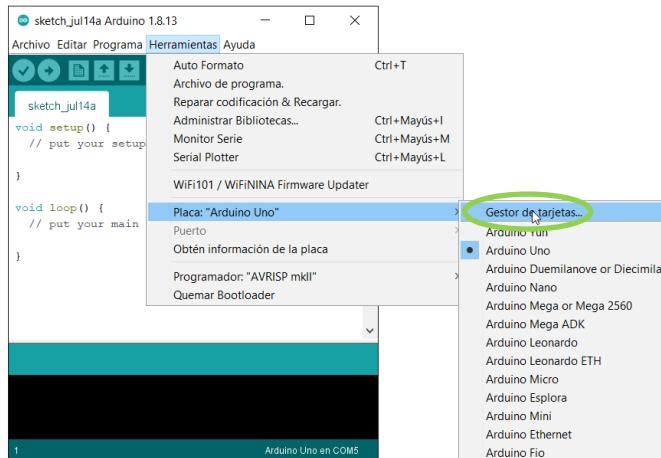
En la ventana que se abre se debe agregar en la opción **Gestor de URLs Adicionales de Tarjetas**, la línea:

https://arduino.esp8266.com/stable/package_esp8266com_index.json

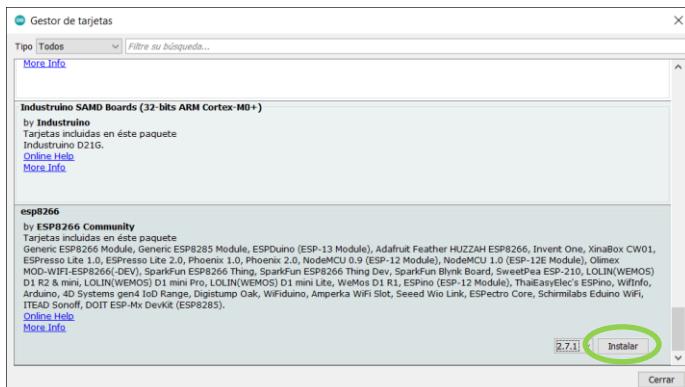
Como se muestra a continuación:



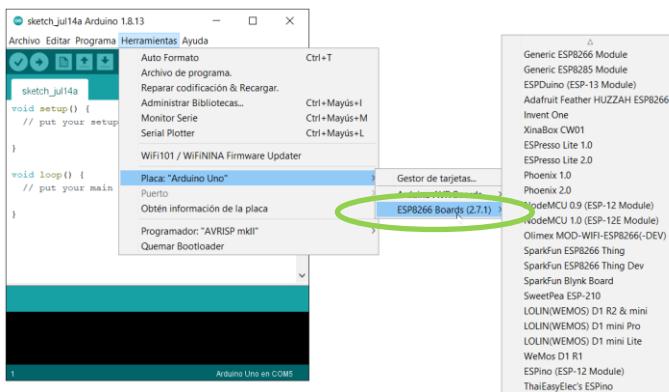
Luego se debe hacer clic en la opción **Placa** dentro del menú **Herramientas**, en el menú que se despliega se debe hacer clic en la opción **Gestor de tarjetas**:



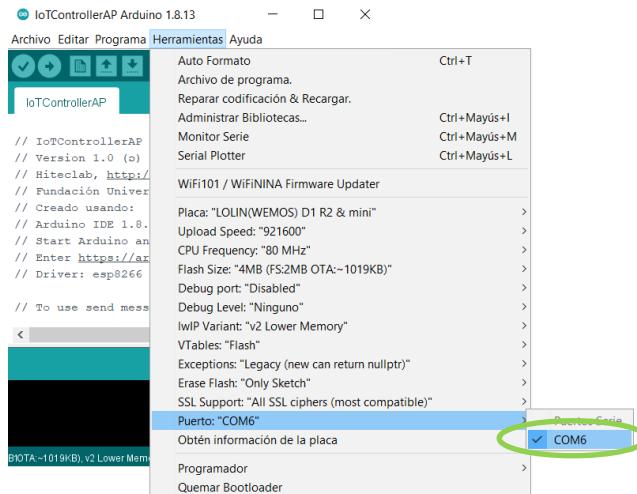
En la ventana que se abre se debe buscar el controlador para la tarjeta **esp8266**, seleccionar la versión **2.7.1** y hacer clic en el botón **Instalar**.



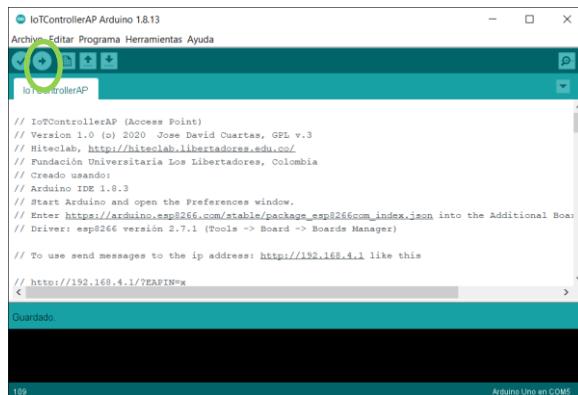
Una vez hecho esto se puede elegir la tarjeta ESP a la que se va a cargar la librería IoTControllerAP.



Luego se debe elegir el puerto al que está conectada la tarjeta ESP.



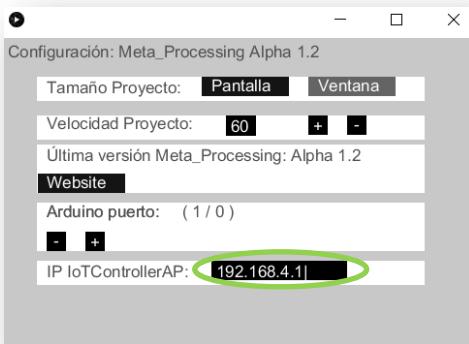
Lo siguiente es descargar la librería IoTControllerAP desde <https://github.com/hiteclab/IoTControllerAP>, luego se debe abrir el código **IoTControllerAP.ino**, y hacer clic en el ícono **subir**.



Una vez cargado el controlador IoTControllerAP en la tarjeta ESP, se debe conectar nuestra computadora vía WiFi a la tarjeta ESP. Se debe buscar en Access Point IoTController y hacer clic en conectar (No requiere contraseña). Luego se debe abrir Meta_Javascript y hacer clic en el ícono **configuración** para abrir la ventana de configuración y poder confirmar que la dirección IP de la Tarjeta ESP, es **192.168.4.1**.



En la ventana que se abre se puede ver que la cuarta opción es: **IP IoTControllerAP**.



Allí se puede cambiar la dirección IP de la tarjeta con la librería IoTControllerAP, por defecto es: **192.168.4.1**

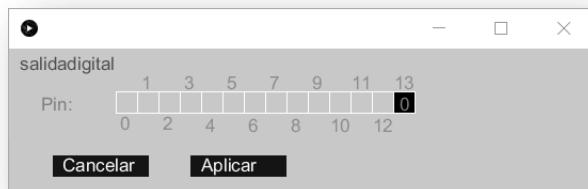
5.2. Instrucciones IoTControllerAP

Las instrucciones de IoTControllerAP que se puede usar con Meta_Javascript Alpha 1.2 son: **salidadigital**, **entradadigital**, **entradaanalógica** y **servo**. A continuación se explicará cómo usar cada una de ellas.

5.2.1. salidadigital

La instrucción **salidadigital** sirve para prender o apagar un pin determinado de la tarjeta ESP. Si se enciende un pin significa que este pin suministrará un voltaje superior a 3 voltios al dispositivo que esté conectado a dicho pin. Por el contrario si se apaga un pin, significa que este pin suministrará un voltaje inferior a 1.5 voltios al dispositivo que esté conectado a dicho pin.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin del ESP que se quiere prender o apagar. Cuando el fondo de la casilla es negra entonces se apaga ese pin, si el color es blanco entonces se prende ese pin. Ejemplo pin 13 apagado:



Ejemplo pin 13 prendido:



Si se quisiera crear un proyecto que prenda el pin 12 cuando se haga clic con el ratón y que se apague cuando se oprime cualquier tecla, en la pestaña **ratón** se pondría el siguiente código:

```
1 salidadigital ( 12, 1 ) IoTControllerAP
```

Y en la pestaña **teclado** se pondría el siguiente código:

```
1 salidadigital ( 12, 0 ) IoTControllerAP
```

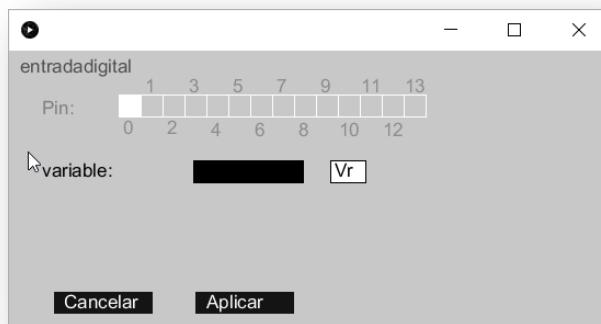
Para usar el anterior código sería necesario conectar un LED al pin 12 de la tarjeta ESP.

5.2.2. entradadigital

La instrucción **entradadigital** sirve para leer el estado de un pin digital de la tarjeta ESP. Esta instrucción fue creada para poder conectar un pulsador directamente a cualquier pin digital, sin necesitar construir un circuito adicional (Pull Up). Uno de los terminales de pulsador deberá estar conectado al pin GND y el otro al pin digital que se desee usar.

Esta instrucción almacena en una variable el estado del pin digital, si es 1 significa que el pulsador está presionado si por el contrario es 0 significa que el pulsador no está siendo presionado.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin del ESP que se quiere leer y en la segunda casilla se debe seleccionar la variable que va a almacenar el estado del pin.



Si se quisiera crear un proyecto con un pulsador que cada vez que se presione el color de fondo cambie a verde, y que cuando no esté presionado se ponga el fondo de color rojo, en la pestaña **principal** se pondría el siguiente código:

```
1 entradadigital ( 4, esp ) IoTControllerAP
2 si ( esp > 0 )
3   fondo ( #FF52D938 ) █
4 Sino fondo ( #FFD24638 ) █
```

Para usar este código de ejemplo se debe tener conectado un terminal del pulsador al pin GND del Arduino y el otro terminal al pin 4 del ESP.

5.2.3. **entradaanalógica**

La instrucción **entradaanalógica** sirve para leer el estado de un pin analógico de la tarjeta ESP. El rango de valores que se puede obtener de esta lectura está entre 0 y 1023.

Esta instrucción almacena en una variable el estado del pin analógico, donde 0 equivaldría a una lectura de un voltaje inferior a 0.0049 voltios y 1023 a una lectura de un voltaje de 5 voltios.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin analógico del ESP que se

quiere leer y en la segunda casilla se debe seleccionar la variable que va a almacenar el estado del pin.



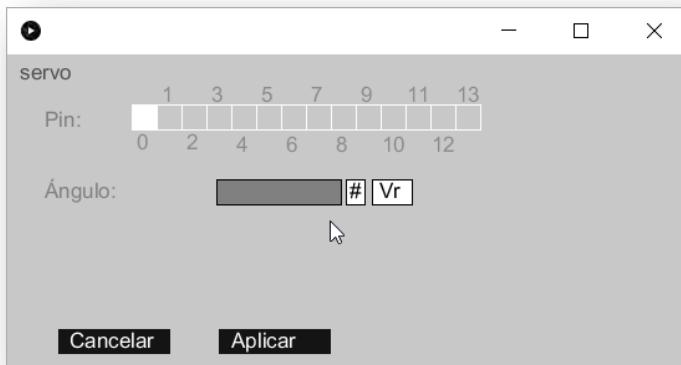
Si se quisiera crear un proyecto con un potenciómetro conectado al pin analógico 0 y que al girarse cambie el color de fondo, en la pestaña **principal** se pondría el siguiente código:

```
1 entradaanalógica ( 0, giro ) IoTControllerAP
2 si ( giro > 100 )
3   fondo ( 218 )
4 si ( giro > 400 )
5   fondo ( 158 )
6 si ( giro > 600 )
7   fondo ( 87 )
```

5.2.4. servo

La instrucción **servo** sirve para controlar el ángulo al cual se desea girar un servomotor conectado a alguno de los pines digitales de la tarjeta ESP.

Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin del ESP al cual está conectado el servomotor, y en la segunda casilla se debe seleccionar el valor del ángulo al que se quiere girar el servomotor.



Si se quisiera crear un proyecto con un servo motor conectado al pin digital 4 para que gire en un sentido cuando se haga clic con el ratón y que gire en el sentido contrario cuando se oprima cualquier tecla se podría usar el siguiente código.

En la pestaña **ratón** se agregaría:

```
1 servo(4,0) IoTControllerAP
```

Y en la pestaña **teclado** se agregaría:

```
1 servo(4,180) IoTControllerAP
```

6. EJEMPLOS DE CÓDIGO CON META_JAVASCRIPT

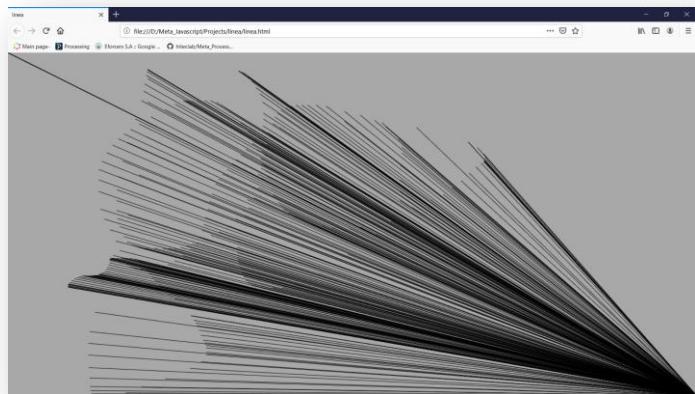
A continuación se presentan una serie de ejemplos de cómo usar Meta_Javascript. El primero de ellos nos muestra cómo hacer dibujos abstractos con líneas. El segundo no permite experimentar con la función teclado. El tercero nos permite dibujar círculos en pantalla al presionar el ratón. El cuarto es un ejemplo de cómo se puede crear una animación. El quinto nos muestra tres formas de programar un piano. Y el último nos muestra cómo crear un sencillo mini juego.

6.1. Dibujos abstractos con líneas

La siguiente línea de código se debe escribir en la pestaña **Principal** y al ejecutar el código se debe mover el cursor del ratón para generar los dibujos abstractos con líneas.

```
1  linea( ancho , alto , ratonX , ratonY )
```

Al ejecutarse se verá en el navegador algo así:



6.2. Ejemplo básico con el Teclado

El siguiente código se debe escribir en la pestaña **Teclado** de manera que se ejecute en el momento que se presione cualquier tecla.

```
1 fondo (#FF0B3B99)
2 tamtexto ( 180 )
3 texto ( Hello World! , 500 , 500 )
```

La idea es que al presionar cualquier tecla, se pone la pantalla azul y aparece en la pantalla un texto en color blanco.

Hello World!

6.3. Ejemplo básico con el Ratón

El siguiente código se debe escribir en la pestaña **Ratón** de manera que se ejecute en el momento que se presione cualquiera de los botones del ratón.

```
1  sinlínea  
2  relleno ( #FFFDD738 )  
3  elipse ( ratonX , ratonY , 80 , 80 )
```

La idea es que al presionar cualquiera de los botones del ratón se dibujen círculos amarillos en pantalla.



6.4. Animación

A continuación se muestra un ejemplo de cómo se puede hacer una animación en Meta_Javascript. Primero se debe crear dos variables una llamada **px** y otra llamada **py**. Se debe inicializar la variable **px** con el valor 10.



Y variable **py** se debe inicializar con el valor 950.



Luego se debe agregar el siguiente código en la pestaña **Principal**:

```
1 fondo ( px )  
2 sumar ( px , 0.5 )  
3 restar ( py , 0.5 )  
4 relleno ( #FFF2E138 )  
5 elipse ( 200 , py , 200 , 200 )  
6 relleno ( #FF0B259A )  
7 rectangulo ( 0 , 800 , ancho , 400 )
```

Al hacer clic en el ícono **ejecutar** se vería que la animación empeza recreando una escena nocturna en

el mar, y lentamente irá amaneciendo hasta tener el firmamento completamente iluminado.



6.5. Piano

A continuación se muestra tres ejemplos de cómo hacer un piano en Meta_Javascript. El primero es un piano simple, el segundo es un piano que además cambia el color de la pantalla y el tercero es un piano que cambia el color de la pantalla y muestra un texto con la nota que suena.

6.5.1. Piano simple

Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:

```
1  si ( tecla = 'a' )
2    tocanota ( C4 )
3  si ( tecla = 's' )
4    tocanota ( D4 )
5  si ( tecla = 'd' )
6    tocanota ( E4 )
7  si ( tecla = 'f' )
8    tocanota ( F4 )
9  si ( tecla = 'g' )
10   tocanota ( G4 )
11  si ( tecla = 'h' )
12   tocanota ( A4 )
13  si ( tecla = 'j' )
14    tocanota ( B4 )
```

Principal | Ratón | Teclado

Con este código se puede tocar las 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

6.5.2. Piano pantalla de colores

Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:

```
1 si ( tecla = 'a')
2   tocanota ( C4 )
3   fondo ( #FF2AD538 )■
4 si ( tecla = 's' )
5   tocanota ( D4 )
6   fondo ( #FFC72538 )■
7 si ( tecla = 'd' )
8   tocanota ( E4 )
9   fondo ( #FF3348AE )■
10 si ( tecla = 'f' )
11   tocanota ( F4 )
12   fondo ( #FFCD138 )■
13 si ( tecla = 'g' )
14   tocanota ( G4 )
15   fondo ( #FF48C0E )■
16 si ( tecla = 'h' )
17   tocanota ( A4 )
18   fondo ( #FFE1EC8 )■
19 si ( tecla = 'j' )
20   tocanota ( B4 )
21   fondo ( #FF57938D )■
```

Principal | Ratón | Teclado

Con este código se puede tocar las 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

6.5.3. Piano colores y notas en pantalla

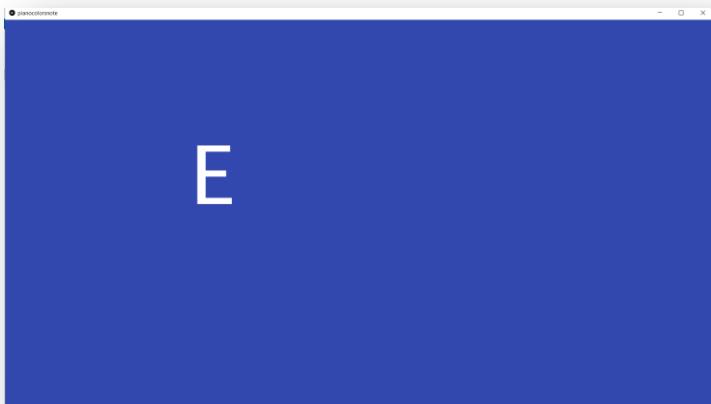
Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:

```
1 tamtexto ( 220 )
2 si ( tecla = 'a' )
3 tocanota ( C4 )
4 fondo ( #FF2AD538 ) █
5 texto ( C , 500 , 500 )
6 si ( tecla = 's' )
7 tocanota ( D4 )
8 fondo ( #FCFC72538 ) █
9 texto ( D , 500 , 500 )
10 si ( tecla = 'd' )
11 tocanota ( E4 )
12 fondo ( #FF3348AE ) █
13 texto ( E , 500 , 500 )
14 si ( tecla = 'f' )
15 tocanota ( F4 )
16 fondo ( #FFFCDB138 ) █
17 texto ( F , 500 , 500 )
18 si ( tecla = 'g' )
19 tocanota ( G4 )
20 fondo ( #FFF48C0E ) █
21 texto ( G , 500 , 500 )
22 si ( tecla = 'h' )
23 tocanota ( A4 )
24 fondo ( #FFEF1EC8 ) █
25 texto ( A , 500 , 500 )
26 si ( tecla = 'j' )
27 tocanota ( B4 )
28 fondo ( #FF68B0A0 ) █
29 texto ( B , 500 , 500 )
```

Principal | Ratón | Teclado

Con este código se puede tocar las 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

El piano al ejecutarse se vería así:



6.6. Mini Juego

A continuación se muestra un ejemplo de cómo se puede hacer un mini juego en Meta_Javascript. En la pestaña **Principal** se debe agregar el siguiente código:

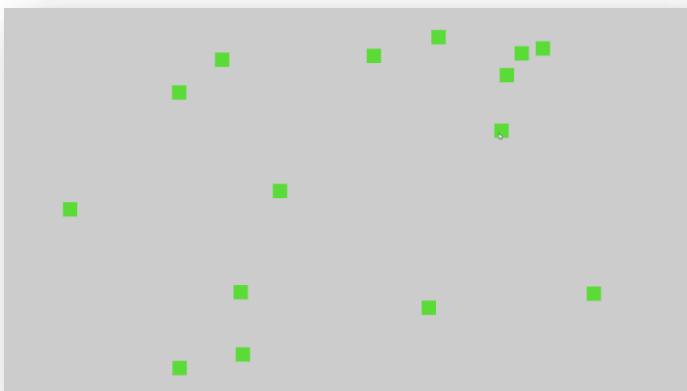
```
1 sinlinea  
2 relleno ( #FF5ADB38 )  
3 sumar ( contador , 0.1 )  
4 si ( contador > 5 )  
5 aleatorio ( px , 1900 )  
6 aleatorio ( py , 1040 )  
7 rectangulo ( px , py , 40 , 40 )  
8 asignar ( contador , 0 )
```

Y en la pestaña **Ratón** se debe agregar el siguiente código:

```
1 relleno ( 205 )  
2 elipse ( ratonX , ratonY , 100 , 100 )  
3 tocanota ( C4 )
```

El juego consiste en una serie de cuadrados verdes que van apareciendo aleatoriamente en la pantalla. El propósito del juego es tratar hacer desaparecer todos los cuadros verdes haciendo clic sobre ellos.

El juego al ejecutarse se vería algo así:



Fuentes de referencia

Cuartas, J.D. (2014). Digitópolis I: Diseño de Aplicaciones Interactivas para Creativos y Comunicadores. Bogotá: Fundación Universitaria Los Libertadores.

Cuartas, J.D. (2017). Programar el mundo en el contexto de las tecnologías libres y las culturas Hacker-Maker. Caso de estudio: Hitec Lab. (Tesis doctoral). Doctorado en Diseño y Creación. Universidad de Caldas, Manizales.

Hitec Lab (2020). Hitec Lab Homepage. Recuperado de <http://hiteclab.libertadores.edu.co/>

Hitec Lab (2020). Meta_Javascript. Recuperado de https://github.com/hiteclab/Meta_Javascript

Hitec Lab (2020). Meta_Javascript. Recuperado de https://github.com/hiteclab/Meta_Javascript

Libertadores, L. (2019). Institución Universitaria Los
Libertadores. Recuperado de
<http://www.ulibertadores.edu.co/>

Processing. (2019). Processing. Recuperado de
<http://www.processing.org/>

Lo que se busca con esta derivación de Meta_Processing es ir creando un ecosistema de

de un mismo lenguaje de programación. En la actualidad es casi indispensable aprender un lenguaje diferente para programar cada plataforma.

Lab) en la Fundación Universitaria Los Libерadores. Es Diseñador visual y doctor en Diseño y Creación de la Universidad de Caldas. Es promotor del uso desarrollo de software libre para los entornos del arte, el diseño y el entretenimiento. Su trabajo investigativo se enfoca en explorar estrategias para promover las culturas Hacker y Maker en el contexto académico universitario, buscando que las personas inexpertas descubran las oportunidades creativas que existen al aprender a programar. Desde el laboratorio Hipermédia es uno de los mentores de la iniciativa llamada: "Mujeres en tecnología", con la que se busca acercar al público femenino a las culturas Hacker y Maker y contribuir en desmitificar las tecnologías. El profesor Cuartas es el autor de la serie de tres libros llamada Digitópolis, la cual está disponible para descarga gratuita en google books. El primer libro de la serie (Digitópolis I), le permite al lector aprender a programar usando el lenguaje Processing. El segundo libro (Digitópolis II), es una guía para aprender a desarrollar videojuegos 2D con Gdevelop. Y el tercer libro (Digitópolis III), es para aquellos interesados en aprender a crear recorridos virtuales en Blender 3D.



A standard linear barcode is located at the bottom of the page, consisting of vertical black bars of varying widths on a white background.