Writing Programs for ExpEYES

Whether you're just starting your programming journey, a curious student eager to learn, or an aspiring engineer, this manual will be your trusty companion on this exciting adventure. With our device's Python library, you'll discover how to easily conduct experiments, collect data, and gain insights into various physical phenomena, all while having fun and building essential skills.

In the upcoming sections, We'll walk you through the Python library's features, using straightforward examples and hands-on exercises. You'll quickly see how you can create experiments, measure data, and even automate tasks to save time and improve accuracy.

Import and connect

the following two lines import the python library, and then attempt to connect to it. the instance `p` will now be used to access all the functions of ExpEYES. It is our gateway to the device.

Connecting to the device

```
from eyes17 import eyes
p = eyes.open()
```

If connected successfully, `p` will be automatically initialized. This process also uploads the unique calibration coefficients from the connected device.

```
In [1]: p
Out[1]: <eyes17.eyes.Interface at 0x7fef91b95120>
```

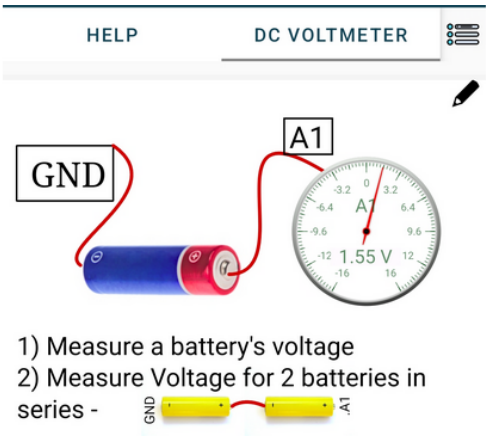
Failure to detect a device

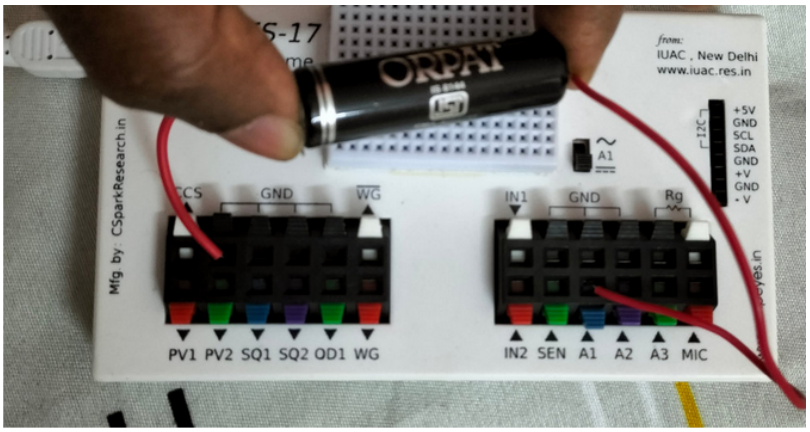
If connection fails, `Device opening Error` will be printed, and the `p.connected` variable will be set to False. After properly connecting, you can either recreate `p`, or call `p.__init__()`

Simple Test

Read A Voltage

```
from eyes17 import eyes
p = eyes.open()
print(p.get_voltage('A1')
```





Functions : Accessing the hardware

Analog | Waveforms | Digital | Sensors

- Analog Measurements: Voltages, Oscilloscope calls etc
  - Voltage Measurement
  - Capture calls
  - Capture configuration such as trigger, select_range etc
  - Code Examples
- Analog Output: Set Voltages
  - PV1, PV2
- Waveform Generators: configure sine, triangle, square wave outputs
  - sine wave frequency, amplitude, shape configuration
  - square wave 1,2 frequency, duty_cycle setting.
- Digital I/O, Timing: Logic Levels, and Timing measurements
  - read and set logic levels on digital pins
  - timing measurements
- Capacitance, Resistance:
  - Measure Capacitance

- ✅ Measure Resistance
- ✅ I2C Sensors: Read data from sensors
  - ✅ I2C Function Calls
  - ☑ Document Sensors

The following is an unsorted list. head over to relevant sections for detailed usage docs and examples

Function call summary

METHOD	DESCRIPTION
get_version	Returns a version string
reconnect	reconnect to the device. Needed if the USB cable was replugged
guess_sensor	scans the I2C bus, and returns a list of sensor options
get_resistance	measure resistance between SEN and GND
capture1	Oscilloscope: Single Channel
capture_action	Oscilloscope: Single Channel with an initial action such as set_state
capture2	Oscilloscope: 2 Channels
capture4	Oscilloscope: 4 Channels
capture1_hr	Oscilloscope: Single Channel . High Resolution
capture_traces	Oscilloscope: Initialization
capture_hr_multiple	Oscilloscope: High Res , multiple channels. Sequential Sampling*
fetch_trace	Fetch oscilloscope buffer from the hardware
oscilloscope_progress	Total samples collected.
configure_trigger	trigger level and channel for the oscilloscope
set_gain	gain for A1 or A2 inputs
select_range	Voltage Range Selection for A1, or A2 . Similar to the range knob on a multimeter
get_voltage	measure voltage . Input channel options A1, A2, A3, IN1, SEN
get_voltage_time	measure voltage as well as a timestamp. Returns a tuple
voltmeter_autorange	autorange the voltmeter

METHOD	DESCRIPTION
get_average_voltage	measure an averaged voltage value
get_high_freq	measure high frequencies on IN2 (>100KHz)
get_freq	measure frequency on IN2
MeasureInterval	timing measurements for digital signals on IN2 or SEN
MeasureMultipleDigitalEdges	timing measurements for digital signals on IN2 or SEN
SinglePinEdges	timing measurements for digital signals on IN2 or SEN
DoublePinEdges	timing measurements for digital signals on IN2 or SEN
get_states	get logic level on inputs SEN, IN2
get_state	get logic level on inputs SEN, IN2
set_state	set output state of OD1 / SQ1 / SQ2 / CS1-4/
stepper_move	Stepper motor movement
stepper_forward	Stepper motor movement
stepper_reverse	Stepper motor movement
set_multiplexer	Set CS1-4 to control analog multiplexers . Only on SEElab3
duty_cycle	measure duty cycle on IN2
r2rtime	Timing measurements on IN2/SEN. Rising Edge to Rising edge
f2ftime	Timing measurements on IN2/SEN. Falling Edge to Falling edge
r2ftime	Timing measurements on IN2/SEN.
f2rtime	Timing measurements on IN2/SEN.
multi_r2rtime	Timing measurements on IN2/SEN. Multiple rising edges.

METHOD	DESCRIPTION
set2rtime	Enable an output such as OD1/SQ1, and then measure time to a rising edge on IN2/SEN
set2ftime	Enable an output such as OD1/SQ1, and then measure time to a falling edge
clr2rtime	Turn off an output such as OD1/SQ1, and then measure time to a rising edge on IN2/SEN
clr2ftime	Turn off an output such as OD1/SQ1, and then measure time to a falling edge
capacitance_via_RC_discharge	Measure capacitance. For values >1uF
get_capacitor_range	estimate capacitance of capacitor connected on IN1-GND
get_capacitance	measure capacitance.
get_temperature	measure CPU temperature
get_ctmu_voltage	-
read_bulk_flash	-
write_bulk_flash	Do not touch. Especially locations 0,3 which are used to store the calibration coefficients.
set_sine	set frequency of sine wave on WG
set_wave	set wave type ‘sine’ / ‘tria’
set_sine_amp	set amplitude of the sine wave output on WG
load_equation	load an arbitrary waveform to WG using an equation
load_table	load a set of 512 points to the wave generator
set_sq1	set frequency of square wave on SQ1
set_sq2	set frequency of square wave on SQ2

METHOD	DESCRIPTION
set_sq1_dc	set the duty cycle of the square wave on SQ1
set_sq2_dc	set the duty cycle of the square wave on SQ2
set_pv1	Set the voltage output on PV1 (-5V to 5V)
set_pv2	Set the voltage output on PV2 (-3V to 3V)
servo	Set the angle of a servo motor connected to SQ1
sr04_distance	Measure distance using the SR04 sensor. SQ2→Trig, IN2←ECHO
sr04_distance_time	Measure Timestamped distance
save	save an array to a csv file

Static Variables

The instance `p` contains some static variables such as p.version . These are described here

Variable List

- connected: Boolean. Indicates the connection status
- version : A version string read from the connected device. Such as EJ-5.1
- version_number: Version Number
- aboutArray : print(p.aboutArray) to find out
- calibrated : Boolean. Indicates calibration status.
- verbose : set to True to get more verbose information
- WaveType : Indicates if WG output is enabled to sine/tria/arbit shapes.
- WaveMode : 'sine'/'tria'/'sqr2' . If sqr2 is enabled, WG output is disabled.
- sinefreq : Last set Frequency of Wave output on WG
- sqr1freq : Last set Frequency of Wave output on SQ1
- sqr1dc : Last set Duty Cycle of Wave output on SQ1
- sqr2freq : Last set Frequency of Wave output on SQ2
- sqr2dc: Last set Duty Cycle of Wave output on WG

Submodules

Access to communication buses such as I2C, SPI are available as submodules of the variable `p` . e.g. `p.I2C.scan()` returns a list of addresses of sensors/devices connected to the I2C bus.

I2C : Access the I2C Communication bus

- config
- start
- stop
- wait
- send
- send_burst
- restart
- simpleRead
- read
- read_repeat
- read_end
- read_status
- readBulk
- writeBulk
- scan

SPI : Access the SPI Communication bus

- set_parameters
- start
- set_cs
- stop
- send8
- send16
- send8_burst
- send16_burst
- xfer
- map_reference_clock