My Doorbell Runs Swift

iOSDevCampDC 2017

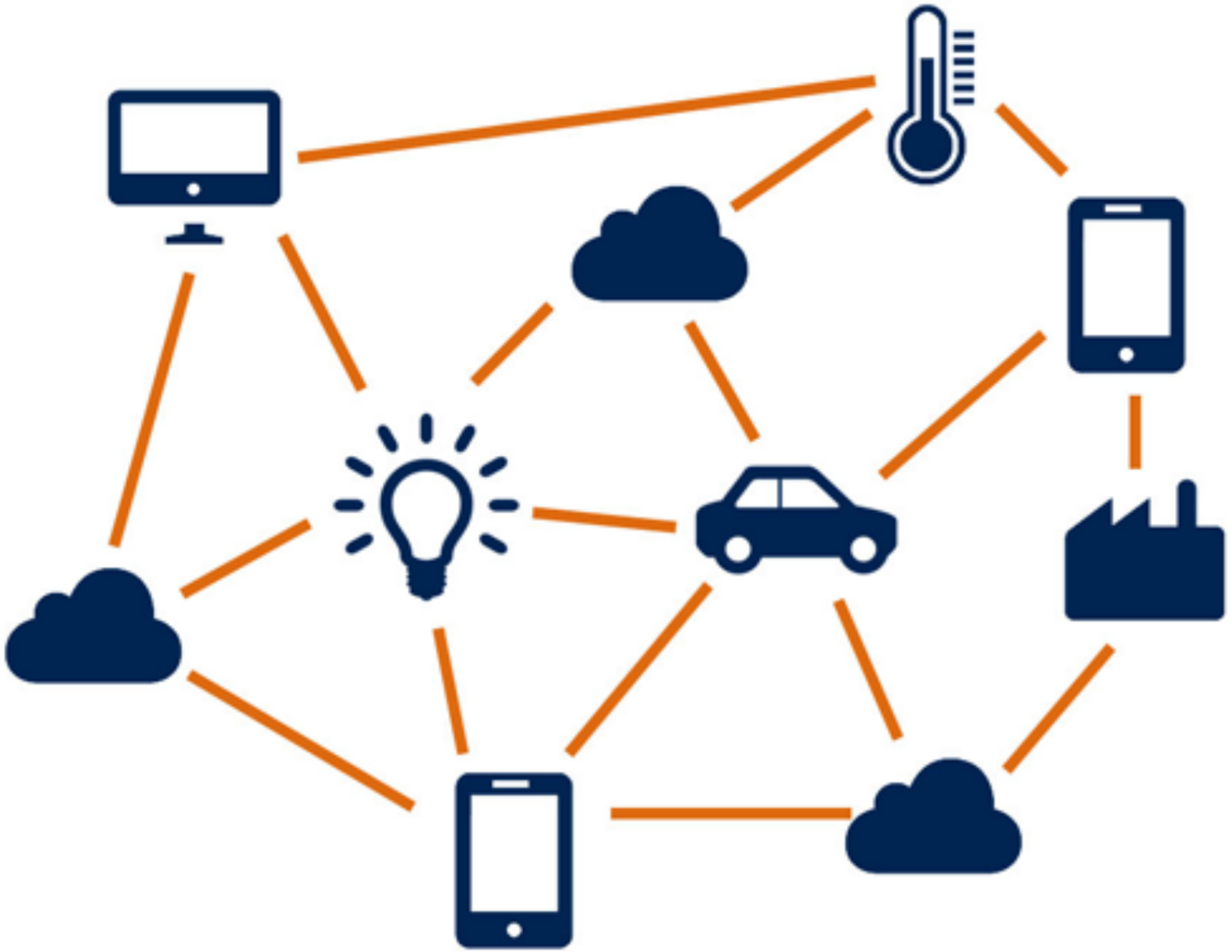
Rob Huebner

 @huebnerob

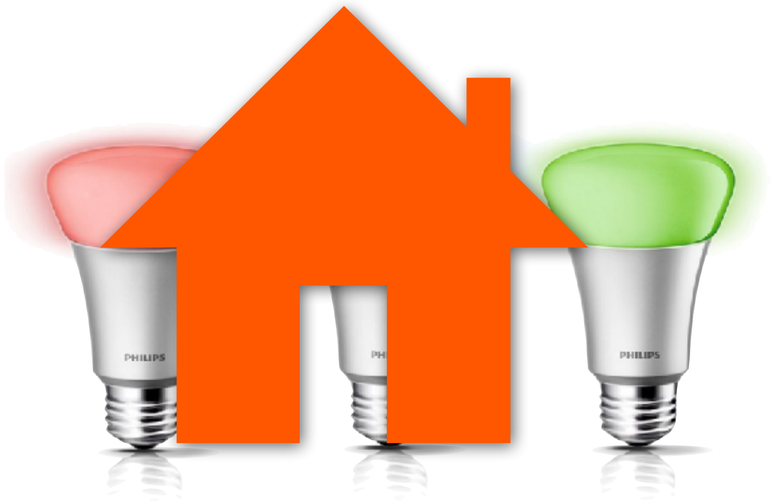
Agenda

- Internet of Things
- DEMO
- Hardware
- Software
- What's next

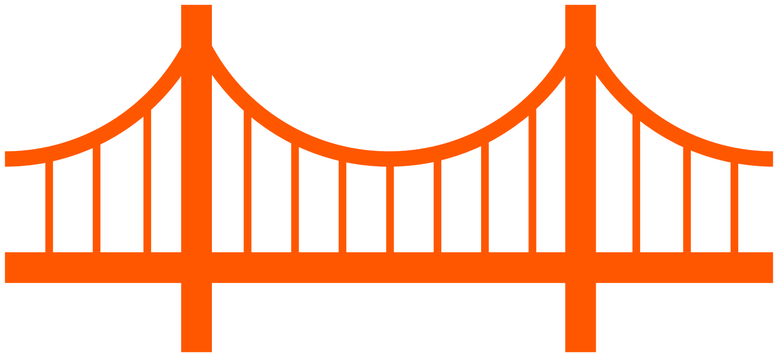
“Internet of Things”



Example: Philips Hue Lights



✅



🤔

✅

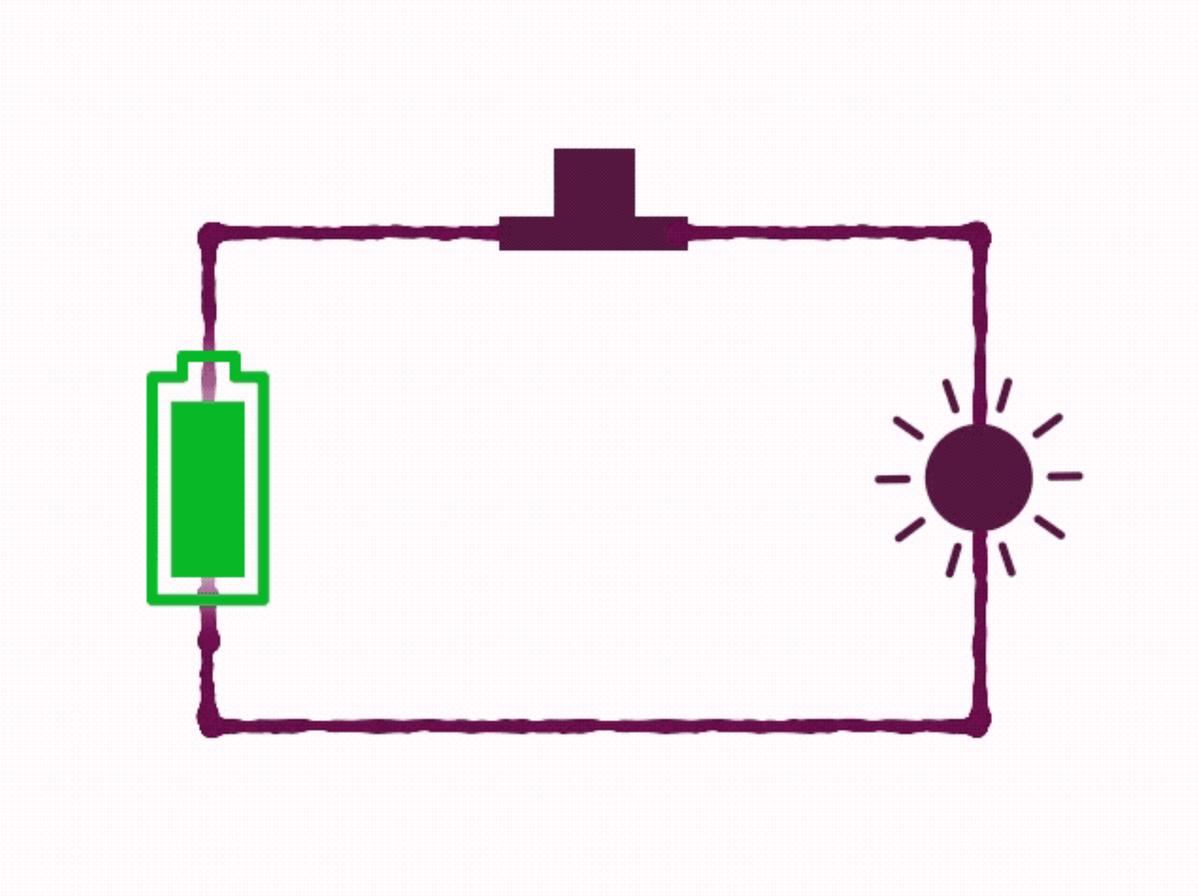

Build Our Own Bridge

But what are “Things™”?

- Controlling systems with electronics
  - Caution ⚠️ High Voltage!
- Model them as simple circuits

Simple Circuits

- Power source
- Input
  - Button
- Output
  - Light



Agenda

- ✓ Internet of Things
- DEMO
- Hardware
- Software
- What's next

DEMO

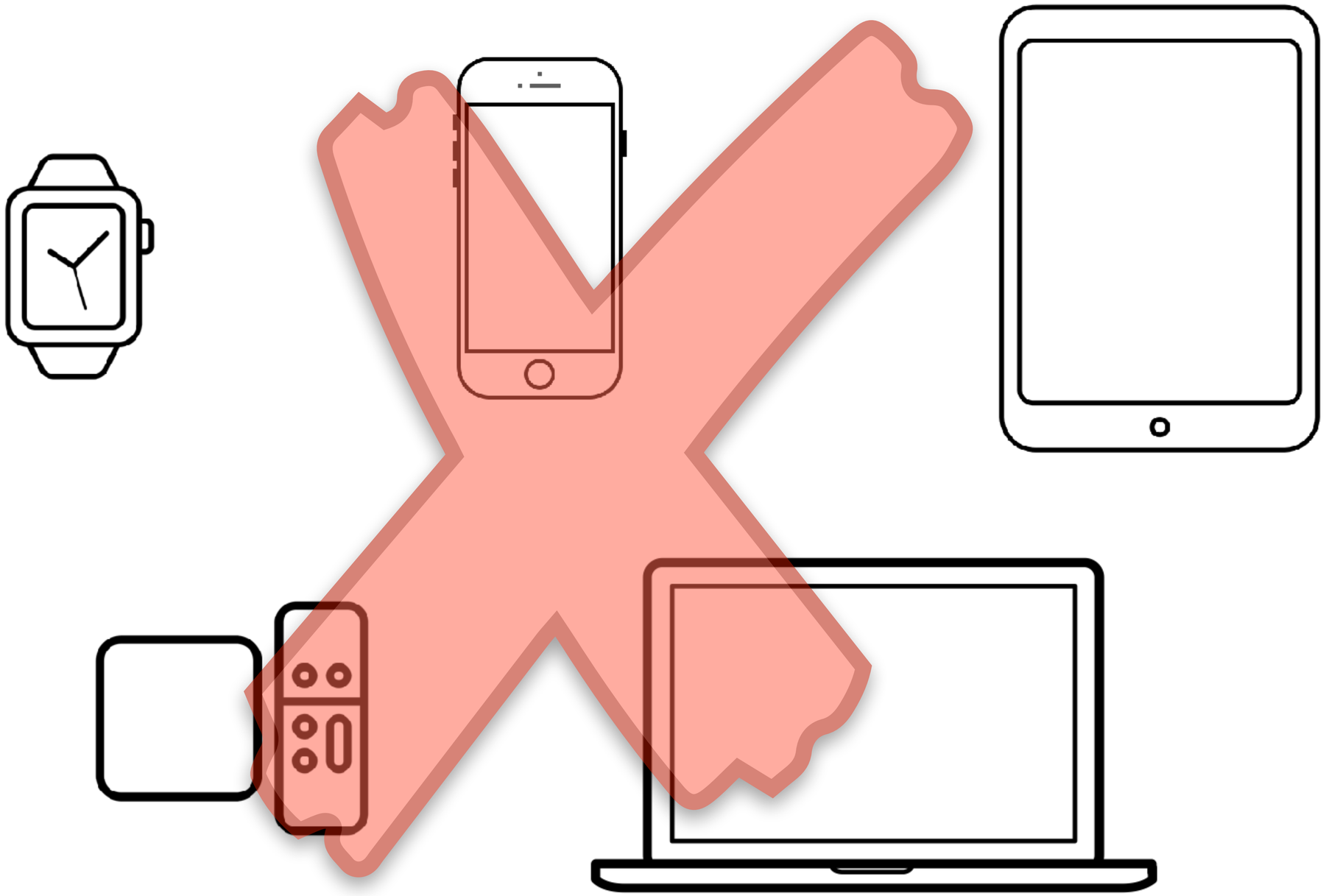

APT 3A
APARTMENT 3B

DEMO

Agenda

- ✓ Internet of Things
- ✓ DEMO
- Hardware
- Software
- What's next

Hardware

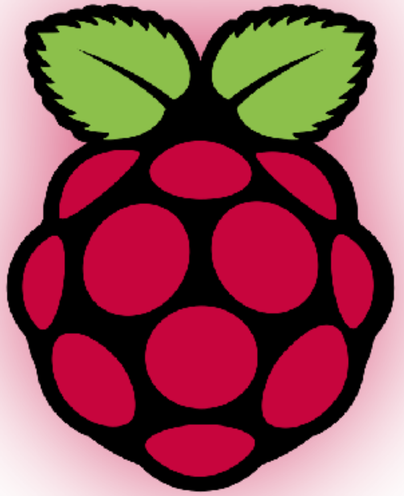


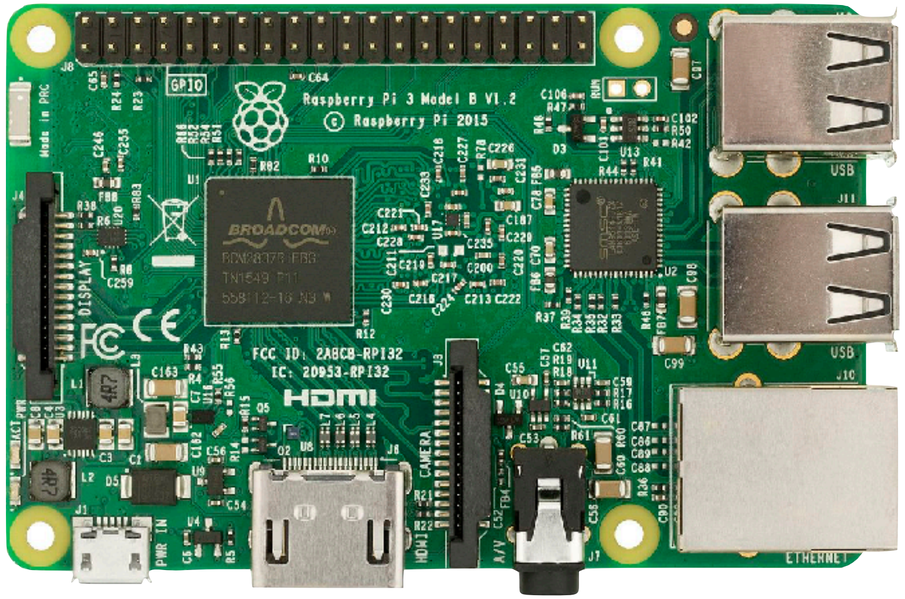
Hardware

- What do we need to build our bridge?
  - Internet connectivity
  - Interface with “Things” (a.k.a. simple circuits)
  - Non-mobile

Raspberry Pi

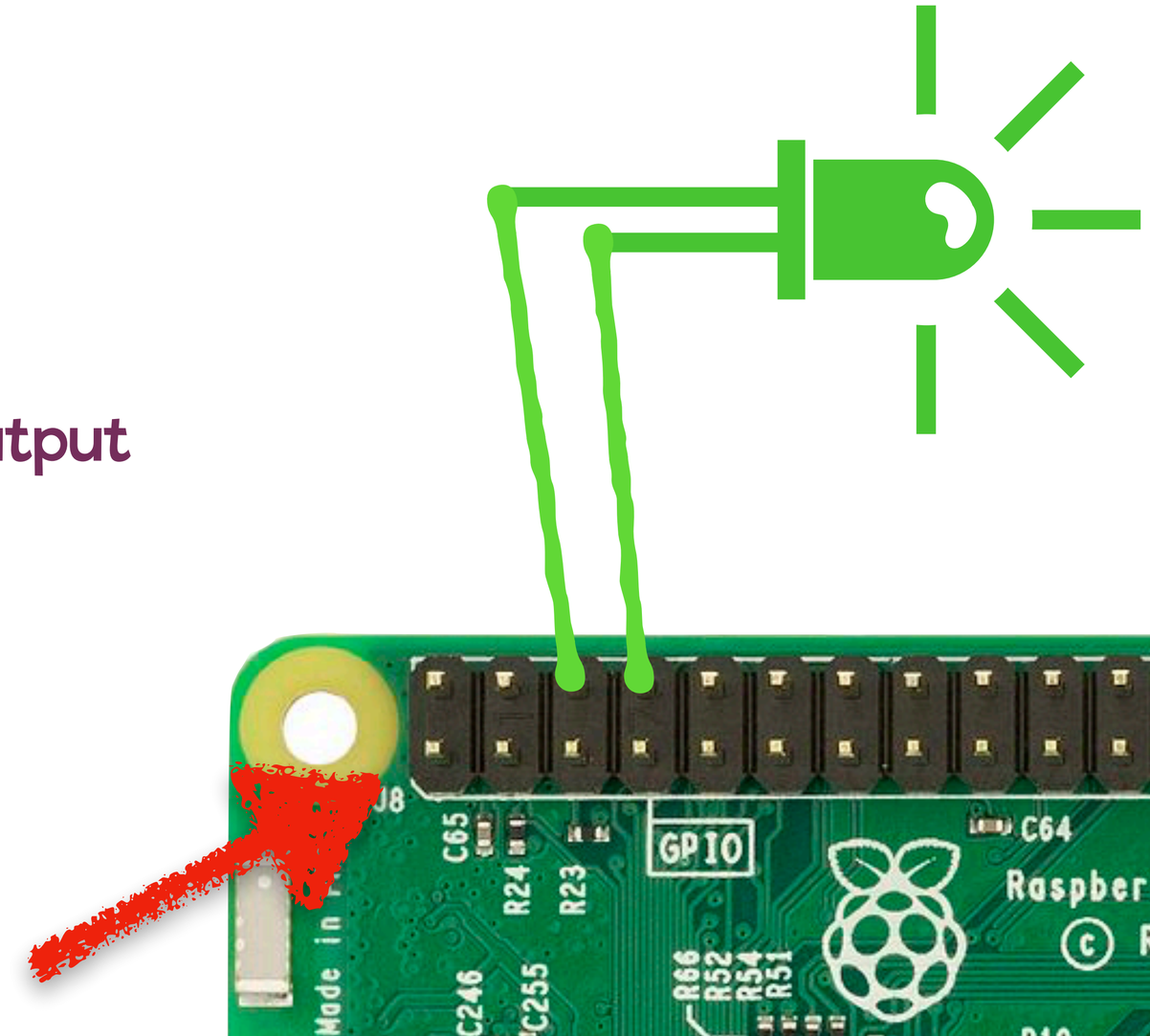
- Small and energy efficient
- Powerful ARM processor
- Mount it anywhere
- Runs Linux (which supports Swift!)
- WiFi and Ethernet
- General Purpose Input and Output (GPIO)



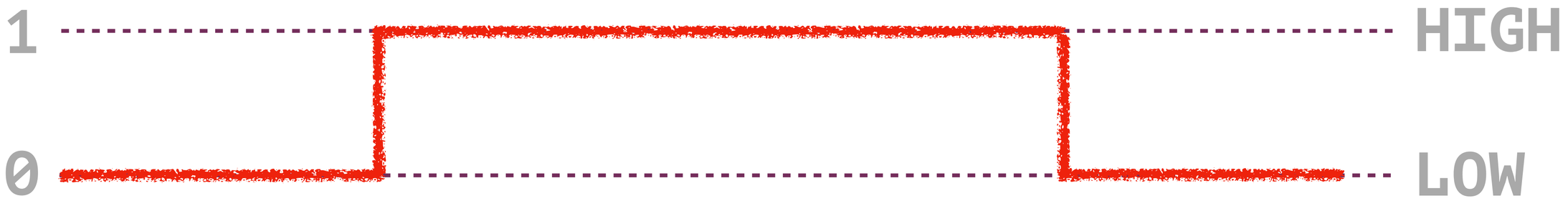


What is GPIO?

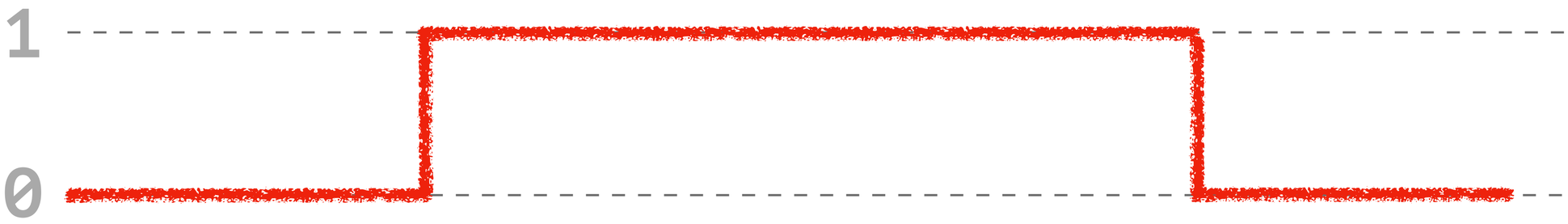
- "API for Electricity"
- GPIO pins connect to circuits
- Each pin can be an input or an output
- Input: Read if it's ON or OFF
- Output: Set it ON or OFF

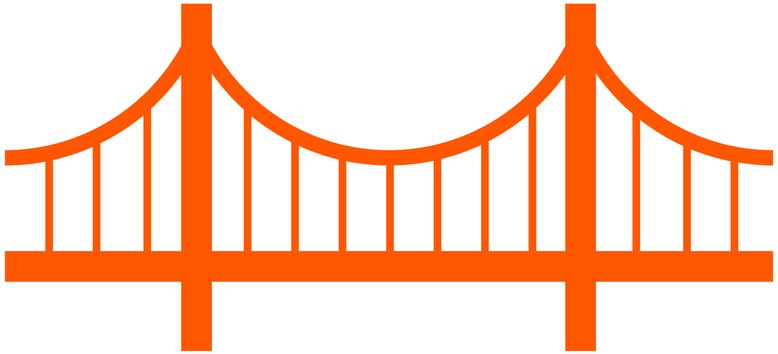


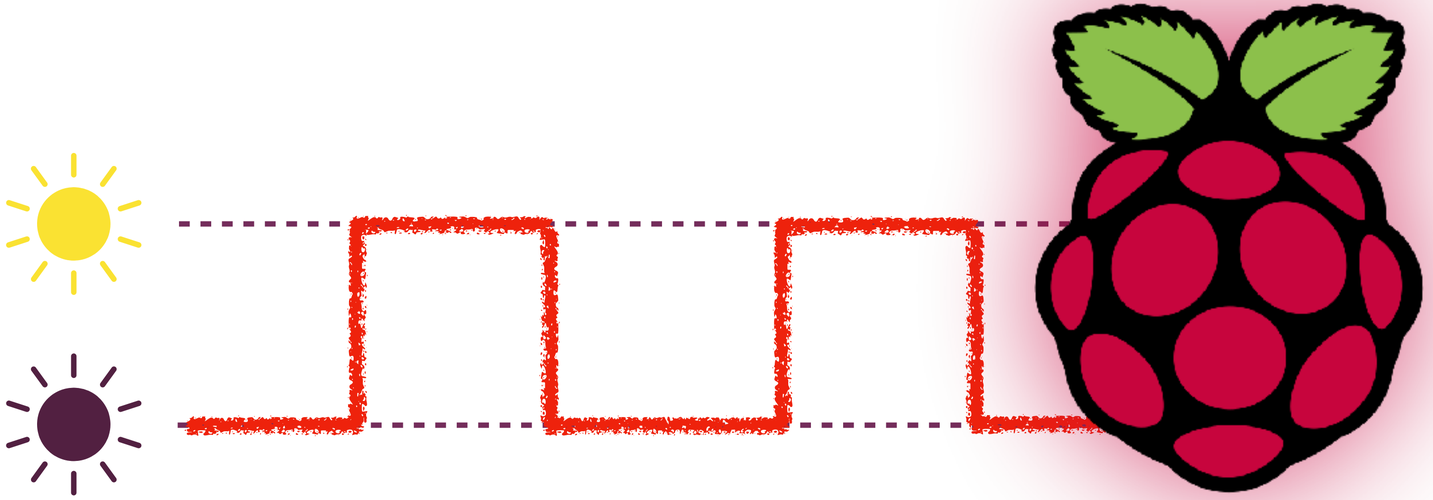
Outputs

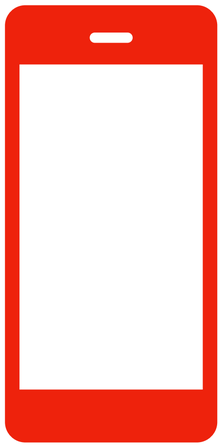


Inputs









Agenda

- ✓ Internet of Things
- ✓ DEMO
- ✓ Hardware
- Software
- What's next

Software

Setting up your Pi

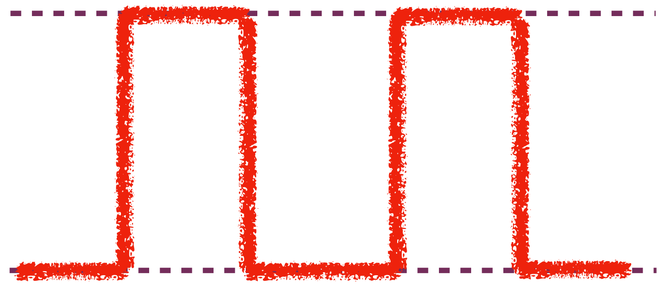
- Ubuntu Linux
- Installing the Swift Toolchain
- Installing dependencies
- Network configuration
- Helper guide
  - goo.gl/P5JKcy

Software Components

- Interacting with Things
  - SwiftyGPIO
- Interacting with the Internet
  - Server-side Swift (Vapor)
  - Push Notifications (Vapor-APNS)

SwiftyGPIO

- Interact with GPIO pins from Swift
- Simple to Use
- Integrates with other Swift libraries well



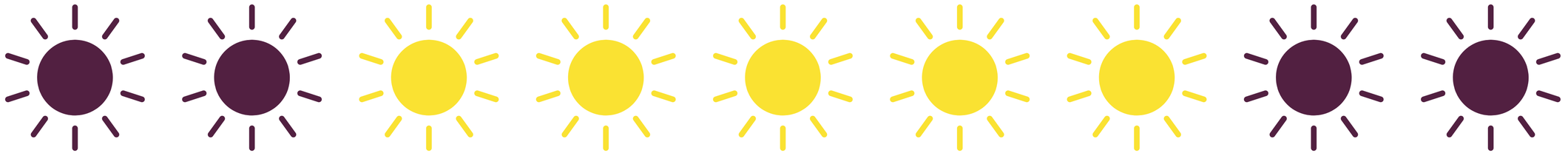
Pin Handling

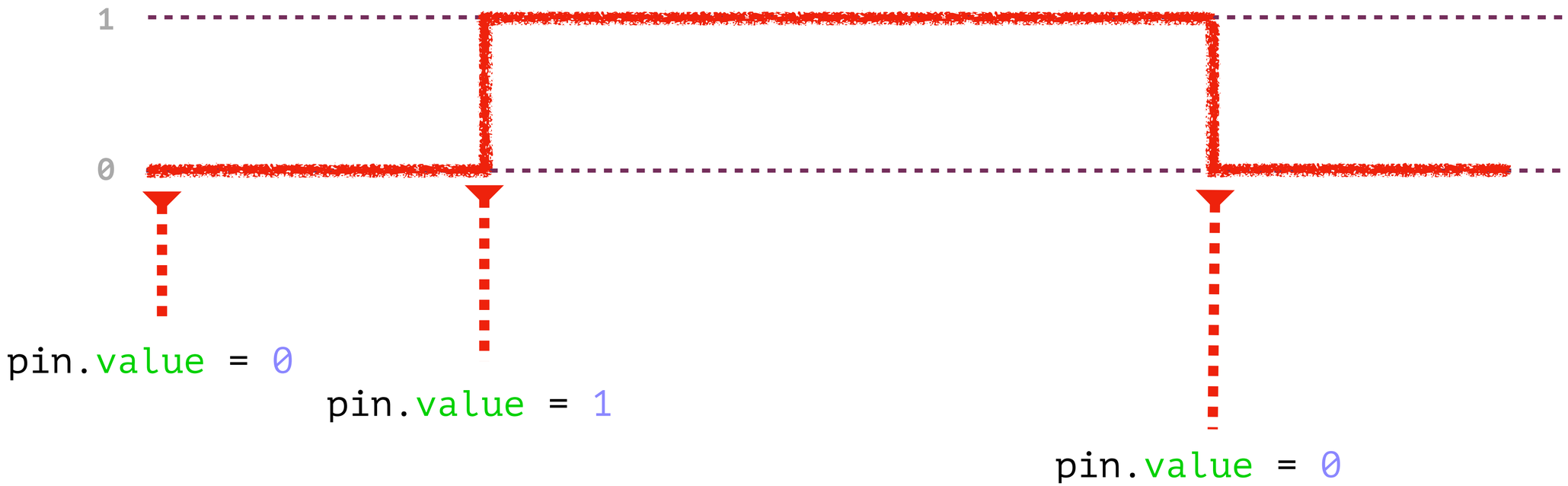
```
let pin = SwiftyGPIO.GPIOs(for: .RaspberryPi3)[.P2]
```

```
pin.direction = .OUT
pin.value = 1
```

```
pin.direction = .IN
print(pin.value) // “0” or “1”
```

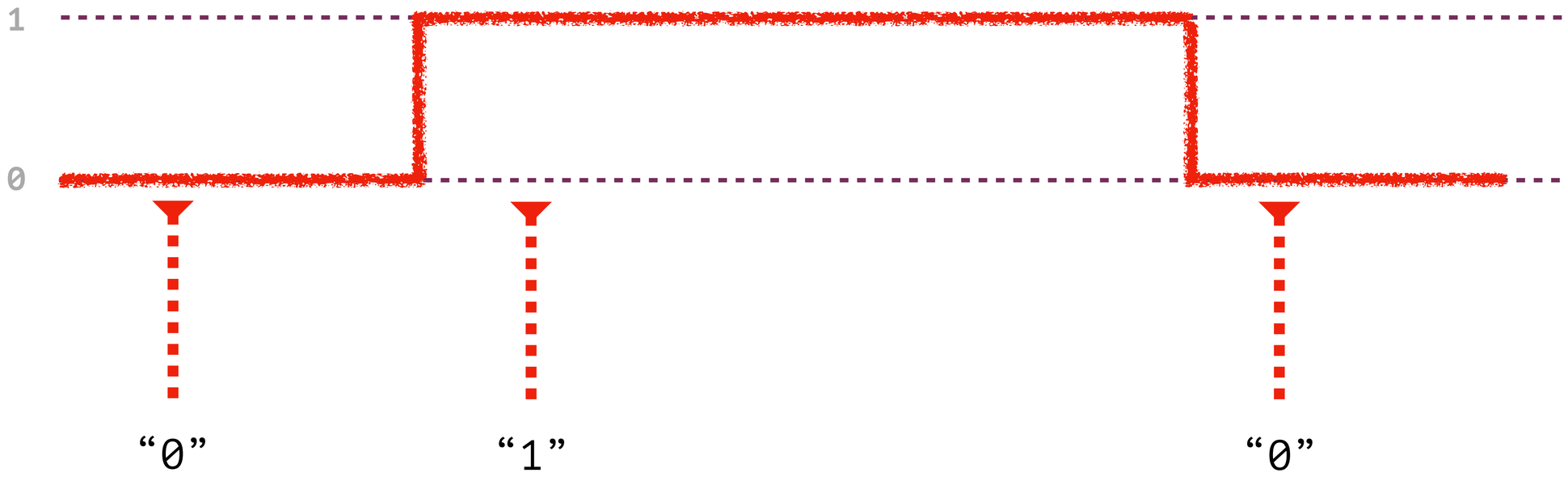
Outputs





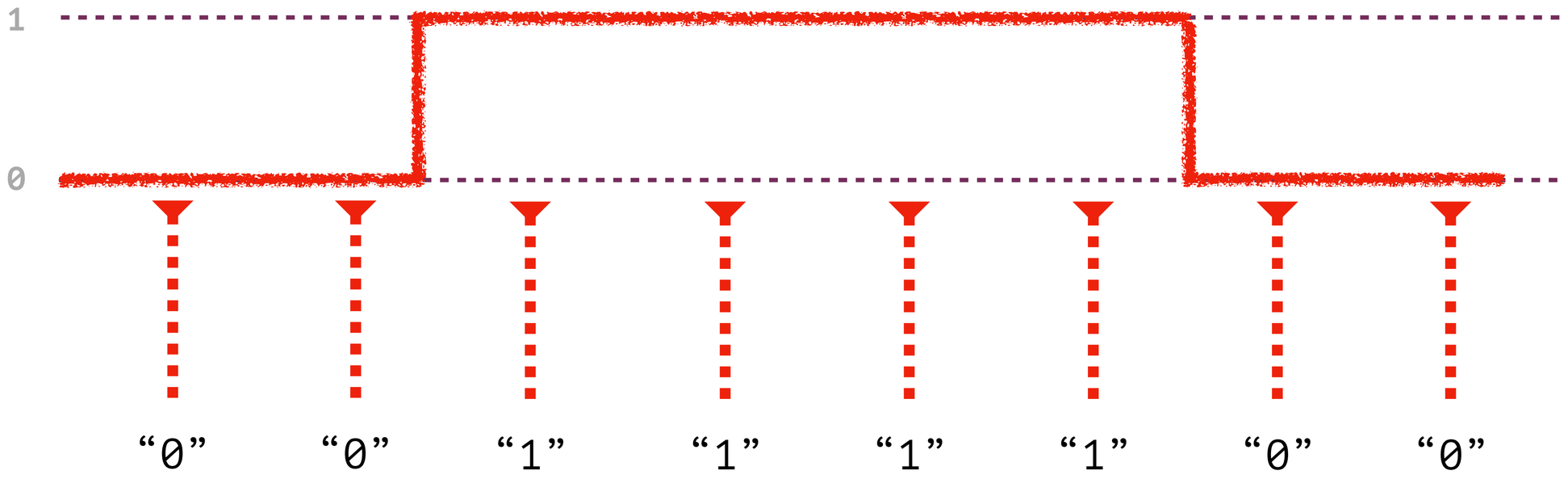
Inputs

```
print(pin.value)
```

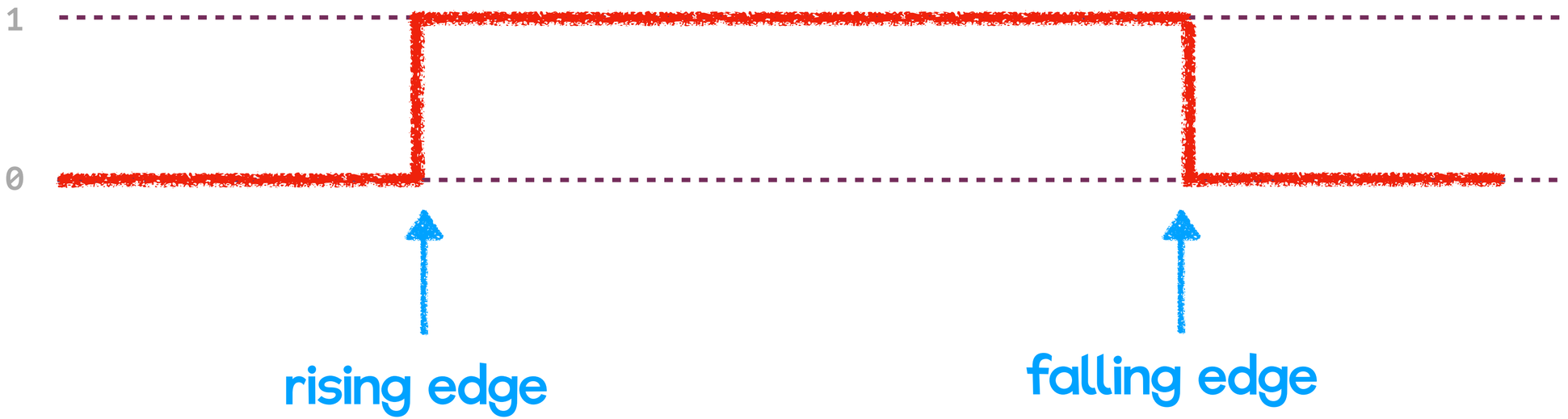


Inputs

```
print(pin.value)
```



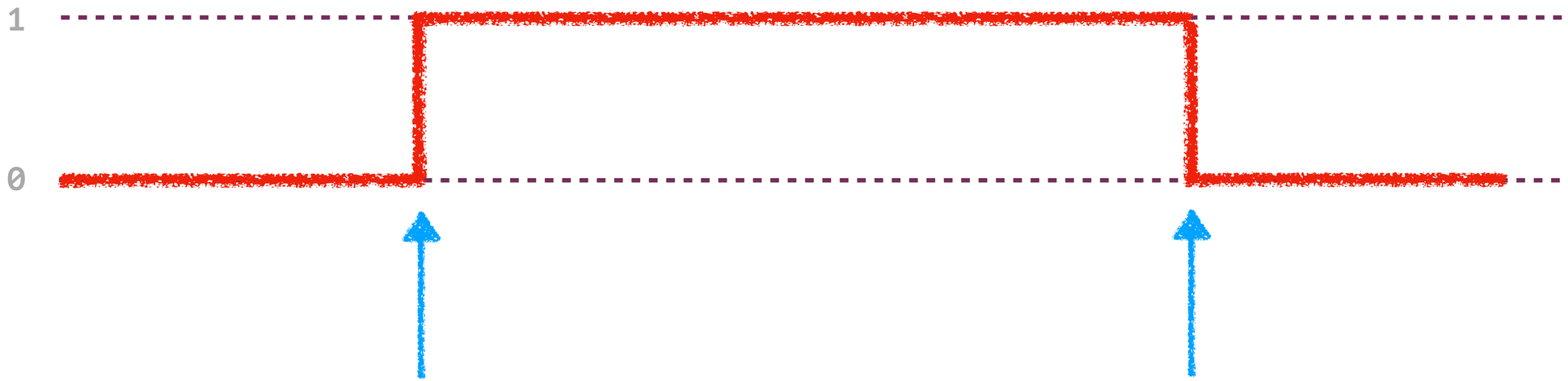
Edges



Input Handling

```
pin.onRaising { pin in . . . }

pin.onFalling { pin in . . . }

pin.onChange  { pin in . . . }
```

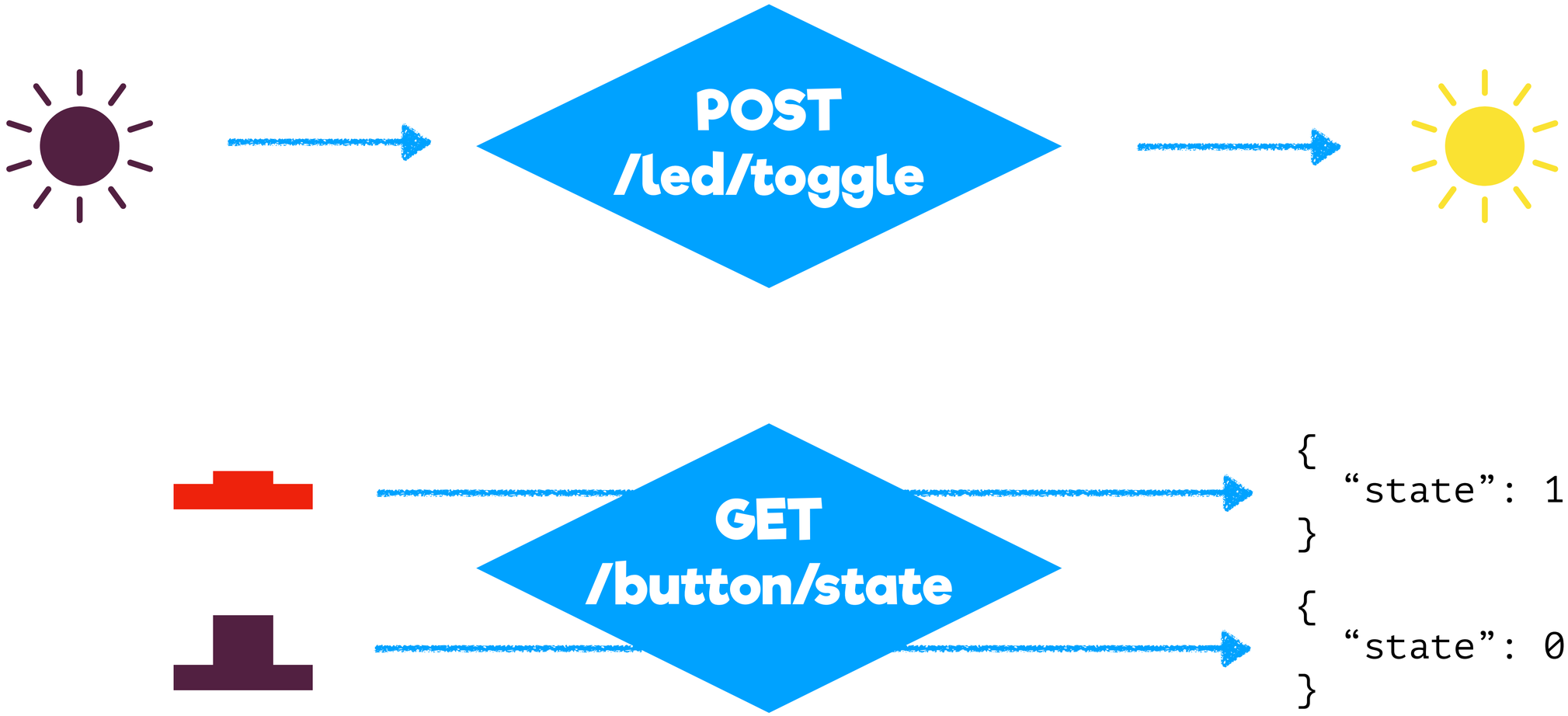
Inputs



Server Side Swift for IoT

- Build a Web Server in Swift
- Fundamentals are relatively constant
  - Routing requests and serving responses
- Vapor
  - Works on Raspberry Pi

Client API



Push Notifications

- Don't poll your state
- Alerts your client application immediately when a notable event occurs
- Swift library: Vapor APNS

Push Process

1. Ask your user for permission
2. Send the client's device token to your bridge, save it
3. When an event happens, create and submit a push payload

Permission

1. Ask your user for permission (in iOS Client)

```
let center = UNUserNotificationCenter.current()

center.requestAuthorization(options: [.alert]) { (granted, error) in

    if !granted { return }

    UIApplication.shared.registerForRemoteNotifications()

}
```

Device Token

2. Send the client's device token to your bridge

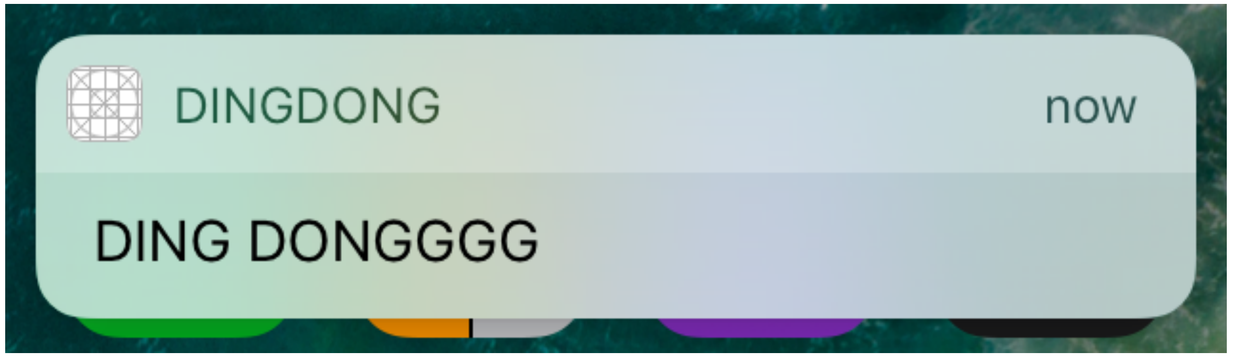
```
func application(_ application: UIApplication,
  didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {

    var urlRequest = URLRequest(url: setTokenURL)
    urlRequest.httpBody = token
    urlRequest.httpMethod = "POST"

    URLSession.shared.dataTask(with: urlRequest) {
        // handle completion
    }.resume()
}
```

Push Payload

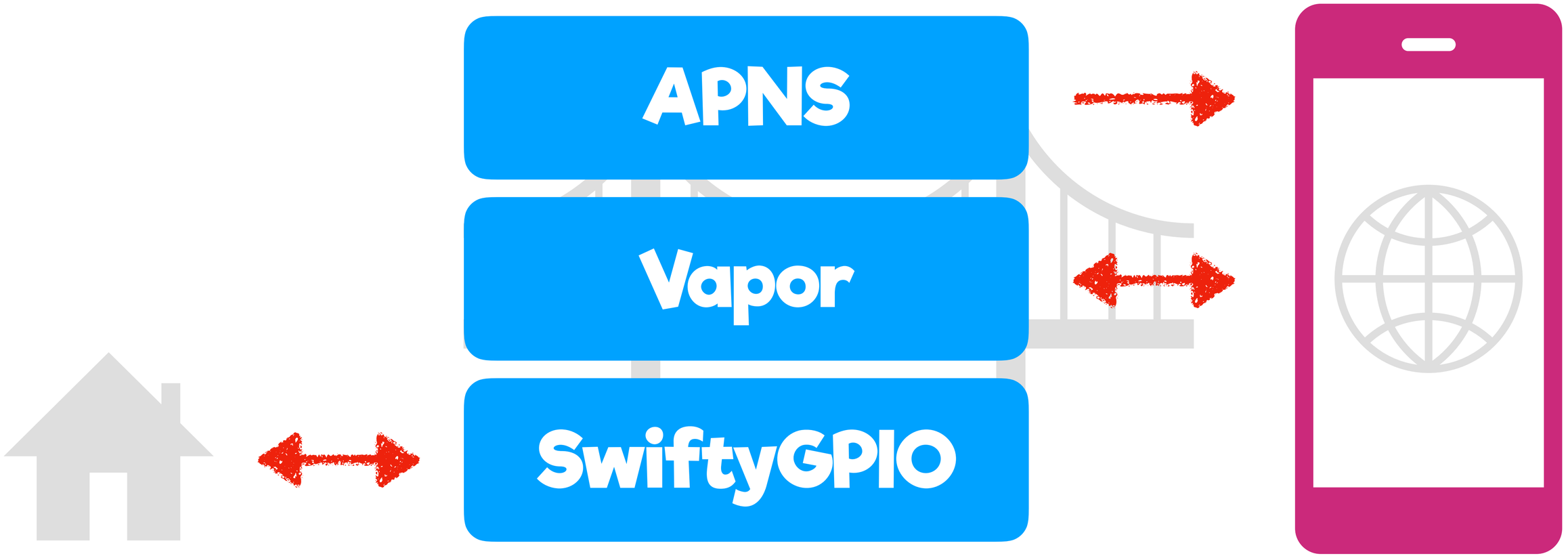
3. When an event happens, send a push payload (on bridge)

```
let message = ApplePushMessage(priority: .immediately,
                               payload: Payload(message: “DingDong!"),
                               sandbox: false)
let result = apns.send(message, to: token)
```

Success!



Software Summary



Swifty Doorbell

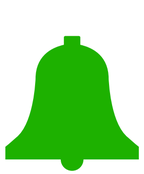
Interaction Flow

1. Doorbell rings
2. Bridge sends push notification
3. User taps “Door Open” button
4. Bridge executes door open process

Electrical Interface

Function	Pin	Direction	ON state	OFF state
Door Open	2	Output	Door Unlocked	Door Locked
Bell Detector	3	Input	Ringing	Quiet

Mapping the Flow



Send Push

POST /door/open

BELL

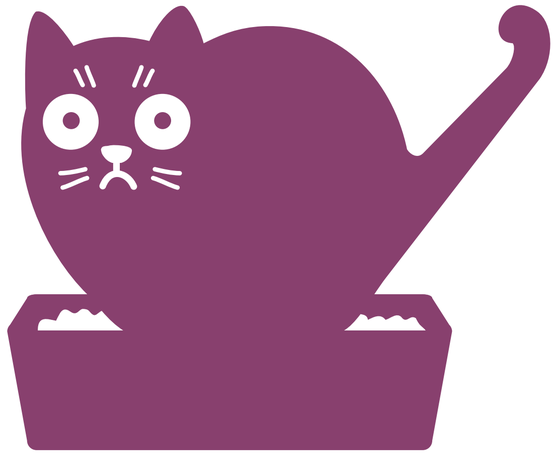
DOOR

Agenda

- ✓ Internet of Things
- ✓ DEMO
- ✓ Hardware
- ✓ Software
- What's next

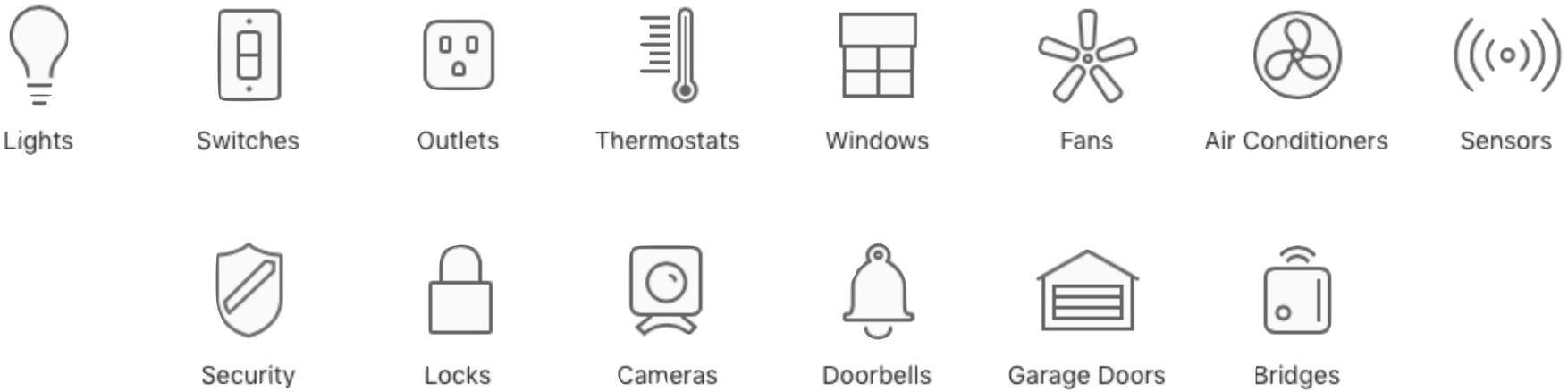
What's Next?

Future Projects?



HomeKit

Works with
Apple HomeKit



HomeKit Accessory Protocol Specification (Non-Commercial Version)

This document describes how to create HomeKit accessories that communicate with Apple products using the HomeKit Accessory Protocol for non-commercial purposes.

Companies that intend to develop or manufacture a HomeKit-enabled accessory that will be distributed or sold must be enrolled in the MFi Program.

Download

developer.apple.com/homekit/specification

Agenda

- ✓ Internet of Things
- ✓ DEMO
- ✓ Hardware
- ✓ Software
- ✓ What's next

Thanks!

 @huebnerob