Guía práctica Día 5

Closed-loop systems

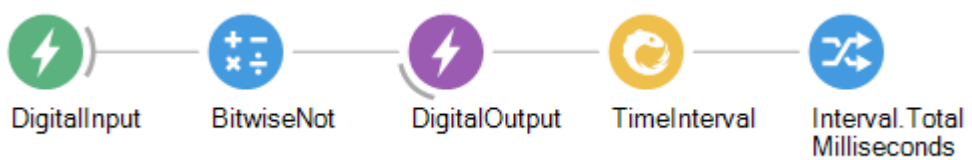
En un experimento de tipo "circuito cerrado", más conocido como *closed loop,* el objetivo es generar en tiempo real una acción como respuesta -o feedback- a una determinada señal. Así, la salida de nuestro sistema, depende de la detección de un evento o señal. En el caso de experimentos comportamentales, esa señal podría, por ejemplo, tratarse de un animal haciendo un determinado comportamiento o estando en cierto lugar de la arena experimental. Con Bonsai es posible diseñar este tipo de experimentos usando un procesamiento en tiempo real de los datos. Los próximos ejercicios sirven para mostrar algunos ejemplos de sistemas *closed-loop*.

Medición de la latencia del sistema *closed-loop*

Uno de los parámetros más importantes para evaluar el rendimiento de un sistema *closed-loop* es la latencia, es decir, el tiempo que tarda en generarse un cambio en la salida como respuesta a un cambio en la entrada.

La manera más fácil de medir la latencia de un sistema *closed-loop* es usar un test de retroalimentación *(feedback)* digital. En este tipo de tests medimos una salida binaria del sistema *closed-loop* y la introducimos directamente en el sensor de entrada. A continuación, registramos una serie de mediciones en las que cambiamos la salida a ALTO si el sensor detecta BAJO, y la cambiamos a BAJO, si el sensor detecta ALTO. El intervalo de tiempo entre las señales de ALTO y BAJO nos dará la latencia total del sistema *closed-loop*, también conocida como *round-trip time*.

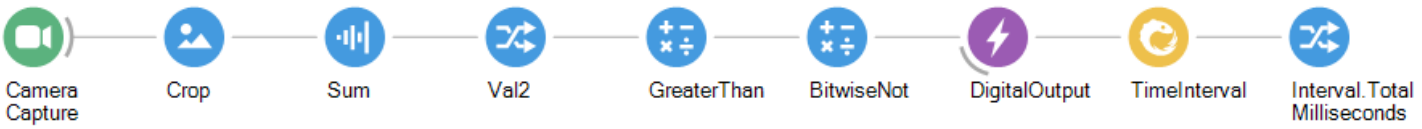
Ejercicio 1. Midiendo la latencia de la comunicación entre puertos.



- Conectá el pin digital 8 del Arduino al pin digital 13 usando un cable macho-macho
- Insertá un nodo ***DigitalInput*** y configurá la propiedad pin al 8.
- Insertá un nodo transformador ***BitwiseNot.***
- Insertá un nodo ***DigitalOutput*** y la propiedad pin al 13.
- Insertá el operador ***TimeInterval***.
- Clic derecho en ***TimeInterval***, seleccioná ***Output*** > ***Interval*** > ***TotalMilliseconds.***
- Corré el *workflow*. Hacé doble click en el nodo ***TotallMilliseconds*** y estimá la latencia o el *round trip time* entre los mensajes de entrada digitales. Podés verlo como gráfico o como lista de valores (haciendo click derecho en ***TotalMilliseconds*** > ***ShowVisualizer*** > ***Time Series Visualizer*** u ***Object Text Visualizer.***

¿Qué te parece que hace el comando *BitwiseNot*?

Ejercicio 2. Midiendo la latencia en la adquisición de video



- Conectá un LED rojo al pin digital 13 del Arduino
- Insertá un nodo ***CameraCapture***
- Insertá un nodo transformador ***Crop***, que te va a permitir cortar una parte del espacio de la imagen que sea de tu interés.
- Para establecer esa región de interés, que en este caso debe englobar el LED, corré el *workflow*. Recordá que haciendo doble clic sobre cada nodo podrás "ver" la información que sale de ese nodo. Como verás, inicialmente la imagen del ***Crop*** es igual a la imagen del ***CameraCapture***. Andá a las propiedades del Crop, clickeá el símbolo "… " de ***RegionOfInterest***. Aparecerá una muestra del espacio de la imagen el la que podrás recuadrar tu región de interés-

  Nota: si estás usando la cámara integrada de una laptop podés pegar con cinta el protoboard a una caja de cartón que la sostenga vertical, de manera que puedas detectar el LED desde tu imagen.

Pista: La calibración visual de la región de interés también puede realizarse haciendo clic derecho en el nodo crop, seleccionando ***ShowDefaultEditor***

- Insertá un nodo ***Sum(Dsp)*** y seleccioná el campo ***Val2*** (para hacerlo recordá cómo habíamos hecho en el caso de querer configurar Val0 para imágenes en escala de grises)

  Nota: En este caso, asumiendo que las imágenes son tomadas en un formato BGR (o RGB), Val2 estará sumando los pixeles en el canal rojo. Val0 y Val1 guardarían los valores de los canales azul y verde respectivamente. En caso de usar otro LED que no sea rojo, debería cambiarse este operador acordemente.

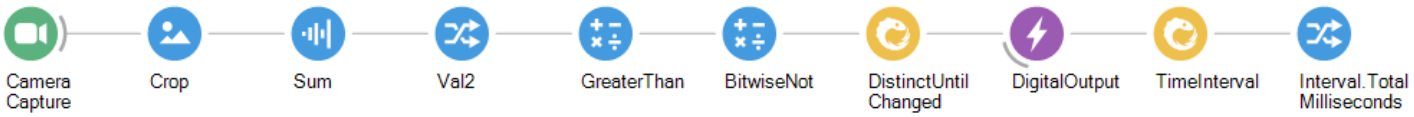
- Insertá un nodo transformador ***GreaterThan***
- Insertá un ***BitwiseNot***
- Insertá un **DigitalOutput** y configurá su propiedad **Pin** al pin 13
- Corré el *workflow* y usá el visualizador de ***Sum*** *(o mejor, el del Val2)* para elegir un umbral apropiado en la propiedad ***Value*** de ***GreaterThan***.

  Nota: Para entender este ejercicio es clave pensar que cuando se inicia el workflow el LED rojo que conectaron al protoboard va a titilar (porque está conectado al pin 13, y como vimos antes, esto es lo que también sucede con el LED integrado al arduino: titila al encenderse). Esa luz roja titilante inicial va a ser la señal detectada con el ***Sum*** (si el Crop estuvo bien hecho sobre el LED, claro). Dado que el ***DigitalOutput*** también está configurado para el pin 13, la salida del workflow va a afectar el encendido apagado del LED rojo-

¿Qué pasa con el LED rojo, más allá del titilado inicial?

Probá ahora deshabilitando el ***BitwiseNot*** (Ctrol-D sobre el nodo). ¿Qué pasa con el LED rojo?

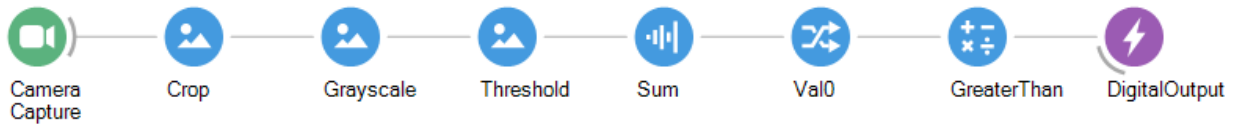
- Ahora volvé a habilitar el ***BitwiseNot*** e ***i***nsertá un nodo ***DistinctUntilCyhanged*** luego del transformador ***BitWiseNot:***



¿Cuál es la consecuencia? ¿Es mejor usar el operador ***DistinctUntilChanged*** para lograr el objetivo del ejercicio?

Control *closed-loop*

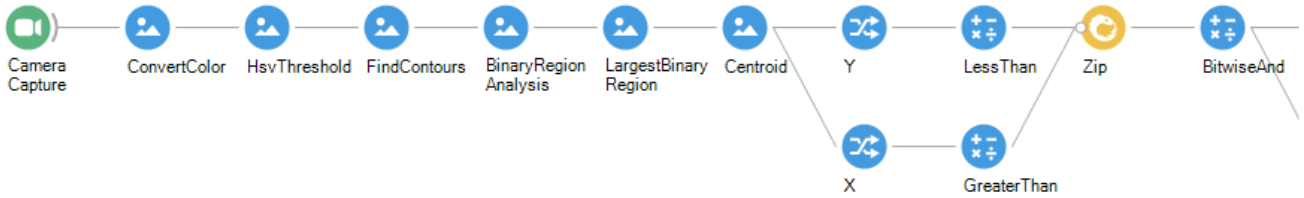
Ejercicio 3. Emitir una señal digital en base a la actividad de una región de interés



- Insertá un nodo ***CameraCapture***
- Insertá un nodo transformador ***Crop***
- Corré el *workflow* y establecé tu región de interés
- Insertá un nodo ***Grayscale*** y un ***Threshold(Vision).*** Estos nodos pasan la imagen a una escala de grises y convierten una imagen en escala de grises a una imagen binaria en base a un umbral, respectivamente.
- Insertá un operador ***Sum(Dsp)*** y seleccioná ***Val0***
- Insertá un nodo **GreaterThan** y configurá la propiedad ***Value.*** Al configurar este valor establecemos a partir de que valor input de **GreaterThan** el output del mismo pasa de cero a uno.
- Insertá un nodo ***DigitalOutput.*** Configuralo al pin 13 del Arduino y al puerto adecuado.
- Corré el *workflow.* Ahora tenemos que establecer un valor de **Threshold** que me permita discriminar la presencia y ausencia de un objeto dado en la región del nodo **Crop**.
- Verificá que si tu objeto entra a la región de interés se genere la emisión de una señal en el Arduino (luz en el LED integrado al Pin 13).

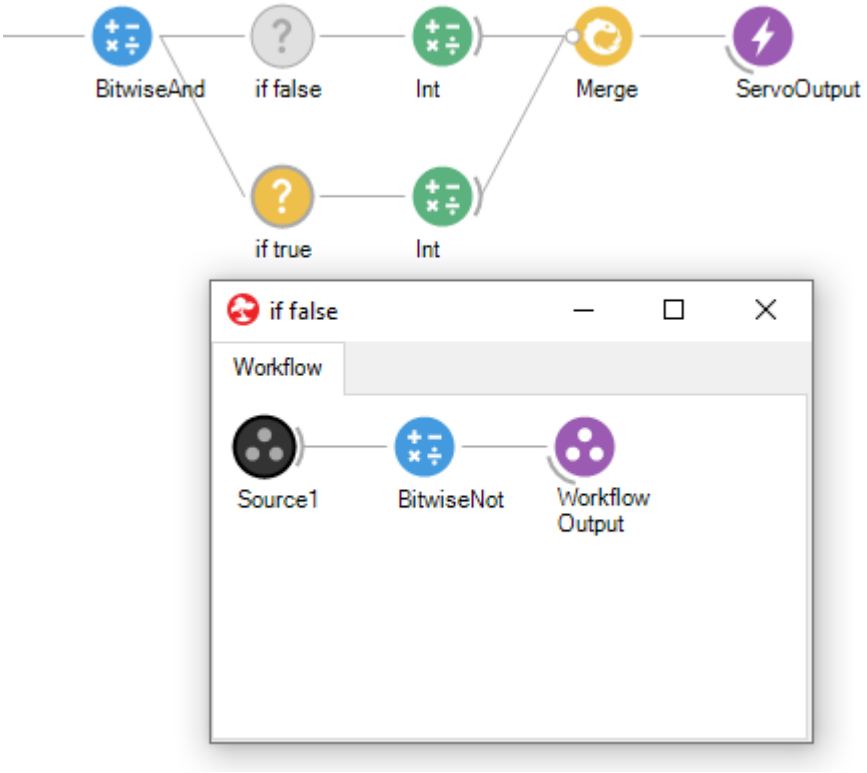
  Extra: reemplazá el transformador Crop por uno ***CropPolygon*** (experimentalmente no siempre vamos a querer regiones de interés rectangulares).

Ejercicio 4. Mover un servo en función de la presencia de un objeto en una región de interés



- Insertá un nodo **CameraCapture** y armá un *workflow* para *trackear* el centroide de un objeto como en el ejercicio 9 de la guía 4.
- Ahora haciendo click derecho en **Centroid** seleccioná el *Output* **X (float)**. Ahora repetí para el **Y (float)**. Estos dos nodos me dan la posición en pixeles del centroide del objeto en coordenadas **X** e **Y**.

- Colocá un nodo **GreaterThan** conectado al **X**. Este nodo nos va a devolver un buleano verdadero cuando la posición en **X** del objeto supere el valor que pongamos en la propiedad **Value** del nodo. Ahora repetí este proceso con el nodo **LessThan** conectado a **Y**. Colocá valores que definan los límites de la región de interés.
- Conectá el *output* de estos dos nodos a un **Zip** y este a un **BitwiseAnd**. Este último nodo nos va a devolver un buleano verdadero cuando se cumplan las condiciones de **LessThan** y **GreaterThan** al mismo tiempo.



- Ahora conectá el *output* del nodo **BitwiseAnd** a un nodo **Condition**. Este nodo detiene los pasos que sigan en el *workflow* hasta que se cumpla la condición dada. En este caso particular el *workflow* sólo sigue si el input es verdadero. Podés modificar la propiedad **Name** para que sea más claro qué hace el nodo, en este caso "*if true*" es bastante explícito.
- Repetí el paso anterior con otro nodo **Condition** pero esta vez agregá un nodo **BitwiseNot** en el *workflow* de este nodo. Esta condición sólo va a ejecutarse cuando su *input* sea falso. Podés modificar el nombre del nodo para que sea explícito, como por ejemplo a "*if false*".
- Colocá un **Int** en el *output* de cada uno de los **Condition**. Asigná a cada Int un valor de posición del servo (por ejemplo 10 y 170).
- Colocá un nodo **Merge** para combinar los dos *outputs* de los **Condition**. Como los nodos **Condition** no pueden cumplirse en simultáneo el valor de **Merge** va a ser siempre uno de los dos valores de Int que colocamos en el paso previo.

- Colocá un nodo **servoOutput** luego del Merge.
- Conectá el servo al pin 9 del Arduino. Podes fijarte en la guía 1 cómo hacerlo.
- Probá que el servo se mueva en función de si el objeto entra o no a la región que delimitaste con los valores de **LessThan** y **GreaterThan**.

Nota: Este ejercicio tiene un workflow más complejo que el resto. Presenta una oportunidad excelente para entender el funcionamiento de varios nodos que controlan el flujo del *workflow* de Bonsai. Recordá que podés ver los outputs de estos nodos con el *workflow* en funcionamiento haciendo doble click en cada uno. Prestá atención a los valores que devuelve cada nodo y en qué momento están dando estos valores.

Ejercicio 5. Mover un servo en función de la posición de un objeto

- Insertá un nodo **CameraCapture** y armá un *workflow* para *trackear* el centroide de un objeto como en la primera parte del ejercicio anterior.
- Ahora haciendo click derecho en **Centroid** seleccioná el *Output* **X (float)**.
- Colocá un nodo **Rescale**, vamos a modificar sus propiedades. Colocá el número de pixeles en X de tu pantalla en **Max** y cero en **Min**. Estos son los límites del *input* que recibe el nodo de **X**. En **RangeMax** colocá 170 y en **RangeMin** 10. Ahora el nodo va a hacer una conversión lineal de valores entre cero y el ancho en píxeles de tu pantalla a un número entre 10 y 170 grados.
- Colocá un nodo **ServoOutput** y conectá un servo al pin 9 del Arduino.
- Ahora verificá que el movimiento del servo acompañe al del objeto a *trackear*.

¿El servo se mueve en la misma dirección que el objeto? ¿Qué nodo deberías modificar en caso que no?

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, el Ing. Marcos Coletti y el Lic. Alejandro Cámera.

Nos apoyan generosamente:

IBRO LARC

SAN SOCIEDAD ARGENTINA DE INVESTIGACIÓN EN NEUROCIENCIAS

TALLERES OPEN SOURCE

Realizado con el apoyo del Fondo Metropolitano de la Cultura, las Artes y las Ciencias

Fondo Metropolitano de la Cultura, las Artes y las Ciencias | Buenos Aires Ciudad