



---

## GETTING STARTED WITH ADHAWK MINDLINK

---

Version 1.0

September 16, 2022

STRICTLY CONFIDENTIAL

Copyright AdHawk Microsystems 2022

AdHawk Microsystems

410 Albert St, unit 102, Waterloo, ON, N2L 3V3, Canada

[info@adhawkmicrosystems.com](mailto:info@adhawkmicrosystems.com)

# Contents

<b>Getting started with AdHawk MindLink</b>	<b>1</b>
<b>What is eye tracking?</b>	<b>2</b>
Human eye movements . . . . .	2
How does eye tracking work? . . . . .	3
Eye tracking applications . . . . .	3
<b>How does AdHawk eye tracking work?</b>	<b>5</b>
Hardware components . . . . .	6
Software components . . . . .	6
AdHawk Backend . . . . .	7
<b>System requirements</b>	<b>8</b>
<b>Installing the SDK and examples</b>	<b>9</b>
<b>Fitting your MindLink glasses</b>	<b>10</b>
Changing the nosepiece . . . . .	10
A proper fit . . . . .	10
Selecting a nosepiece . . . . .	12
Installing a nosepiece . . . . .	12
Using ear hooks . . . . .	12
Prescription lenses . . . . .	14
<b>Quick Start</b>	<b>15</b>
<b>Auto-tune</b>	<b>16</b>
<b>Calibration</b>	<b>17</b>
<b>Device calibration</b>	<b>18</b>
<b>Testing the quality of your setup with validation</b>	<b>19</b>
<b>Creating an eye tracking app</b>	<b>20</b>
AdHawk Python API . . . . .	20
Return codes . . . . .	20
Dependencies . . . . .	20
Example code . . . . .	21
Data streams . . . . .	22
Asynchronous vs synchronous operation . . . . .	23
<b>What are the important steps within AdHawk eye tracking?</b>	<b>24</b>

<b>Troubleshooting</b>	<b>25</b>
MindLink camera preview is not working properly . . . . .	25
One or both eyes aren't found during a Quick Start or Auto-tune. . . . .	25
<b>Glossary</b>	<b>26</b>



# Getting started with AdHawk MindLink

# What is eye tracking?

Eye tracking is the process of observing eye behavior so you can learn about the point of gaze. Eye tracking can be used to:

- learn what someone is looking at
- learn about aspects of someone's mental state (e.g., fatigue, concussion, level of focus, etc.)

## Human eye movements

Before diving into the AdHawk SDK and writing an application that incorporates eye tracking data, it helps to know about some of the more common eye movements.

### Fixations

One of the most common eye events occurs when the eyes aren't moving. *Fixations* occur when we focus on something for a period of time. Fixations range in duration from tens of milliseconds to several seconds.

### Saccades

*Saccades* occur when our eyes move between fixations. These movements are the fastest movements the human body can perform; they can reach a peak velocity of 500 degrees/second (although the amplitude of saccades are usually in the 4 to 20 degree range).

Saccades are an event within the AdHawk API—you can register to receive notifications when they occur.

### Smooth pursuit

*Smooth pursuit* is another form of eye movement but one which requires something to follow (unlike saccades which can happen without stimuli). Consider a bird flying across the sky: our brain enters a feedback loop where we can smoothly track the bird as it moves. Smooth pursuit velocities range from 10 to 30 degrees/second (compared with the much faster saccades).

### Blinks

Blinking closes the eyelids to moisten the eye. We often think of blinking as not involving eyeball movement, but while fixating on something straight ahead, the eyeball actually rotates downward slightly during a blink.

Blinks are an event within the AdHawk API; you can register to receive notifications when they occur.

### Vestibulo-ocular reflex (VOR)

The vestibulo-ocular reflex acts to stabilize your gaze (on a visual target) while your head is moving. This helps stabilize the image on your retina as the environment may be changing. To see VOR in action, fixate

on a word in this sentence and move your head from right to left; notice how your gaze remains locked to the word while your head moves?

One of the calibration methods supported by AdHawk relies on VOR ('fixed gaze').

### Vergence

*Vergence* involves eyes moving in opposite directions. This is done to converge or diverge, usually to focus on items that are close up or far away, respectively.

Vergence information is available as part of the **gaze data stream**.

## How does eye tracking work?

You can generally tell where someone is looking by looking at their eyes. Eye tracking is nothing more than a really accurate, really fast way of the same thing.

Some eye tracking systems use a combination of small cameras pointed at your eyes and image analysis to figure out where you're looking.

AdHawk eye tracking doesn't use cameras and image processing (both require a fair bit of power), instead using incredibly small (MEMS-scale) mirrors to shine a lower power IR light across your eyes. Reflections from your eyes are captured by photodetectors and feed into a mathematical model. This model allows us to determine where you're looking very accurately ( $< 1$  degree MAE) and at high rates (up to 500 Hz).

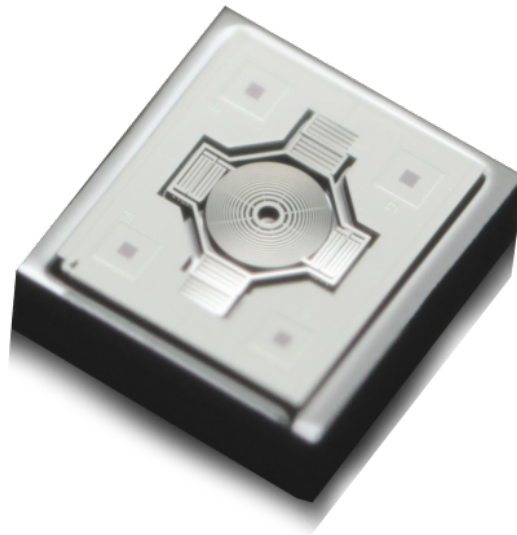


Figure 1: MEMS mirror

## Eye tracking applications

Eye tracking can be used in several different ways:

- **Interaction.** In the same way as a mouse or your finger can move across a screen, so too can your gaze act as a pointer. Consider actions like blinks to be like mouse clicks or taps. This sort of direct interaction can be as rich as other modalities, and can allow us to communicate our interest and intent through eye behavior.

- **Observation.** Eye behavior is becoming increasingly valuable in unlocking mental and cognitive states. Eye tracking can be used to observe and collect information about these states in an unobtrusive way. Data collected can be used to help identify, diagnose, or assess recovery for certain medical conditions (e.g., concussion, dementia, etc.).
- **Research.** Although a specialized form of observation, eye tracking has been used in market research (to learn more about people's behavior in retail environments, for example) and lab settings (e.g., psychology, neuroscience, usability, etc.) for a long time.

We've posted a few interesting examples of eye tracking on our [Hack the North website](#).



## How does AdHawk eye tracking work?

Traditional eye tracking systems point a camera at your eye and use the image of your pupil to determine where you're looking.

The AdHawk eye tracking system does not use a camera to calculate your gaze. Instead, a path of low power IR light is drawn on your eyeball and photodetectors capture the reflections. Those glints tell enough about the position and orientation of your eyes to figure out where you're looking.

Another key component of the MindLink glasses is the front-facing camera. This captures the scene and helps with correlating your gaze with something you may be looking at (this is especially important during calibration, but also great for seeing what someone is looking at or hoping to interact with).

With this approach we can calculate your gaze (where you're looking), the vergence angle (angle between your individual eye gaze vectors), and pupil size and position.



## Hardware components

The hardware components within the AdHawk eye tracking system are:

1. Scanner. A MEMS device that directs low power IR light across your eyes.
2. Photodetectors. Sensors that pick up on the IR light reflections.
3. Front-facing camera. Also known as a world or scene camera; captures the view of the MindLink wearer so that gaze can be overlaid, markers can be detected, etc.

Some important features and specifications of the AdHawk MindLink system include:

Feature	Specification
Calibrated range	40 × 25 degrees (width × height)
Gaze Mean Absolute Error	<1.0 degrees
Gaze data rate	Up to 500 outputs per second
Calibration points	user configurable from 1–9
Latency	3 ms
Weight	27 g

Eye tracking features:

- binocular gaze tracking and output
- vergence output
- pupil size output
- robust tracking under sunlight
- single click eye tracking
- slip tolerant eye tracking
- export eye tracking video
- export eye tracking data
- gaze validation function

Front-facing camera:

- 82 degree FOV
- 16:9 aspect ratio
- supports 720p, 1080p @ 30 FPS

Prescription lenses:

- a limited number of prescription lenses (standard diopter, non-astigmatism) are available from AdHawk folks at their sponsor booth

## Software components

Depending upon which platform you're building with, your software stack will look a bit different.

With an AdHawk MindLink, your minimal stack will include:

- your application, which relies in part on...
- the AdHawk python SDK, which communicates with...
- the AdHawk Backend

You may find you use other libraries or code to help with things like a GUI or visualization. Some of the sample applications we've provided rely on Qt (for the interface), or other third party libraries (e.g., numpy, OpenCV). Those can be installed using the included `requirements.txt` file for each example.

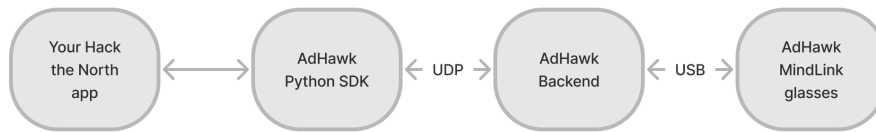



Figure 2: Python architecture

## AdHawk Backend

The AdHawk Backend program acts as a go-between for the MindLink glasses and applications who want to use eye tracking. In order to receive eye tracking data streams in your application, AdHawk Backend needs to be running.

Run AdHawk Backend from your computer. After launching, the AdHawk Backend icon  will appear in the system tray. Keep the AdHawk Backend service running in the background while eye tracking.

The AdHawk Backend will continue to run until you close it. You can launch, run, and close down your application repeatedly with the same instance of AdHawk Backend running. The service can be shutdown by right clicking on the system tray icon, and selecting *Quit AdHawk Backend*.

# System requirements

To create an eye tracking application using the AdHawk MindLink glasses, you will need a computer that:

- has an Intel Core i5 or equivalent CPU
- has 8 GB RAM or more
- has 1 GB disk storage free
- has 1 free USB port (adapters and cables provided as part of the kit)
- runs Windows 10, Windows 11, or Linux

# Installing the SDK and examples

To install the Python SDK:

```
pip install adhawk
```

This will install the contents of the SDK to your python `site-packages` directory. From there, you can `import adhawkapi`, `import adhawkapi.frontend`, etc. into your python code.

To install the example programs, visit the [AdHawk github page](#). Clone the repo so you've got the code locally.

- we recommend you work within a [Python virtual environment](#)
- don't forget to install the dependencies of a given sample application (see the `requirements.txt` in each example's directory); this is also described by the [README.md](#) for the github repo

# Fitting your MindLink glasses

Eye tracking systems are largely comprised of physical sensors positioned near the eyes themselves. To ensure great eye tracking, you need to optimize the position of those sensors.

You can adjust the fitting of MindLink glasses by: \* changing the nosepiece \* using ear hooks

If you can't seem to get a good fit with any of the nosepieces or suggestions, visit the AdHawk sponsor booth.

## Changing the nosepiece

The MindLink kit ships with a set of four different nosepieces. You can tell the nosepieces from each other by their size, but also the small colour marking. You should use the correct nose piece for each user to improve eye tracking.



Figure 3: Side view of four MindLink nosepieces

## A proper fit

When fitting the glasses, your goal is to have the pupil located midway between the top and bottom of the frame.

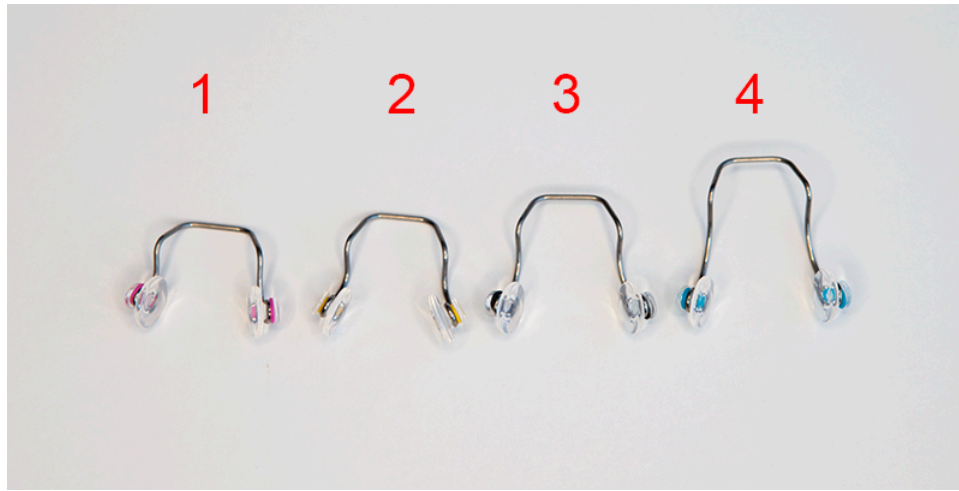


Figure 4: Back view of four MindLink nosepieces



Figure 5: Correct Fit Example: Front

## Selecting a nosepiece

Each nosepiece can: \* change the height at which the glasses sit on your face \* change the distance the glasses sit from your eye (the ‘relief’) \* change both the height and relief

As you increase the nosepiece number from 1 (pink) up to 4 (blue), the glasses will get higher and farther away from your face. The table below describes the change in glasses fit between the four nosepieces.

Nosepiece	Purpose
1 (pink)	Reduce height
2 (yellow)	Reduce relief (distance from face)
3 (black)	This is the baseline nosepiece
4 (blue)	Increase height

The nosepieces are designed to accommodate different face and nose shapes, to maintain a consistent glasses position relative to the eyeball. They follow the trend shown below:

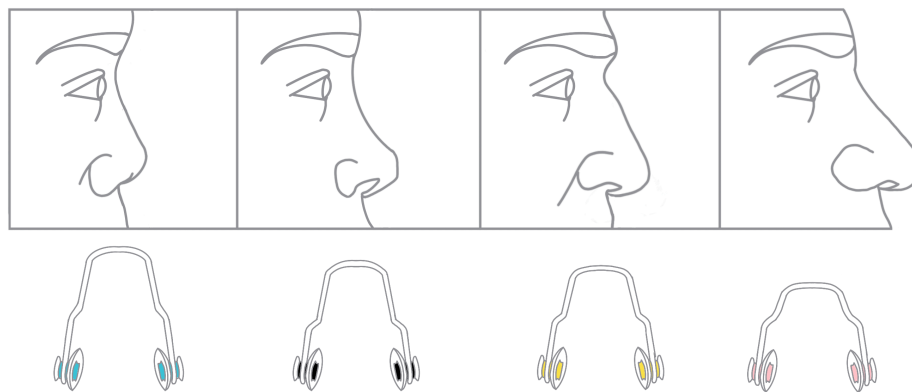


Figure 6: Face shapes and recommended nosepieces

## Installing a nosepiece

You can swap the nosepiece by gently pinching the nose pads towards each other and sliding the bridge of the nosepiece into or out of the channel on the glasses. Use your index finger of the opposite hand to guide the nosepiece into place. The nosepiece will snap in place once inserted fully.

## Using ear hooks

Your MindLink kit includes two ear hooks. Install the ear hooks on the arms of the glasses to reduce slipping. The ear hooks should work to maintain an optimal fitting, not work against your nosepiece selection.

To install the ear hooks: \* Slide the ear hook opening over the end of the glasses’ arm, starting with the pointed upper edge of the arm. \* Rotate the ear hook downwards to fully enclose the end of the arm with the ear hook. \* Slide and wiggle the ear hook until you’ve moved it to the desired location on the arm.



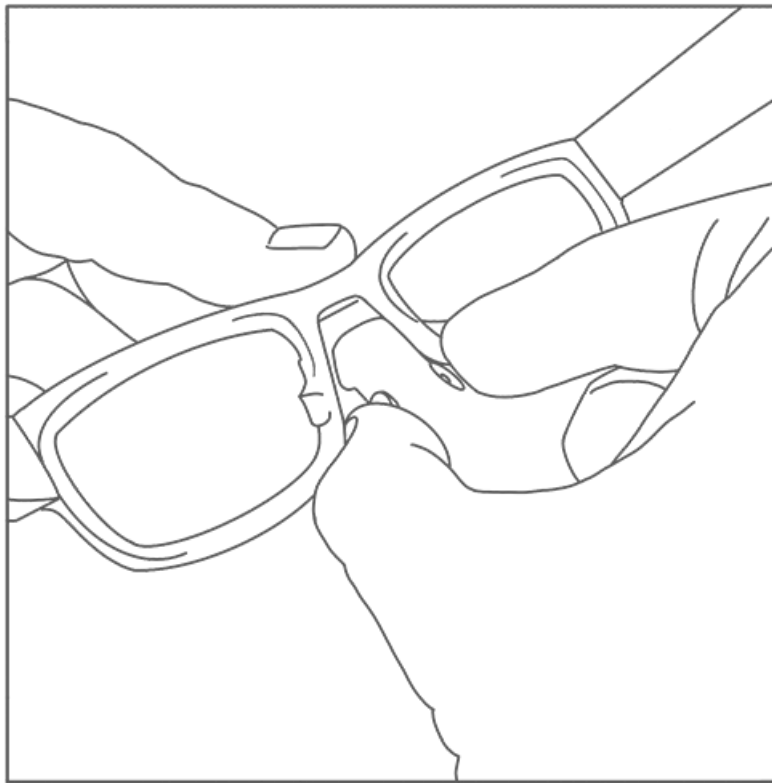


Figure 7: Installing a nosepiece by pinching it

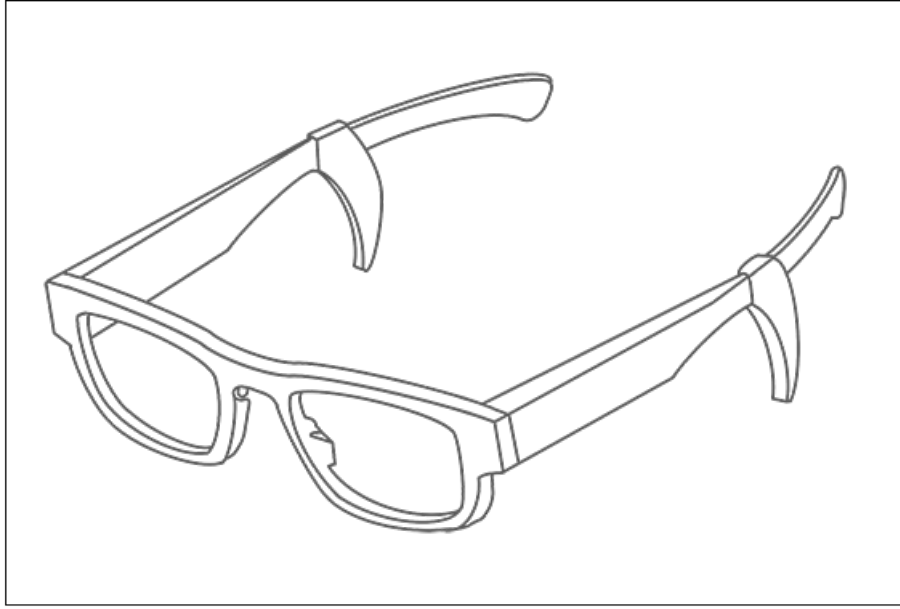


Figure 8: MindLink glasses with ear hooks installed

## Prescription lenses

AdHawk MindLink glasses can have their non-prescription lenses swapped out for prescription lenses. A limited number of prescription lenses (standard diopter, non-astigmatism) are available from AdHawk folks at their sponsor booth if you need them.

# Quick Start

The *Quick Start* activity is intended to get your eye tracking up and running quickly. It works best with a saved calibration.

# Auto-tune

The *Auto-tune* activity will scan your eye, then tweak the size and position of the scan box used during eye tracking. Auto-tune is part of Quick Start, but can also be run independently.

# Calibration

*Calibration* reconciles the gaze with the real world. To calibrate, a marker—something to fixate on—is shown on a monitor. Using the front-facing camera, we can determine where it is. Then we capture the gaze when someone is looking at the marker. We create a model that allows us to map a measured gaze with a position in the real world.

# Device calibration

Infrequently, the eye tracker needs to be calibrated against a known environment or set of inputs. This is different than regular calibration (see above) and involves a calibration *fixture*. If you need to perform a device calibration (e.g., you get a return code of 17 when calling into the API), visit AdHawk's sponsor booth.

# Testing the quality of your setup with validation

If calibration allows us to create a model to map gaze to real world objects, validation is a way of testing that model to see how accurate it is. The process of validation is very similar to calibration in that it involves looking at markers in a known location in the real world.

Validation does not need to be done prior to performing eye tracking, but it is a good way to see if you're set up properly. Aim for a mean absolute error (MAE) of 1 degree or less for the best experience with eye tracking.

# Creating an eye tracking app

Your application can use the AdHawk MindLink glasses by using the Python software development kit (SDK). The SDK includes APIs to trigger a calibration, register for gaze streams, and so on.

## AdHawk Python API

### Return codes

When using the Python SDK, return codes may be sent back to your calling code. These provide status information about the call you've made or the system in general.

If a call is successful, the ack code will be SUCCESS (0). Non-zero ack codes indicate something went wrong or a situation where additional setup work is required.

```
SUCCESS = 0
FAILURE = 1
INVALID_ARGUMENT = 2
TRACKER_NOT_READY = 3
EYES_NOT_FOUND = 4
RIGHT_EYE_NOT_FOUND = 5
LEFT_EYE_NOT_FOUND = 6
NOT_CALIBRATED = 7
NOT_SUPPORTED = 8
SESSION_ALREADY_RUNNING = 9
NO_CURRENT_SESSION = 10
REQUEST_TIMEOUT = 11
UNEXPECTED_RESPONSE = 12
HARDWARE_FAULT = 13
CAMERA_FAULT = 14
BUSY = 15
COMMUNICATION_ERROR = 16
DEVICE_CALIBRATION_REQUIRED = 17
```

You can find a full list of ack codes in the [API documentation](#).

A **DEVICE\_CALIBRATION\_REQUIRED** (17) ack code indicates that the AdHawk MindLink device itself needs to be calibrated. This doesn't happen often and is different than a calibration that occurs when setting up eye tracking for a specific user. If you receive this ack code, bring your glasses to the AdHawk sponsor booth so they can be calibrated.

### Dependencies

You may need to install dependencies as you use the example programs or build out your application. Some common libraries include:



- Qt (via PySide2; we use version 5.15.12)
- `pyqtgraph`; we use version 0.12.1
- `numpy`; we use version 1.20.3
- OpenCV (via `opencv-contrib-python-headless`; we use version 4.5.2.52)

## Example code

There are three sample applications to show you how to use key parts of the AdHawk MindLink python SDK:

### Simple data streaming (`stream_register_example.py`)

This example demonstrates:

- how to subscribe to gaze and event streams
- how to and handle data from gaze and event streams; data is dumped to the console

The example is written in Python and only depends on the AdHawk Python SDK. Code can be found on [AdHawk's github](#).

### Gaze in image (`camera_gaze_example.py`)

This example demonstrates:

- how to handle video from the MindLink front-facing camera
- how to overlay a gaze marker onto the video stream properly

The example is written in Python and depends on the AdHawk Python SDK and PySide2 (for Qt/the GUI). Code can be found on [AdHawk's github](#).

**Parallax** Parallax correction aims to bring the projected gaze and the camera image into the same focal plane. This ensures that the gaze position is accurate when mapped onto that image. If your calibration seems good but your gaze marker when mapped into the camera image seems off, you may need to adjust the gaze depth.

There are two settings that relate to parallax; both are adjusted using `set_camera_user_settings()`.

`GAZE_DEPTH` (where value is the gaze depth in metres)

```
api.set_camera_user_settings(adhawkapi.CameraUserSettings.GAZE_DEPTH, 0.5)
```

`PARALLAX_CORRECTION` (where 1 is enable—and the default—and 0 is disable)

```
api.set_camera_user_settings(adhawkapi.CameraUserSettings.PARALLAX_CORRECTION, 1)
```

See the [API documentation](#) for more information about these calls.

### Screen tracking (`screen_tracking_example.py`)

This example demonstrates:

- how to track gaze on a monitor
- how to generate ArUco markers and place them on screen (these are used by the MindLink camera for spatial positioning)

n.b. You may need to change the camera index in the call to `start_camera_capture`. The index you need to use will be system dependent based on which other webcams, etc. you have attached, but is often 0 or 1.

The example is written in Python and depends on OpenCV (for marker generation and recognition), and PySide2 (for Qt/the GUI). Code can be found on [AdHawk's github](#)

## Data streams

There are several streams of eye tracking data available. Some streams are always broadcast, such as the signal indicating if the eye tracker is ready. Other streams need to be subscribed to individually (like the gaze stream).

### Gaze stream

The best way to learn how to use the gaze stream data is by looking at `stream_register_example.py` (for a simple example of how to register and receive the gaze data stream).

```
def handle_gaze_data(*data):
    timestamp, xpos, ypos, zpos, vergence = data

self._api.register_stream_handler(
    adhawkapi.PacketType.GAZE,
    self._handle_gaze_data)

self._api.set_stream_control(
    adhawkapi.PacketType.GAZE,
    1,
    callback=(lambda *_args: None))
```

Gaze data is comprised of: - timestamp since the system started, in seconds (float32) - x position, in meters (float32) - y position, in meters (float32) - z position, in meters (float32)

Note that x, y, and z positions are the estimated coordinates of the user's gaze point relative to the *midpoint of the scanners*. More information about the AdHawk coordinate system and its relationship to the AdHawk MindLink glasses can be found in the [API documentation](#).

### Event stream

Several higher level eye events are recognized and made available as part of the event stream.

- **Blink.** This event is triggered as soon as either of the two blinking eyes opens. The blink event indicates the time window where both left and right eye blink events overlap in time.

Blink events include a timestamp and duration (ms). See the [API documentation](#) for more details. Blink events are also used in `stream_register_example.py`, one of the example programs.

- **Saccade.** This event is triggered as soon as either of the saccades (left or right in a combined-eye saccade) ends. The saccade event indicates the time window where both left and right saccade events overlap in time. This event is triggered as soon as the saccade in any of the eyes ends.

Saccade events include a timestamp (of the end of the saccade), duration (ms), and an amplitude (degrees). For more information, see the [API documentation](#).

These events can be used for interaction or captured as a way to record eye behavior.

### Gaze in screen stream

This stream provides normalized (x, y) coordinates of the gaze inside the screen. The screen is defined by a set of ArUco markers, often in the corners of the screen. x and y are float values (0–1) where (0, 0) is the top-left corner and (1, 1) is the bottom-right corner of the screen.

```
def handler(*data):
    timestamp, xpos, ypos = data

api.register_stream_handler(adhawkapi.PacketType.GAZE_IN_SCREEN, handler)
```

For more information, see the API documentation or the `screen_tracking_example.py` example source code.

### IMU stream

The MindLink glasses contain an inertial measurement unit (IMU). There are two data streams associated with the IMU:

- IMU data stream
- IMU rotation data stream

For more information about the IMU data streams, see the [API documentation](#).

### Asynchronous vs synchronous operation

You can chose to make Python API calls as either blocking or non-blocking.

To perform a non-blocking (or asynchronous) operation, provide a callback function as a parameter to be executed when the operation is complete. This mode of operation is useful for GUI applications.

If a callback function is not provided, the call blocks until a response is received. On success, the response is returned. On failure, an exception is raised.

For examples of both types of calling, see the [API documentation](#).

# What are the important steps within AdHawk eye tracking?

There are a few steps required to get eye tracking working well within your application:

1. Look at the section on fitting your MindLink glasses. You may need to tweak things as you go, but at least find a nosepiece that seems reasonable.
2. Connect your MindLink glasses to your computer.
3. Start AdHawk Backend. This application runs in the background but will be accessible via the System Tray (on Windows).
4. Establish a connection with AdHawk Backend (in code; see the example programs for the best way to do this). This step will allow you to communicate with the eye tracker itself.
5. Run a Quick Start and/or calibration to create a model that can be used for eye tracking.
6. Start developing your application!

# Troubleshooting

## MindLink camera preview is not working properly

In order to start capturing frames from the front-facing MindLink camera, you need to correctly indicate its camera device index. If there are multiple cameras connected (e.g., a webcam in addition to the MindLink glasses), the device index might need to be discovered by trial and error.

```
api.start_camera_capture(camera_device_index, resolution_index, undistort_image, callback=handle_camera,
```

If the MindLink camera images are not displayed using a specific device index, then another device index should be tried.

## One or both eyes aren't found during a Quick Start or Auto-tune.

More often than not, when this occurs, your glasses aren't fitting properly. Check that you're using the correct nosepiece **for you**, that your eyes are centered within the lenses (when viewed from the front), and that the glasses aren't slipping down your nose.

Occasionally, the windows covering the scanner will be dirty. Gently clean the window surface with the provided cloth.



Figure 9: Sensor windows

# Glossary

**AdHawk MindLink.** AdHawk MindLink glasses include a full eye tracking system within the frames.

**Anti-saccade.** A saccade *away* from a stimulus.

**ArUco markers.** Specialized (graphical) markers designed to facilitate computer vision. They are blocky black and white grids with a black border.

**Auto-tune.** A process to optimize where the eye tracking system will scan on someone's eyes.

**Calibration.** The process by which the relationship between a user's eye tracking setup—and thus their gaze—and the world is calculated, allowing for the 'real world' gaze target to be determined.

**Device calibration.** Infrequently, the eye tracker needs to be calibrated against a known environment or set of inputs. This is different than regular calibration (see above) and involves a calibration *fixture*. If you need to perform a device calibration (e.g., you get a return code of 17 when calling into the API), visit AdHawk's sponsor booth.

**Eye tracking.** The process of inferring the gaze point (and, in general, eye behavior) by using technology to observe someone's eyes.

**Fixation.** Fixations are when we focus on something for a period of time. Fixations range in duration from tens of milliseconds to several seconds.

**IMU.** Inertial measurement unit. The IMU is a sensor used to measure orientation and acceleration.

**IPD.** Interpupillary distance. The distance between the centres of your eyes/pupils (whilst looking straight ahead, far into the distance). This value is often part of your eyeglass prescription.

**MAE.** Mean absolute error. Used to characterize the quality of a calibration (the higher the MAE, the worse the calibration). MAE is calculated as part of a validation. The acceptable range of MAE depends on the type of calibration performed: the more points used during the calibration, the higher the expectations for its quality (and thus a lower MAE). For a 1 point calibration, the largest MAE 'accepted' is 2°; for a 9 point calibration, a 1° MAE is the upper limit.

**Saccade.** The rapid movement of both eyes to shift the centre of gaze to a new portion of the visual field.

**Tracker.** The collective term for the hardware and software used to track a single eye.

**Quick Start.** A process to get the eye tracker set up quickly. You can trigger a Quick Start via the API.

**Validation.** The process of having a user look at a series of targets post-calibration. By comparing the calculated gaze to known target position, the validity of the calibration can be assessed.

**VOR.** Vestibulo-ocular reflex. Reflex triggered by the vestibular system (related to the inner ear and orientation sensing) to ensure that the visual signal to the eye is stabilized during movement. One of the calibration modes ('VOR') we use involves keeping your gaze fixed on a point and moving your head to align a 'cursor' to that fixed marker; this takes advantage of the VOR. See [https://en.wikipedia.org/wiki/Vestibulo%20ocular\\_reflex](https://en.wikipedia.org/wiki/Vestibulo%20ocular_reflex)