Designing an ‘In-Meeting’ Indicator with Raku

Joelle Tori Maslak
(She/Her/Hers)

Raku Devroom
esLibre 2021

Who is Joelle?

- Raku Developer for 5 years
- Author of several Raku modules, most proud of Raku Net::BGP
- Network Engineer @ Netflix
- Previously: Experience at government, telecom, and dotcom

The Problem - Definition

Working From Home Problem

- I share a house with my wife (that isn’t a problem!)
- Sometimes I’m okay with interruptions
- Other times I’m not okay with interruptions
- My wife doesn’t like to appear on video unexpectedly
- I want this to be automatic

Defining the Solution

- Basically, I need an “on air” type of light
- Must work with Linux
- Must integrate with Google Calendar
- Should monitor webcam state
- Should be controllable via the terminal

Defining the Solution

- Basically, I need an “on air” type of light
- Must work with Linux
- Must integrate with Google Calendar
- Should turn on when webcam is in use
- Should be controllable via the terminal


LUXAFOR

How to Drive a USB Device?

- Raku is a great choice if I need computerized duct tape
- Travis Gibson did the hard work already!

  “USB Devices: How to Drive Your Own”

  Conference in the Cloud 2020

  (LibUSB from Raku)
- Yes, there were gotchas still…
  - What USB commands do I need?
  - There was a Python script available that did this!

Defining the Solution

- Basically, I need an “on air” type of light
- Must work with Linux
- Must integrate with Google Calendar
- Should turn on when webcam is in use
- Should be controllable via the terminal

gcalcli

```
[0] sandbox:work$ gcalcli --nocolor --calendar "Test Calendar" agenda --military --tsv --nodeclined
2020-10-10      17:15   2020-10-10      19:00   Example Meeting
2020-10-10      19:00   2020-10-10      20:00   Example Meeting #2
```

Defining the Solution

- Basically, I need an “on air” type of light
- Must work with Linux
- Must integrate with Google Calendar
- Should turn on when webcam is in use
- Should be controllable via the terminal

Webcam NOT in use:

```
[0] sandbox:work$ cat /proc/modules | egrep ^uvcvideo
uvcvideo 98304 0 - Live 0x0000000000000000
```

Webcam IS in use:

```
[0] sandbox:work$ cat /proc/modules | egrep ^uvcvideo
uvcvideo 98304 1 - Live 0x0000000000000000
```

Webcam NOT in use:

```
[0] sandbox:work$ cat /proc/modules | egrep ^uvcvideo
uvcvideo 98304 0 - Live 0x0000000000000000
```

Webcam IS in use:

```
[0] sandbox:work$ cat /proc/modules | egrep ^uvcvideo
uvcvideo 98304 1 - Live 0x0000000000000000
```

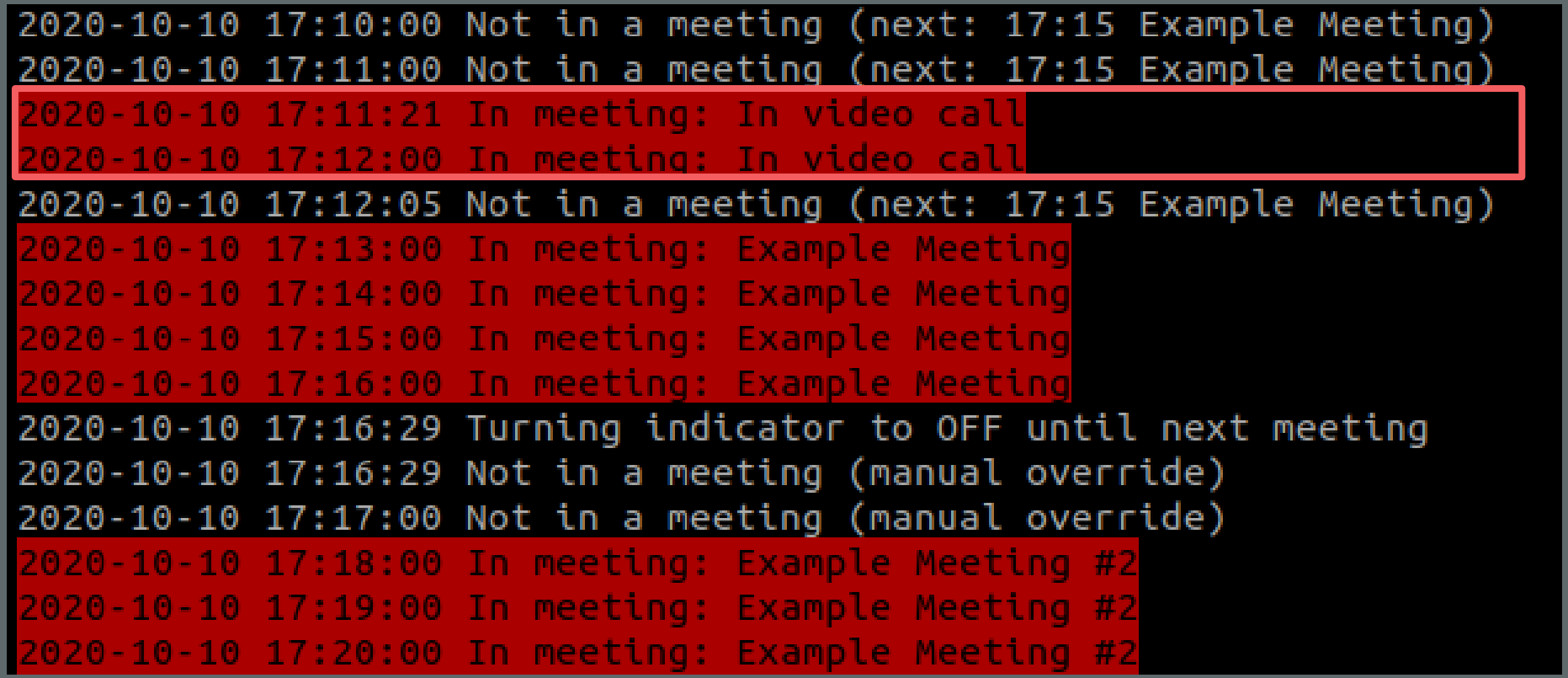
Defining the Solution

- Basically, I need an “on air” type of light
- Must work with Linux
- Must integrate with Google Calendar
- Should turn on when webcam is in use
- Should be controllable via the terminal

```
2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2
```

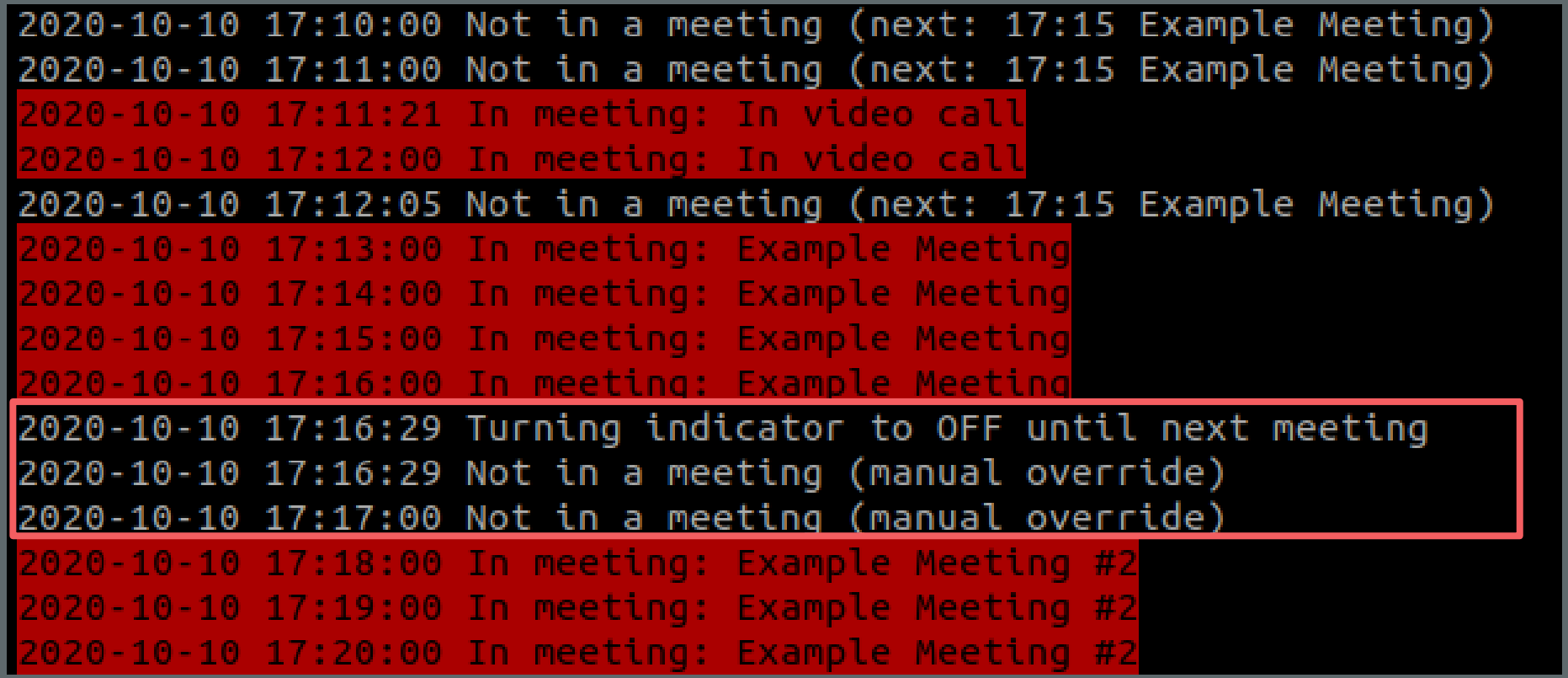
Putting Everything Together

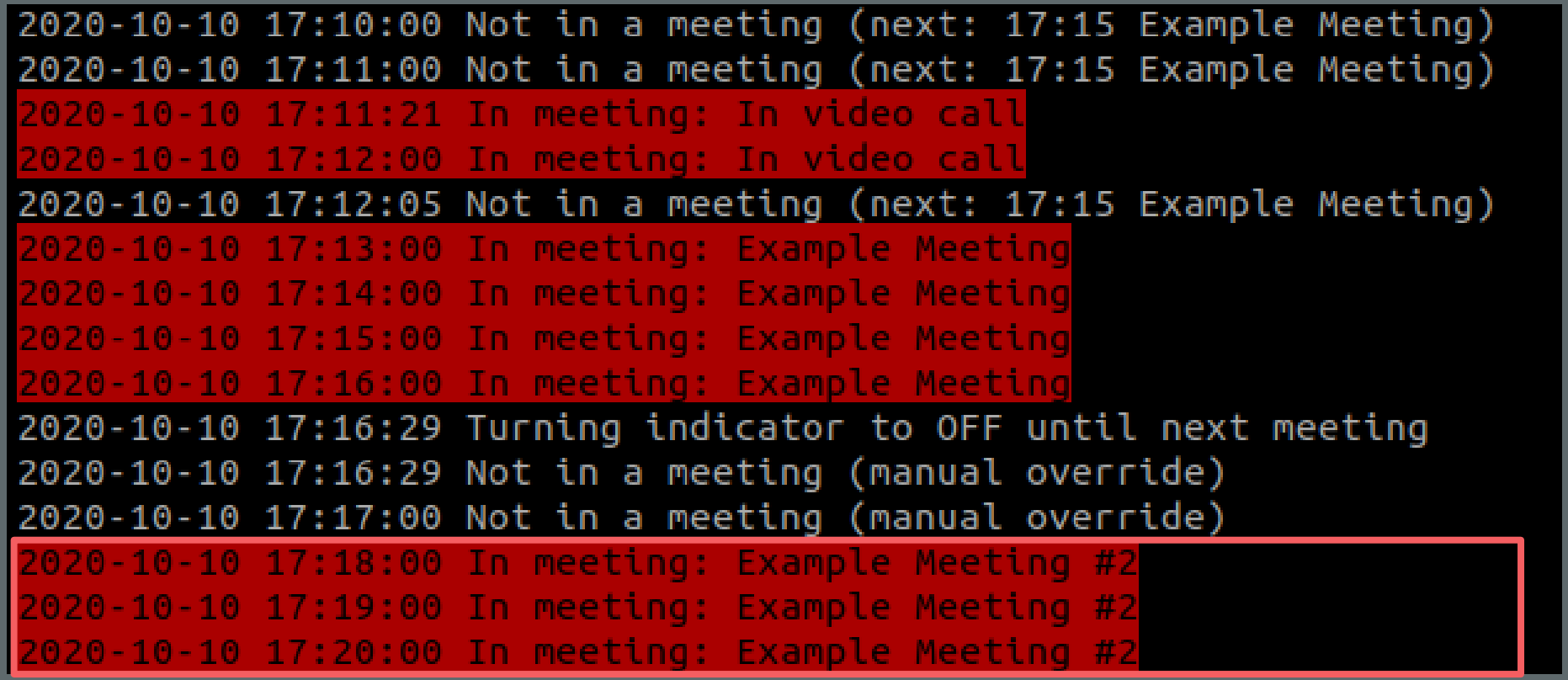
```
2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2
```


2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2

```
2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2
```

```
2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2
```


2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2

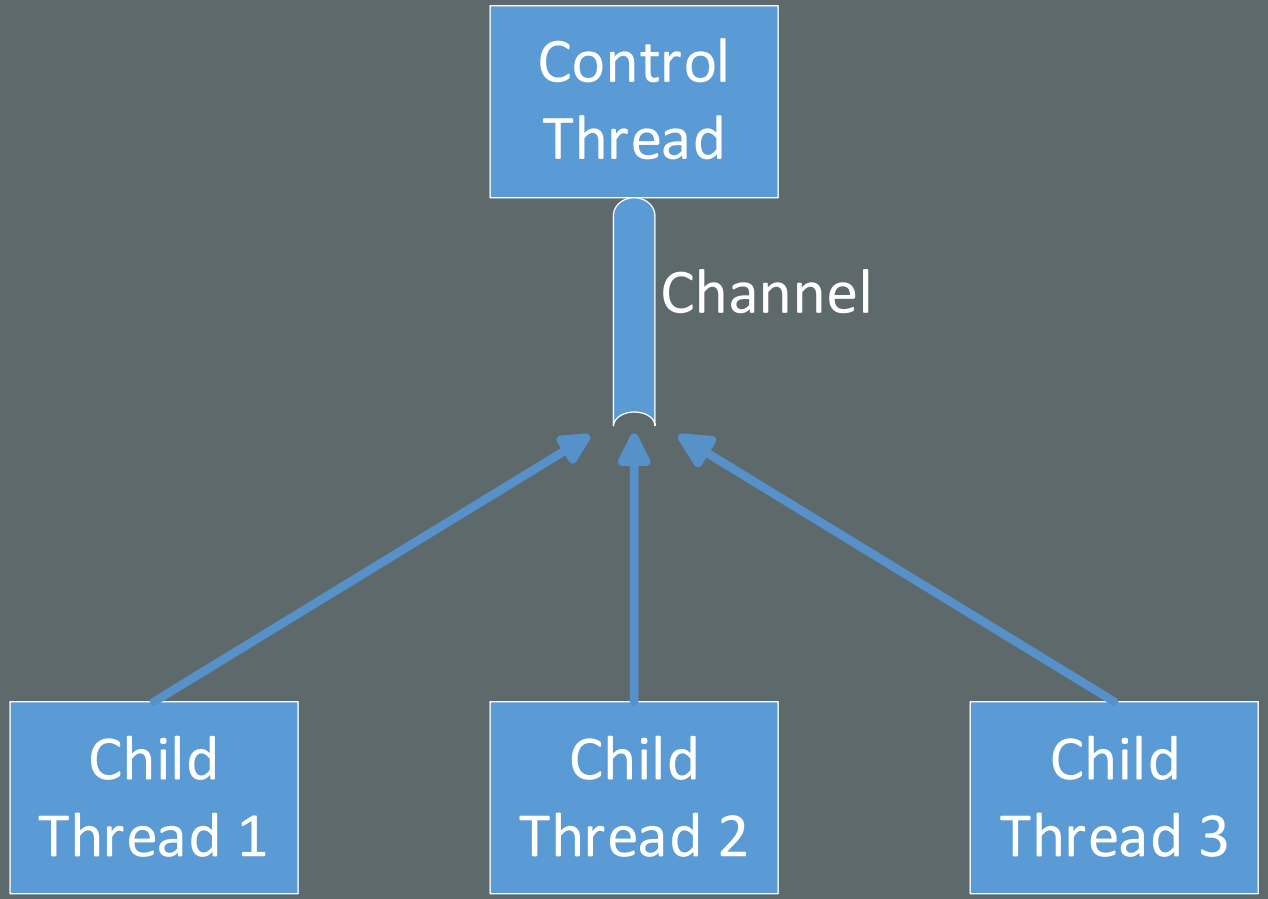

2020-10-10 17:10:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:00 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:11:21 In meeting: In video call
2020-10-10 17:12:00 In meeting: In video call
2020-10-10 17:12:05 Not in a meeting (next: 17:15 Example Meeting)
2020-10-10 17:13:00 In meeting: Example Meeting
2020-10-10 17:14:00 In meeting: Example Meeting
2020-10-10 17:15:00 In meeting: Example Meeting
2020-10-10 17:16:00 In meeting: Example Meeting
2020-10-10 17:16:29 Turning indicator to OFF until next meeting
2020-10-10 17:16:29 Not in a meeting (manual override)
2020-10-10 17:17:00 Not in a meeting (manual override)
2020-10-10 17:18:00 In meeting: Example Meeting #2
2020-10-10 17:19:00 In meeting: Example Meeting #2
2020-10-10 17:20:00 In meeting: Example Meeting #2

Iterating on the Design

- What if I want the Luxafor on a different computer than the camera? (Network support!)
- Also, what if I wanted to control the indicator from the other computer?
- Can I remind *me* of meetings?

Basic Design

- “Main” Thread: Listens to a Raku *Channel*
- Other threads send messages to that channel
- Examples of other threads:
  - Clock Ticker (once a minute sends a tick)
  - Fetch updates to Google Calendar
  - Keyboard Reader
  - Camera Monitor
  - Later, Network Thread


Control Thread
Channel
Child Thread 1
Child Thread 2
Child Thread 3

Control Thread

```
my Channel:D $channel = Channel.new;

… # Start background tasks

react {
    whenever $channel -> $message {
        self.process-message($message);
    }
}
```

Messages
(Sent Across the Channel)

```
class Message-Keypress {
    has Str:D $.key is required;
}

class Message-Tick { }

class Message-Camera {
    has Bool:D $.state is required;
}

class Message-Remote {
    has Bool:D $.state is required;
}

class Message-Appointments {
    has @.appointments is required;
}
```

Background Threads

```
start { background-ticks($channel, $interval)                }
start { background-google($channel, $interval, @calendar) }
start { background-network($channel, $port)                  }
start { background-camera($channel)                          }
start { background-keypress($channel)                        }
start { background-timezone($channel)                        }
```

Example Thread: “Ticks”

```
react {
    whenever Supply.interval($interval) {
        $channel.send(Message-Tick.new);
    }
}
```

Example Thread: Network Listener

```
my $camera = 0;
my $socket = IO::Socket::Async.bind-udp('::', $port);

react {
    whenever $socket.Supply -> $v {
        if $v eq "CAMERA ON" {
            # We need TWO camera "on" events before we change state
            if $camera++ == 2 {
                $channel.send: Message-Remote.new(state => True);
            }
        } elsif $v eq "CAMERA OFF" {
            if $camera ≠ 0 {
                $camera = 0;
                $channel.send: Message-Remote.new(state => False);
            }
        } elsif $v ~~ m/ ^ "KEY " (.) $/ {
            $channel.send: Message-Keypress.new(key => $0.Str.fc);
        }
    }
}
```

```
my $camera = 0;
my $socket = IO::Socket::Async.bind-udp('::', $port);

react {
    whenever $socket.Supply -> $v {
        if $v eq "CAMERA ON" {
            # We need TWO camera "on" events before we change state
            if $camera++ == 2 {
                $channel.send: Message-Remote.new(state => True);
            }
        } elsif $v eq "CAMERA OFF" {
            if $camera ≠ 0 {
                $camera = 0;
                $channel.send: Message-Remote.new(state => False);
            }
        } elsif $v ~~ m/ ^ "KEY " (.) $/ {
            $channel.send: Message-Keypress.new(key => $0.Str.fc);
        }
    }
}
```

```
my $camera = 0;
my $socket = IO::Socket::Async.bind-udp('::', $port);

react {
    whenever $socket.Supply -> $v {
        if $v eq "CAMERA ON" {
            # We need TWO camera "on" events before we change state
            if $camera++ == 2 {
                $channel.send: Message-Remote.new(state => True);
            }
        } elsif $v eq "CAMERA OFF" {
            if $camera ≠ 0 {
                $camera = 0;
                $channel.send: Message-Remote.new(state => False);
            }
        } elsif $v ~~ m/ ^ "KEY " (.) $/ {
            $channel.send: Message-Keypress.new(key => $0.Str.fc);
        }
    }
}
```

```
my $camera = 0;
my $socket = IO::Socket::Async.bind-udp('::', $port);

react {
    whenever $socket.Supply -> $v {
        if $v eq "CAMERA ON" {
            # We need TWO camera "on" events before we change state
            if $camera++ == 2 {
                $channel.send: Message-Remote.new(state => True);
            }
        } elsif $v eq "CAMERA OFF" {
            if $camera ≠ 0 {
                $camera = 0;
                $channel.send: Message-Remote.new(state => False);
            }
        } elsif $v ~~ m/ ^ "KEY " (.) $/ {
            $channel.send: Message-Keypress.new(key => $0.Str.fc);
        }
    }
}
```

Control Thread (Revisited)

```
my Channel:D $channel = Channel.new;

… # Start background tasks

react {
    whenever $channel -> $message {
        self.process-message($message);
    }
}
```

```
method process-message($message) {
    given $message {
        when Message-Tick          { self.display if $!google-success   }
        when Message-Keypress      { self.handle-keypress($message)     }
        when Message-Camera        { self.handle-camera($message)       }
        when Message-Remote        { self.handle-remote($message)       }
        when Message-Appointments { self.handle-appointments($message) }
        when Message-Offset        { self.handle-offset($message)       }
        when Message-Offset-Error { self.handle-offset-error           }
        default                    { die("Unknown command type")        }
    }
}
```

Learnings

- I hate wireless!
- Raku is great internet duct tape!
  - Files!
  - Processes!
  - Networking!
- Multi-threaded Raku is pretty simple
- The pattern of a control thread receiving events from child threads is very flexible

Questions?


HAPPY
PRIDE!