



Guía práctica Día 4


Instalación de Bonsai	2
Utilizando Arduino desde Bonsai	3
Adquisición de video	5
Adquisición de audio (Extra)	6
Seguimiento de un objeto en movimiento	7

En esta guía comenzaremos a familiarizarnos con el lenguaje de programación visual Bonsai: <https://bonsai-rx.org/> . Este lenguaje permite combinar el funcionamiento de varios dispositivos de código abierto en paralelo y diseñar soluciones experimentales personalizadas integradas con enfoques flexibles para el estudio del comportamiento, incluyendo el trackeo de los sujetos experimentales, registros de electrofisiología e incluso con la posibilidad de realizar experimentos de tipo bucle cerrado o, más conocidos como *closed loop*.





Bonsai tiene la ventaja de ser una plataforma relativamente simple de usar, con requerimientos mínimos de conocimientos de programación para la adquisición de datos. Se caracteriza también, como otras herramientas abiertas, por su comunidad activa de usuarios dispuestos a entender y armar sus propias herramientas de trabajo, así como también a ayudar a otros miembros a lograr sus objetivos experimentales. Link a la comunidad: <https://bonsai-rx.org/community/> .

En esta guía, en primer lugar utilizaremos Bonsai para lograr objetivos similares a los de la guía 1 (encender un LED, mover un servo), utilizando una placa Arduino. En vez de usar “sketches” del IDE de Arduino, programaremos la placa directamente desde el software Bonsai. En segundo lugar, abordaremos algunas de las ventajas propias de este software, como es el trackeo de objetos, entre otras.

En <https://bonsai-rx.org/docs/observables/> podrás acceder a contenidos teóricos del lenguaje de programación visual Bonsai. Para un desarrollo más fluido de esta guía durante el curso, a continuación te comentamos algunos de los conceptos claves.

Categoría del operador	Descripción
 Source (fuente)	Genera flujos de datos desde dispositivos o desde archivos



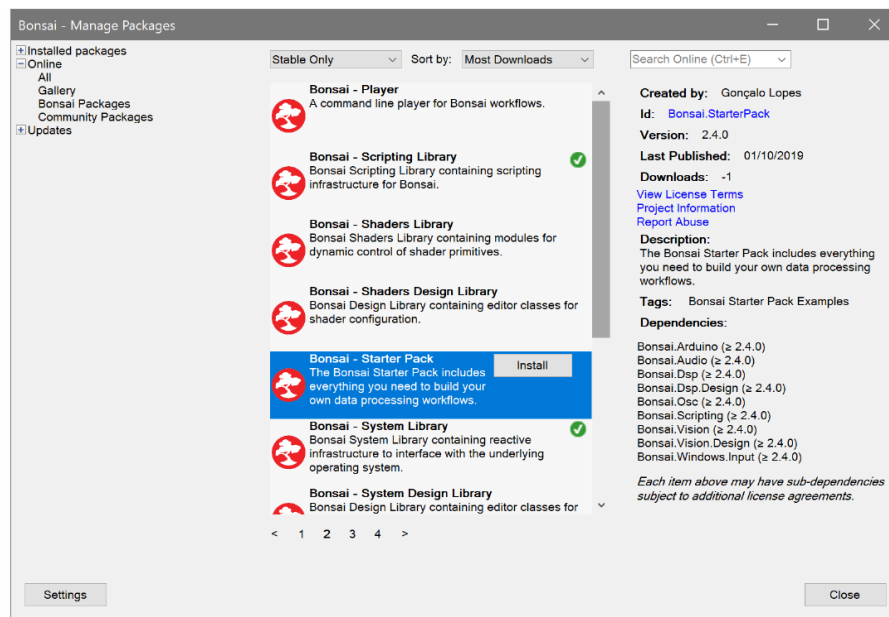
	Transform (Transformar)	Convierte o procesa elementos individuales de datos
	Condition (condición)	Filtra elementos de datos que coincidan con alguna condición específica
	Sink (sumidero)	Guarda datos o activa salidas externas
	Combinator (combinador)	Administrar el flujo de información o sincroniza entradas paralelas

Sugerencias y comandos útiles:

- Los operadores pueden encontrarse en la ventana **Toolbox**. Doble clic en el operador de interés y aparecerá en el espacio **Workflow**.
- Para conectar nodos, clic izquierdo sobre uno > **Create Connection**, y clic sobre el nodo que querés unir al anterior.
- Al hacer clic izquierdo sobre un nodo, se despliega un menú de opciones, entre las cuales están habilitar (Ctrl D)/deshabilitar (Ctrl+Shift+D) dicho nodo y externalizar alguna de sus salidas o propiedades.
- En un *workflow* que esté corriendo, al hacer doble clic sobre un nodo, se abre una ventana con su salida.

Instalación de Bonsai

1. Descargar **Bonsai** de <http://bonsai-rx.org>.
2. Instalar **Bonsai - Starter Pack** desde el administrador de paquetes (*Tools > Manage Packages*).



3. Cliqueá en la pestaña de **Updates** en el extremo izquierdo de la pantalla e instalá cualquier actualización que esté disponible.
4. Podés encontrar una introducción sobre cómo usar la interfaz con el usuario en: <http://bonsai-rx.org/docs/editor>
5. En <https://bonsai-rx.org/docs/editor/#commands-and-shortcuts> encontrarás una lista de los comandos más usados.

Utilizando Arduino desde Bonsai

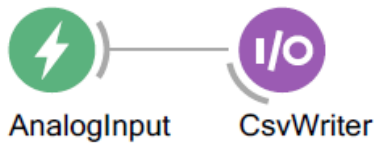
Para poder interactuar con la placa de arduino desde Bonsai es necesario programar el microcontrolador para enviar y recibir datos binarios desde y hacia la computadora vía el cable USB. El IDE de Arduino ya posee un protocolo, llamado *Firmata*, que justamente permite la comunicación eficiente con aplicaciones externas como Bonsai.

Configurar Arduino para la comunicación con Bonsai en tiempo real:

- Abrir el **IDE de Arduino**
- Subir el archivo **StandardFirmata** al Arduino. El código está en Archivo > Ejemplos > Firmata > StandardFirmata.



Ejercicio 1: Guardar datos analógicos



- Insertá el nodo **AnalogInput**.
- Haciendo clic izquierdo sobre el nodo de interés, podrás configurar sus propiedades. Configuraré la propiedad **PortName** para que indique el puerto USB correcto donde está conectado el arduino. Podés corroborar cuál es este puerto desde el IDE de Arduino en Herramientas > puerto.
- Corré el *workflow* y mirá el *output* de la fuente análoga, haciendo doble clic sobre el nodo. ¿Qué ves?

Extra: Conecta un sensor a uno de los pines analógicos del Arduino, por ejemplo un botón y fijate que pasa.

- Insertá el operador **CsvWriter**. Este operador guarda los datos que le ingresan en un archivo de texto. Conectalo al nodo AnalogInput (clic izquierdo en nodo **AnalogInput** > *Create Connection*, clic izquierdo en nodo **CsvWriter**. ¡Ahora ya sabés conectar nodos! :)
- Configuraré la propiedad **FileName** del operador **CsvWriter** con un nombre de archivo terminado en .csv.
- Corré el *workflow*, registrá alguna señal interesante y después abrí el .csv que generaste.

Ejercicio 2: Controlar un LED



- Insertá un nodo **Boolean (o Buleano)**.
- Insertá un nodo **DigitalOutput**.
- Configuraré la propiedad **Pin** del nodo **DigitalOutput** en 13 (el pin que corresponde al LED que viene integrado en la placa Arduino).
- Configuraré la propiedad **PortName** del nodo **DigitalOutput**.
- Corré el *workflow* y cambiá la propiedad **Value** de operador **Buleano**.
- **Extra:** ¡Usa tu mouse para controlar el LED! Reemplazá el operador **Buleano** por el nodo **MouseMove**. Pista: vas a tener que externalizar un output de **MouseMove (Window.Input)**, con clic izquierdo sobre el nodo y usar los nodos **GreaterThan**,



LessThan u operadores equivalentes para conectar uno de los ejes del mouse con el **DigitalOutput**.

Ejercicio 3: Controlar el movimiento de un servo



- Insertá un nodo **Timer (Reactive)**. Configuraré la propiedad **Period** en 500 ms (00:00:00.5).
- Insertá un nodo **Take**. Configuraré la propiedad **Count** en 10.
- Insertá un nodo **Rescale**. Configuraré su propiedad **Max** en 10, y su **RangeMax** en 180.
- Insertá un nodo Repeat.
- Insertá un nodo **ServoOutput**. Configuraré su propiedad **Pin** al 9, y asegúrate que el PortName siga siendo el correcto.
- Conectá el servo al pin 9 del Arduino. Podés fijarte en la guía 1 cómo hacerlo.
- Corré el *workflow*.

Adquisición de video

Bonsai puede ser usado para adquirir y registrar datos utilizando diferentes dispositivos. Los siguientes ejercicios harán que te familiarices con los tipos de datos más comunes usados en Bonsai. El primer tipo de datos con el que vamos a trabajar es **imagen**, que se representa como una matriz 2D de píxeles. Cuando se trata de una imagen en blanco y negro, cada pixel representa un valor de brillo, mientras que si se trata de una imagen a color, cada pixel representa un valor de RGB (composición del color en intensidades de rojo, verde y azul -Red, Green, Blue-).

Ejercicio 4: Grabar un vídeo en tiempo real

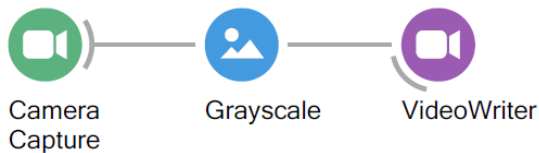


- Insertá el nodo **CameraCapture**. Necesitas tener alguna cámara conectada a tu computadora. Si es una laptop podés usar la cámara integrada.
- Insertá el nodo **VideoWriter**.



- Configuraré la propiedad **FileName** del nodo **VideoWriter** con un nombre de archivo terminado en **.avi**. Podés apretar los tres puntos al costado de **FileName** para setear la dirección donde querés guardar el video.
- Corré el *workflow* y controlá que se genere un vídeo válido.

Ejercicio 5: Guardar un video en escala de grises



- Insertá un transformador **Grayscale** entre los nodos **CameraCapture** y **VideoWriter**.
- Corré el *workflow*. ¿Cómo se ve el video guardado?
- Modificá el workflow para que ahora puedas guardar en simultáneo un video en color y en escala de grises.

Adquisición de audio (Extra)

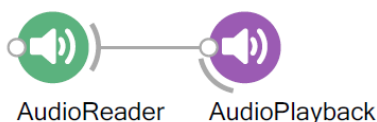
Los datos de audio se registran a frecuencias de muestreo mucho más altas que las del vídeo. Sin embargo, los datos suelen almacenarse en fragmentos con múltiples muestras antes de ser enviados a la computadora. Los datos multi-canal suelen representarse como una matriz 2D, donde las filas representan los canales y las columnas el tiempo.

Ejercicio 6: Adquirir y guardar un archivo de audio .WAV



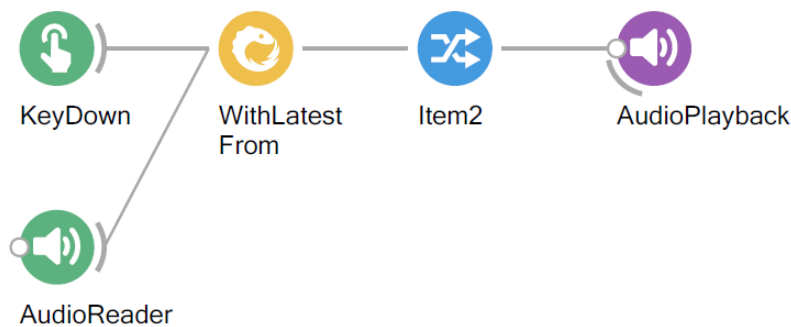
- Insertá un nodo **AudioCapture**
- Insertá un nodo **AudioWriter** y conectalo al **AudioCapture**. Configuraré la propiedad de **FileName** del **AudioWriter**. Acordate de agregar al nombre la extensión **.WAV**.
- Revisá que las frecuencias de registro de ambos nodos sean iguales
- Corré el *workflow* por unos segundos y chequeá que el archivo de audio se haya guardado correctamente.

Ejercicio 7: Emitir un estímulo auditivo





- Insertá un nodo fuente **AudioReader**.
- Configurá la propiedad **FileName** para que sea el archivo de audio que grabaste en el ejercicio 6.
- Insertá un nodo sumidero **AudioPlayback**.
- Corré el workflow y chequea que el sonido esté siendo emitido correctamente.



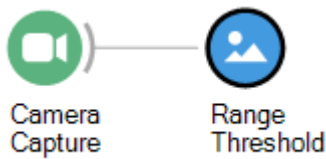
- Insertá un nodo fuente **KeyDown**.
- Seteá la propiedad **BufferLength** de **AudioReader** en 0, así todos los datos del audio van a ser leídos en un único buffer.
- Combiná **KeyDown (Windows.Input)** con **AudioReader** usando el nodo combinador **WithLatestFrom**.
- Hacé doble clic en el nodo **WithLatestFrom**. Seleccioná **Tuple > Item2** del menú.
- Mové el nodo sumidero **AudioPlayback** para que quede a continuación del **Item2**.
- Corré el *workflow* y presiona una tecla. ¿Qué pasa si presionas una tecla varias veces?

Seguimiento de un objeto en movimiento

Bonsai permite procesar vídeos en tiempo real (o desde archivos de video). Así, es posible obtener medidas tales como posiciones de objetos moviéndose, entre otras. Los siguientes ejercicios muestran algunas de las capacidades de Bonsai en este aspecto.

(Nota: para realizar los siguientes ejercicios podés usar como objeto de interés el pollito que enviamos. En cuanto a las cámaras, podés usar una webcam externa, o, si usas laptop, también podés usar la cámara integrada).

Ejercicio 8: Segmentación de un objeto de color

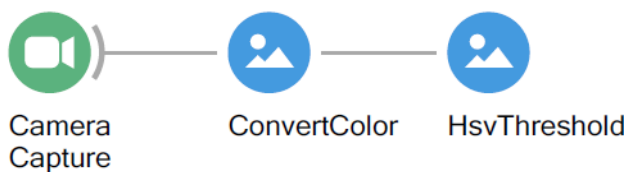


- Insertá el nodo **CameraCapture**.
- Insertá el nodo **RangeThreshold**.
- Abrió el visualizador del operador **RangeThreshold** haciendo doble clic sobre ese nodo.
- Configuraré las propiedades **Lower** y **Upper** del **RangeThreshold** para detectar solo tu objeto de color.

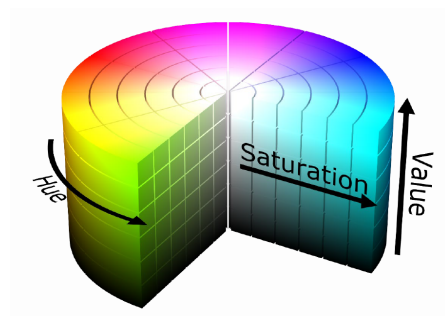
Pista 1: al cliclear en el símbolo + que se encuentra a la izquierda de cada propiedad, se expanden sus valores individuales.

Pista 2: haciendo doble clic sobre cualquier nodo podrás visualizar la información que sale de ese nodo. Te recomendamos, al correr cada *workflow*, que hagas doble clic sobre los nodos, al menos sobre los más relevantes, para una inspección del proceso. Además, si haces clic izquierdo sobre la ventana de visualización de los nodos de imagen, en el margen inferior de la ventana verás no solo la posición xy del cursor sino también los valores de color en esa posición.

Este método segmenta los objetos de color estableciendo límites directamente en el espacio de color RGB. Se considera que este espacio de color es pobre para la segmentación de un color ¿Te das cuenta por qué?



- Reemplazá el operador **RangeThreshold** por el **ConvertColor**. Este transformador convierte tu imagen desde el espacio de color RGB a al espacio de color *Hue-Saturation-Value* (HSV, Tono - Saturación - Valor).



- Insertá un nodo **HsvThreshold**.
- Configurá las propiedades **Lower** y **Upper** de **HSVThreshold** para detectar solo a tu objeto de color.
- Corroborá el seguimiento de tu objeto de color bajo distintas condiciones lumínicas.

Ejercicio 9: Seguimiento en tiempo real



- Usando el workflow del ejercicio anterior, insertá un nodo **FindContours**. Este operador traza los contornos de todos los objetos en una imagen blanco y negro. Definimos como objeto a una región de la imagen conectada por pixeles blancos.
- Insertá un transformador **BinaryRegionAnalysis**. Este nodo permite calcular área, posición del centro de masa, orientación de todos los contornos detectados en la imagen.
- Insertá un nodo **LargestBinaryRegion**. Este operador extrae el objeto más grande de los presentes en el nodo anterior. Haciendo clic izquierdo, seleccioná como *output* al *centroid*, que indica la posición del centro de masa del objeto (*Connected Component* > *Centroid*). Si arrastras el nodo Centroid hacia la ventana de visualización del nodo **CameraCapture** o **ConvertColor**, podrás ver el centroide marcado con un punto rojo, acompañando el movimiento del objeto. Si haces clic sobre el video, se marcará la trayectoria del centroide.
- Grabá en un .csv la posición del centro de masa usando un nodo **CsvWriter**.

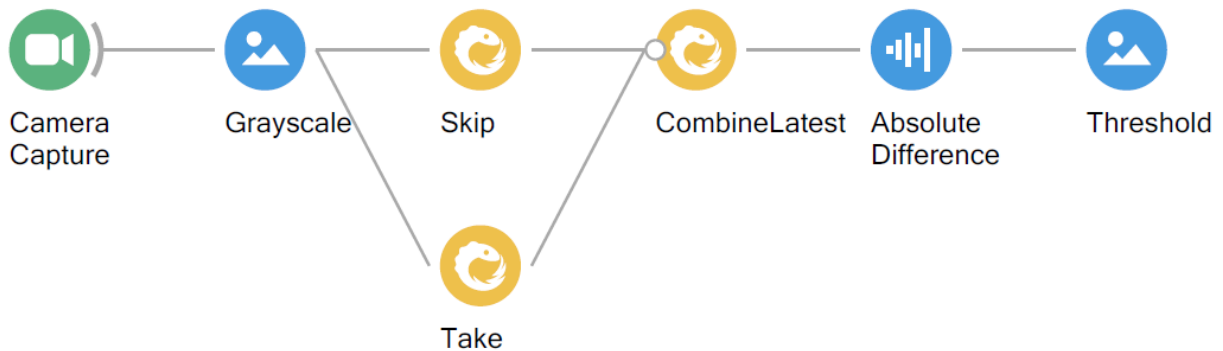
Extra 1: Abrí el archivo generado en el programa que prefieras (Excel, Python, Matlab) y graficá la trayectoria del objeto.

Extra 2: El seguimiento de objetos no necesariamente debe ser con videos adquiridos *online*, sino que puede hacerse un análisis *offline* usando videos ya



grabados. Usando el operador **FileCapture** realizá el ejercicio anterior pero usando un video pregrabado.

Ejercicio 10: Sustracción del fondo y segmentación del movimiento



- Creá un video en escala de grises como hiciste en el ejercicio 5.
- Insertá un operador **Skip**. Configurá su propiedad **Count** en 1.
- En una nueva rama insertá un operador **Take**. Configurá su propiedad **Count** en 1.
- Combiná las imágenes de las dos ramas usando el nodo combinador **CombineLatest**.
- Insertá el nodo **AbsoluteDifference** después de **CombineLatest**.
- Insertá un nodo transformador **Threshold**. Visualiza la salida de ese nodo y ajusta la propiedad **ThresholdValue**.
- Probá el resultado del workflow variando la propiedad Count en **Take** y **Skip**.

El operador skip produce datos luego de #”Count” frames. Si el count es 1, empieza a producir datos 1 frame después de que comenzó el video original. En cambio el operador Take toma los datos desde el primer frame hasta el frame número “Count”.

Describí con tus palabras qué pasa en el *workflow* anterior. ¿Cuál es la diferencia conceptual más evidente en comparación con el *workflow* del Ejercicio 9?

- Reemplazá el operador **CombineLatest** por el nodo **Zip**.
- Eliminá el operador **Take**.
- Cambiá la propiedad Count de Skip a 60 para entender qué está haciendo este nodo.
- Corré el *workflow*.
- Insertá luego del **Threshold** un operador **Sum**. Este operador va a sumar el valor de todos los píxeles de la imagen.
- Clic derecho sobre el operador **Sum**. Seleccioná *Scalar > Val0* desde el menú.
- Visualizá la propiedad **Val0** del operador **Sum**.

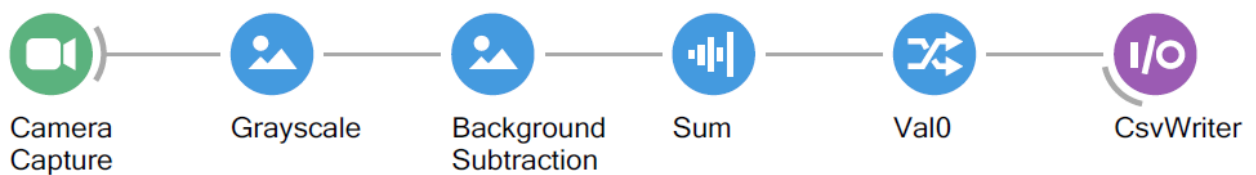


Nota: El operador **Sum**, suma los valores de los píxeles en todos los canales de color de la imagen. Sin embargo, en el caso de las imágenes binarias en escala de grises, sólo hay un canal activo y su suma se almacena en el campo **Val0**.

Describí con tus palabras qué pasa en el *workflow* al quitar el *Take*.

- Ahora volvé a configurar la propiedad **Count** de **Skip** a 1. ¿Qué pasa ahora?

Ejercicio 11: Medición de movimiento



- Creá un video en escala de grises como hiciste en el ejercicio 5.
- Insertá el nodo **BackgroundSubtraction**. Configurá su propiedad **AdaptationRate** en 1. Ajustá la propiedad **ThresholdValue** para quitar el ruido de fondo.
- Insertá el operador **Sum**.
- Corré el *workflow*, apunta tu cámara a un objeto en movimiento y mirá la salida del operador **Sum**. Compará movimientos pequeños con movimientos más marcados. ¿Qué pasa con la señal cuando el objeto se queda perfectamente quieto?
- Clic derecho sobre el operador **Sum**. Seleccioná **Scalar > Val0** desde el menú.
- Registrá el movimiento de un objeto usando el nodo **CsvWriter**.



Este contenido es una traducción y adaptación de parte del material del curso “Visual Reactive Programming with Bonsai”, del Cajal NeuroKit Advanced Neuroscience Training Programme”, que fue desarrollado por NeuroGEARS, Ltd, que está disponible en <https://neurogears.org/vrp-2021/>. Fueron realizadas para el Taller práctico abierto IBRO-LARC: Adquisición de señales neuronales, por la Lic. Azul Silva, la Dra. Verónica de la Fuente, la Lic. Cecilia Herbert y el Ing. Marcos Coletti.

Este contenido está bajo licencia [Atribución/Reconocimiento-CompartirIgual 4.0 Internacional \(CC-BY-SA 4.0\)](https://creativecommons.org/licenses/by-sa/4.0/). Esto significa que:

Sos libre de...

- **Compartir** — copiar y redistribuir el material en cualquier medio o formato
- **Adaptar** — remezclar, transformar y construir a partir del material para cualquier propósito, incluso comercialmente.

Bajo los siguientes términos:

- **Atribución** — Debés dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Podés hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que vos o tu uso tienen el apoyo de la licenciante.
- **CompartirIgual** — Si remezclás, transformás o creás a partir del material, debés distribuir tu contribución bajo la misma licencia del original.
- **No hay restricciones adicionales** — No podés aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier uso permitido por la licencia.

Este resumen legible por humanos no es un sustituto de [la licencia](https://creativecommons.org/licenses/by-sa/4.0/). Para ver una copia de esta licencia, visitá el siguiente vínculo: <https://creativecommons.org/licenses/by-sa/4.0/>.

Nos apoyan generosamente:

