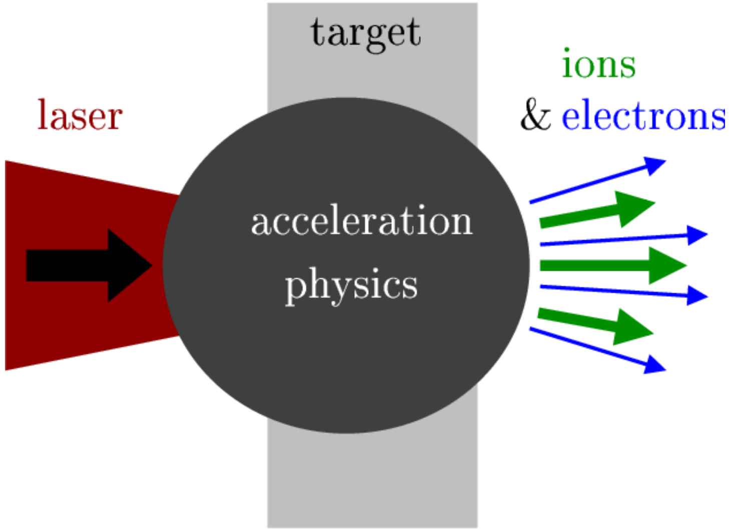
Scientific Instrumentation and Control with Web of Things

Vignesh Vaidyanathan (M.Sc. - Software Developer)

About Myself

- Software developer @ CALA, LMU, Garching, Germany
  - B.Sc. Electrical Engineering (2016) (India)
  - M.Sc. Physics (2023) (TUM)

- Centre for Advanced Laser Applications
  - Petawatt Peak Power Laser used for Ion Acceleration



Macchi, Andrea. “A Review of Laser-Plasma Ion Acceleration.” *arXiv: Plasma Physics* (2017): 2, Fig. 1

Requirement of Data-Acquisition

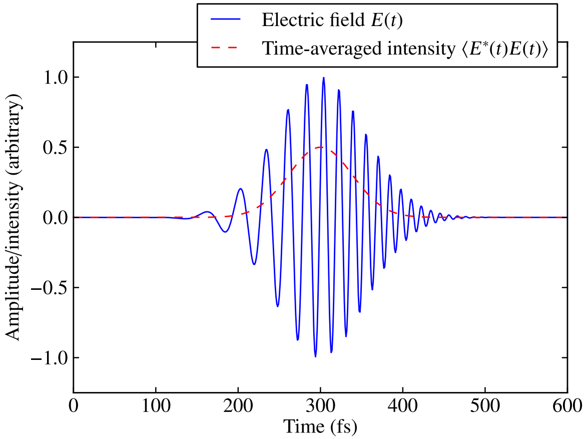
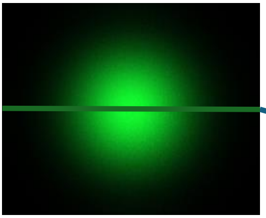


Fig. Light Pulse

Photodiode

Fig. Picoscope



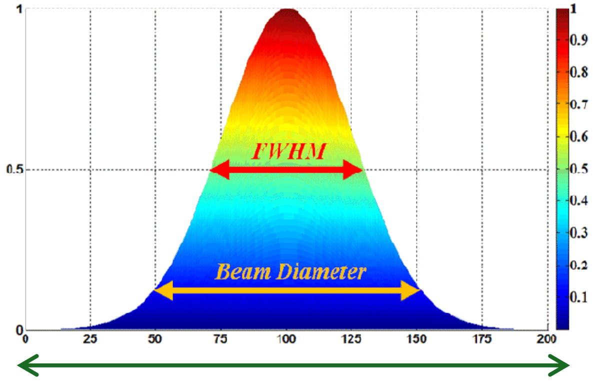


Fig. Pulse Parameters

Implementation - Data acquisition customized to experiment

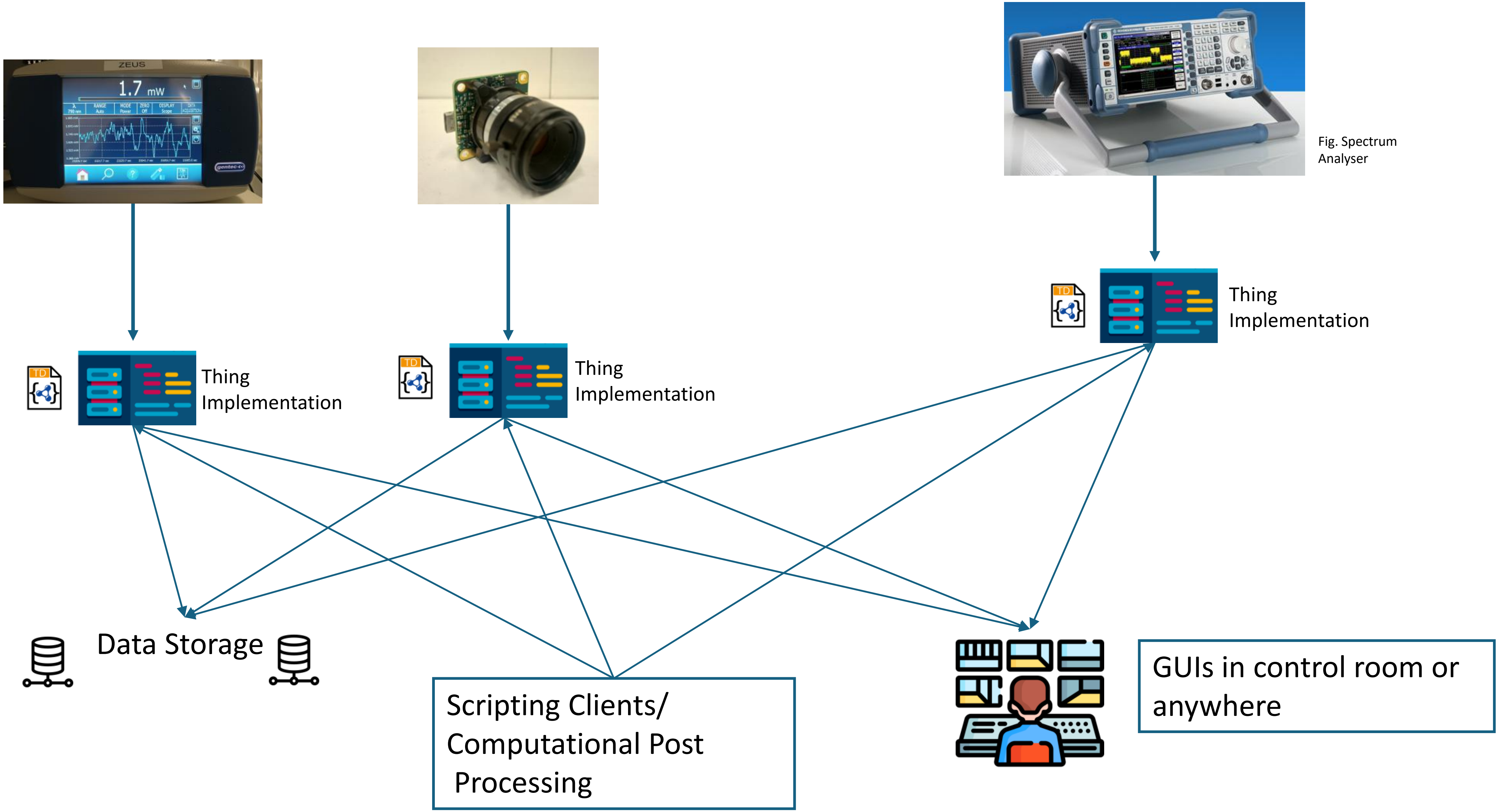


Fig. Spectrum Analyser

Thing Implementation

Thing Implementation

Thing Implementation

Data Storage

Scripting Clients/ Computational Post Processing

GUIs in control room or anywhere

Ultrafast Lasers & High Power Lasers

Pulse Characteristics

- Picosecond or femtosecond pulses
  (1 second to approximately 31.69 million years)
- Pulses per second – 1Hz, 10Hz, also, order of Millions
  (80 Million pulses per second for 80MHz)

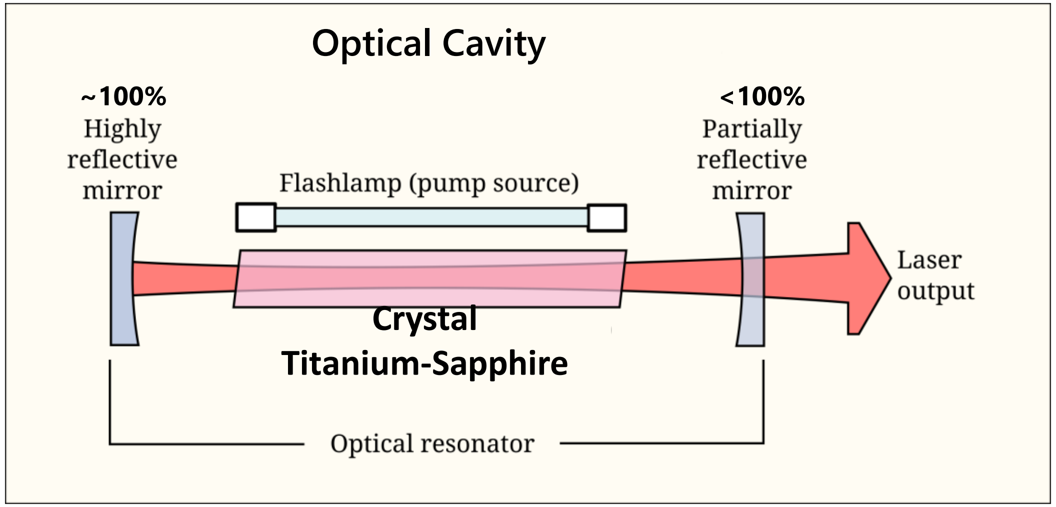


Fig. Optical Cavity

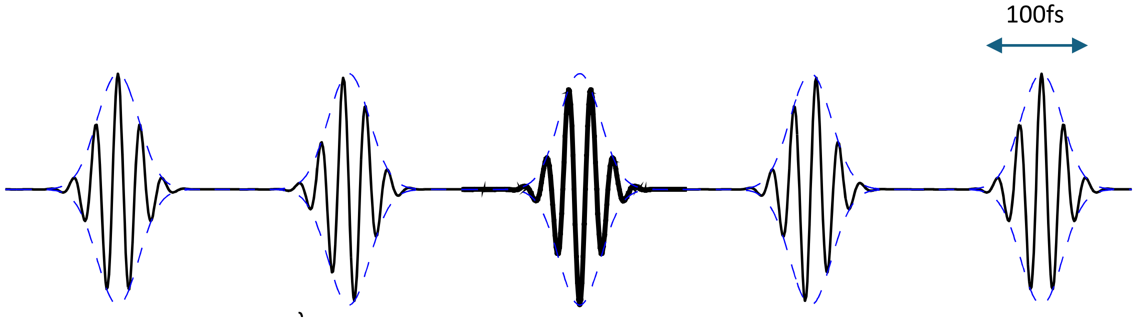
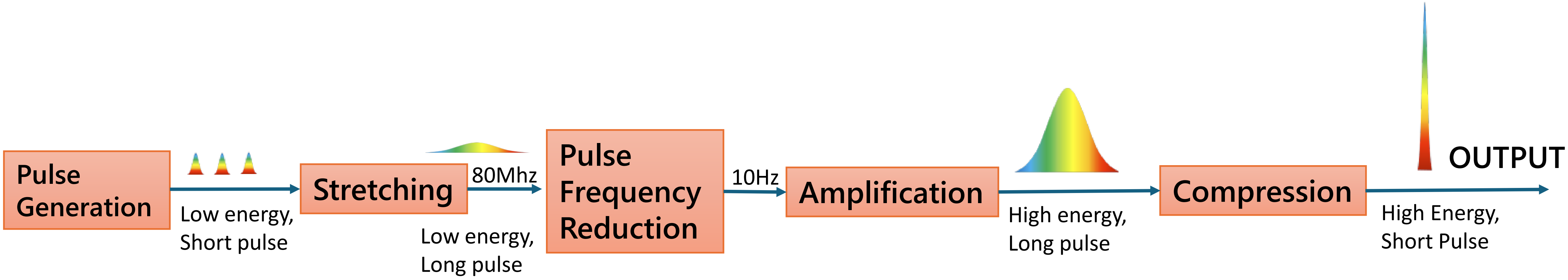


Fig. Pulse Train

Atomic Clocks, Spectroscopy

High Power Lasers

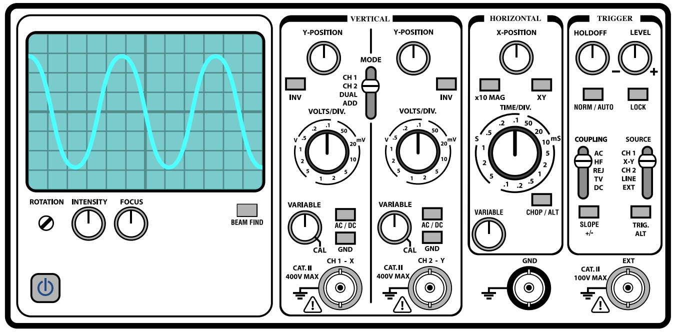


More often - **Custom built** to research or application requirements

- Ion Acceleration/Ionoacoustics
- Fundamental Physics

Coding/Implementation

- WoT Interaction Affordances (Example – Oscilloscope):
  - Properties
    - Time Resolution, Time Interval, Trigger Level, Range
  - Actions
    - Start Acquisition, Stop Acquisition
  - Events
    - Measurement Data Points or Measurement “Ready”

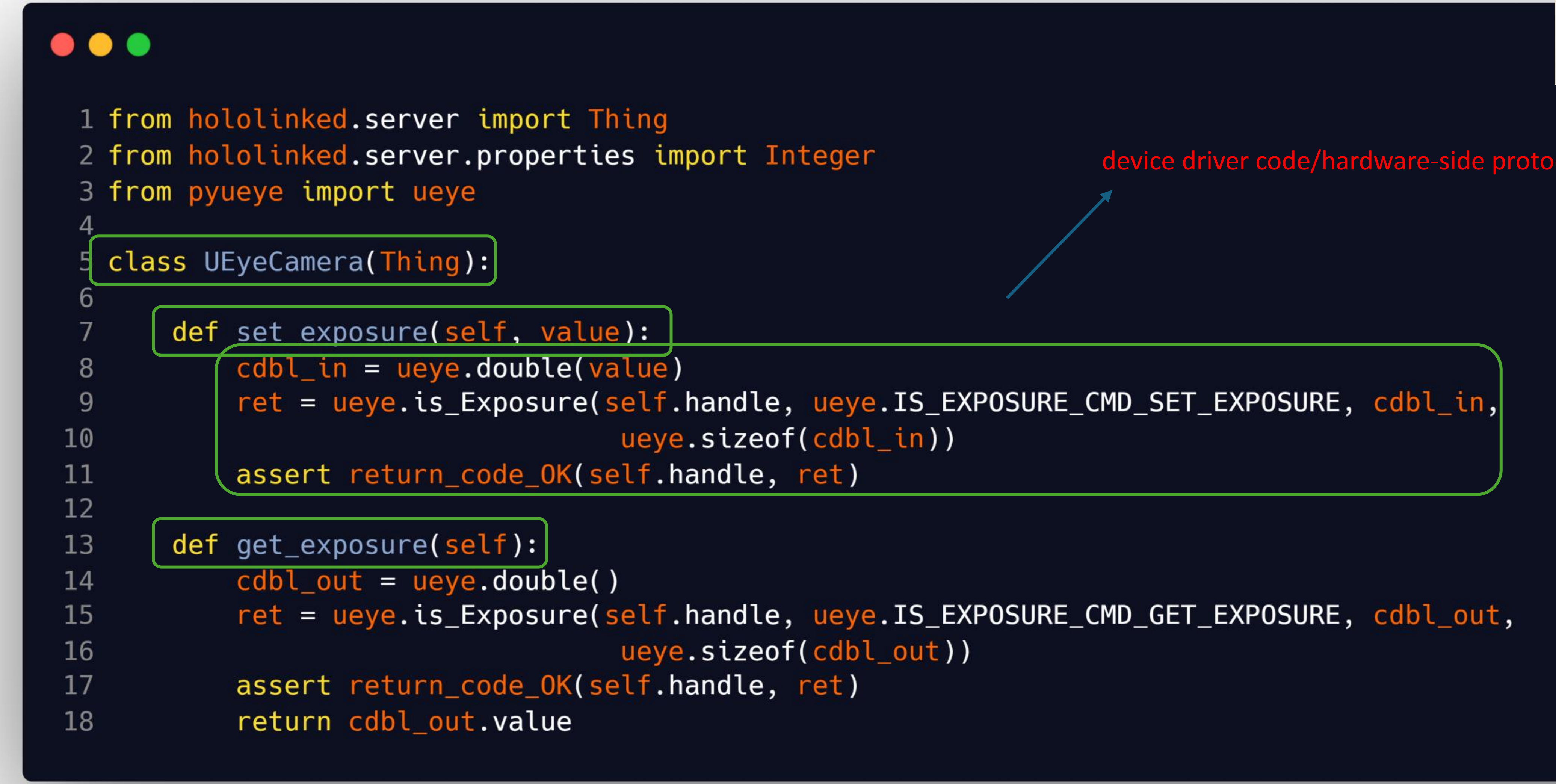


Cloudo, CC0, via Wikimedia Commons

- Sometimes – a state machine
  - If camera is capturing video, don’t allow (GUI) client to change exposure time (property write)
  - If acquisition is already running, don’t start acquisition again (action invoke)

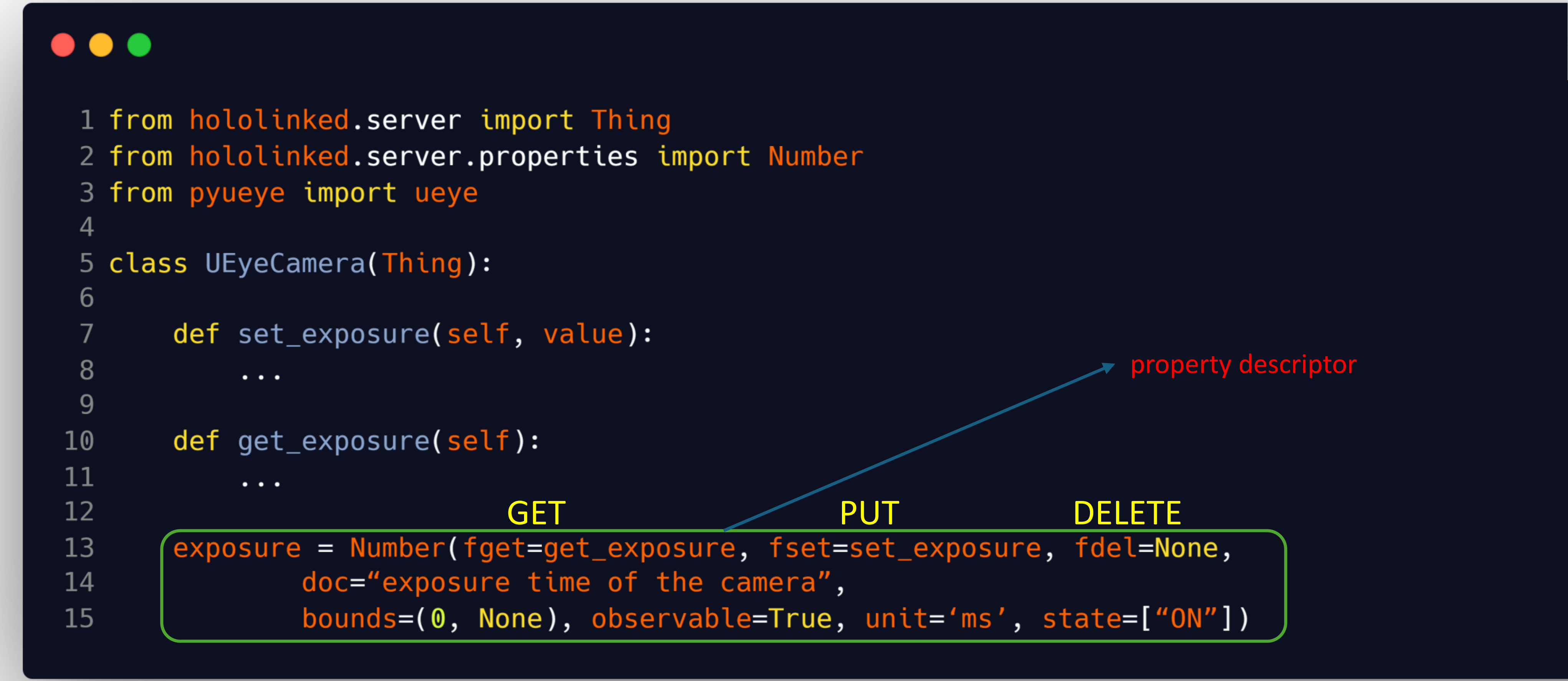
- Code Organization/Design
  - Object Oriented

Properties



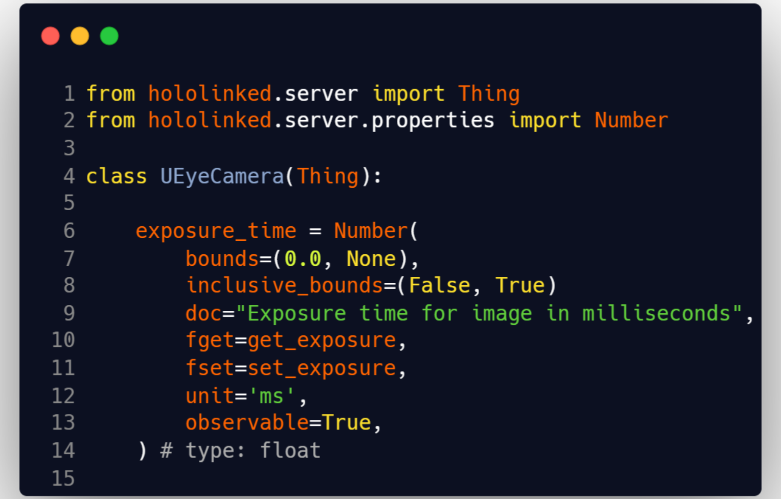
```python
from hololinked.server import Thing
from hololinked.server.properties import Integer
from pyueye import ueye

class UEyeCamera(Thing):

    def set_exposure(self, value):
        cdbl_in = ueye.double(value)
        ret = ueye.is_Exposure(self.handle, ueye.IS_EXPOSURE_CMD_SET_EXPOSURE, cdbl_in,
                                ueye.sizeof(cdbl_in))
        assert return_code_OK(self.handle, ret)

    def get_exposure(self):
        cdbl_out = ueye.double( )
        ret = ueye.is_Exposure(self.handle, ueye.IS_EXPOSURE_CMD_GET_EXPOSURE, cdbl_out,
                                ueye.sizeof(cdbl_out))
        assert return_code_OK(self.handle, ret)
        return cdbl_out.value
```

device driver code/hardware-side protocol

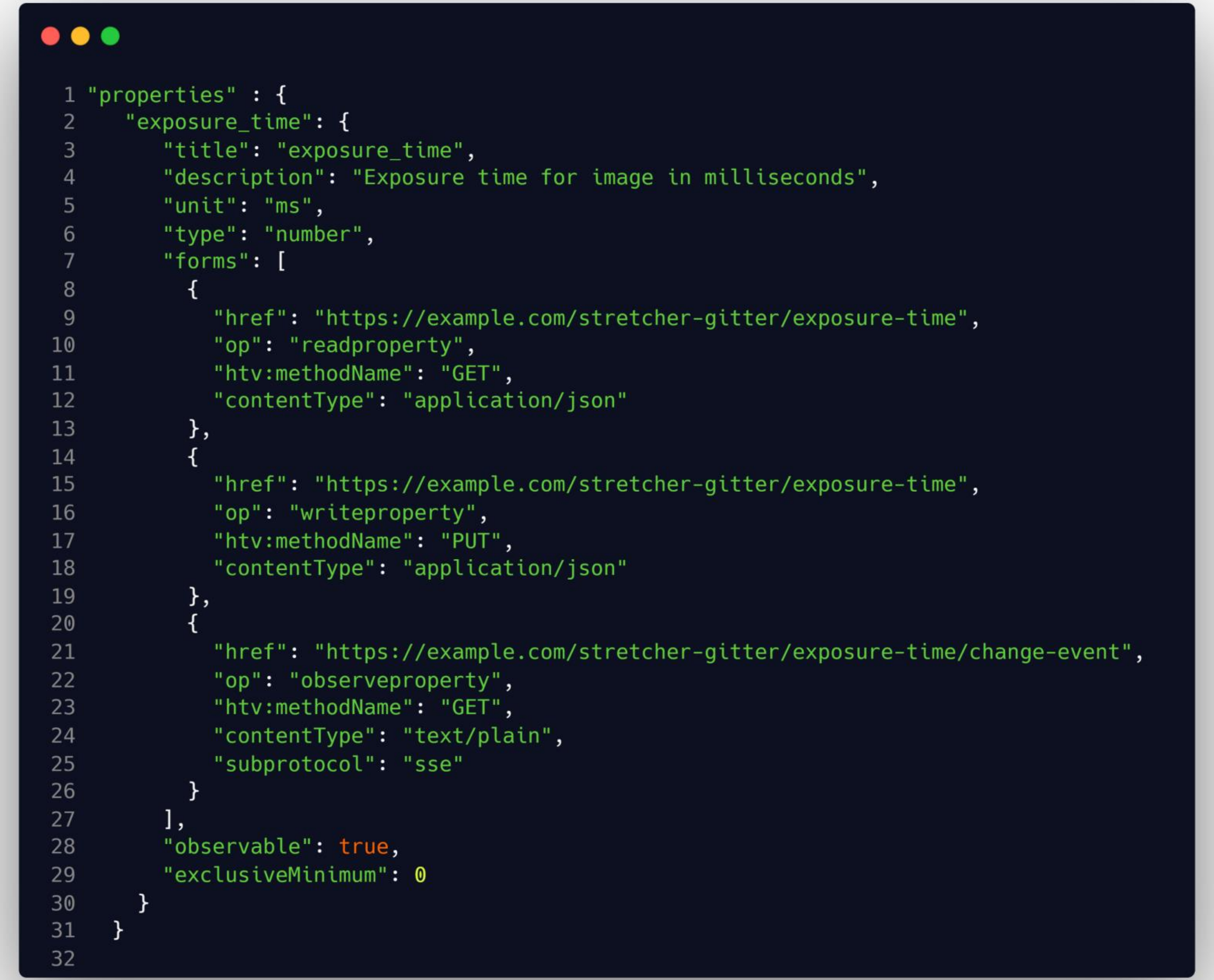


```python
from hololinked.server import Thing
from hololinked.server.properties import Number
from pyueye import ueye

class UEyeCamera(Thing):

    def set_exposure(self, value):
        ...

    def get_exposure(self):
        ...

    exposure = Number(fget=get_exposure, fset=set_exposure, fdel=None,
            doc=“exposure time of the camera”,
            bounds=(0, None), observable=True, unit=‘ms’, state=[“ON”])
```

```python
cam = UEyeCamera()
cam.exposure = 5000 # ms
```

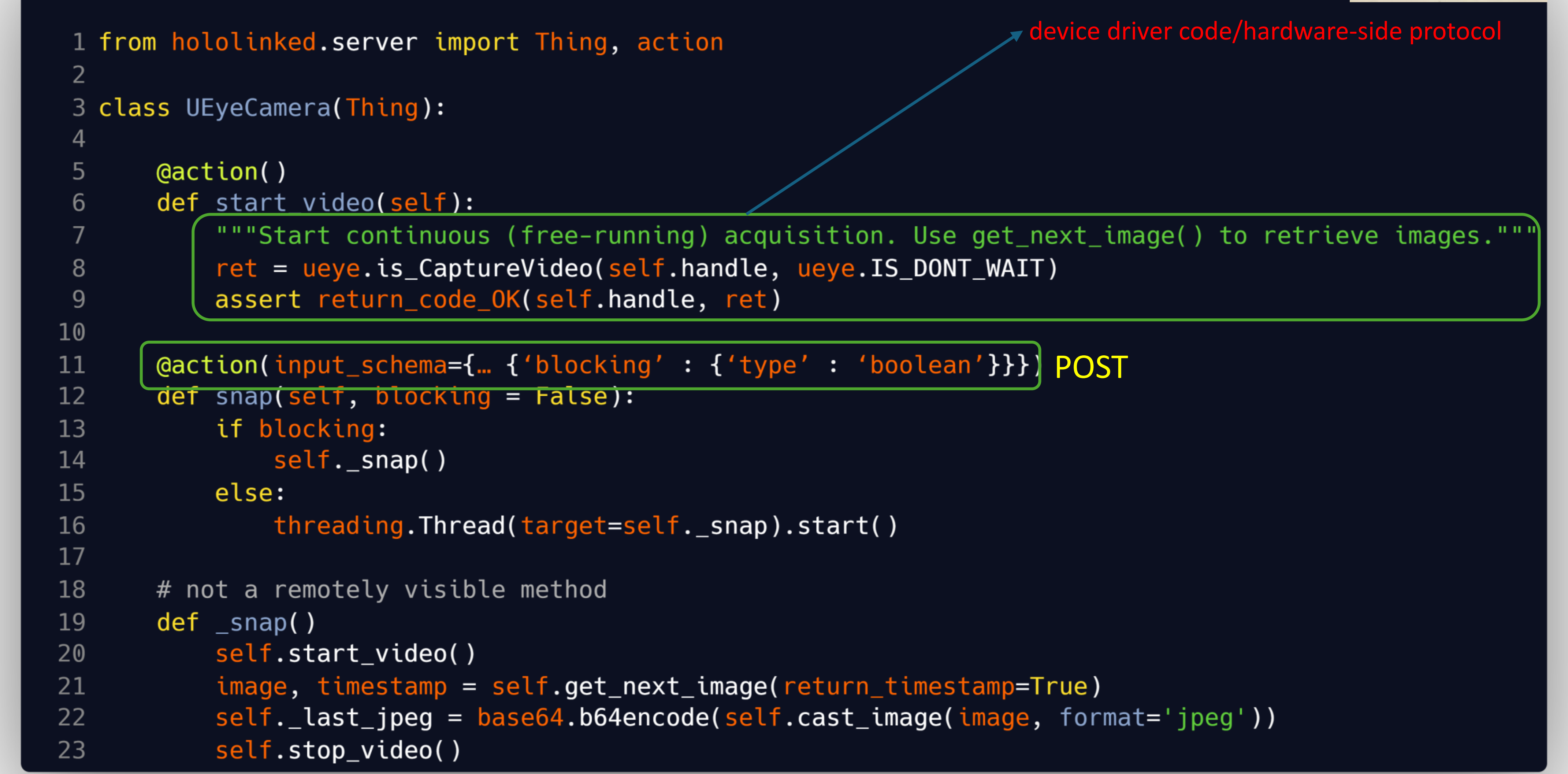


```python
from hololinked.server import Thing
from hololinked.server.properties import Number

class UEyeCamera(Thing):

    exposure_time = Number(
        bounds=(0.0, None),
        inclusive_bounds=(False, True)
        doc="Exposure time for image in milliseconds",
        fget=get_exposure,
        fset=set_exposure,
        unit='ms',
        observable=True,
    ) # type: float

```



```json
"properties" : {
   "exposure_time": {
      "title": "exposure_time",
      "description": "Exposure time for image in milliseconds",
      "unit": "ms",
      "type": "number",
      "forms": [
        {
          "href": "https://example.com/stretcher-gitter/exposure-time",
          "op": "readproperty",
          "htv:methodName": "GET",
          "contentType": "application/json"
        },
        {
          "href": "https://example.com/stretcher-gitter/exposure-time",
          "op": "writeproperty",
          "htv:methodName": "PUT",
          "contentType": "application/json"
        },
        {
          "href": "https://example.com/stretcher-gitter/exposure-time/change-event",
          "op": "observeproperty",
          "htv:methodName": "GET",
          "contentType": "text/plain",
          "subprotocol": "sse"
        }
      ],
      "observable": true,
      "exclusiveMinimum": 0
    }
  }

```

Actions

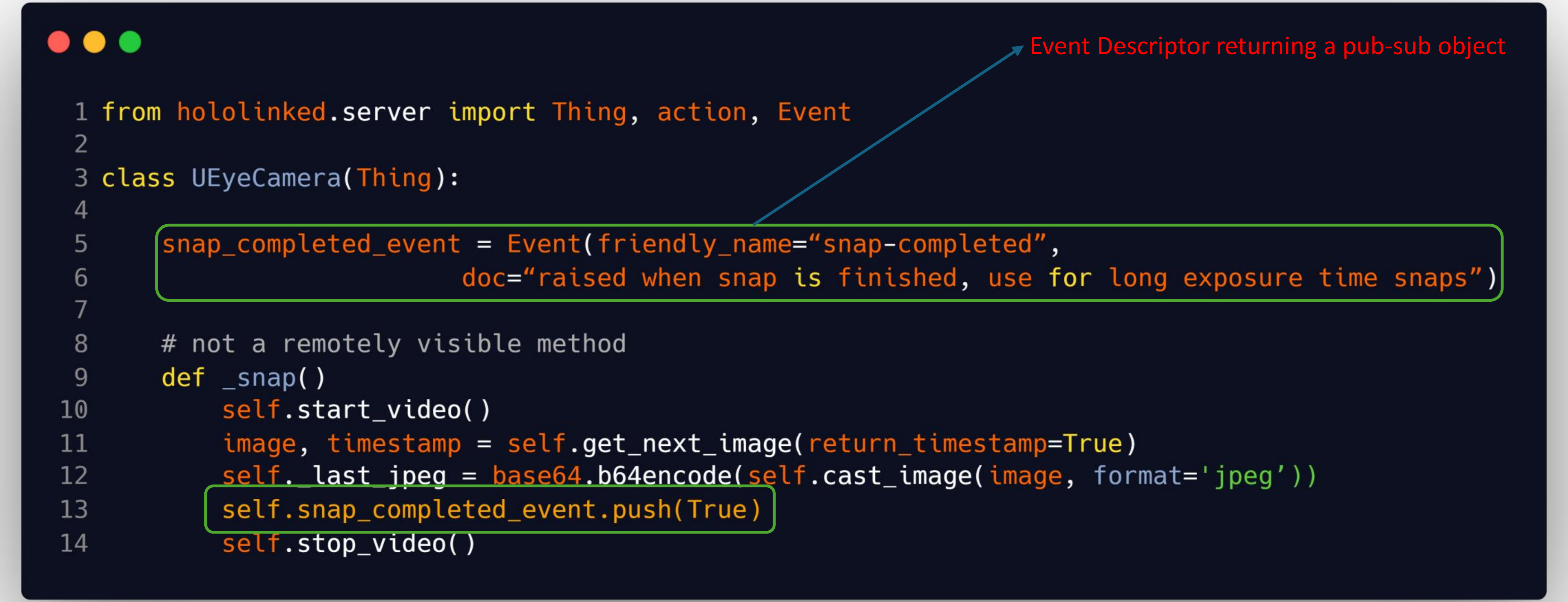


device driver code/hardware-side protocol

POST

```python
from hololinked.server import Thing, action

class UEyeCamera(Thing):

    @action()
    def start_video(self):
        """Start continuous (free-running) acquisition. Use get_next_image() to retrieve images."""
        ret = ueye.is_CaptureVideo(self.handle, ueye.IS_DONT_WAIT)
        assert return_code_OK(self.handle, ret)

    @action(input_schema={… {‘blocking’ : {‘type’ : ‘boolean’}}})
    def snap(self, blocking = False):
        if blocking:
            self._snap()
        else:
            threading.Thread(target=self._snap).start()

    # not a remotely visible method
    def _snap()
        self.start_video()
        image, timestamp = self.get_next_image(return_timestamp=True)
        self._last_jpeg = base64.b64encode(self.cast_image(image, format='jpeg'))
        self.stop_video()
```

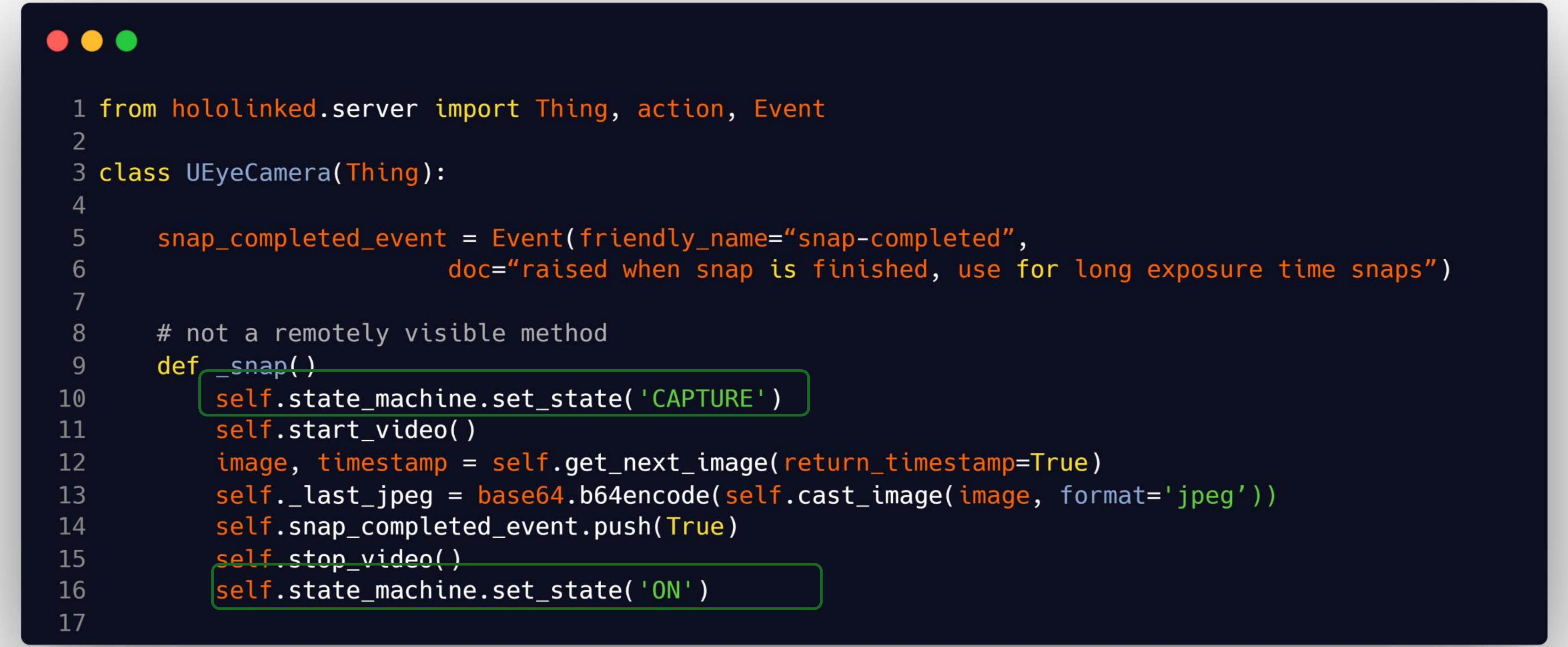
Events



Event Descriptor returning a pub-sub object

```
from hololinked.server import Thing, action, Event

class UEyeCamera(Thing):

    snap_completed_event = Event(friendly_name=“snap-completed”,
                        doc=“raised when snap is finished, use for long exposure time snaps”)

    # not a remotely visible method
    def _snap()
        self.start_video()
        image, timestamp = self.get_next_image(return_timestamp=True)
        self._last_jpeg = base64.b64encode(self.cast_image(image, format='jpeg’))
        self.snap_completed_event.push(True)
        self.stop_video()
```

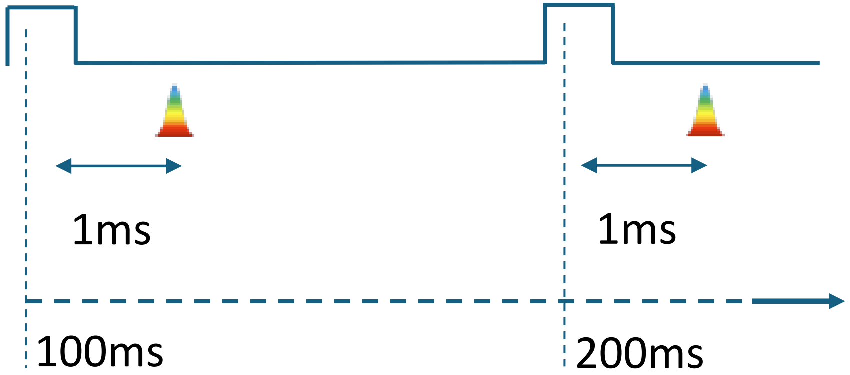
State Machine



```python
from hololinked.server import Thing, action, Event

class UEyeCamera(Thing):

    snap_completed_event = Event(friendly_name=“snap-completed”,
                        doc=“raised when snap is finished, use for long exposure time snaps”)

    # not a remotely visible method
    def _snap()
        self.state_machine.set_state('CAPTURE')
        self.start_video()
        image, timestamp = self.get_next_image(return_timestamp=True)
        self._last_jpeg = base64.b64encode(self.cast_image(image, format='jpeg’))
        self.snap_completed_event.push(True)
        self.stop_video()
        self.state_machine.set_state('ON')

```

Fig 7. – Signal Generator

Laser Control Electronics/Pulse Frequency Reduction

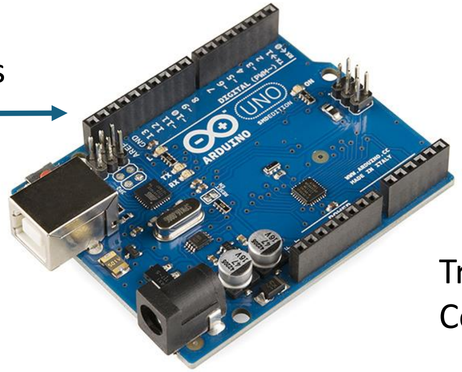


GUI

Request to Shoot Pulses

Start/Stop Counting Triggers

Fig 8. – Arduino Uno



Trigger Counter

10Hz

Trigger read as software event

Optically triggered

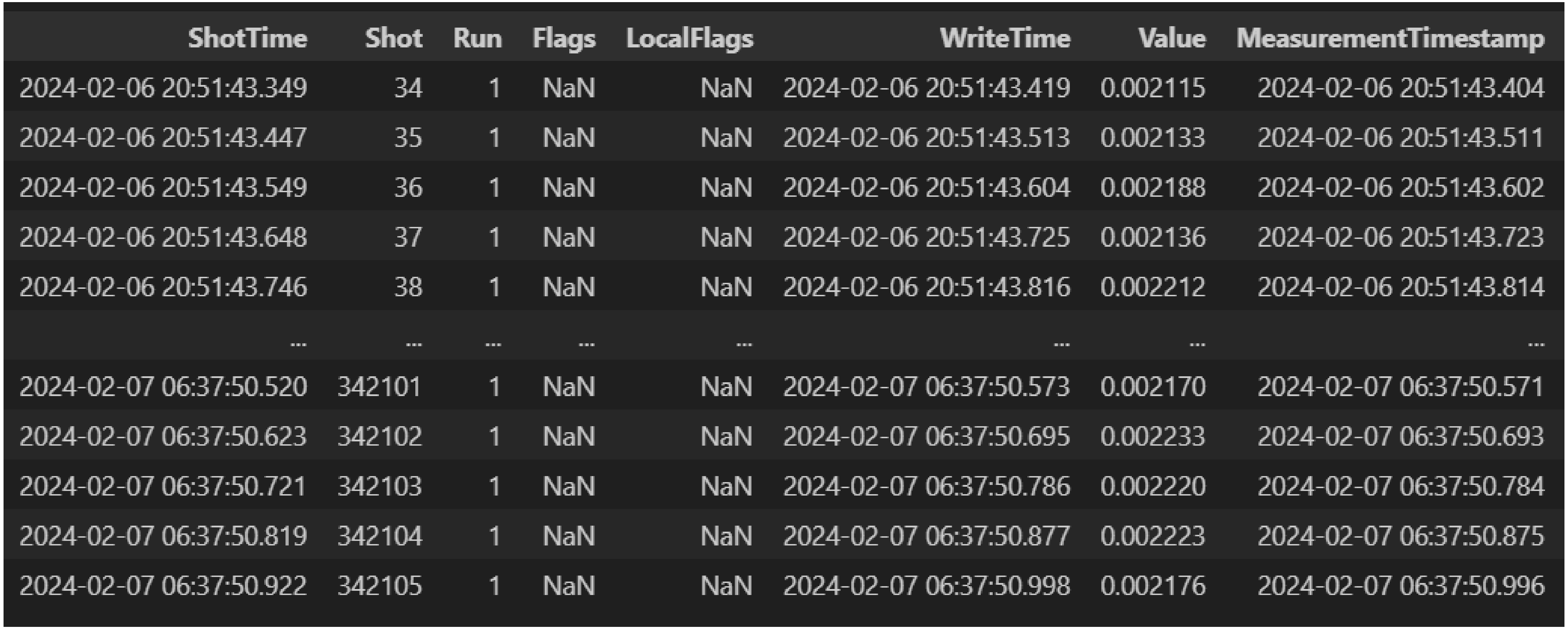
Rising edge

Rising edge

```
{
    “shot_number”: 3000
    “shot_time” : “16:00:00.100T2024-10-01”
}
```

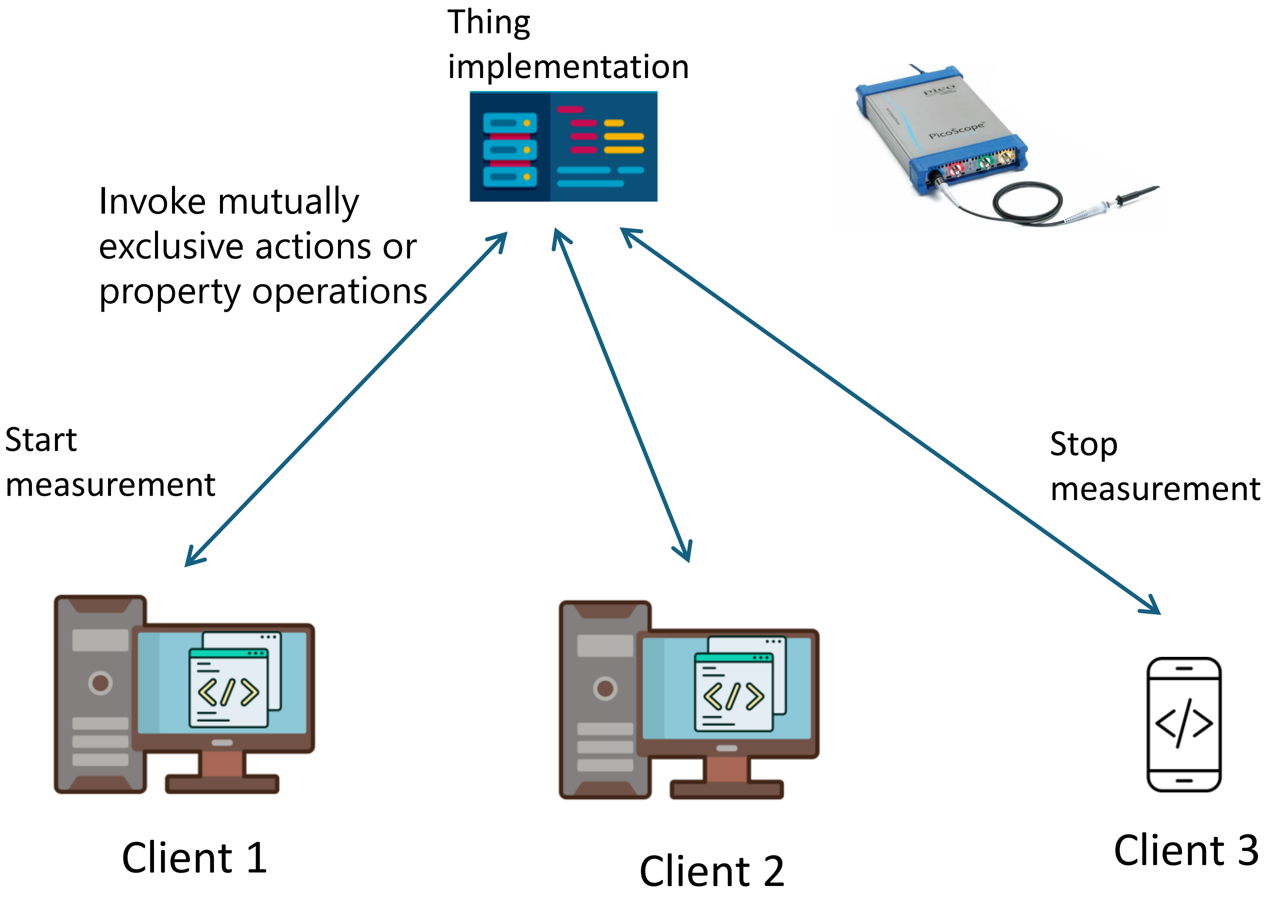
Correlate information coming from event with local measurement

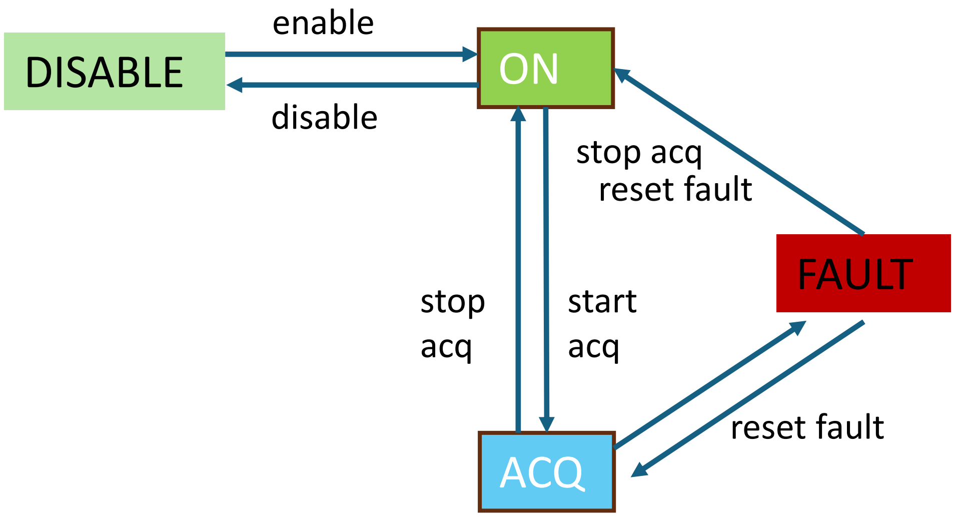
Energy Meter Data



ShotTime	Shot	Run	Flags	LocalFlags	WriteTime	Value	MeasurementTimestamp
2024-02-06 20:51:43.349	34	1	NaN	NaN	2024-02-06 20:51:43.419	0.002115	2024-02-06 20:51:43.404
2024-02-06 20:51:43.447	35	1	NaN	NaN	2024-02-06 20:51:43.513	0.002133	2024-02-06 20:51:43.511
2024-02-06 20:51:43.549	36	1	NaN	NaN	2024-02-06 20:51:43.604	0.002188	2024-02-06 20:51:43.602
2024-02-06 20:51:43.648	37	1	NaN	NaN	2024-02-06 20:51:43.725	0.002136	2024-02-06 20:51:43.723
2024-02-06 20:51:43.746	38	1	NaN	NaN	2024-02-06 20:51:43.816	0.002212	2024-02-06 20:51:43.814
...	...	...	...	...	...	...	...
2024-02-07 06:37:50.520	342101	1	NaN	NaN	2024-02-07 06:37:50.573	0.002170	2024-02-07 06:37:50.571
2024-02-07 06:37:50.623	342102	1	NaN	NaN	2024-02-07 06:37:50.695	0.002233	2024-02-07 06:37:50.693
2024-02-07 06:37:50.721	342103	1	NaN	NaN	2024-02-07 06:37:50.786	0.002220	2024-02-07 06:37:50.784
2024-02-07 06:37:50.819	342104	1	NaN	NaN	2024-02-07 06:37:50.877	0.002223	2024-02-07 06:37:50.875
2024-02-07 06:37:50.922	342105	1	NaN	NaN	2024-02-07 06:37:50.998	0.002176	2024-02-07 06:37:50.996

How to synchronise GUIs

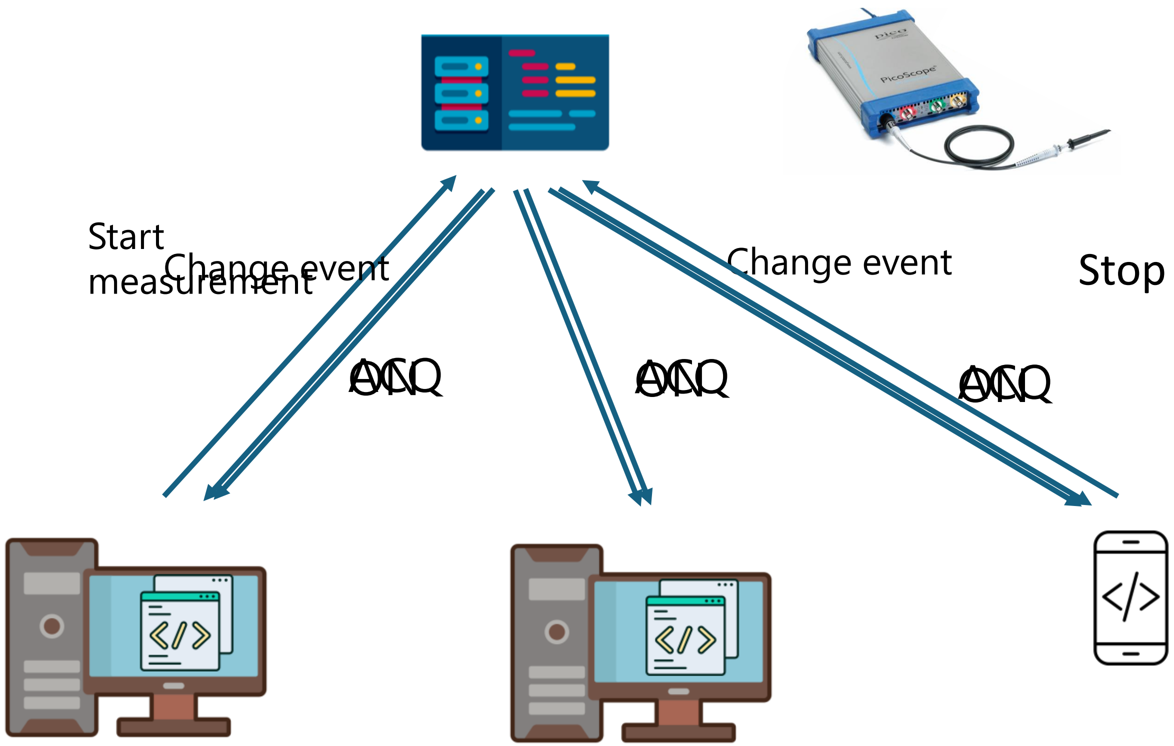




1) Only certain actions (& property read-write) can be issued in certain states

2) Actions can change states

Promote State Machine State to an observable property



Start measurement

Change event

ACQ

ON

ACQ

ON

Change event

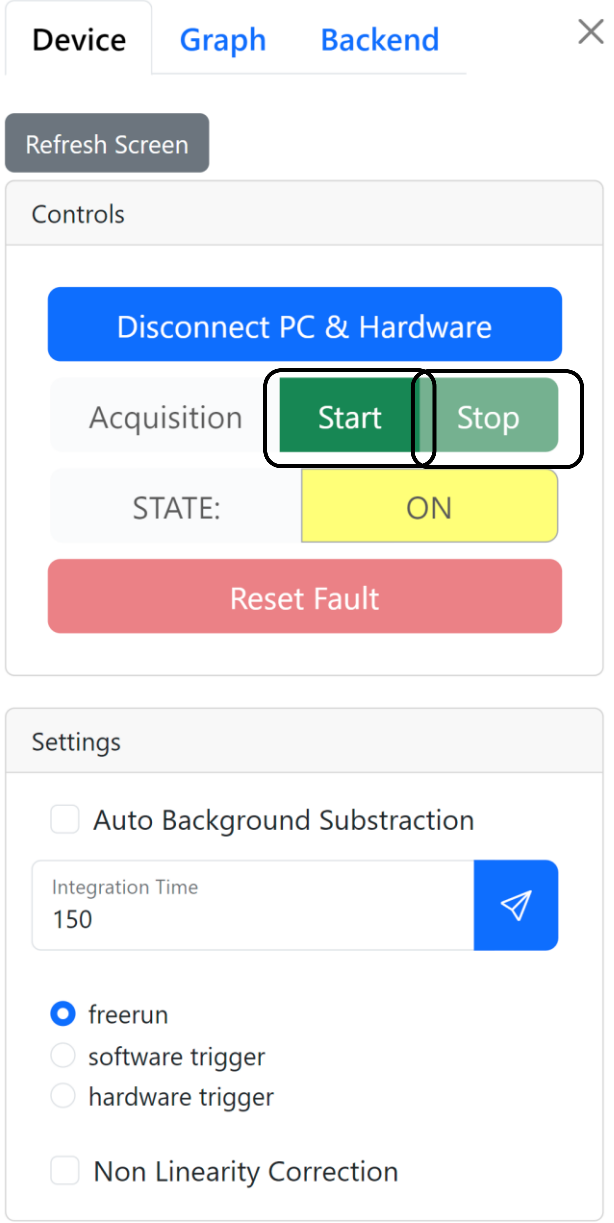
ACQ

ON

Stop measurement

PC 1

PC 2



Device Graph Backend

Refresh Screen

Controls

Disconnect PC & Hardware

Acquisition Start Stop

STATE: ON

Reset Fault

Settings

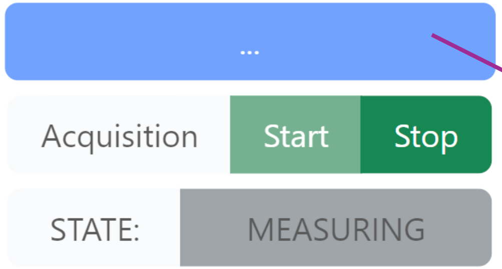
□ Auto Background Substraction

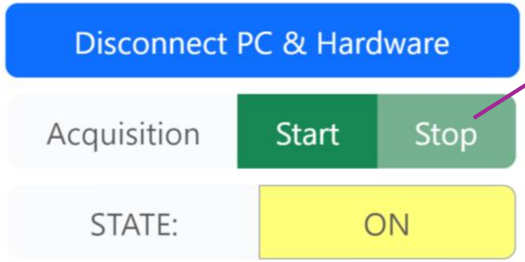
Integration Time
150

● freerun
○ software trigger
○ hardware trigger

□ Non Linearity Correction

Bind State Machine State to GUI Elements’ Props





```
{
    ON : {
        text : ‘Disconnect PC & Hardware’,
        backgroundColor : ‘blue’
    },
    MEASURING : {
        text : ‘...’,
        backgroundColor: ‘light blue’
    },
    DISCONNECTED : {
        text : ‘Connect PC & Hardware’,
        backgroundColor : ‘blue’
    }

}
```

```
{
    ON : { disable : true }
    MEASURING : { disable : false }
}
```

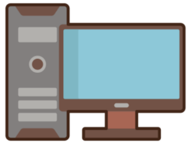
To auto-apply in React & Svelte

1) State Property in context API
2) Make components react to change to state (useEffect hook)

Special Thanks

- Prof. Dr. Jörg Schreiber, Associate Professor, LMU
  - Chair for Experimental Physics – Medical Physics and Laser Ion Acceleration
- Anna Schmidt, M.Sc.
  - Ph.D. student – Ion Acoustics and Laser Ion Acceleration
  - Co-organiser for hardware

Attribution

- Fig. – Optical Cavity - Lakkasuo, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
- Fig. – Pulse Train - HartmutG, CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia
- Fig. – Picoscope - Pico Technology, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
- Fig. – Beam cross section - AdamJones88, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons < https://commons.wikimedia.org/wiki/File:Gaussian_Beam_Laser_Pulsed.png >
- Fig. – Light Pulse - By Zueignung - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22226803
- https://www.flaticon.com/free-icons/backend" Backend icons created by Flat Icons – Flaticon
- https://www.flaticon.com/free-icons/data - Data icons created by Freepik – Flaticon
-  https://www.flaticon.com/free-icons/control-room - Control room icons created by Freepik – Flaticon
- Fig. – Spectrum Analyzer – Rohde & Schwarz GmbH & Co. KG, Mühldorfstraße 15, D-81671 München, http://www.rohde-schwarz.com, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
- Egekorkan, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- Fig 7. – Signal Generator - Mu301, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
- Fig 8. – Arduino Uno - By SparkFun Electronics from Boulder, USA - Arduino Uno – R3, CC BY 2.0,https://commons.wikimedia.org/w/index.php?curid=26785892

Ultrafast Lasers & High Power Lasers

Pulse Characteristics

- Picosecond or femtosecond pulses
  (1 second to approximately 31.69 million years)
- Pulses per second – order of Millions
  (80 Million pulses per second for 80MHz)

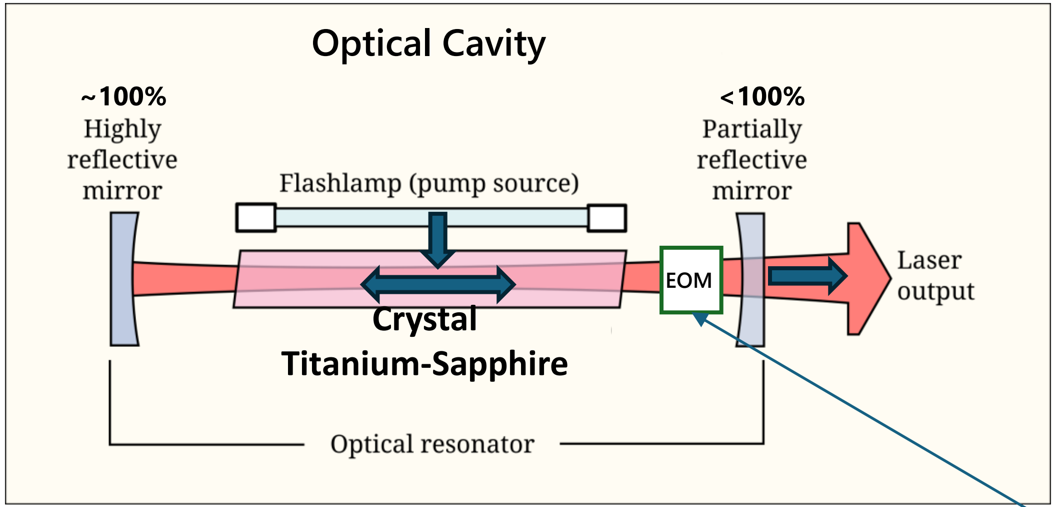


Fig 1. Optical Cavity

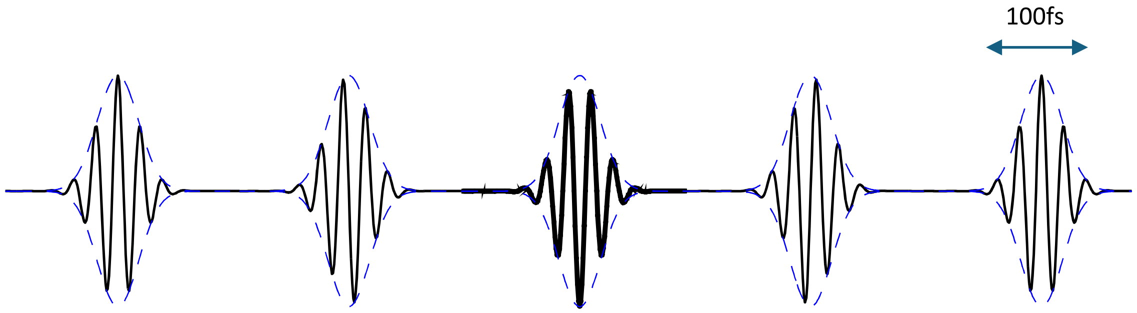


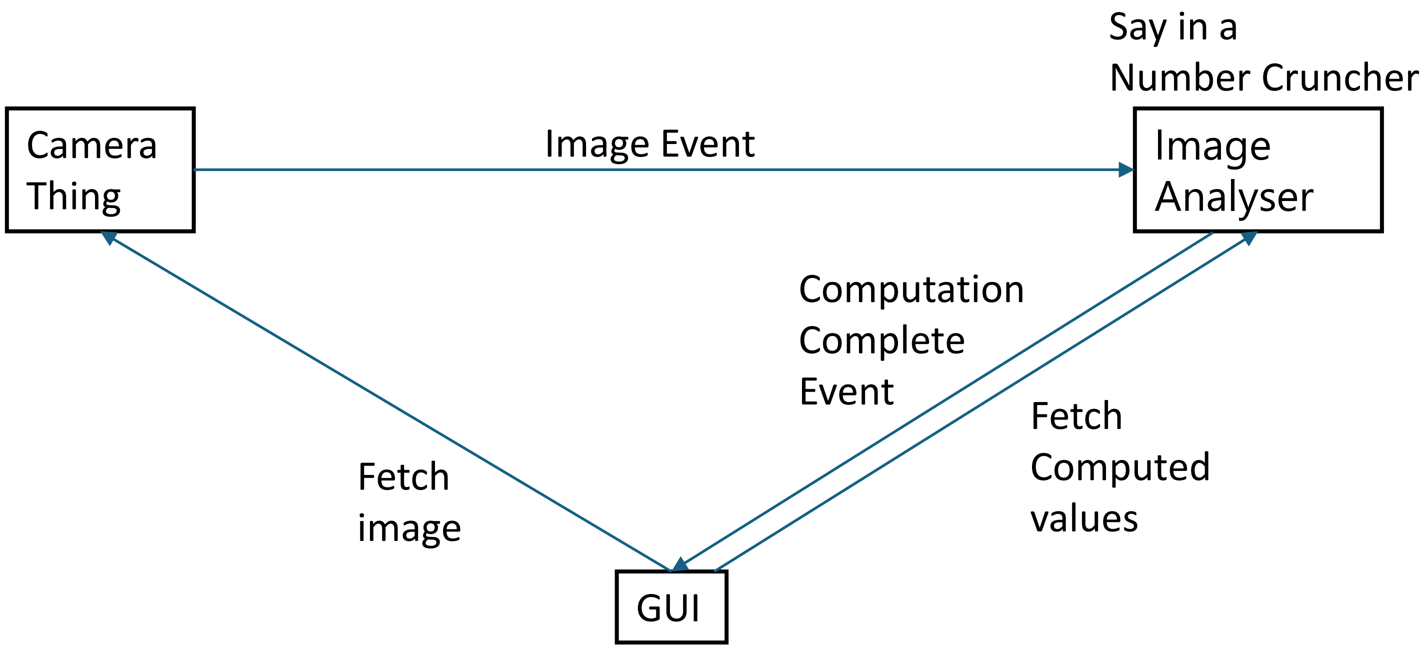
Fig 2. Pulse Train

Radio Frequency
Optical Switch (say 80MHz)

Atomic Clocks, Spectroscopy

Computational “Virtual” Thing

- Properties - computed beam diameter, centre of mass (beam pointing)
- Actions - start computation, take control action
- Events - computation complete


Say in a
Number Cruncher
Camera Thing
Image Event
Image Analyser
Computation Complete Event
Fetch Computed values
Fetch image
GUI