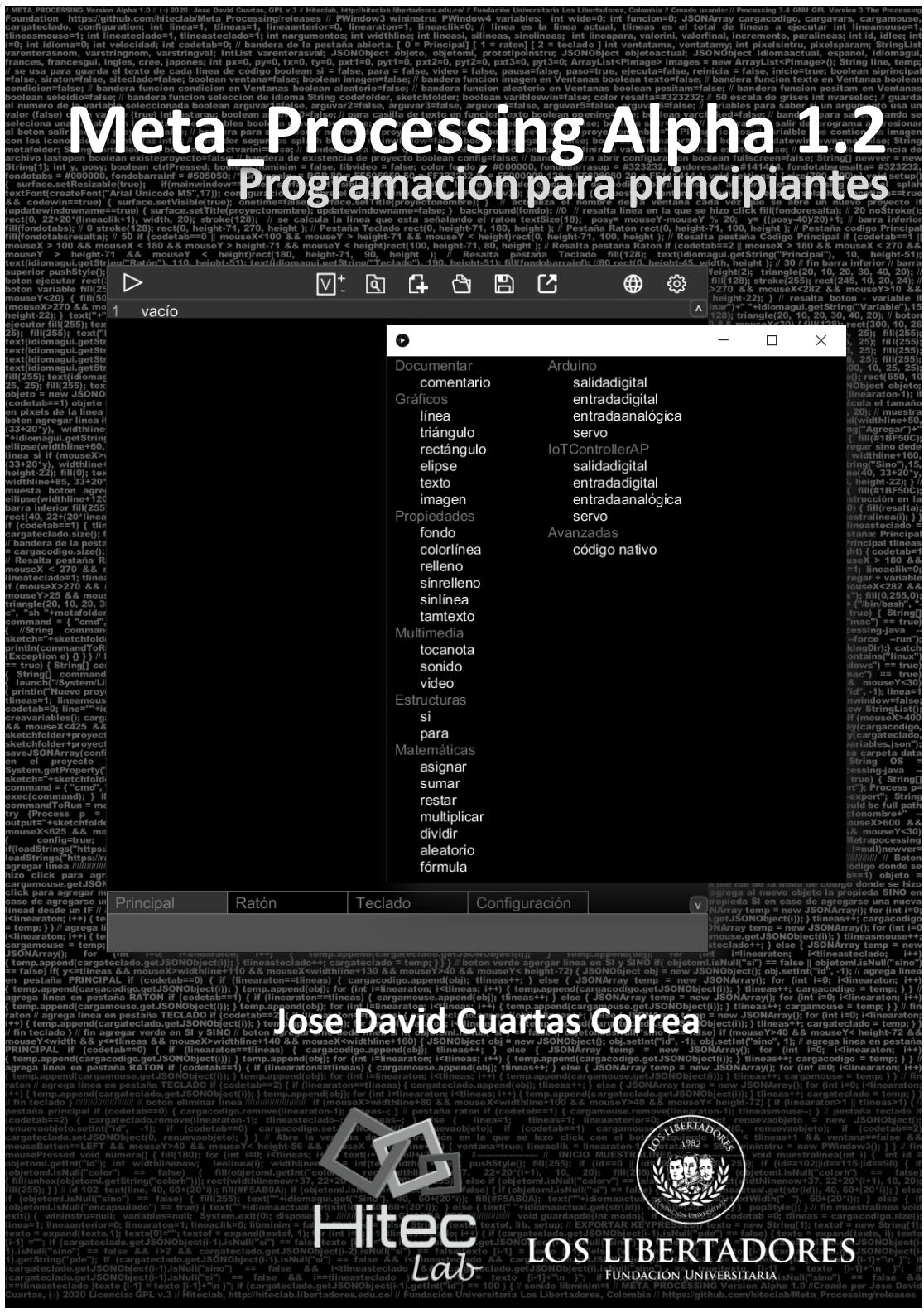


Meta Processing Alpha 1.2

Programación para principiantes



Meta_Processing Alpha 1.2

Programación para principiantes

Jose David Cuartas Correa



LOS LIBERTADORES
FUNDACIÓN UNIVERSITARIA

Meta_Processing 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_Processing, un lenguaje de meta-programación que desarrollé para el principiante de la programación. Es una iniciativa personal que es influenciada por mi trabajo como director del Laboratorio Hipermedia¹ (Hitec Lab) de la Fundación Universitaria Los Libertadores (Bogotá, Colombia).

La idea de crear Meta_Processing surge durante el desarrollo de mis estudios de 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). Allí destaque la importancia de que los diseñadores, artistas y creativos aprendan a programar, y pude presentar evidencias de la gran variedad de oportunidades creativas que este conocimiento les puede brindar a los estudiantes curiosos e inquietos.

Durante este tiempo también escribí el libro "Digitópolis I: Diseño de Aplicaciones Interactivas para Creativos y Comunicadores" el cual era una guía introductoria al lenguaje de programación Procesing. Con este libro busqué promover el interés por aprender programación en los estudiantes de diseño gráfico, publicidad, comunicación. Sin embargo para aquellos que no tenían buenas bases de inglés, se les dificultaba recordar las palabras claves del lenguaje.

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

Meta_Processing es un entorno de programación diseñado para evitar que el usuario principiante cometa errores comunes de sintaxis. Es un lenguaje de meta-programación basado en el lenguaje Processing, y todo el código creado con Meta_Processing es exportado como código Processing. Con Meta_Processing se puede escribir y leer el mismo código en diferentes idiomas, como Español, Francés, Hindi, Japonés, Italiano, Chino, Portugués e Inglés.

El concepto de meta-programación hace referencia a un programa que es capaz de escribir o manipular otros programas. El concepto Meta viene de la preposición griega que significa: "después" o "más allá" pero en este caso se usa en la acepción más contemporánea que hace referencia al prefijo "sobre". Un buen ejemplo es cuando se usa en la palabra "meta-cognición" que significaría "cognición sobre la cognición".

Así pues, Meta-Processing lo defino como un lenguaje de meta-programación que funciona sobre Processing. Es un lenguaje de más alto nivel que Processing, pero que se traduce y se ejecuta como código Procesing. En otras palabras así como Processing es una abstracción de Java, Meta_Processing es una abstracción de Processing. Así pues, Meta_Processing continúa con la tradición del Medialab del MIT y de la misma manera como John Maeda se apoyó en los hombros de Java para crear Design by numbers, y de la misma manera que Casey Reas y Ben Fry se inspiraron en Design by numbers para crear Procesing, igualmente Meta_Procesing se apoya en los hombros de Processing para ofrecer una experiencia de programación mucho más amigable con el principiante de programación.

Este lenguaje de meta-programación también surge motivado por la reflexiones hechas por Bret Victor en sus conferencias: Inventing on a Principle (2012) y Stop Drawing Dead Fish (2013), en donde Victor demuestra la urgente necesidad de construir nuevas herramientas que le permita a los creadores, explotar el potencial que tienen por ofrecernos las computadoras. Victor habla de la necesidad de alejarnos del paradigma algebraico y textual que es el más usado a la hora de programar, y propone un paradigma basado en manipulaciones geométricas. Metra_Processing busca hacer otro tipo de aproximación mezclando la metáfora de programación grafica (tipo Scratch) con la metáfora de programación basada en texto (tipo Processing), en una herramienta hibrida que toma las mejores características de ambas, para ofrecer una experiencia amigable que no aleje al usuario del paradigma predominante (que es el textual), pero que le evite algunos momentos de frustración causados por errores insignificantes de sintaxis, que comúnmente le sucede a los principiantes.

Desde hace años existen iniciativas fantásticas de lenguajes de programación para niños como los son Logo y Scratch (desarrollados en Estados Unidos por el MIT) o Pilas Bloques (desarrollado en Argentina por la iniciativa program.ar). Sin embargo Meta_Processing le apunta a otro tipo de usuarios que quieren aprender a programar sin sentir que están usando herramientas diseñadas para niños. También podría llegar a ser usado por niños y personas jóvenes que no quieren usar herramientas con interfaces de estilo infantil.

Introducción Alpha 1.1.

En esta versión se ofrece soporte para comunicarse con tarjetas Arduino usando Firmata y permite desplazar las líneas de código con el mouse. 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. 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**.

Introducción Alpha 1.2.

Para esta nueva versión de Meta_Processing se agrega la pestaña Configuración y se adicionan los idiomas Punjabi, Kannada, Bengali, Tamil, Koreano, Russo y Alemán. Se agrega soporte para comunicarse con tarjetas ESP usando la librería IoTControllerAP². Se adiciona la instrucción *para* dentro de la categoría Estructuras. Y 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

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

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_PROCESSING PRIMEROS PASOS	13
1.1. ¿Cómo abrir Meta_Procesing?	13
1.2. Ventanas que se abren al iniciar Meta_Processing	16
1.3. Elementos básicos de la interface	17
1.4. ¿Cómo seleccionar Idiomas?	18
1.5. ¿Cómo ejecutar el código?	19
1.6. ¿Cómo agregar una línea de código?.....	19
1.7. ¿Cómo eliminar una línea de código?	20
1.8. ¿Cómo agregar instrucciones?.....	20
1.9. ¿Cuál es la estructura de archivos y carpetas en Meta_Processing?	22
1.10. ¿Cómo abrir la carpeta data del proyecto actual?	24
1.11. ¿Cómo crear un nuevo proyecto?.....	25
1.12. ¿Cómo abrir un proyecto?.....	25
1.13. ¿Cómo guardar el proyecto actual?	26
1.14. ¿Cómo exportar el proyecto actual como aplicación?	26
1.15. Opciones de configuración.....	27
1.16. Funciones: Principal, Teclado, Ratón y Configuración	29
1.16.1. Principal	29
1.16.2. Ratón	30
1.16.3. Teclado	31
1.16.4. Configuración	31
1.17. Atajos de teclado: ctrl+c, ctrl+x, ctrl+v, ctrl+z	32
2. INSTRUCCIONES BÁSICAS	34
2.1. Documentar el código	35
2.2. Coordenadas de pantalla	36
2.3. Instrucciones para gráficos en pantalla	38
2.3.1. fondo	38

2.3.2. línea	39
2.3.3. rectángulo.....	39
2.3.4. elipse	40
2.3.5. triángulo	41
2.3.6. tamtexto.....	41
2.3.7. texto	42
2.3.8. imagen	43
2.3.9. colorlinea.....	43
2.3.10. relleno.....	44
2.3.11. sinrelleno.....	45
2.3.12. sin linea.....	46
2.4. Instrucciones multimedia	46
2.4.1. tocanota	46
2.4.2. sonido	47
2.4.3. video	47
2.5. Instrucciones Avanzadas	48
2.5.1. código nativo	48
3. VARIABLES, CONDICIONES Y CICLOS	50
3.1. Variables.....	50
3.1.1. ¿Cómo se mira el listado de variables?	51
3.1.2. ¿Cómo se crea una variable?	51
3.1.3. ¿Cómo se elimina una variable?	52
3.1.4. ¿Cómo se inicializa una variable?	53
3.1.5. ¿Cómo asignarle un nuevo valor a una variable?	54
3.1.6. ¿Cómo se le suma un valor a una variable?	55
3.1.7. ¿Cómo se le resta un valor a una variable?	56
3.1.8. ¿Cómo multiplicar una variable?.....	57
3.1.9. ¿Cómo dividir una variable?	58

3.1.10. ¿Cómo asignar un valor aleatorio a una variable?	59
3.1.11. ¿Cómo agregar una fórmula matemática ?	60
3.2. Condiciones	61
3.1. Ciclos	65
4. USANDO ARDUINO CON META_PROCESSING	69
4.1. Instalación de la librería Firmata en una tarjeta Arduino	69
4.2. Instrucciones Arduino	74
4.2.1. salidadigital	74
4.2.2. entradadigital	77
4.2.3. entradaanalógica	79
4.2.4. servo	81
5. USANDO ESP CON META_PROCESSING	83
5.1. Instalación de la librería IoTControllerAP en una tarjeta ESP ..	84
5.2. Instrucciones IoTControllerAP	89
5.2.1. salidadigital	89
5.2.2. entradadigital	91
5.2.3. entradaanalógica	92
5.2.4. servo	94
6. EJEMPLOS DE CÓDIGO CON META_PROCESSING	96
6.1. Ejemplo básico con el Teclado	96
6.2. Ejemplo básico con el Ratón	97
6.3. Animación	98
6.4. Piano	100
6.4.1. Piano simple	100
6.4.2. Piano pantalla de colores	101
6.4.3. Piano colores y notas en pantalla	102
6.5. Mini Juego	104
Fuentes de referencia	107

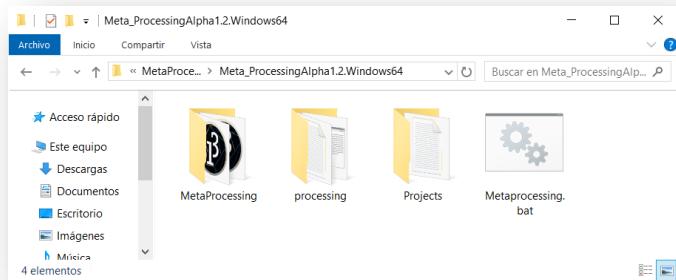
1. META_PROCESSING PRIMEROS PASOS

1.1. ¿Cómo abrir Meta_Procesing?

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

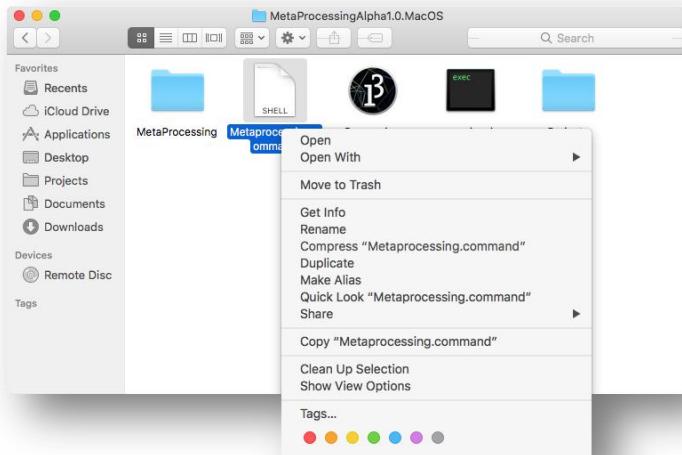
Windows

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

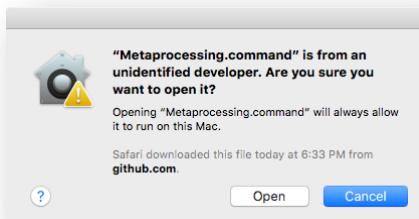


Mac OS

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

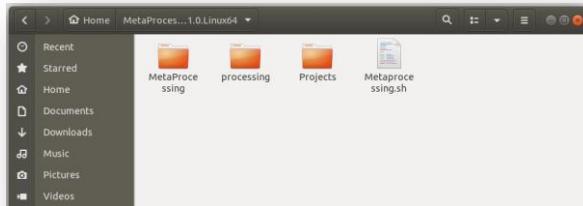


En la ventana que se abre se debe seleccionar la opción: **Open** o **Abrir**

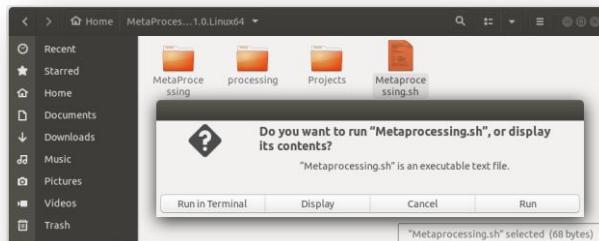


GNU/Linux

En GNU/Linux se debe hacer doble clic en el archivo **Metaprocessing.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_Processing)

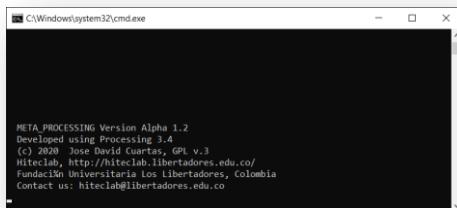


Si el script no abre al hacer doble clic sobre este, se puede ejecutar el siguiente comando en el terminal de Linux, para activar el anterior cuadro de dialogo.

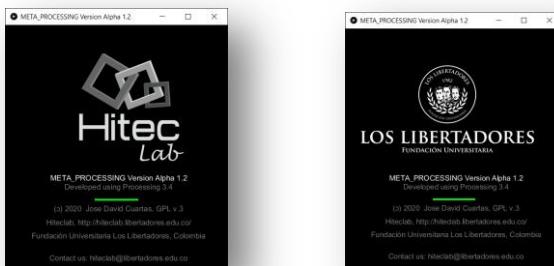
```
gsettings set org.gnome.nautilus.preferences executable-text-activation ask
```

1.2. Ventanas que se abren al iniciar Meta_Processing

Una vez se hace ejecuta el archivo Meta_Processing se abre la ventana terminal donde se pueden ver mensajes que provienen de la ventana principal de Meta_Processing.

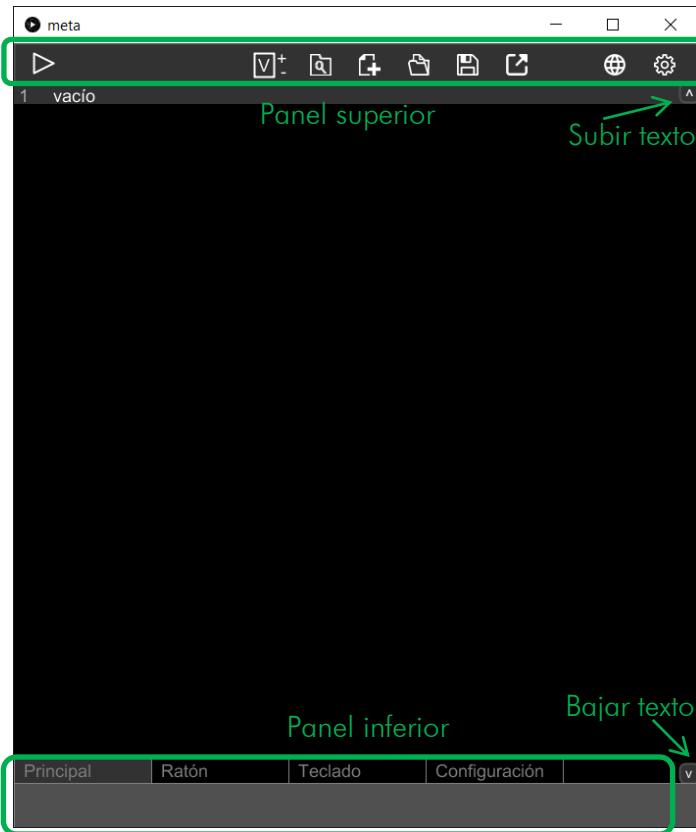


Luego se abre una segunda ventana animada de bienvenida a Meta_Processing. Al hacer clic sobre esta ventana o al cerrarla se abre la ventana principal de Meta_Processing.



1.3. Elementos básicos de la interface

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



En el panel inferior están 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_Processing** 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 ejecutar el código?

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



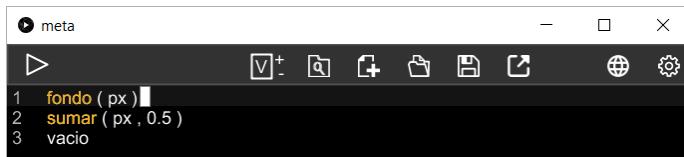
Se espera uno segundos y debe aparecer una nueva ventana en la que se ejecutará el código creado.

1.6. ¿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.

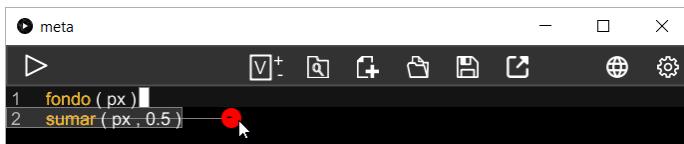


Aparecerá una nueva línea de código una vez se haga clic en el círculo verde.

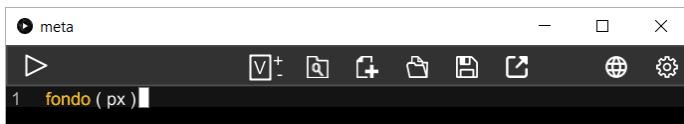


1.7. ¿Cómo eliminar una línea de código?

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.

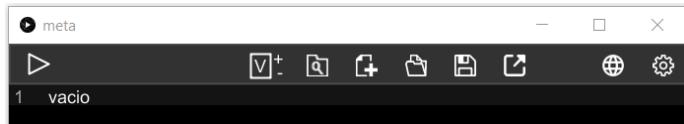


La línea desaparece una vez se hace clic.

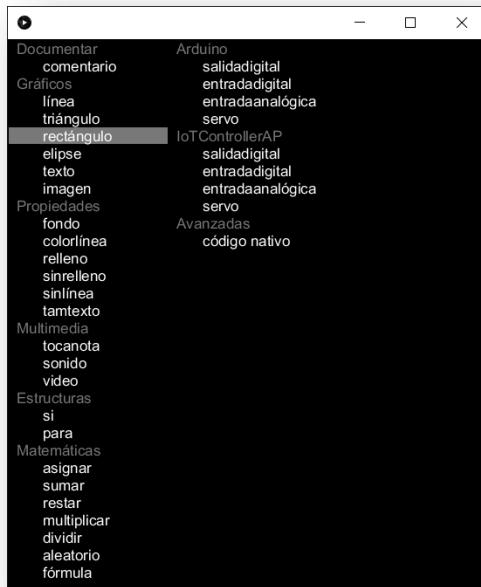


1.8. ¿Cómo agregar instrucciones?

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



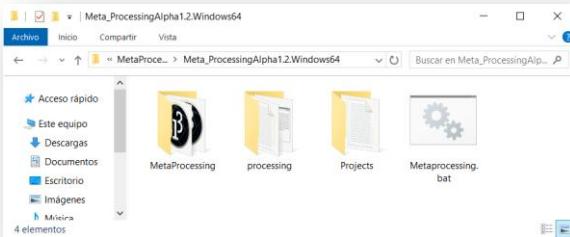
Una vez se hace esto, se abrirá una nueva ventana en donde aparecerán todas las instrucciones disponibles en Meta_Processing organizadas por categorías así:



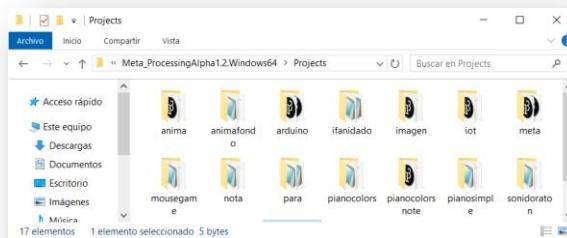
Cuando se ubica el cursor del ratón sobre alguna de estas instrucciones se resalta la instrucción, es este ejemplo se puede ver cómo se resalta la instrucción **rectángulo**. Al hacer clic se abrirá una nueva ventana en la que se podrán ingresar las propiedades de cada instrucción. La descripción de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

1.9. ¿Cuál es la estructura de archivos y carpetas en Meta_Processing?

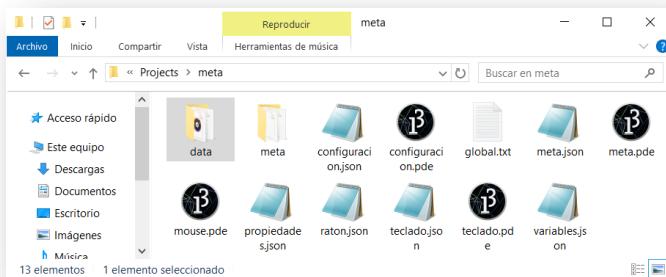
Dentro de la carpeta de **Meta_Processing** se encuentra el archivo que ejecuta **Meta_Processing** y tres subcarpetas. La carpeta **MetaProcessing** contiene los archivos que le permiten funcionar. En la carpeta **processing** hay una distribución de este lenguaje que se usa para compilar y ejecutar cada uno de los proyectos que sean creados por el usuario.



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

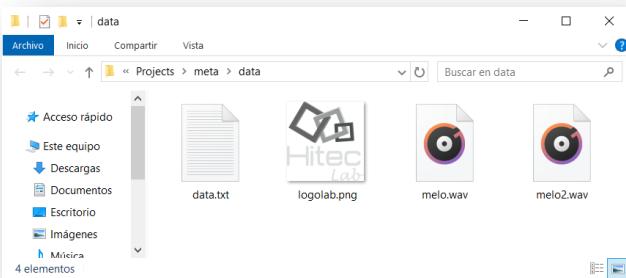


En la carpeta de cada proyecto se encuentran archivos .json y archivos .pde. Los .json contienen las instrucciones en lenguaje Meta_Processing y los archivos .pde contienen el código processing, y son generados cada vez se hace clic en el ícono ejecutar.



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.

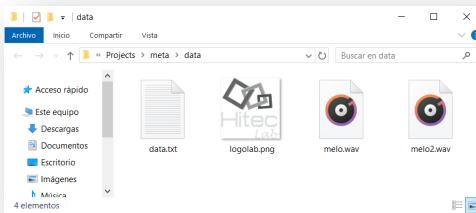


1.10. ¿Cómo abrir la carpeta data del proyecto actual?

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



1.11. ¿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.

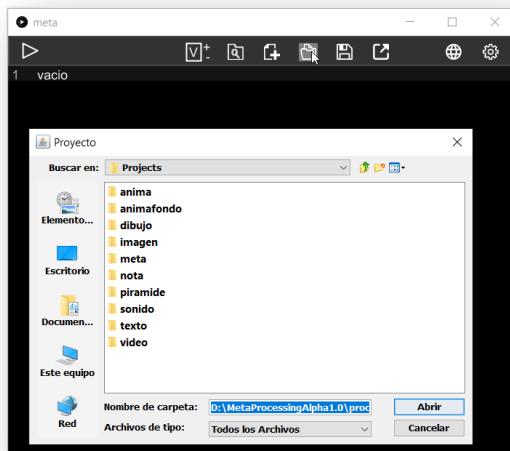


1.12. ¿Cómo abrir un proyecto?

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.



1.13. ¿Cómo guardar el proyecto actual?

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



1.14. ¿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.



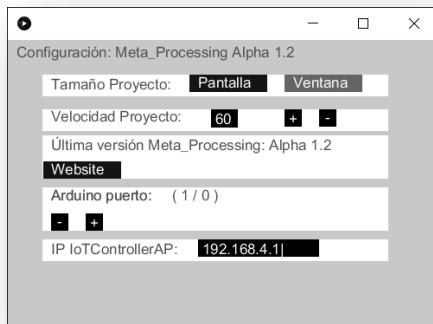
La aplicación se guarda en la subcarpeta llamada **app** dentro de la carpeta del proyecto actual.

1.15. Opciones de configuración

Para cambiar las opciones de configuración haga clic en el ícono **configuración** en la barra superior.



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



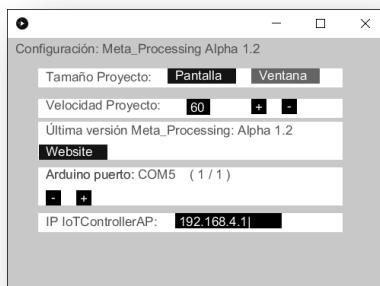
La opción **Tamaño Proyecto** tiene dos botones, **Pantalla** para hacer que la aplicación se ejecute en pantalla

completa o el botón **Ventana** para que la aplicación se abra en una ventana, a la cual se le puede cambiar el tamaño manualmente.

La opción **Velocidad** permite cambiar la velocidad del proyecto (cuadros por segundo).

La opción **Nuevo Metaprocessing** permite revisar última versión de 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 al sitio web de Meta_Processing.

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)



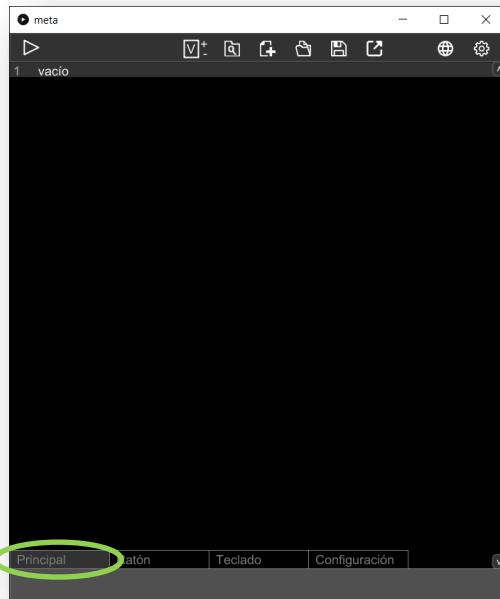
La opción **IP IoTControllerAP** permite cambiar la dirección IP de la tarjeta ESP corriendo IoTControllerAP.

1.16. Funciones: Principal, Teclado, Ratón y Configuración

Para escribir el código en Meta_Processing se pueden usar cuatro (4) funciones: Principal, Ratón, Teclado y Configuración, cada una de ellas se selecciona haciendo clic en su pestaña correspondiente.

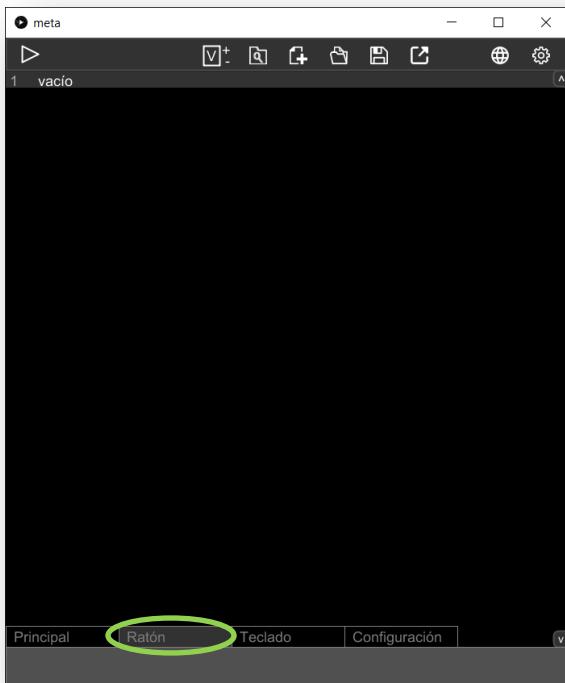
1.16.1. Principal

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



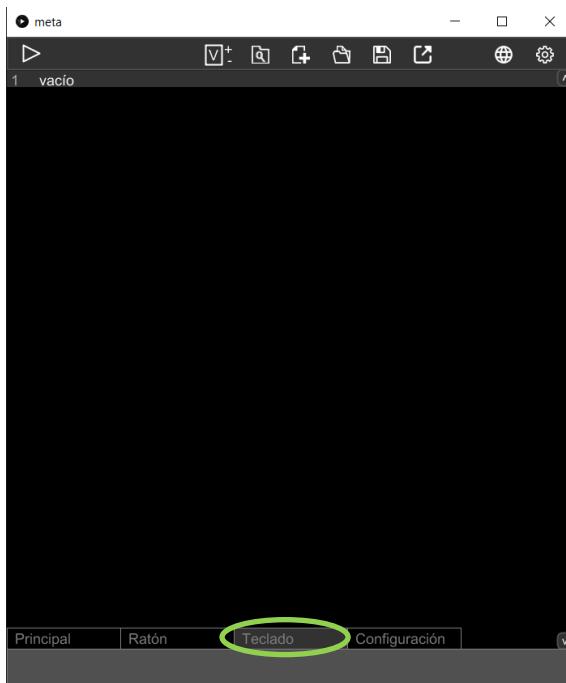
1.16.2. Ratón

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



1.16.3. Teclado

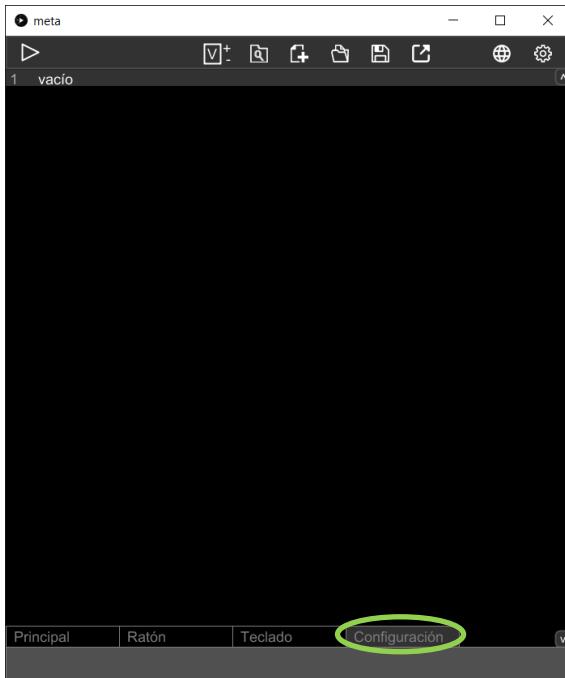
El código que se escribe en la pestaña **Teclado** se ejecuta en el momento que se presiona cualquier tecla.



1.16.4. 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).



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

Para esta versión Meta_Processing 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.

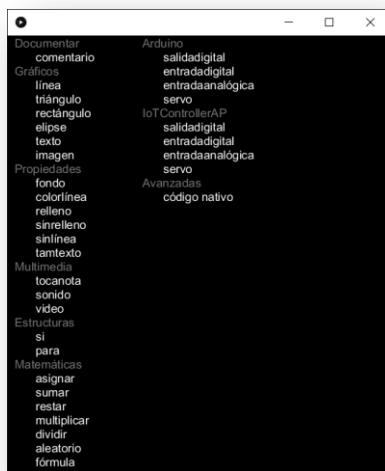
El atajo **ctrl+z** permite deshacer eliminar línea. Si se elimina una línea por equivocación, se puede recuperar la línea borrada presionando **ctrl+z**. Solo aplica para la última línea borrada.

2. INSTRUCCIONES BÁSICAS

En esta sección se abordarán las instrucciones básicas para programar con Meta_Processing. 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_Processing 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**.



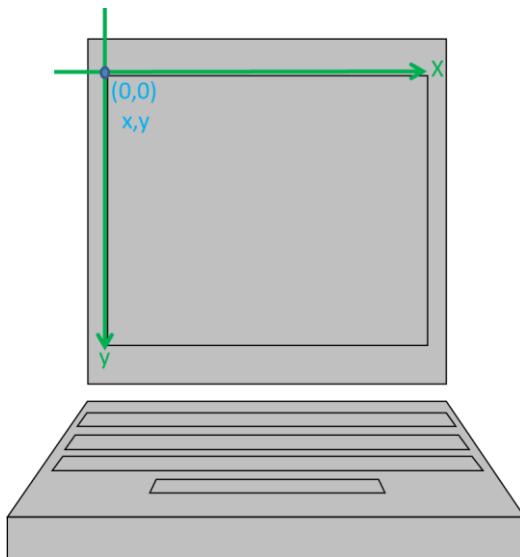
Entonces se abrirá una nueva ventana en la cual se escribe el comentario y se hace clic en **aplicar**.



2.2. Coordenadas de pantalla

Para poder hacer gráficos en pantalla es necesario primero conocer cómo funcionan las coordenadas de pantalla en Meta_Processing. 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.

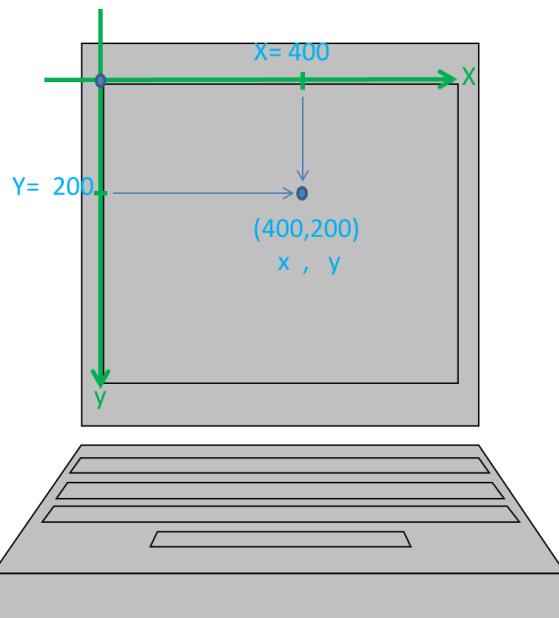
Como se puede observar en la gráfica 1, la esquina superior izquierda de la pantalla es el punto de origen del sistema de coordenadas. Este punto es la posición 0 en el eje X y la posición 0 en el eje Y. Las coordenadas siempre se organizan primero el valor en el eje X, luego una coma y después el valor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



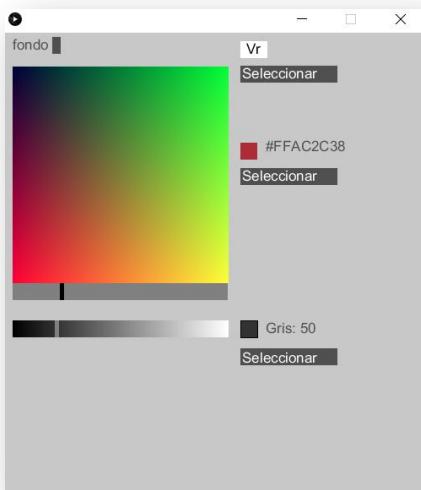
Gráfica 2 Punto en la posición 400 en X y 200 en Y de la pantalla

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 trazar 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.



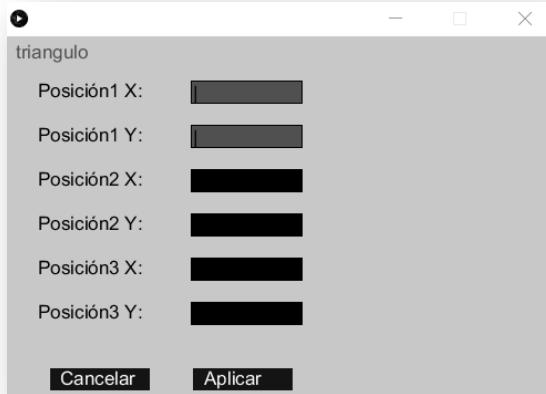
2.3.4. elipse

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

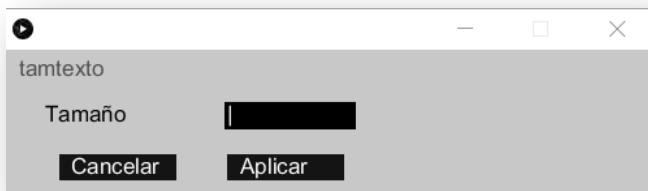
La instrucción **triángulo** sirve para dibujar un triángulo en pantalla. Para dibujar cualquier triángulo se requiere definir los tres puntos correspondientes a cada una de sus esquinas. Para usar esta instrucción se debe definir en las dos primeras casillas la posición del primer punto, en las dos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

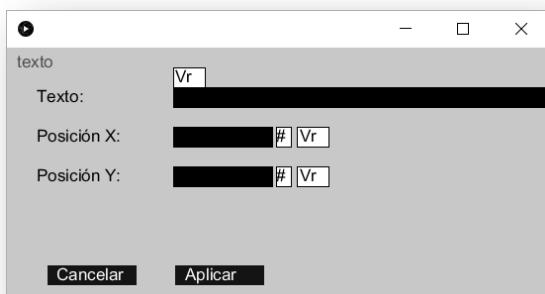
La instrucción **tamtexto** sirve para definir el tamaño de la fuente a la hora de mostrar un texto en pantalla. Para usar esta instrucción se debe llenar la casilla de tamaño con el número correspondiente al tamaño de fuente que se desea aplicar.

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. Si se quiere mostrar el contenido de una variable se puede hacer clic en el botón vr que encuentra sobre la primera casilla.



2.3.8. **imagen**

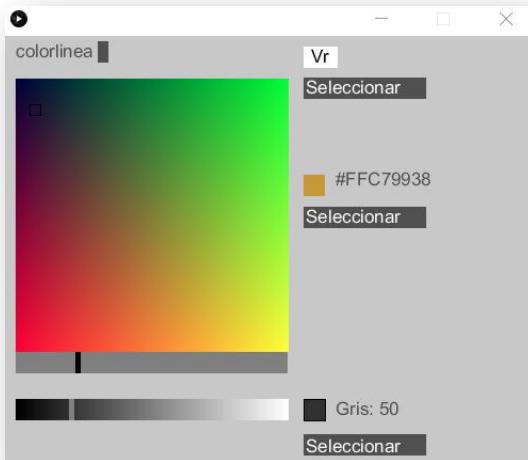
La instrucción **imagen** sirve para mostrar una imagen en pantalla. Para usar esta instrucción primero se debe guardar dentro de la carpeta **data** del proyecto la imagen que se quiere usar, luego se debe seleccionar la imagen que fue guardada en la carpeta data y por último definir en las dos siguientes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



2.3.9. **colorlinea**

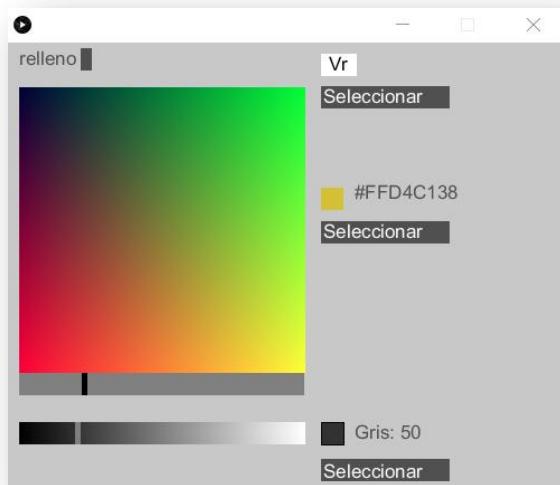
La instrucción **colorlinea** sirve para definir el color de las líneas y color del borde de las figuraras rectángulo, elipse y triángulo. Para asignar el color de línea usando esta instrucción se puede, utilizar el selector de color, o el selector de escala de grises y hacer clic en el botón **Aplicar**.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



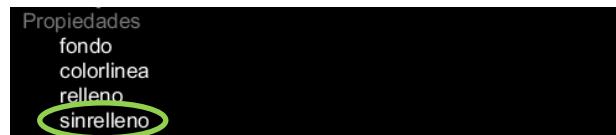
2.3.10. **relleno**

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.



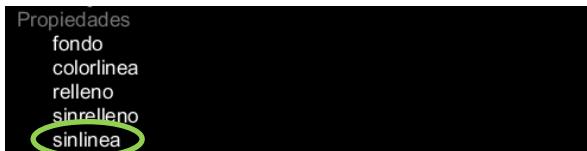
2.3.11. **sinrelleno**

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.

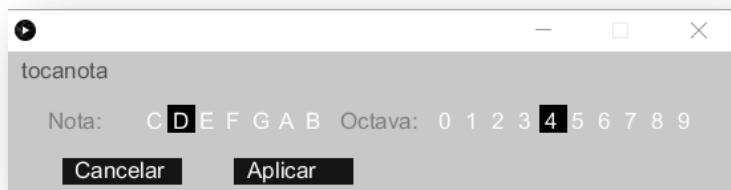


2.4. Instrucciones multimedia

Entre algunas de las instrucciones multimedia que se puede usar con Meta_Processing 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**.



2.4.3. video

La instrucción **video** sirve para reproducir en pantalla un archivo de video en formato mov, avi o mpg. Para usar esta instrucción primero se debe guardar dentro de

la carpeta **data** del proyecto, el archivo de video que se quiere usar, luego se debe seleccionar el archivo que fue guardado en la carpeta data y por último hacer clic en el botón **Aplicar**.



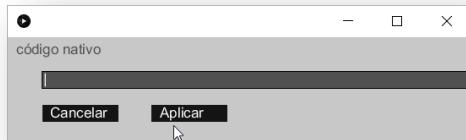
2.5. Instrucciones Avanzadas

Por el momento la única instrucción avanzada disponible para Meta_Processing es: **código nativo**. A continuación se explicará cómo usarla.

2.5.1. código nativo

La instrucción **código nativo** sirve para agregar líneas de código que no están disponibles con Meta_Processing, pero que son nativas de los lenguajes Processing o Java. Para usar esta instrucción se debe escribir en la primera casilla, la línea de código que se

quiere agregar y luego se debe hacer clic en el botón **Aplicar**.



Un ejemplo puede ser usar la instrucción de Processing llamada **println** que permite mostrar texto en el terminal. Si se agregaría la expresión:

```
println(mouseX);
```

Entonces se mostraría en la ventana terminal de Meta_Processing la posición actual del ratón en el eje x.

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_Processing se hay dos tipos de variables: Las variables del sistema y las variables creadas por el usuario.

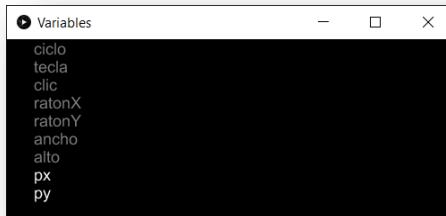
Las variables del sistema son: **ciclo**, **tecla**, **click**, **ratonX**, **ratonY**, **ancho** y **alto**. La variable **ciclo** se usa para contar el número de ciclos dentro de la estructura repetitiva `para`. La variable **tecla** almacena el valor de la última tecla presionada en el teclado. La 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**.

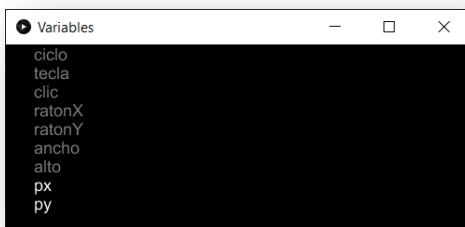


3.1.3. ¿Cómo se elimina una variable?

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.

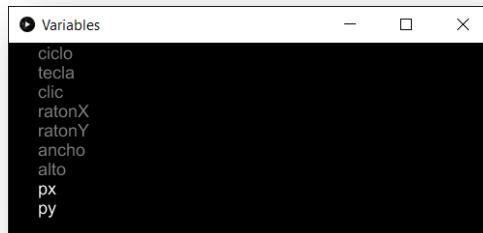


3.1.4. ¿Cómo se inicializa una variable?

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.

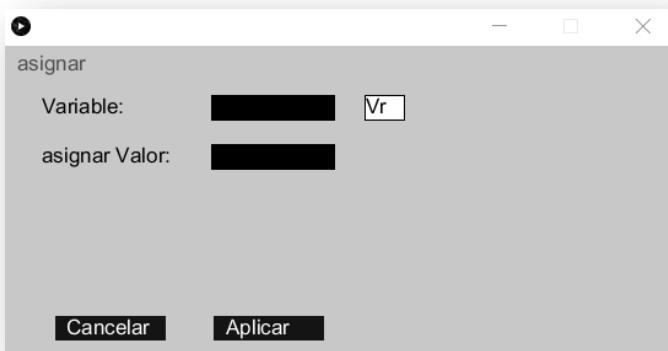


3.1.5. ¿Cómo asignarle un nuevo valor a una variable?

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.

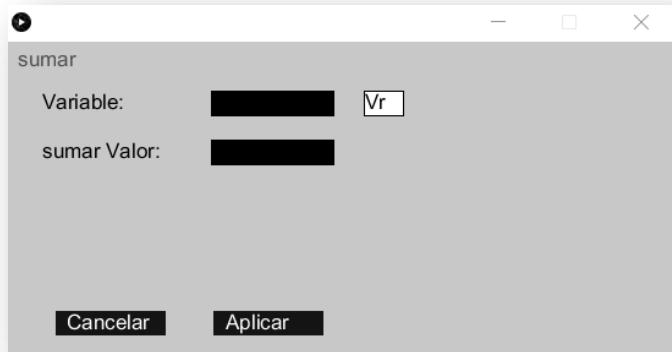


3.1.6. ¿Cómo se le suma un valor a una variable?

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.

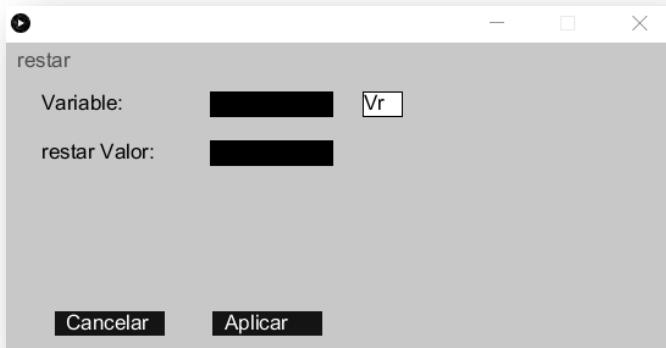


3.1.7. ¿Cómo se le resta un valor a una variable?

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.

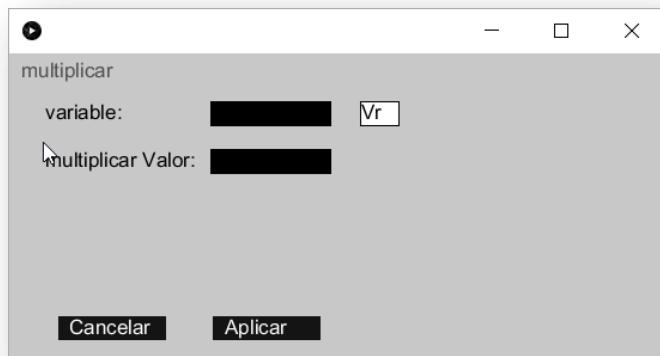


3.1.8. ¿Cómo multiplicar una variable?

Dentro del código se puede multiplicar una variable por un valor, para ello se debe agregar la instrucción **multiplicar** 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 que se le quiere multiplicar el valor y en la segunda casilla se escribe el valor por el que se va a multiplicar.

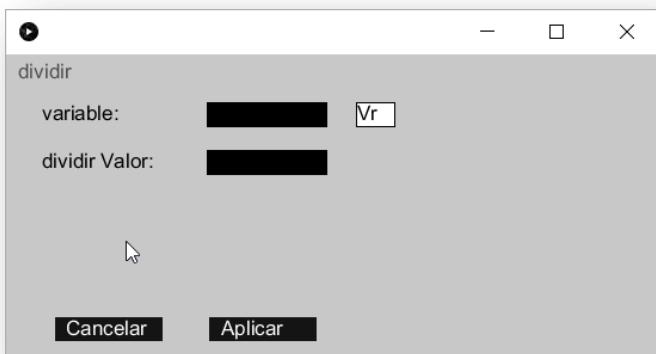


3.1.9. ¿Cómo dividir una variable?

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

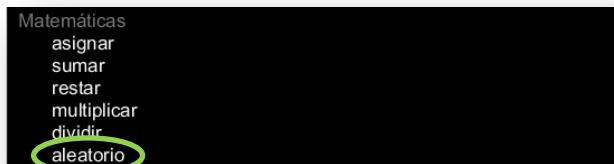


Luego en la ventana que se abre, en la primera casilla se selecciona la variable que se quiere dividir y en la segunda casilla se escribe el valor por el que se va a dividir la variable.

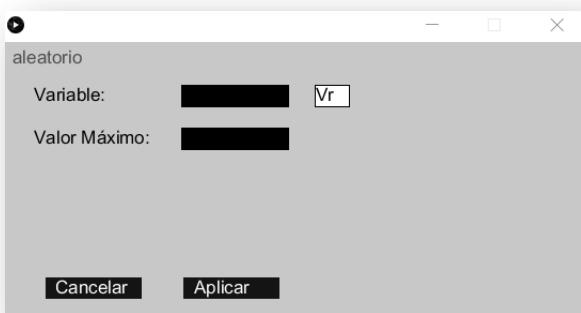


3.1.10. ¿Cómo asignar un valor aleatorio a una variable?

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.

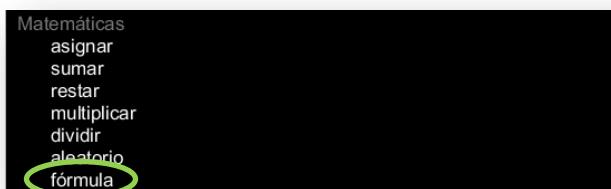


Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere asignar el valor aleatorio y en la segunda casilla se escribe el valor máximo que se podría generar aleatoriamente. De manera que se generaría un valor aleatorio entre 0 y el valor máximo que se le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



3.1.11. ¿Cómo agregar una fórmula matemática ?

Dentro del código se puede agregar una fórmula matemática para hacer operaciones más avanzadas, para ello se debe agregar la instrucción **fórmula** que se encuentra dentro de la categoría **Matemáticas** en la ventana de agregar instrucción.



Luego en la ventana que se abre, se debe escribir en la primera casilla, la fórmula que se quiere agregar y luego se debe hacer clic en el botón **Aplicar**.



Un ejemplo podría ser calcular el promedio entre tres valores:

```
promedio = (valor1 + valor2 + valor3)/3;
```

Entonces en la variable promedio se almacenaría el resultado de esta operación matemática.

3.2. Condiciones

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

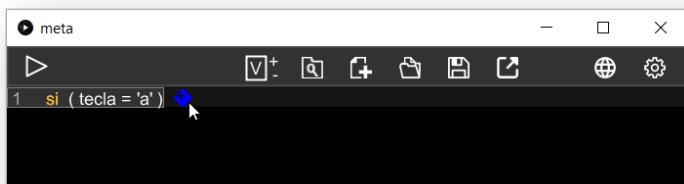
Para agregar una condición se debe seleccionar la instrucción **si** que se encuentra dentro de la categoría Estructuras en la ventana de agregar instrucción.



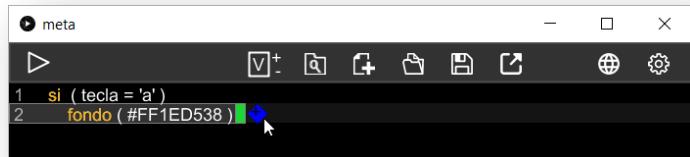
Luego en la ventana que se abre en la primera casilla se puede escribir un valor o seleccionar una variable. Luego se debe seleccionar un operador, el cual puede ser igual (=), menor que (<), mayor que (>) o diferente (!=). Y en la segunda casilla se puede escribir un valor, seleccionar una variable, elegir una tecla o elegir uno de los botones del ratón.



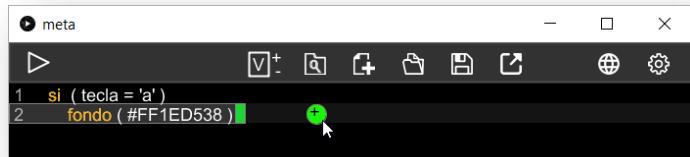
Una vez se hace clic en el botón **Aplicar** se podrá ver la condición en la ventana de código de Meta_Processing. Para agregar una instrucción dentro de la condición, se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.



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.

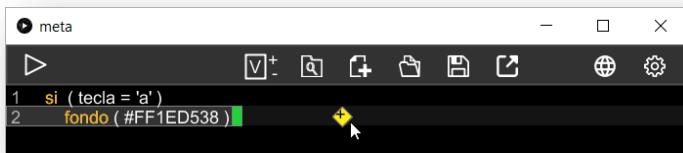


Si una vez se termina de agregar las instrucciones dentro de la condición, lo que se quiere es agregar otra línea de código pero por fuera de la condición, entonces se debe mover el cursor del ratón hasta que aparezca un circulo verde con el carácter más (+) en su interior.

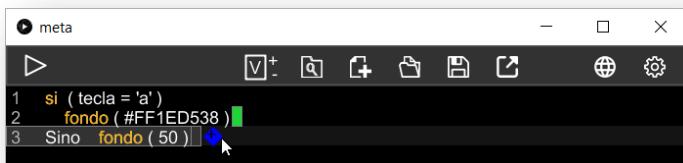


Si por el contrario lo que se quiere es agregar una línea para el caso cuando no se cumpla dicha condición entonces se podrán agregar líneas dentro del sino. Para

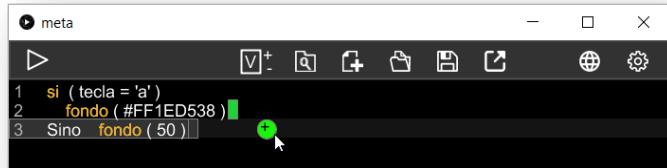
esto se debe mover el cursor del ratón hasta que aparezca un rombo amarillo con el carácter más (+) en su interior.



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.

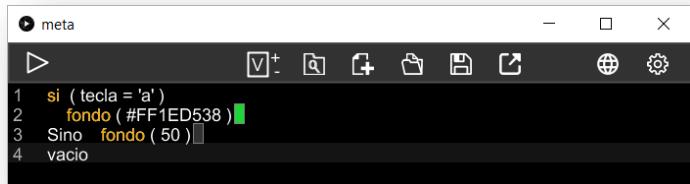


Si por el contrario se quiere es agregar otra línea de código pero por fuera del **Sino**, entonces se debe mover el cursor del ratón hasta que aparezca un circulo verde con el carácter más (+) en su interior.



```
meta
> V+ < < + < < < < <
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
```

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



```
meta
> V+ < < + < < < <
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
4 vacio
```

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 cuadrícula, 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 se vea 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.



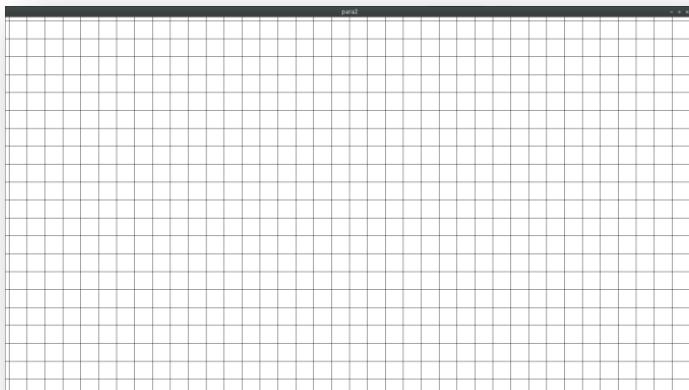
Luego en la ventana que se abre en la primera casilla se puede escribir el valor inicial de la variable del sistema llamada **ciclo**. Debajo de este casilla se puede seleccionar el operador para comparar el valor que contiene la variable ciclo con un valor que se debe agregar en la segunda casilla.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

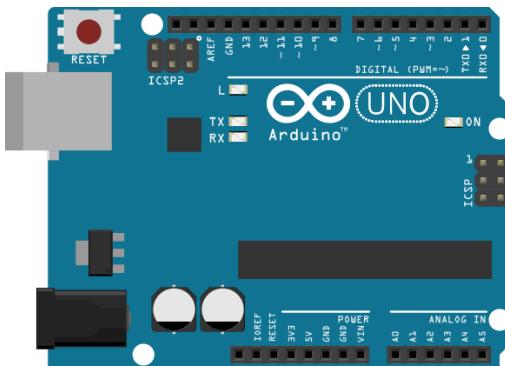
```
1 fondo ( 255 )
2 para ( desde 10 ; mientras ciclo < 1920 ; Incremento 50 )
3   línea ( ciclo , 0 , ciclo , alto )
4   para ( desde 10 ; mientras ciclo < 1080 ; Incremento 50 )
5     línea ( 0 , ciclo , ancho , ciclo )
```

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



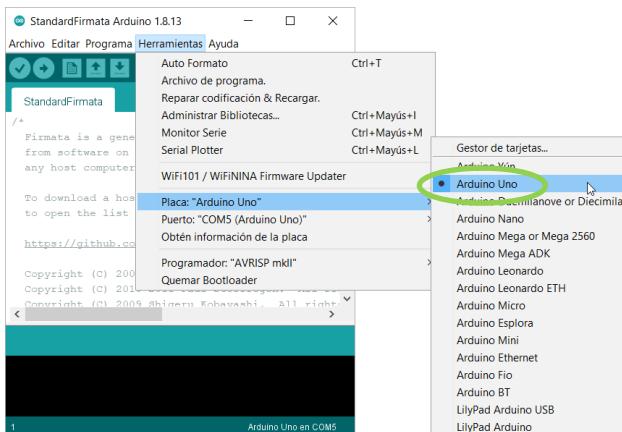
4. USANDO ARDUINO CON META_PROCESSING

Para usar una tarjeta Arduino con Meta_Processing 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.

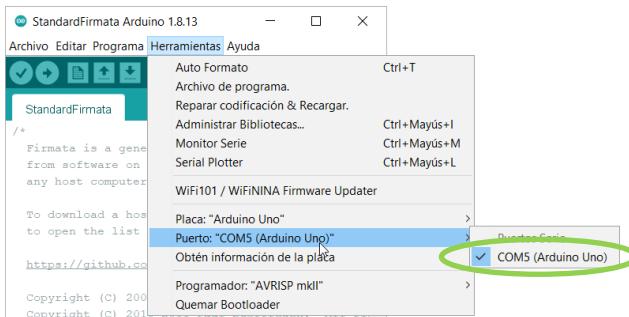


4.1. Instalación de la librería Firmata en una tarjeta Arduino

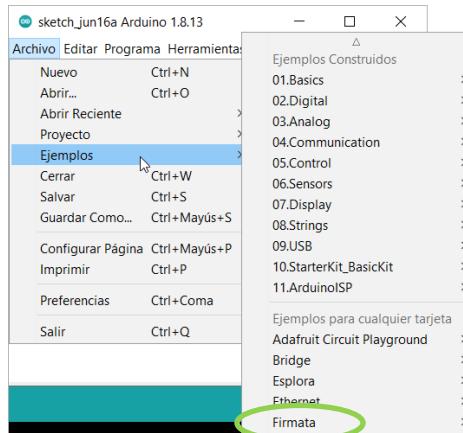
Para cargar la librería Firmata en la tarjeta el primer paso es abrir el IDE de Arduino y elegir la tarjeta Arduino a la que se va a cargar la librería Firmata.



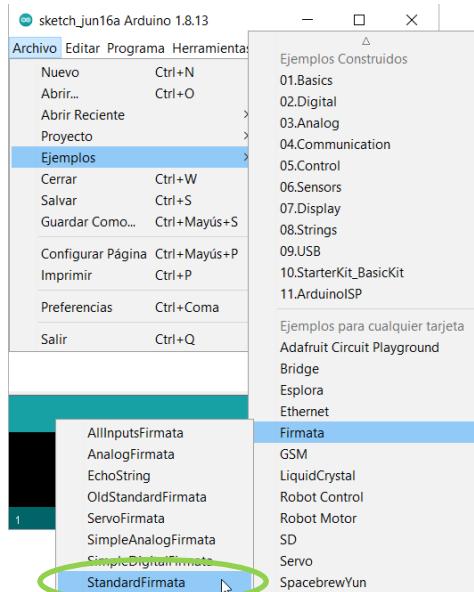
Luego se debe elegir el puerto al que está conectada la tarjeta Arduino.



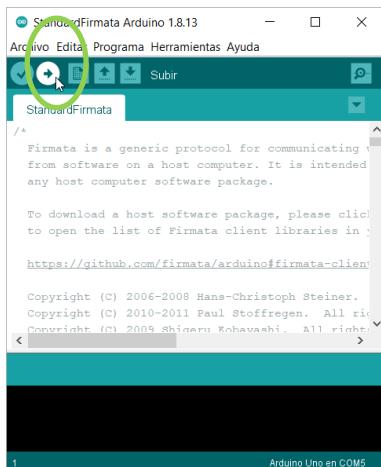
Luego se debe abrir los **ejemplos** de la librería Firmata, los cuales se acceden desde del menú **archivo**.



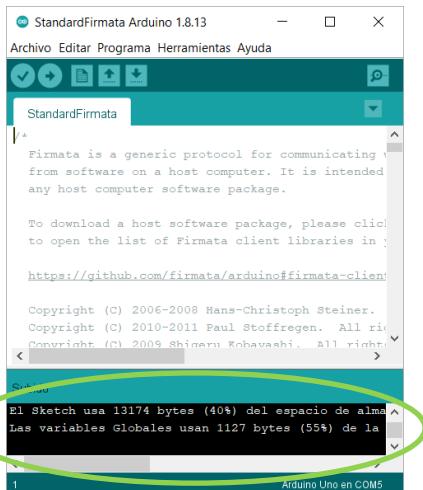
Después se debe hacer clic en la opción **Firmata** para ver todos los archivos de ejemplo de esta librería, y abrir el ejemplo: **StandardFirmata**



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



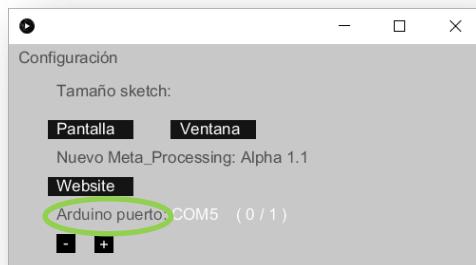
Cuando se termina de subir la librería a la tarjeta Arduino se debe ver algo así:



El siguiente paso es abrir Meta_Processing 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_Processing 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:

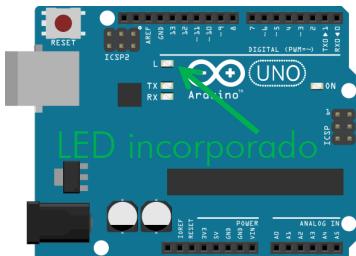


Y en la pestaña **teclado** se pondría el siguiente código:

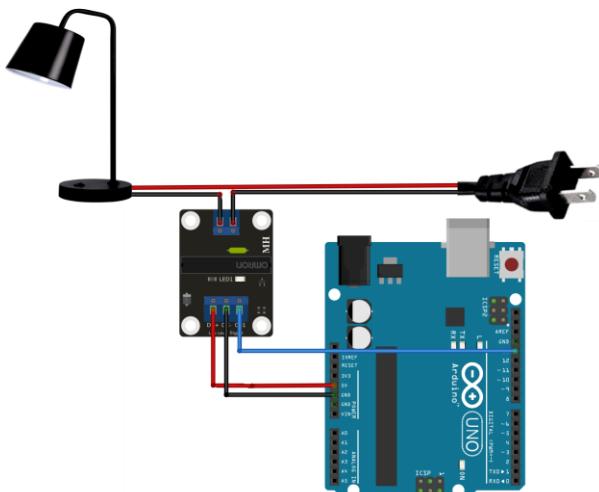


Para usar el anterior código no sería necesario construir ningún circuito con el Arduino, ya que se usaría el Led

que está incorporado en todas las tarjetas Arduino Uno y que por defecto está conectado al pin 13.

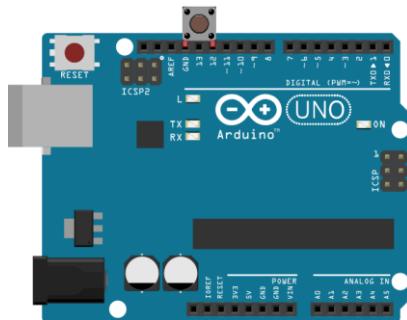


A continuación se muestra un circuito que se podría construir para prender y apagar una lámpara conectada a los 110 voltios, haciendo uso del mismo código. Para esto sería necesario un módulo relé de estado sólido para Arduino.



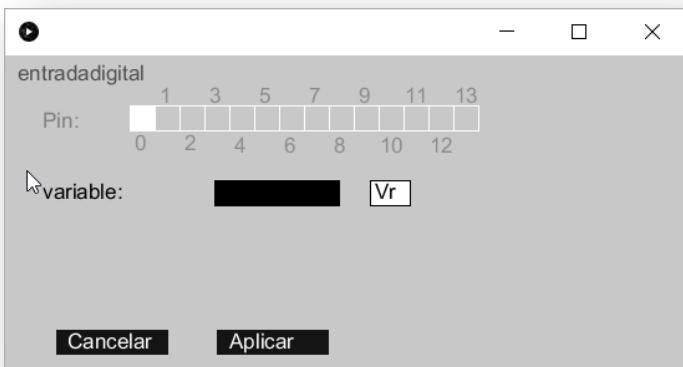
4.2.2. entradadigital

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:

```
▶ [ ] V+ [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] 
1 entradaDigital ( 12, x )
2 si ( x = 1 )
3   fondo ( #FF15ED38 ) [green square]
4 Sino fondo ( #FFDC3338 ) [red square]
```

The image shows a screenshot of a Processing code editor. At the top, there is a toolbar with various icons. Below the toolbar, the code is written in pseudocode. The code defines a digital input on pin 12 and checks if the value is 1. If true, it changes the background color to green (#FF15ED38). If false, it changes the background color to red (#FFDC3338). The code is numbered 1 through 4.

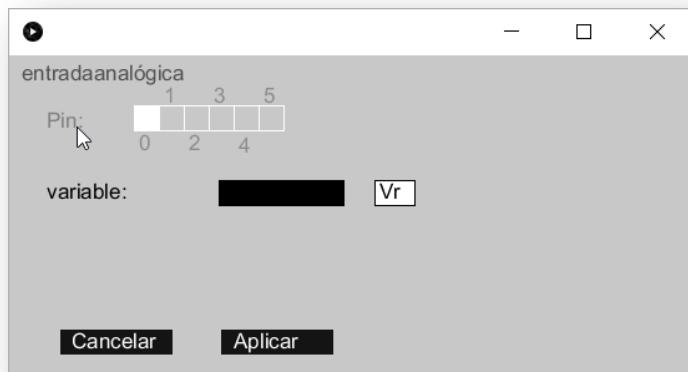
Para usar este código de ejemplo se debe tener conectado un terminal del pulsador al pin GND del Arduino y el otro terminal al pin 12 del Arduino.

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.

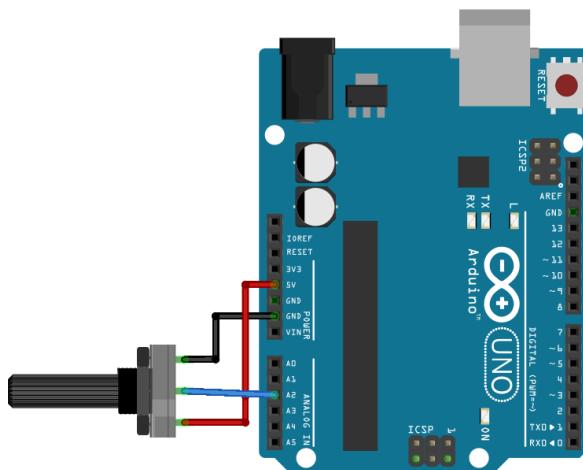
Para usar esta instrucción se debe hacer clic en la casilla correspondiente al pin analógico del Arduino que se quiere leer y en la segunda casilla se debe seleccionar la variable que va a almacenar el estado del pin.



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:

```
▶ [V+] [?] [+/-] [!] [!] [!] [!] [!] [!]
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.



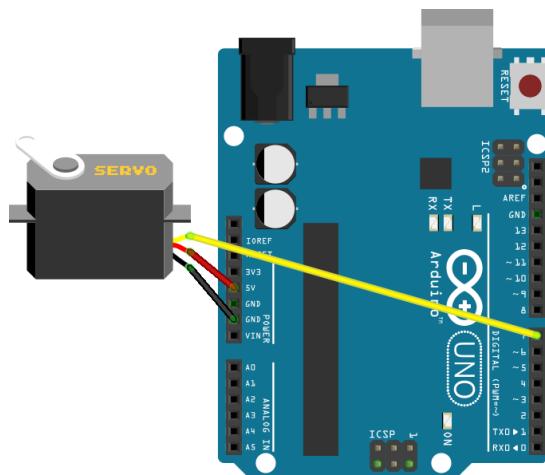
Si se quisiera crear un proyecto con un servomotor conectado al pin digital 7 para que gire en un sentido cuando se haga clic con el ratón y que gire en el sentido contrario cuando se oprima cualquier tecla, en la pestaña **ratón** se pondría el siguiente código:



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

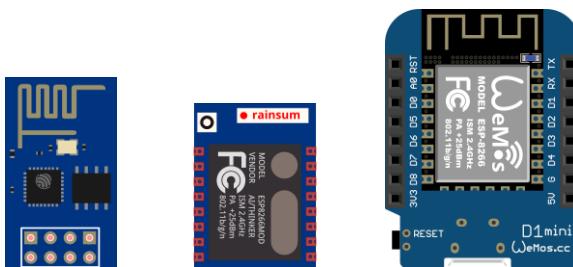


El circuito necesario para usar el anterior código de ejemplo sería el siguiente:



5. USANDO ESP CON META_PROCESSING

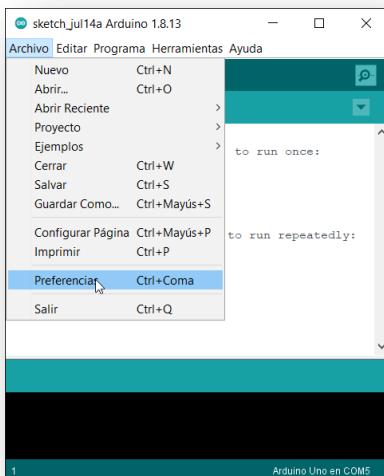
Para usar una tarjeta ESP con Meta_Processing se requiere que a la ESP se le cargue la librería IoTControllerAP³. Esta librería permite que la tarjeta ESP funcione como un Access Point con nombre *IoTController* (no requiere contraseña). De esta forma se puede establecer una comunicación WiFi directa 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.



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

5.1. Instalación de la librería IoTControllerAP en una tarjeta ESP

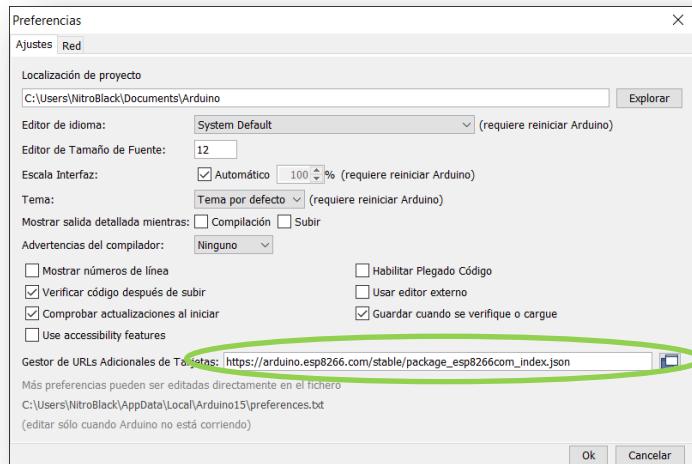
Para cargar la librería IoTControllerAP en la tarjeta ESP el primer paso es abrir el IDE de Arduino e instalar el controlador de estas tarjetas. Para esto es necesario hacer clic en la opción **Preferencias** dentro del menú principal **Archivo**.



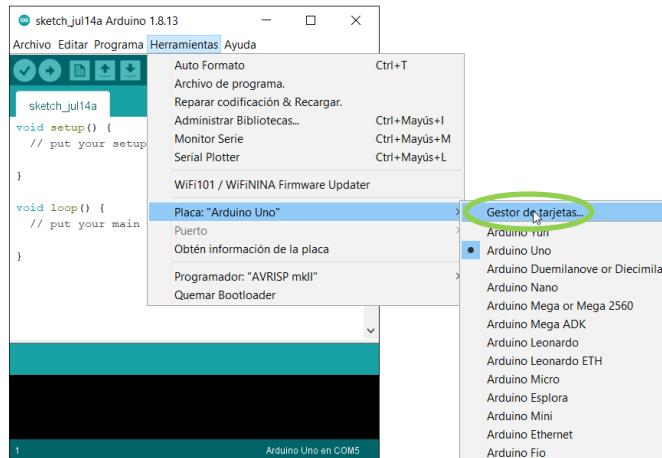
En la ventana que se abre se debe agregar en la opción **Gestor de URLs Adicionales de Tarjetas**, la línea:

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

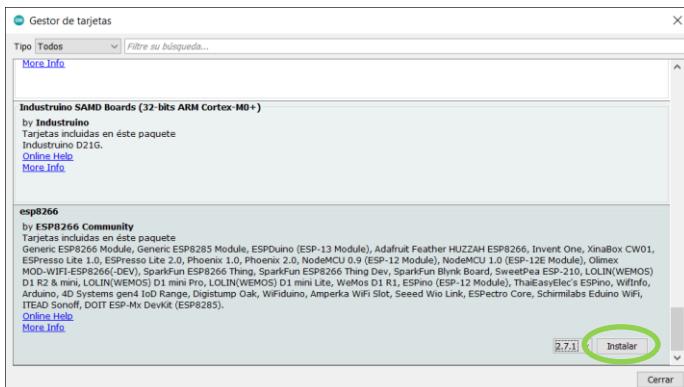
Como se muestra a continuación:



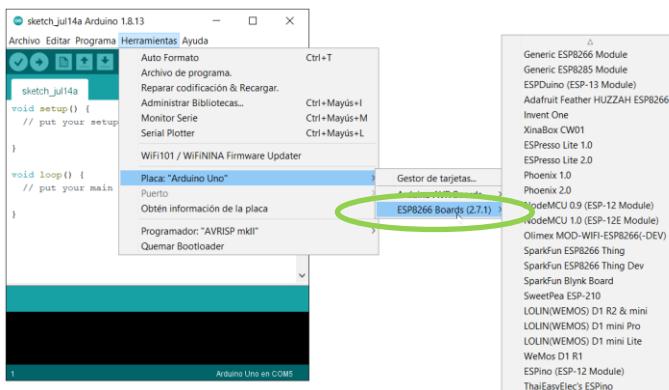
Luego se debe hacer clic en la opción **Placa** dentro del menú **Herramientas**, en el menú que se despliega se debe hacer clic en la opción **Gestor de tarjetas**:



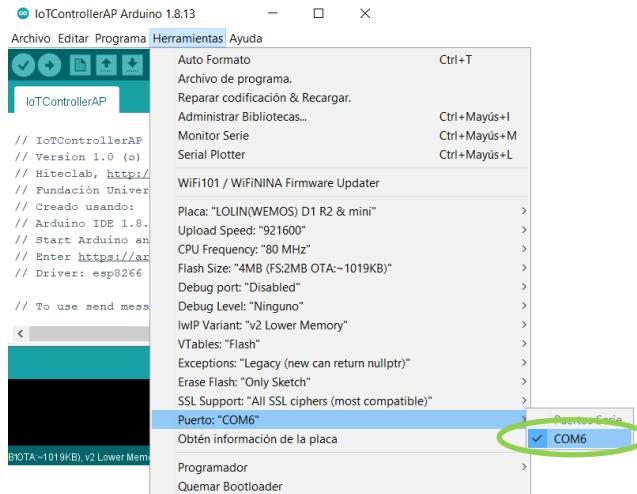
En la ventana que se abre se debe buscar el controlador para la tarjeta **esp8266**, seleccionar la versión **2.7.1** y hacer clic en el botón **Instalar**.



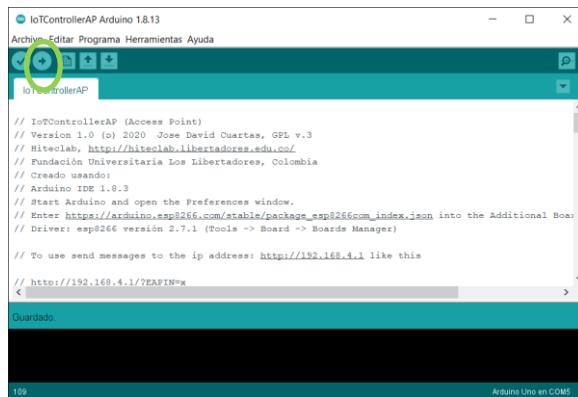
Una vez hecho esto se puede elegir la tarjeta ESP a la que se va a cargar la librería IoTControllerAP.



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



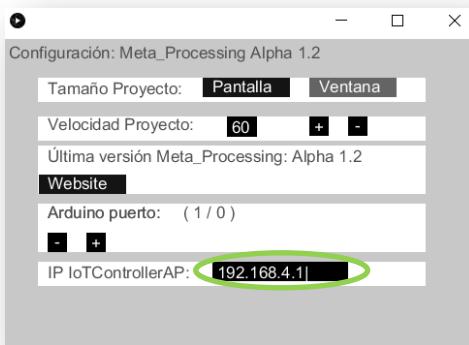
Lo siguiente es descargar la librería IoTControllerAP desde <https://github.com/hiteclab/IoTControllerAP>, luego se debe abrir el código **IoTControllerAP.ino**, y hacer clic en el ícono **subir**.



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_Processing 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**

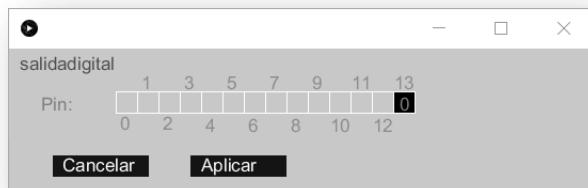
5.2. Instrucciones IoTControllerAP

Las instrucciones de IoTControllerAP que se puede usar con Meta_Processing 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
```

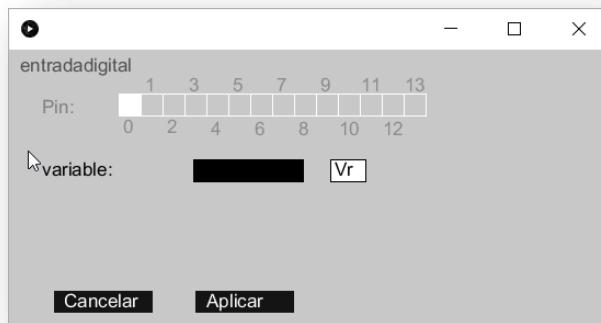
Para usar el anterior código sería necesario conectar un LED al pin 12 de la tarjeta ESP.

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.



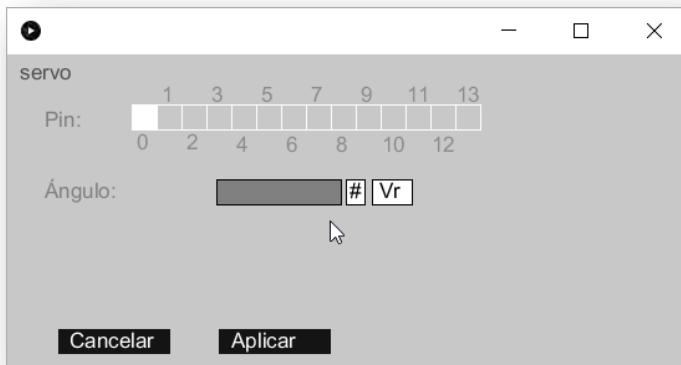
Si se quisiera crear un proyecto con un potenciómetro conectado al pin analógico 0 y que al girarse cambie el color de fondo, en la pestaña **principal** se pondría el siguiente código:

```
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:

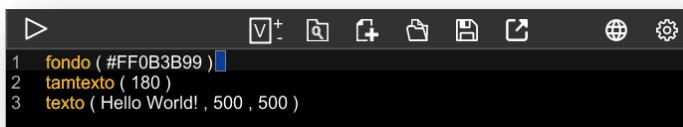
```
1 servo ( 4, 180 ) IoTControllerAP
```

6. EJEMPLOS DE CÓDIGO CON META_PROCESSING

A continuación se presentan una serie de ejemplos de cómo usar Meta_Processing. El primero de ellos nos permite experimentar con la función teclado, el segundo nos permite dibujar círculos en pantalla al presionar el ratón. El tercero es un ejemplo de cómo se puede crear una animación. El cuarto nos muestra tres formas de programar un piano. Y el último nos muestra cómo crear un sencillo mini juego.

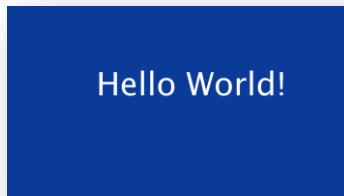
6.1. Ejemplo básico con el Teclado

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.



6.2. 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  ellipse ( ratonX , ratonY , 80 , 80 )
```

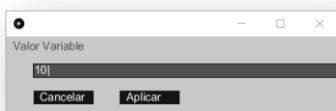
The screenshot shows the Processing IDE interface. At the top, there's a toolbar with various icons: play, stop, run, save, and settings. Below the toolbar is a menu bar with options like File, Edit, Sketch, Help, and View. The main area contains three lines of code:
1 sinlínea
2 relleno (#FFFDD738)
3 ellipse (ratonX , ratonY , 80 , 80)
The code uses the 'relleno' command to set the fill color to yellow (#FFFDD738) and the 'ellipse' command to draw a circle at the mouse position with a diameter of 80 pixels.

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

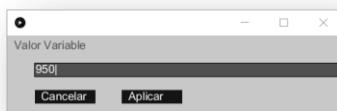


6.3. Animación

A continuación se muestra un ejemplo de cómo se puede hacer una animación en Meta_Processing. 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 ellipse ( 200 , py , 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.



6.4. Piano

A continuación se muestra tres ejemplos de cómo hacer un piano en Meta_Processing. 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.

6.4.1. Piano simple

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



The screenshot shows the Processing IDE interface. The top menu bar includes options like File, Edit, Sketch, Help, and a magnifying glass search icon. Below the menu is a toolbar with icons for selection, zoom, and file operations. The main workspace contains the following pseudocode:

```
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 )
```

At the bottom of the workspace, there are three tabs: Principal, Ratón, and Teclado. The Teclado tab is currently selected.

Con este código se puede tocar la 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.

6.4.2. Piano pantalla de colores

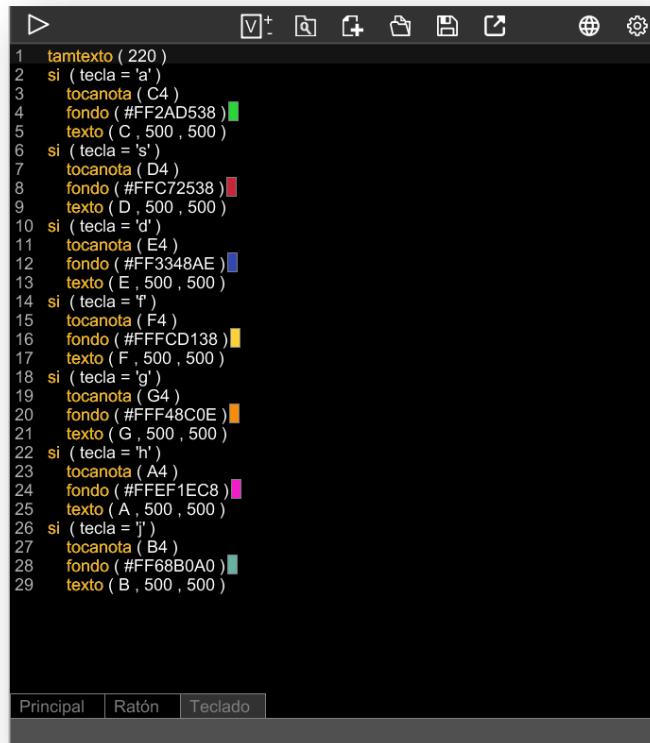
Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:

```
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 ( #FFFCDD138 ) ■
13 si ( tecla = 'g' )
14   tocanota ( G4 )
15   fondo ( #FF48C0E ) ■
16 si ( tecla = 'h' )
17   tocanota ( A4 )
18   fondo ( #FFEF1EC8 ) ■
19 si ( tecla = 'j' )
20   tocanota ( B4 )
21   fondo ( #FF57938D ) ■
```

Con este código se puede tocar la 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.

6.4.3. Piano colores y notas en pantalla

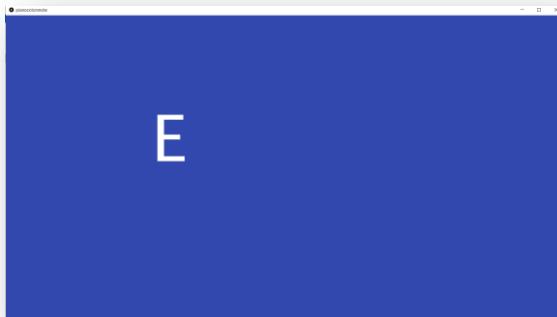
Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:



```
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(#FFC72538)
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(#FFFCB138)
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)
```

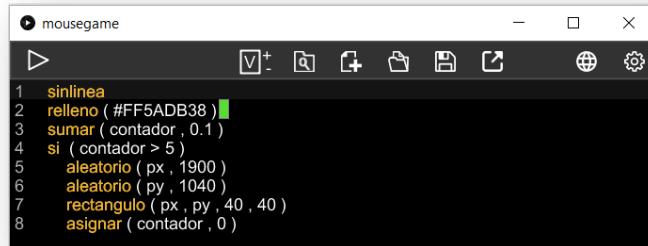
Con este código se puede tocar las 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

El piano al ejecutarse se vería así:



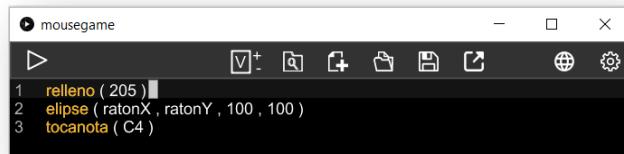
6.5. Mini Juego

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



```
mousegame
>
1 sinilinea
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 )
```

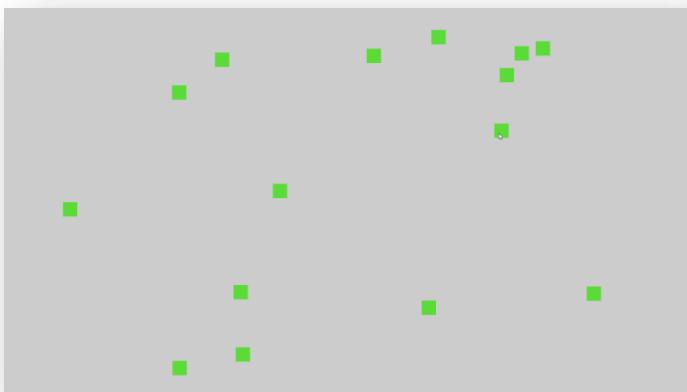
Y en la pestaña **Ratón** se debe agregar el siguiente código:



```
mousegame
>
1 relleno ( 205 )
2 ellipse ( 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

Arduino (2020). Arduino - Home. Recuperado de
<https://www.arduino.cc/>

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.

Firmata (2020). Firmata firmware for Arduino.
Recuperado de
<https://github.com/firmata/arduino>

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

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

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

Maeda, J. (1999). Design by numbers. Cambridge,
Mass: MIT Press.

MIT Medialab (20203). Design by numbers.
Recuperado de <https://dbn.media.mit.edu/>

MIT Medialab (2020). Scratch - Imagine, Program,
Share. Recuperado de <https://scratch.mit.edu/>

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

Program.ar (2020). Pilas Bloques. Recuperado de
<http://program.ar/pilas-bloques-secundaria/>

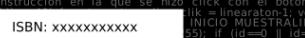
Victor, B. (2012). Bret Victor - Inventing on a Principle
[Archivo de video]. Recuperado de
<https://www.youtube.com/watch?v=a-OyoVcbwWE&feature=youtu.be>

Victor, B. (2013). Bret Victor - Stop Drawing Dead Fish
[Archivo de video]. Recuperado de
<https://www.youtube.com/watch?v=ZfytHvgHybA>

Meta_Processing, es un entorno de meta-programación basado en el lenguaje Processing. Fue creado por el autor de este libro con el propósito de brindarle al principiante en la programación una herramienta para crear código usando su idioma nativo. Con Meta_Processing se puede escribir y leer el mismo código en 14 idiomas diferentes: Español, Francés, Hindi, Japonés, Italiano, Chino, Portugués, Inglés, Punjabi, Kannada, Bengali, Tamil, Koreano, Ruso y Aleman. Es un entorno de programación diseñado para ofrecer una experiencia amigable, que le evite al usuario cometer errores comunes de sintaxis. Meta_Processing es un software libre publicado bajo licencia GPL v3.

Desde hace años existen iniciativas fantásticas de lenguajes de programación para niños como los son Logo y Scratch (desarrollados en Estados Unidos por el MIT) o Pilas Bloques (desarrollado en Argentina por la iniciativa Program.ar). Sin embargo Meta_Processing le apunta a otro tipo de usuarios que quieren aprender a programar sin sentir que están usando herramientas diseñadas para niños. También podría llegar a ser usado por niños y personas jóvenes que no quieren usar herramientas con interfaces de apariencia infantil.

El profesor Jose David Cuartas es el director y fundador del laboratorio Hipermedia (Hitec Lab) en la Fundación Universitaria Los Libertadores. 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 Hipermedia 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.

ISBN: xxxxxxxxxxxx


Meta_Processing

Este libro es una introducción a la programación en Processing y se divide en tres partes principales:

- Parte I: Introducción a la programación en Processing**: Muestra las principales características de Processing y cómo trabajar con su entorno de desarrollo.
- Parte II: Programación en Processing**: Muestra cómo crear programas sencillos en Processing, incluyendo operaciones básicas, bucles, condicionales y funciones.
- Parte III: Avanzado y aplicaciones**: Muestra cómo aplicar lo aprendido para crear aplicaciones más complejas, incluyendo videojuegos y sistemas interactivos.

Este libro es una guía práctica que combina teoría y práctica, permitiendo a los lectores aprender a programar y crear sus propios proyectos.