Communicating with External Hardware using PsychoPy

by Kimberley Dundas

What kind of external hardware?

- PsychoPy is currently able to communicate with:
- EEG recording hardware
- fMRI/MRI scanners
- Eye-trackers
- Microcontrollers (e.g. an **Arduino board**)
- fNIRS – there's a really clear **video tutorial** from NIRx

EEG recording hardware - caveats

- Timing issues and synchronisation
  - Timing Mega-study (Bridges et al., 2020)
  - Dropped frames
  - Millisecond precision
  - Use Builder!
  - Test, test, test:
    - Monitor refresh rates – Poth et al. (2018)
    - ‘Drawing’ from top to bottom
  - Lag and variability

How do I send triggers/markers?

Parallel port component

8 bit communication

Sends values in parallel - faster

Not all computers have a parallel port…

Serial port component

Sends values serially – slower?

Adapters are available for USB, e.g.:

LabHackers USB2TTL8

LabJack U3

Sending EEG triggers in PsychoPy

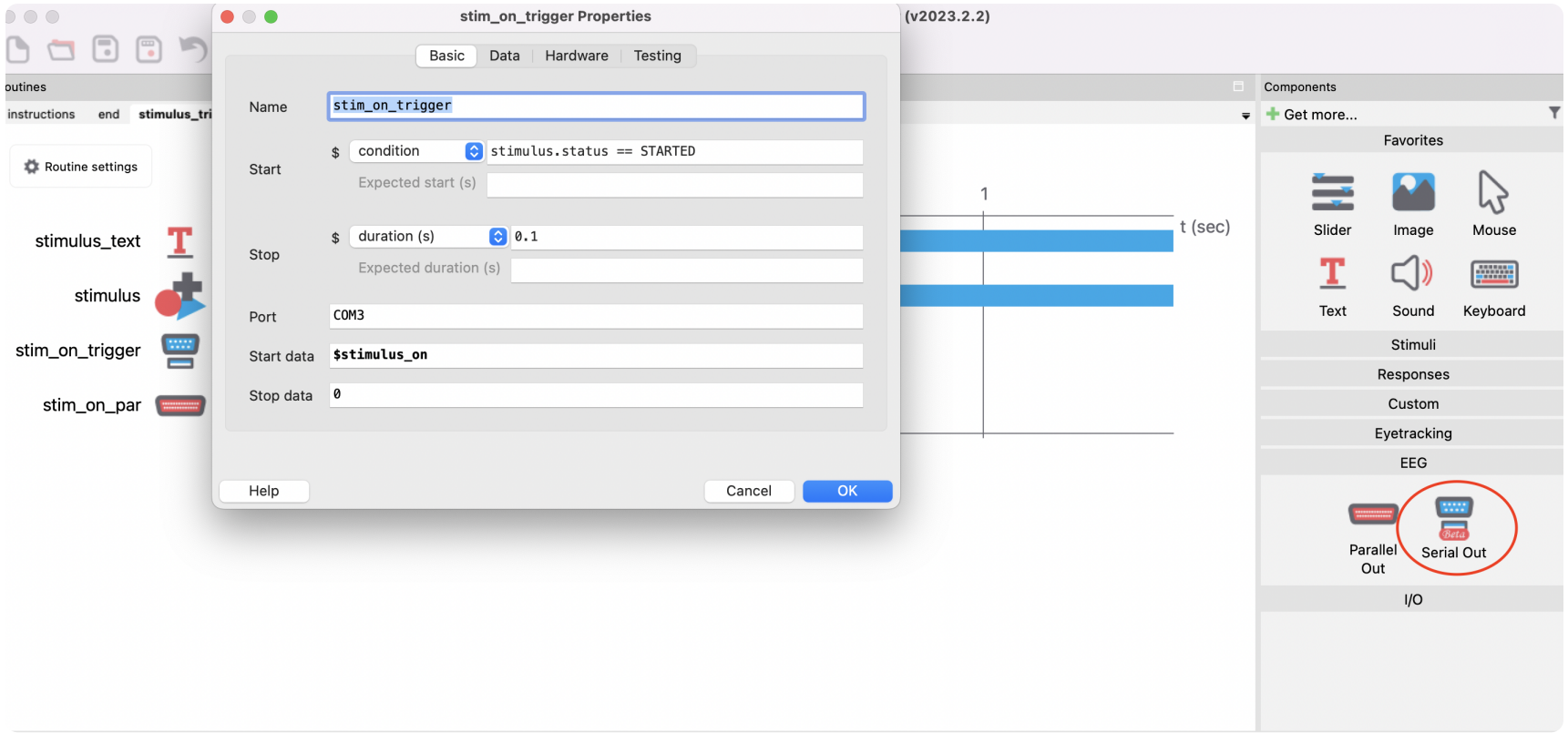
The biggest difficulty that's faced when trying to send triggers in PsychoPy is working out:

1. Which port address your EEG acquisition system is connected to
2. How your EEG acquisition system *wants* to receive the triggers you send

Once we know these things - sending triggers is relatively simple!

> ⚠ Be sure to bear the timing caveats in mind when designing your EEG experiments

Serial Port Component



Insert a **Serial Out** component to the routine that you'd like to add triggers to

Start refers to when you want the trigger to be sent; this can be at a certain time or frame, or we can set the trigger to be sent when a certain condition is met (more on this later)

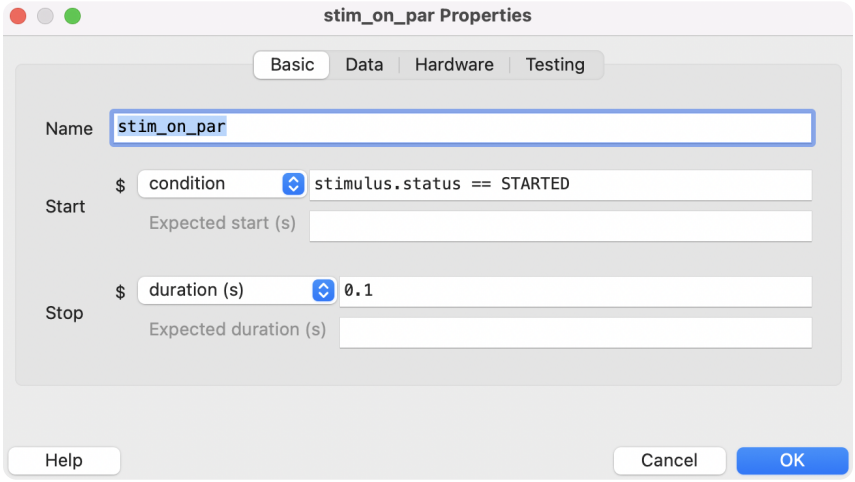
Port refers to the address of the serial port your device is connected to

Start data refers to the value that you want to actually send to your acquisition system - exactly what you'd like to send will depend on what your system wants to receive. This can take a variable just like most other fields so that you can send different triggers for different stimuli etc.

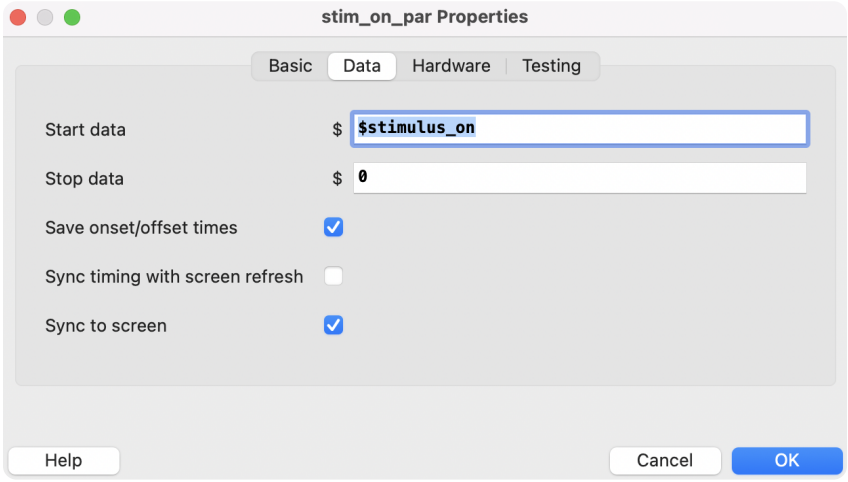
Parallel Port Component

The parallel port component is essentially set up in the same way as the serial port component, but the fields are spread across separate tabs:

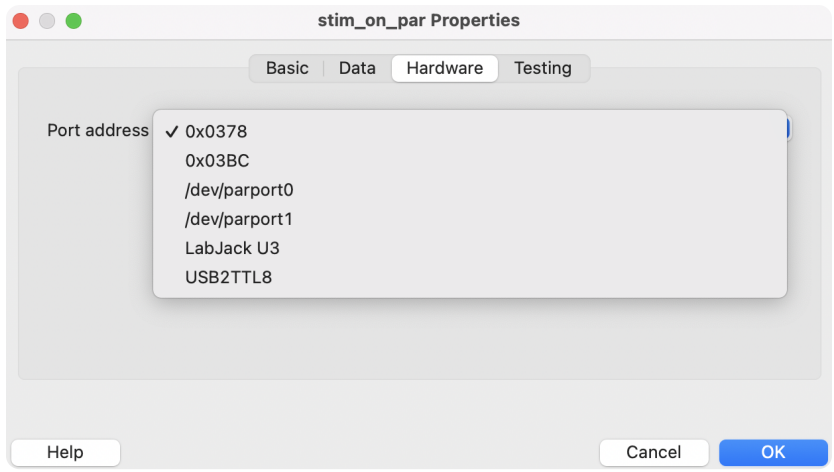
Start is set in the Basic tab just as with the serial port component:



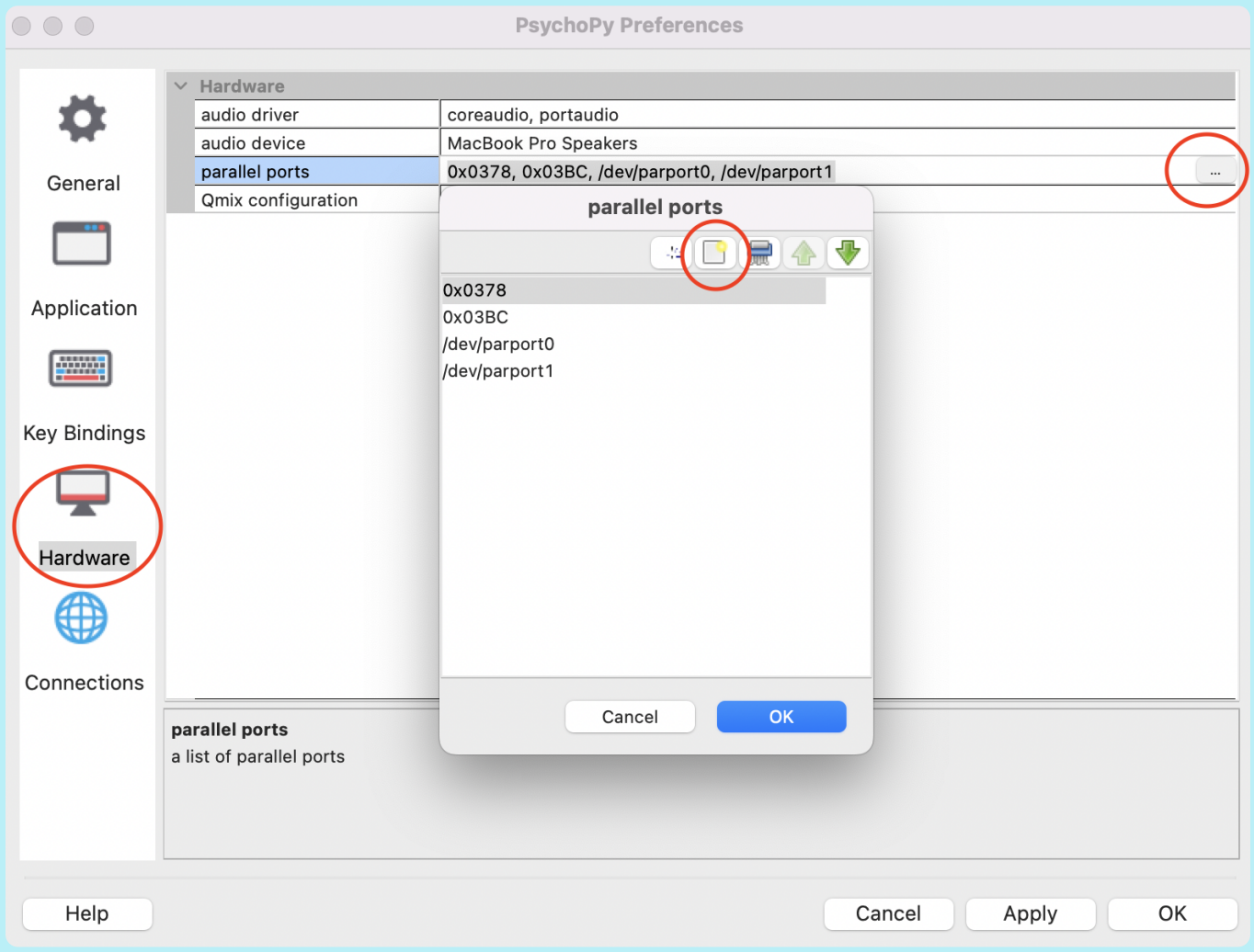
Start data can be found in the Data tab



Port address is found in the Hardware tab

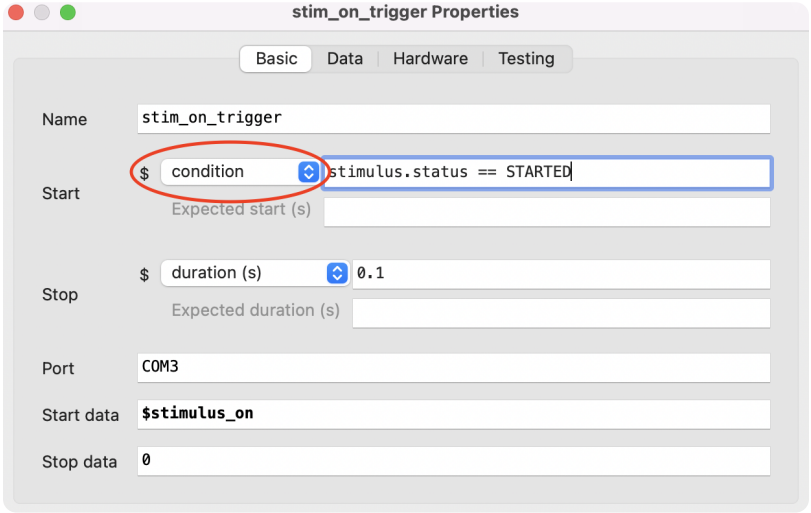


If you don't see your port address listed, you can add it by navigating to File > Preferences > Hardware > Parallel Ports > … > New:



Sending triggers to mark events

Exactly *when* your trigger will be sent to your acquisition device will depend on how you set the **Start** of the serial *or* parallel port component:



We might want to send our trigger on a certain frame or time, but typically we want to send a trigger to mark a given event. To do this we have to set the Start to **Condition** (rather than Duration)

To send on the **onset of something** (e.g., a visual stimulus) set that condition to be (where stimulus is the name of the component you want to yoke your trigger to):

```
stimulus.status == STARTED
```

To send on a **keypress,** keep the Start as Condition and set the condition to be (where key_response is the name of your keyboard component):

```
key_response.keys
```

To send when a stimulus is **clicked by a mouse** (mouse here refers to the name of your mouse component, and stimulus_mouse refers to the stimulus you want to be clicked):

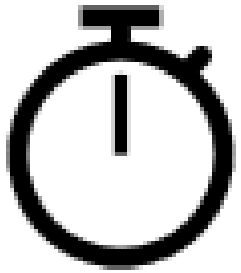
```
mouse.isPressedIn(stimulus_mouse)
```

Using MRI/fMRI with PsychoPy

- Keypress
- Serial/parallel communication
- Find out more (including code to copy and paste) **here**.

Non-slip timing – what is it and why do we care?

Relative timing

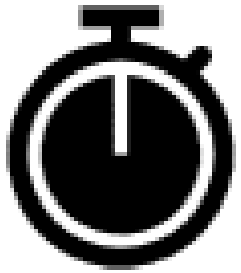


Set a timer to **0** at the start of each **trial**

Present a stimulus

Check on each frame whether the timer exceeds our intended duration

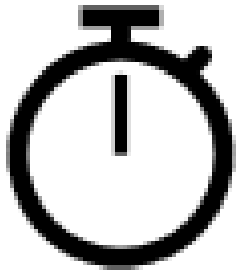
If it hasn't, present the stimulus for this frame too



Stop on the frame where the timer **exceeds** our intended duration

Potentially leading to an overshoot, for example of 10ms

Non-slip timing



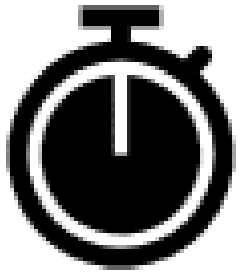
Set a timer that is **not** reset at the start of each **trial**

Instead, any residual value from the previous trial is added to the intended duration of this trial, e.g. -10ms

Present a stimulus

Check on each frame whether the timer exceeds our intended duration

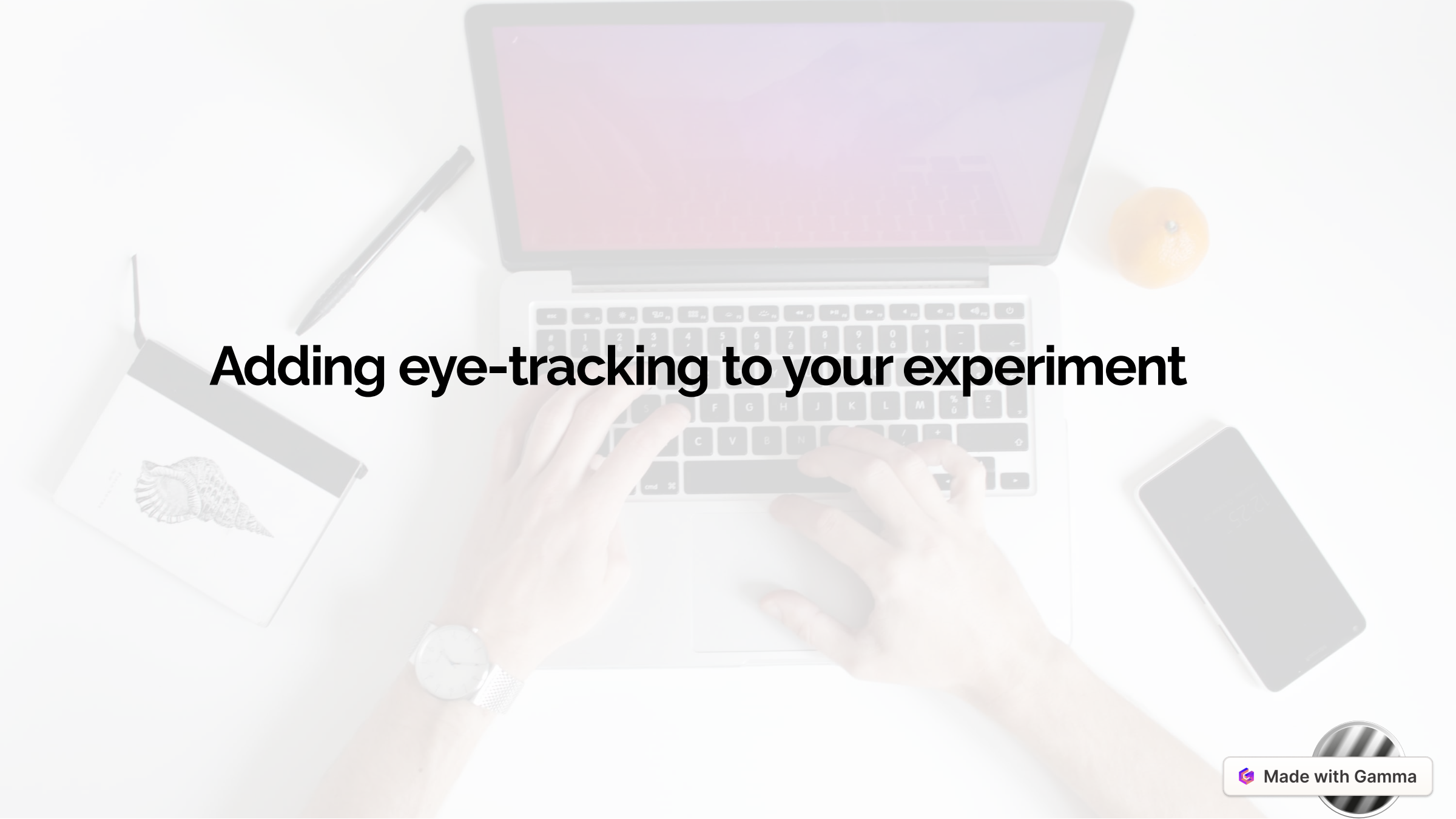
If it hasn't, present the stimulus for this frame too



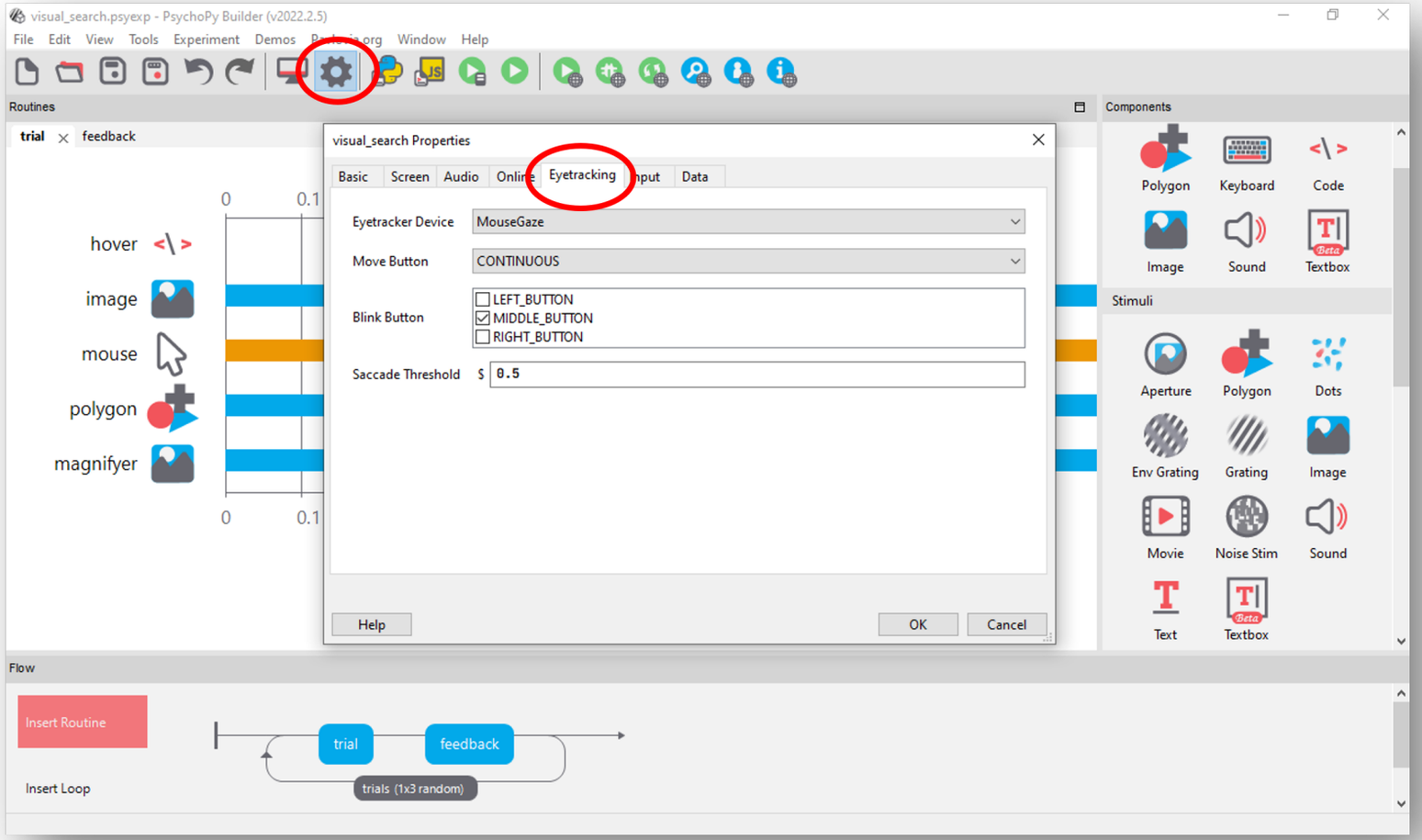
Stop on the frame where the timer **exceeds** our intended duration

If this was 10ms too long, our residual is -10ms, which is then added to the next trial

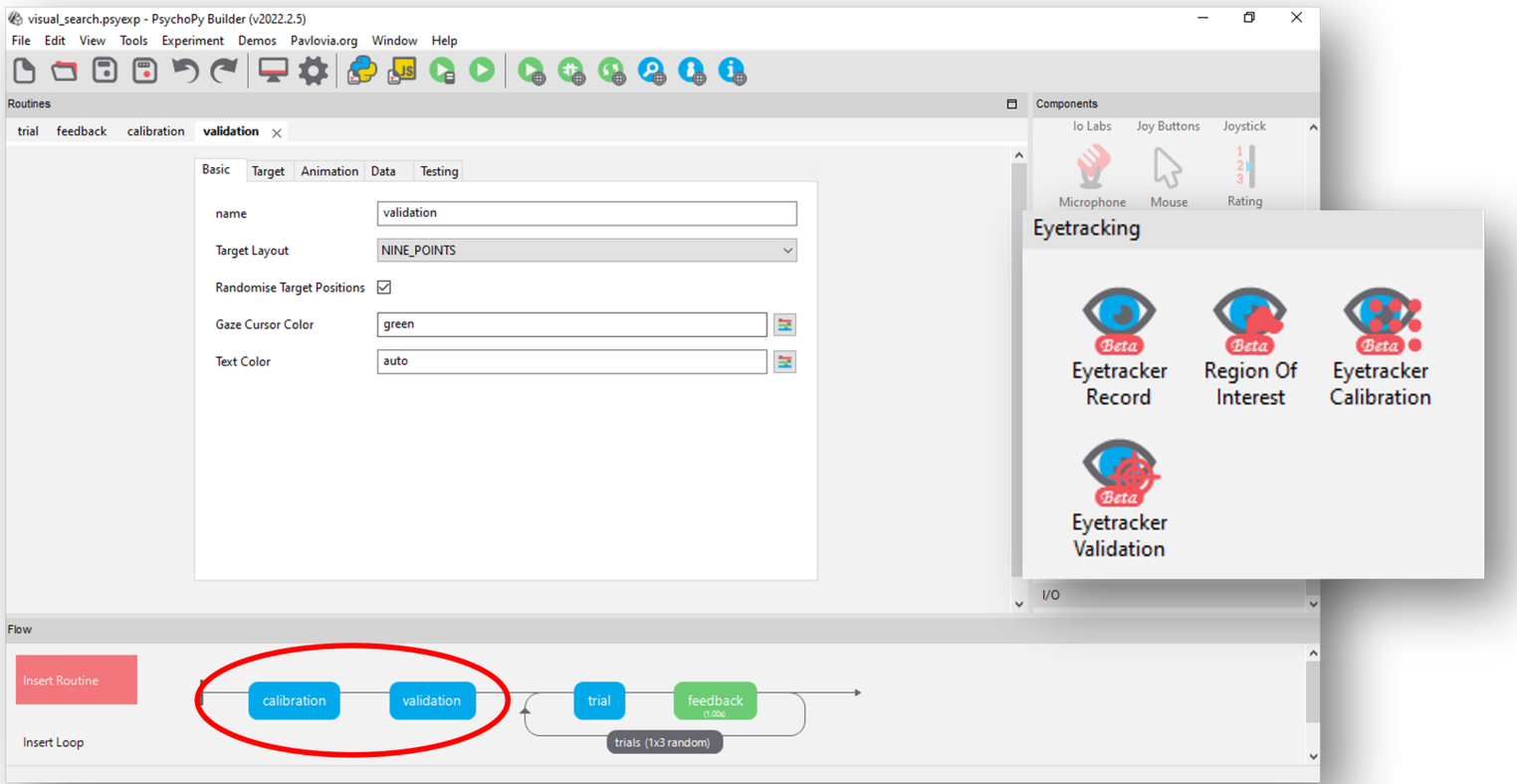
Adding eye-tracking to your experiment



Mouse Gaze

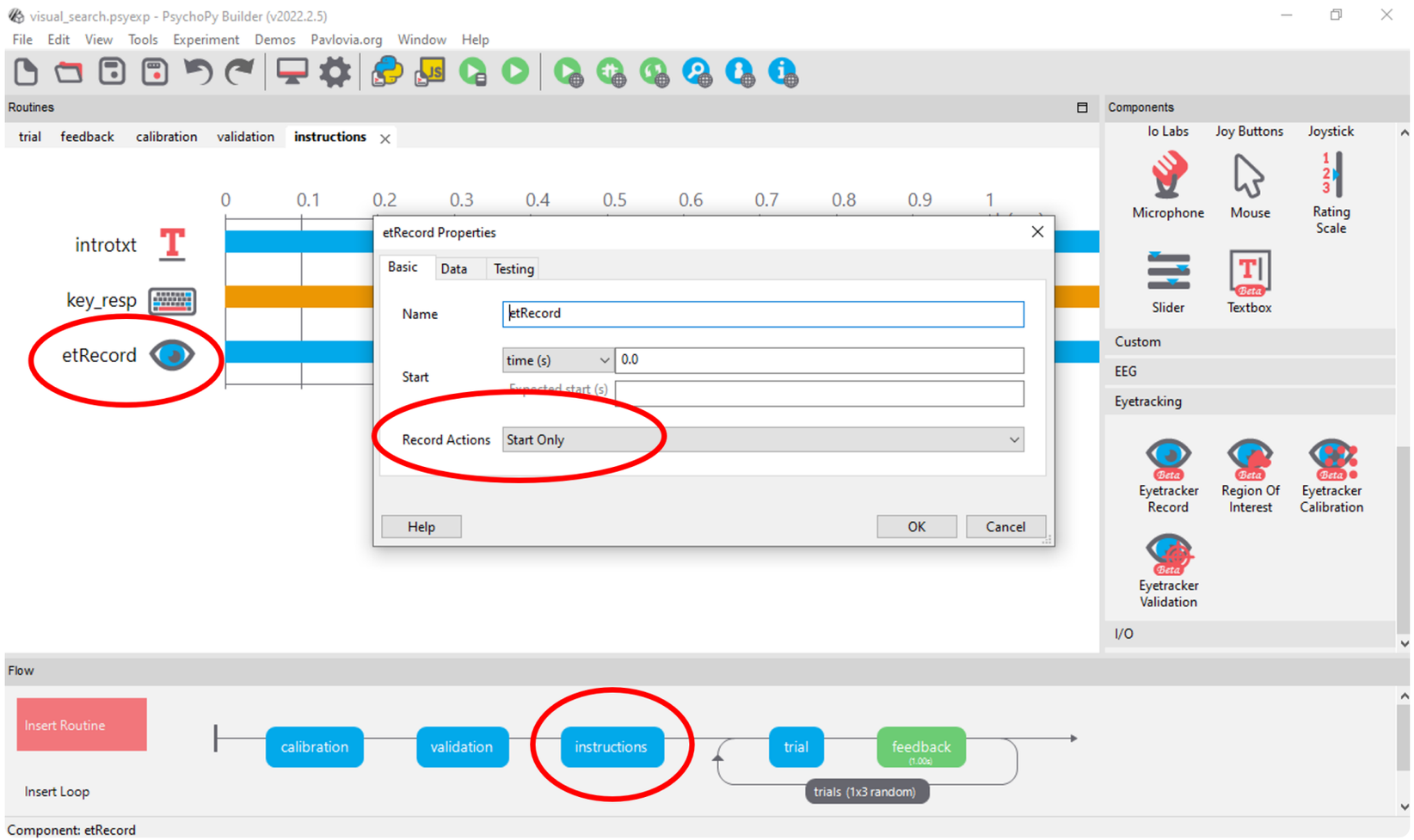


Calibration and Validation routines



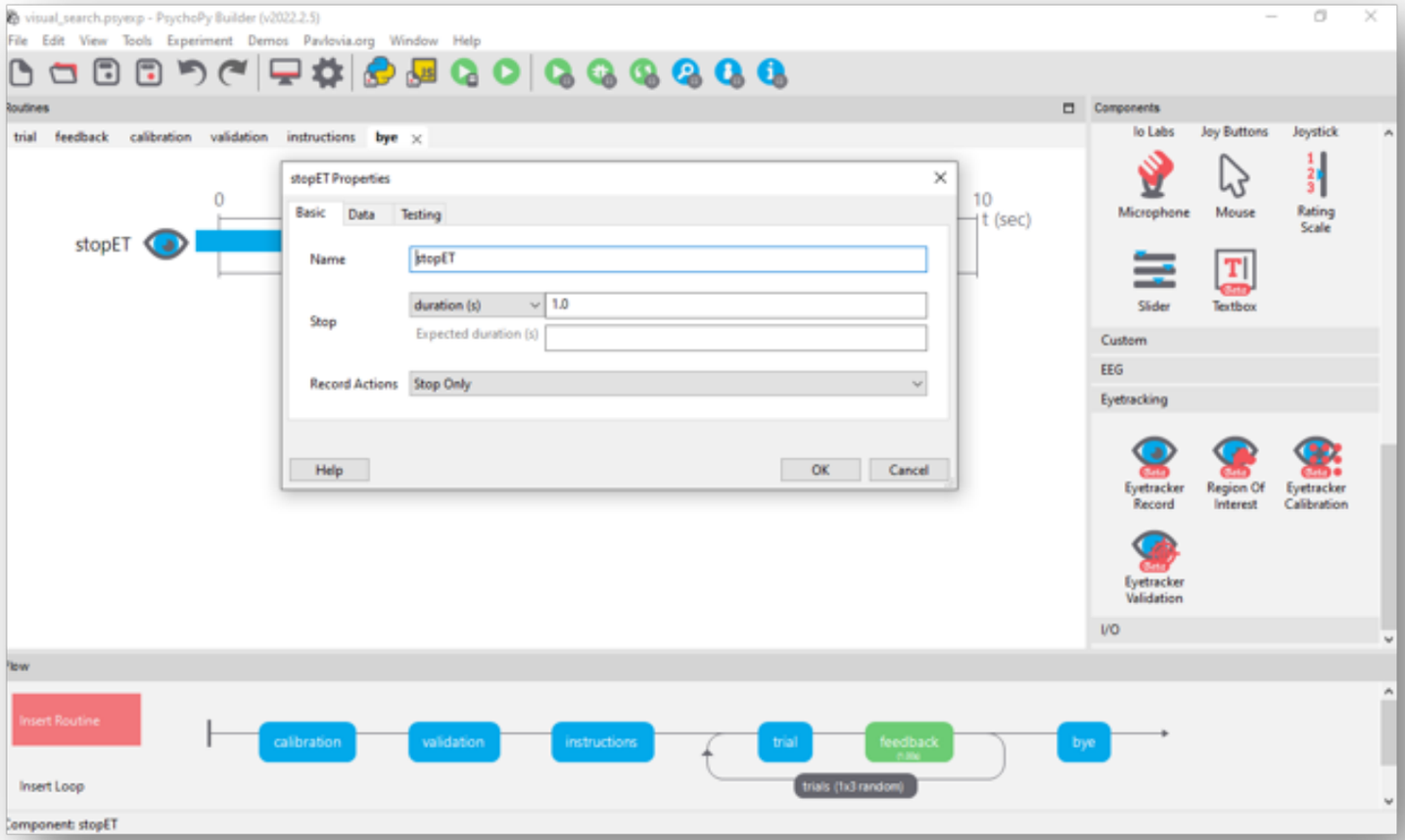
⚠ These components create “standalone routines” rather than components. Once selected you select “insert routine” and add the calibration routine where you want it on your flow.

Start the eye tracker



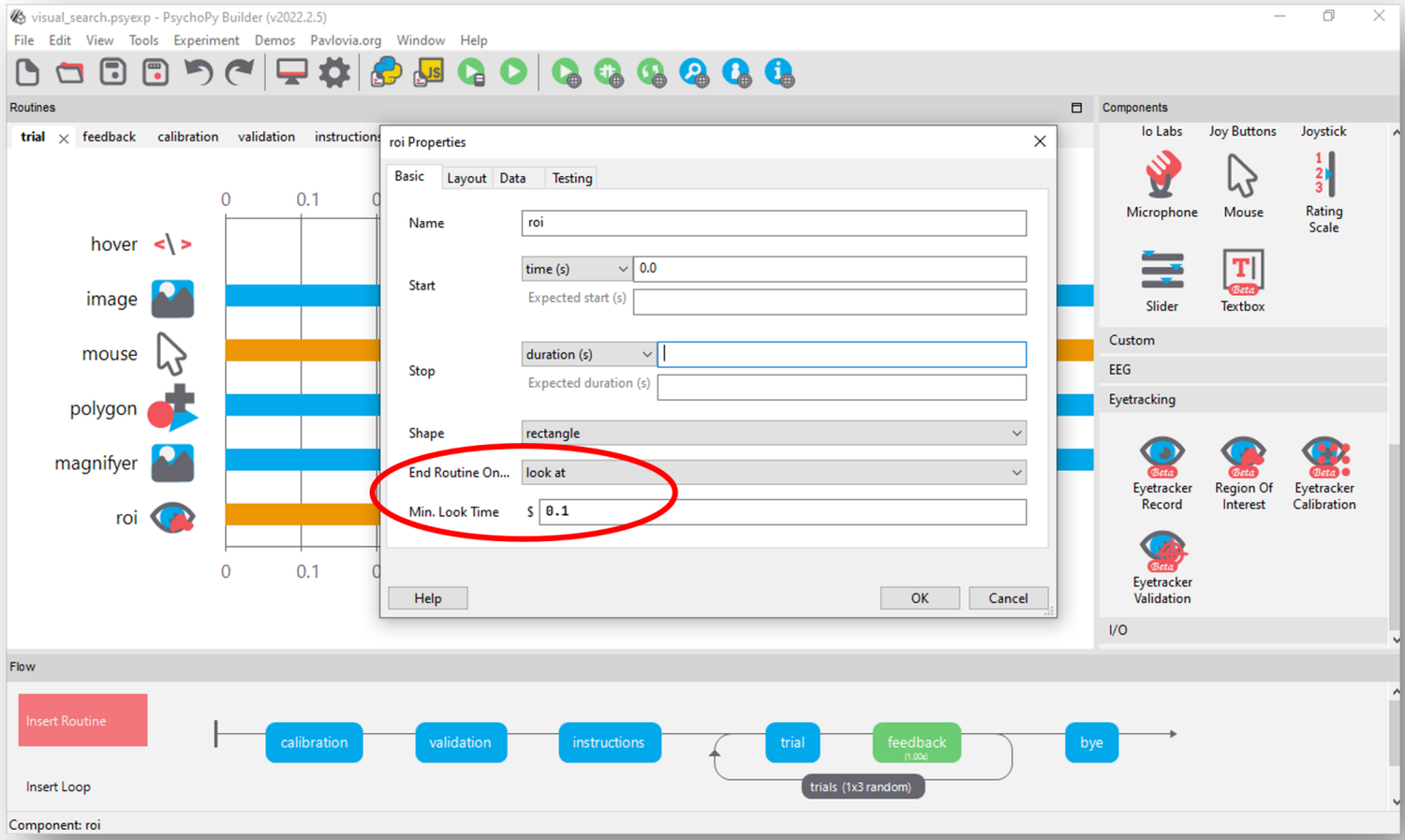
You can start/stop the eye tracker in a single trial – or start the tracker before your trials and stop it afterwards.

Stop the eye tracker



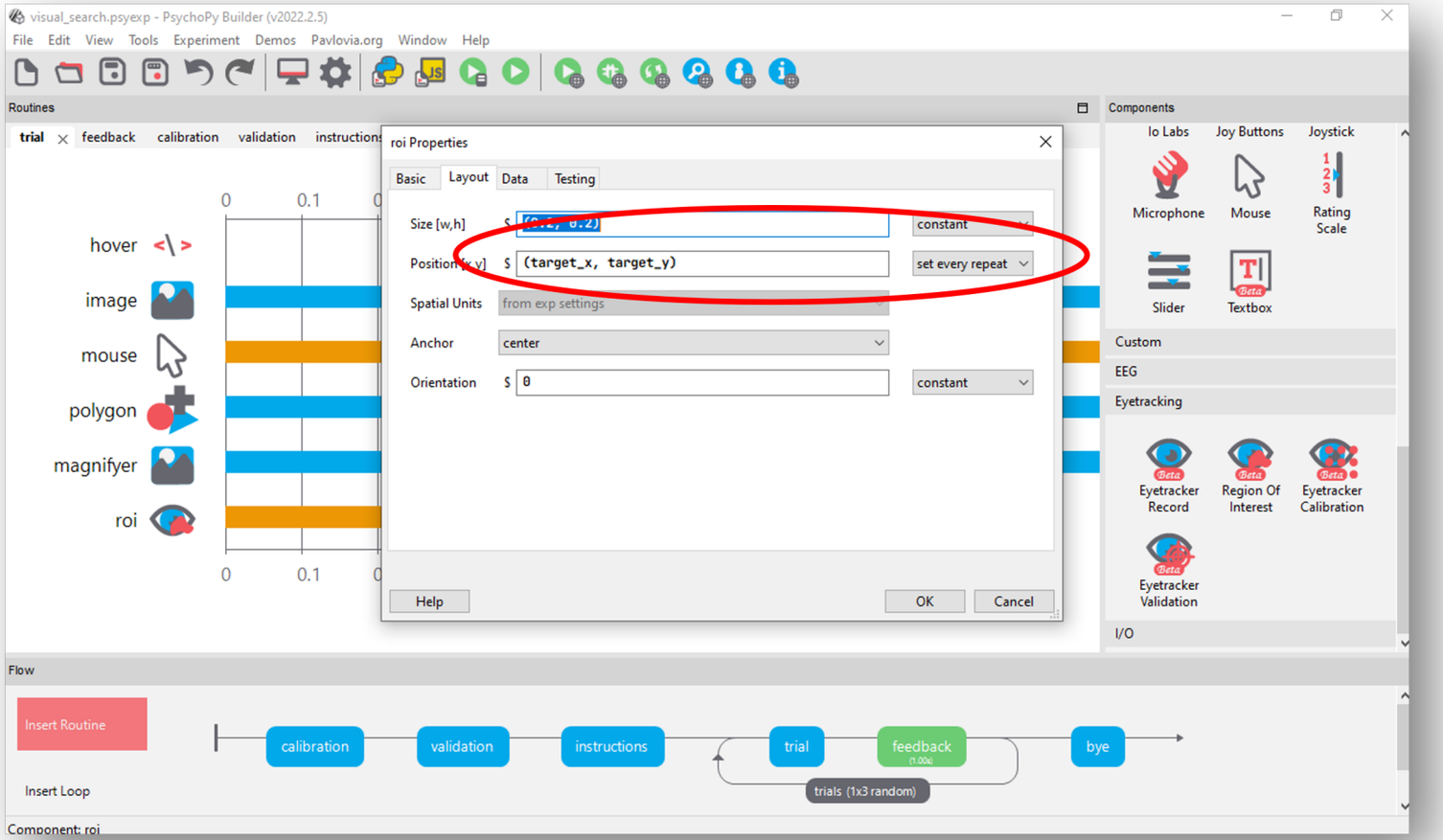
Here we start the tracker in the instructions and stop it in the “bye” routine.

Using ROI components



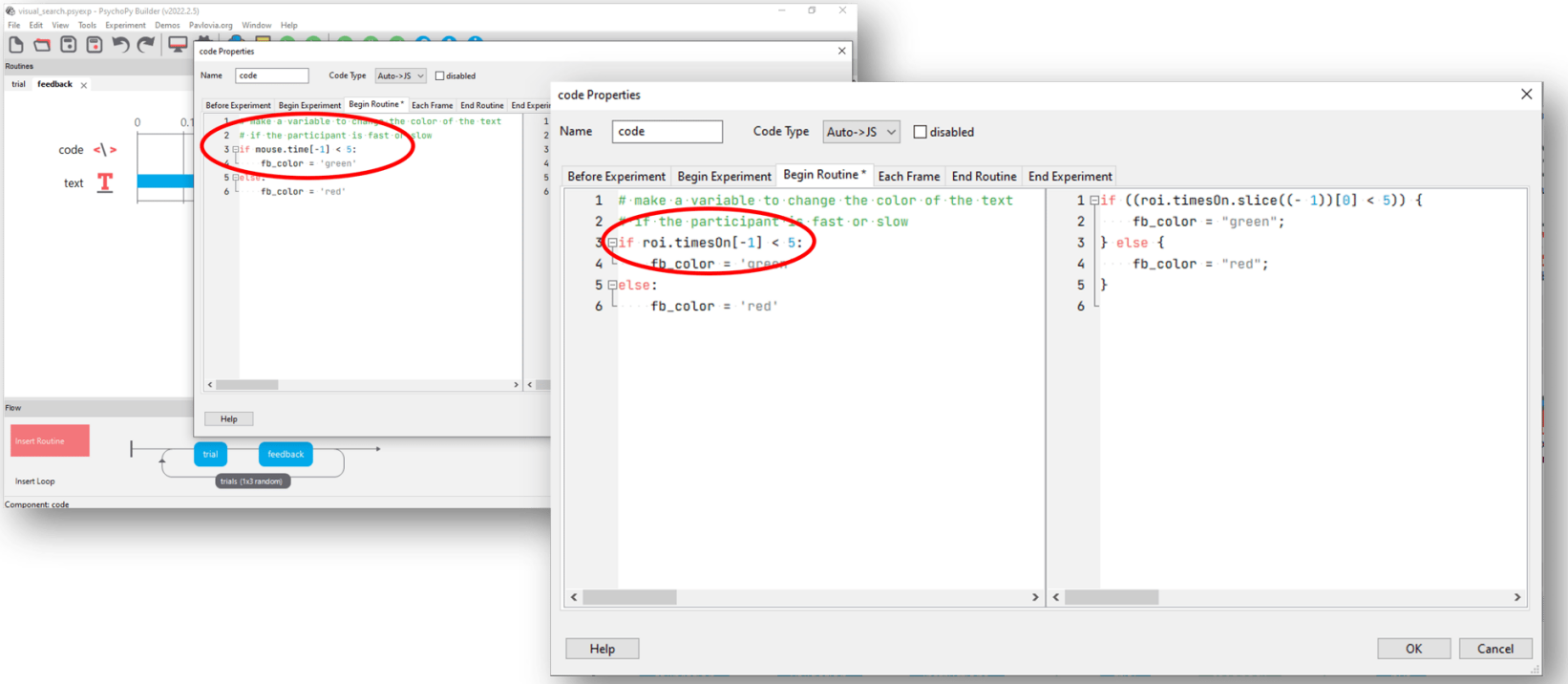
We can set our trial to end when the participant looks at the ROI for a set period of time…

Using ROI components

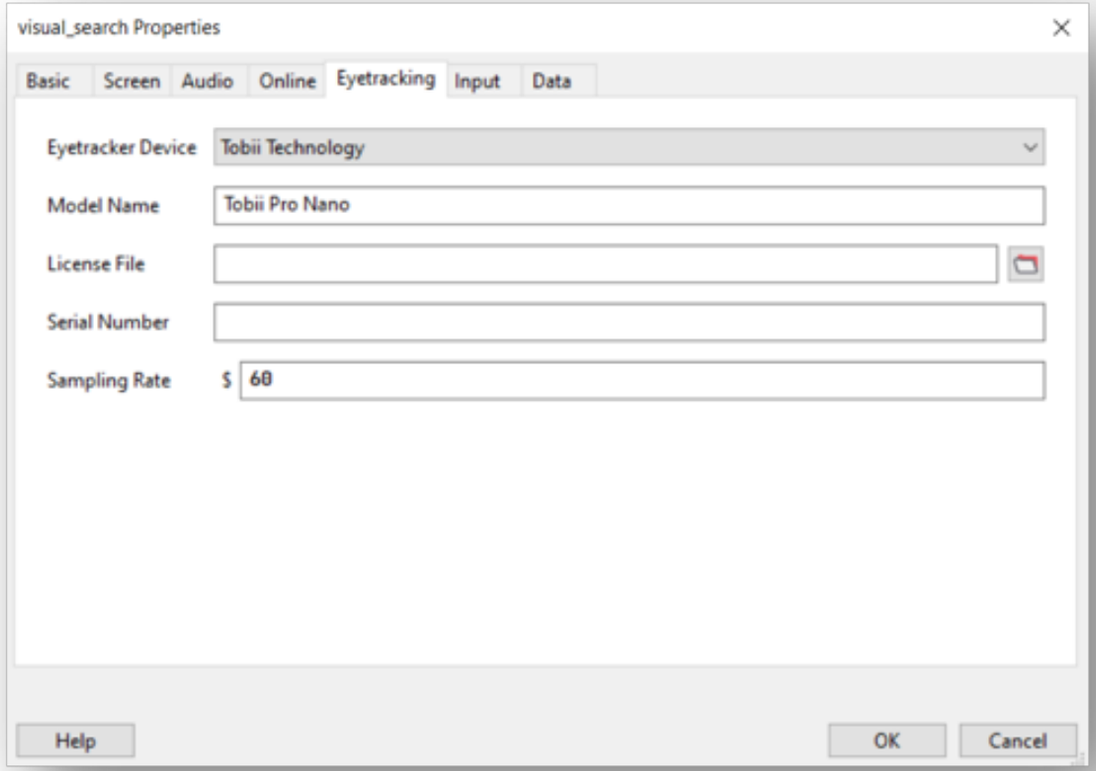


Since we know where the target is on every trial, we can update the region of interest location on every trial too...

Feedback based on ROI data



Configure your experiment to your eye tracker



Exercise (10 mins)

- Make the picture of the magnifying glass track/move with your eye position.