Creating J.A.R.V.I.S.

Creating J.A.R.V.I.S.

An Awesome Guide To Automating Your Home

By Scott Preston

About Me

What’s J.A.R.V.I.S. stand for?

J.A.R.V.I.S.

J – Just

A – A

R – Rather

V – Very

I – Intelligent

S - System

About J.A.R.V.I.S.

- Ubuntu Server 12.04
- Core i5 / 16GB Ram / 240GB SSD
- 1GB Ethernet / 1 Bluetooth
- 4 USB Serial Ports
- 4 Slave Raspberry PIs w/LIRC & Temp Sensors
- 9 Network Cameras
- 1 Windows Machine for Skype
- @jarvispreston (Twitter Handle)

How did I get here?

Summer 2008

Initial Goals

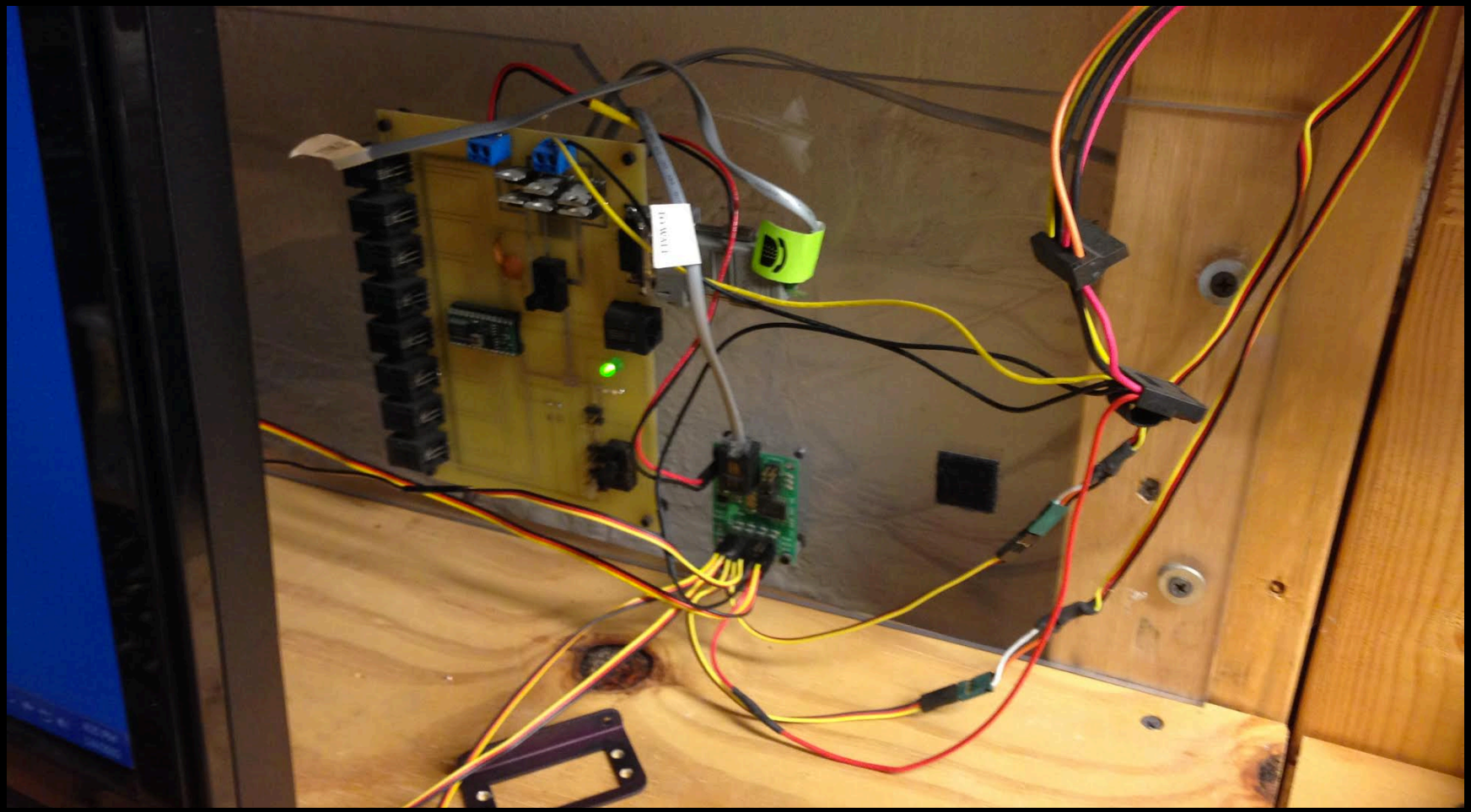
- Turn lights on/off Sunrise / Sunset
- Wake me up with weather report

I had an old robot (JR-MK2)

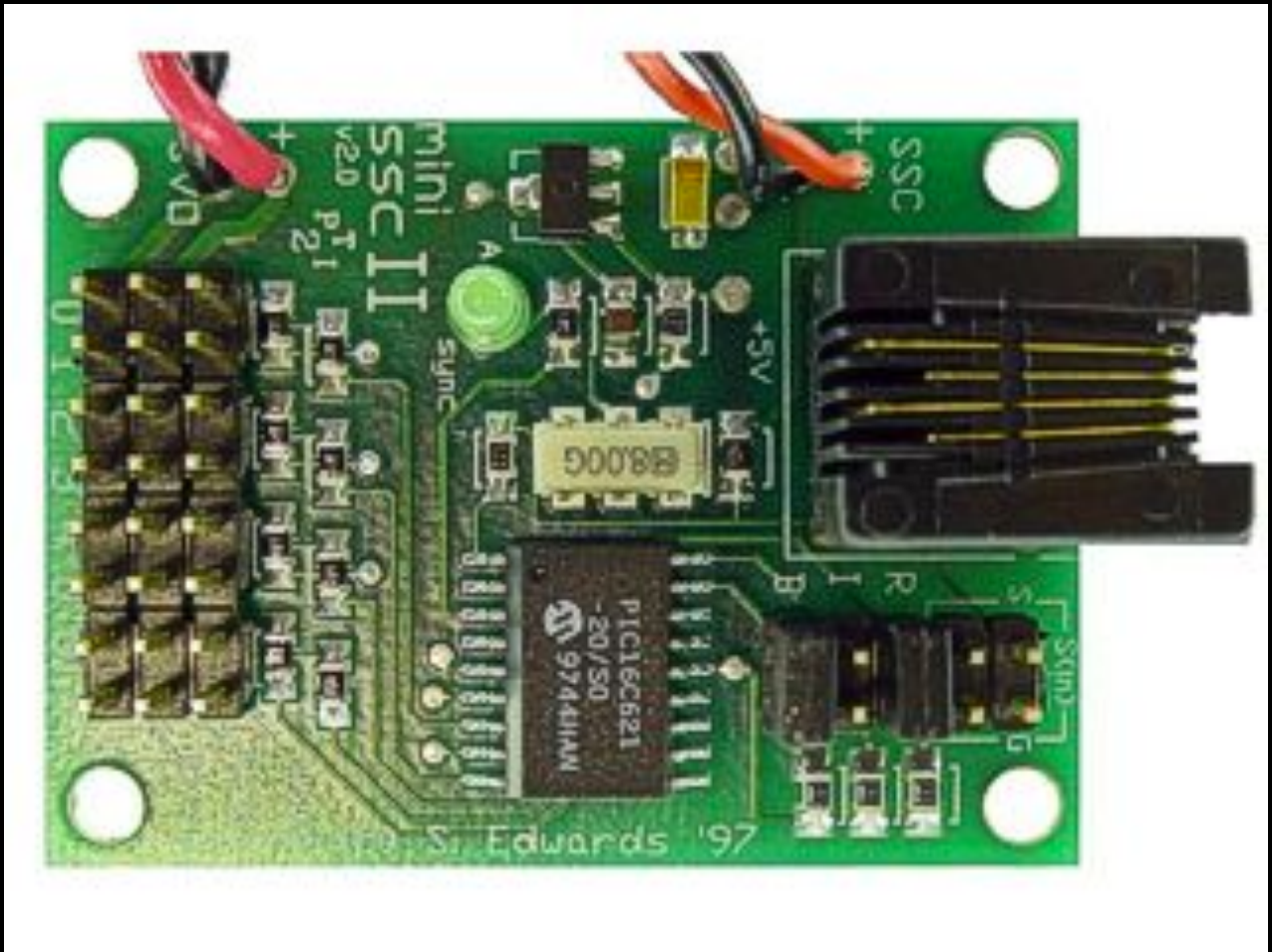
Robot Had 2 Parts I needed

- Basic Stamp & Distribution Board
- Serial Servo Controller

Stamp & Controller Board



SSC


mini
SSC II
v2.0
SVO
SSC
+5V
sync
PIC16C621
-20/SO
9744HAN
St Edwards '97

Before X10 ~ $7 Switch

Used SSC (Still Used)

- Lights On

http://192.168.1.60:8001/?pin=2&pos=200

http://192.168.1.60:8001/?pin=3&pos=70

- Lights Off

http://192.168.1.60:8001/?pin=2&pos=127

http://192.168.1.60:8001/?pin=3&pos=127

X10 – External Lights

What's X10?

- Developed in 1975 by Pico Electronics (Scotland)
- Digital Carrier transmitted onto a 120hz carrier existing power lines.
- Limitations – GCFI Outlets, Older Technology, Reliability, # of Devices
- Pros – Cheap, Nothing else around in 2008, Microcontrollers have built-in support

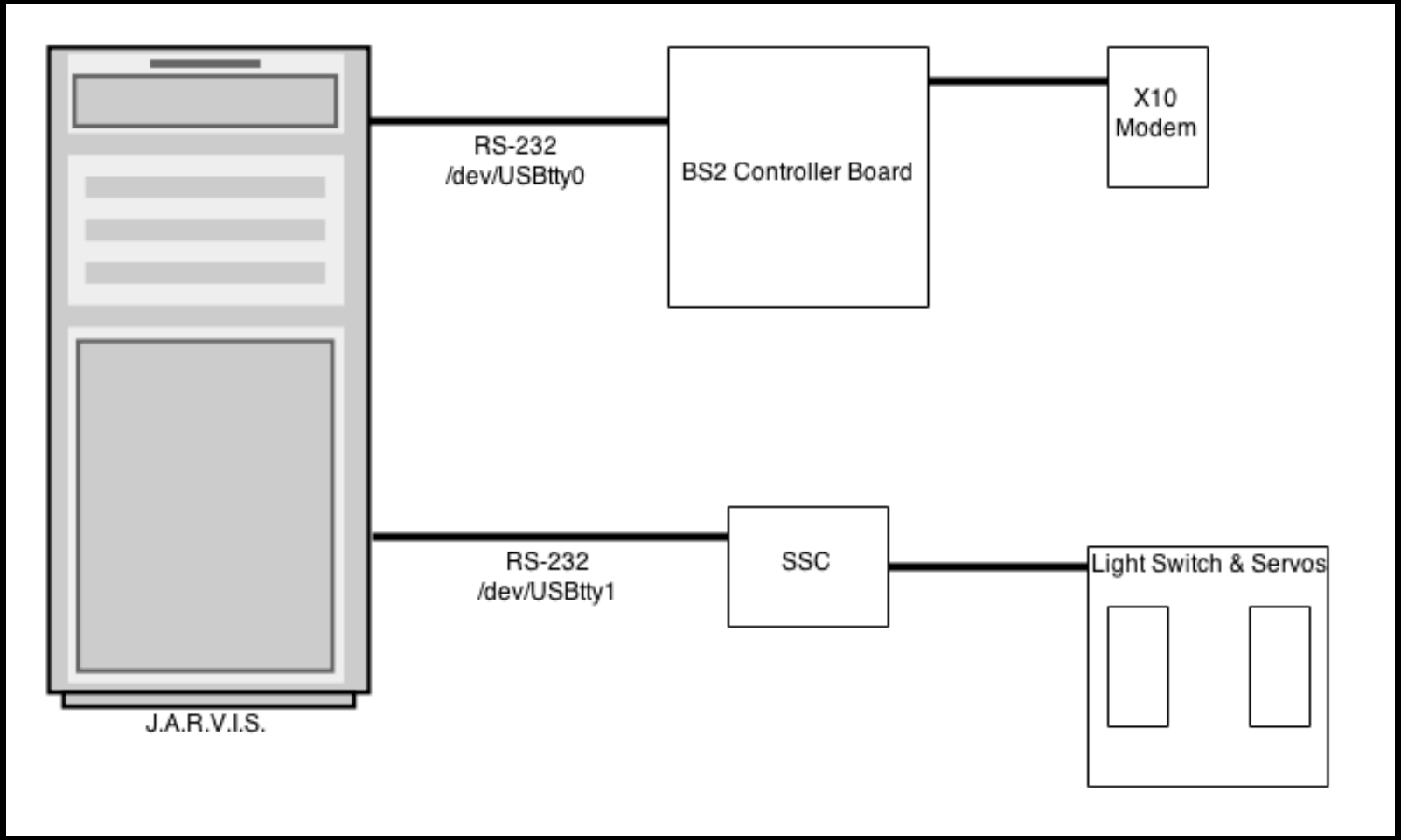
X10 – TW523 Interface~$60-$80

X10 – WS467 $16.99

X10 – Protocol (via BS2)

- XOUT mPin, zPin, [houseCode\unitCode]
- XOUT mPin, xPin, [houseCode\onORoff]
- houseCode = 1-16
- unitCode = 1-16
- onORoff = 0, 1

Initial Layout



Node Set-Up

- npm install serial-port (Serial Ports)
- npm install serial-servo (Serial Servo Controllers)

X10.js – SerialPort

```
var http = require('http');
var url = require('url');
var port = 8000;

var usbport = "/dev/ttyUSB0";
var SerialPort = require("serialport").SerialPort;
var sp = new SerialPort(usbport, {
    baudrate: 9600
});
```

Serial Port Setup

X10.js – Server Code

```
var server = http.createServer(function (request,
response) {
    var url_parts = url.parse(request.url, true);
    var house = url_parts.query.house;
    var device = url_parts.query.device;
    var onoff = url_parts.query.onoff;
    write(house,device,onoff);
});
```

Http Parms to Serial Port

X10.js – Serial Writing

```
function write(house, device, onoff) {
    if (house >= 0 && device >= 0 && onoff >= 0) {
        var b = [100, house, device, onoff];
        sp.write(b);
    }
}
```

Byte[] sent to serial port

Sunrise & Sunset

- http://api.wunderground.com/api/USER_KEY/astronomy/q/43212.json
- Gets sun-rise
- Gets sun-set
- CRON runs each minute and checks for time match

Weather Underground – REST API

- http://api.wunderground.com/api/USER_KEY/conditions/q/43212.json
- Schedule Via CRON (hourly)
- Populate to database via wrapper.

Putting It All Together

- Old Robot Parts w/Serial & X10
- X10 Switches
- Weather Underground w/Sunrise & Sunset Data
- CRON Scheduler
- Results: Robot that can turn on/off lights for me.
- Missing: Text-To-Speech (later)

More Devices

Since X10 is 40 years old

Kinds Of Devices (non-X10)

- Proprietary (Lots of these) – On their own
- Wifi – Many devices, limited APIs & Smart Phones
- BLE – Blue Tooth Low Energy (Smart Phone Only)
- Z-Wave – Need Interface (Wireless)
- Insteon – Need Interface module Wireless + X10

Kinds Of Devices (non-X10)

- Proprietary (Lots of these) – On their own
- Wifi – Many devices, limited APIs & Smart Phones
- BLE – Blue Tooth Low Energy (Smart Phone Only)
- Z-Wave – Need Interface (Wireless)
- Insteon – Need Interface module Wireless + X10

Kinds Of Devices (non-X10)

- Proprietary (Lots of these) – On their own
- Wifi – Many devices, limited APIs & Smart Phones
- BLE – Blue Tooth Low Energy (Smart Phone Only)
- Z-Wave – Need Interface (Wireless)
- Insteon – Need Interface module Wireless + X10

Kinds Of Devices (non-X10)

- Proprietary (Lots of these) – On their own
- Wifi – Many devices, limited APIs & Smart Phones
- BLE – Blue Tooth Low Energy (Smart Phone Only)
- Z-Wave – Need Interface (Wireless)
- Insteon – Need Interface module Wireless + X10

Device Types

- Controllers
- Locks
- Thermostats
- Sensors
- Lights
- Switches
- Smart Meters

Proprietary - IRIS by Lowes

Some devices Z-Wave Compatible!

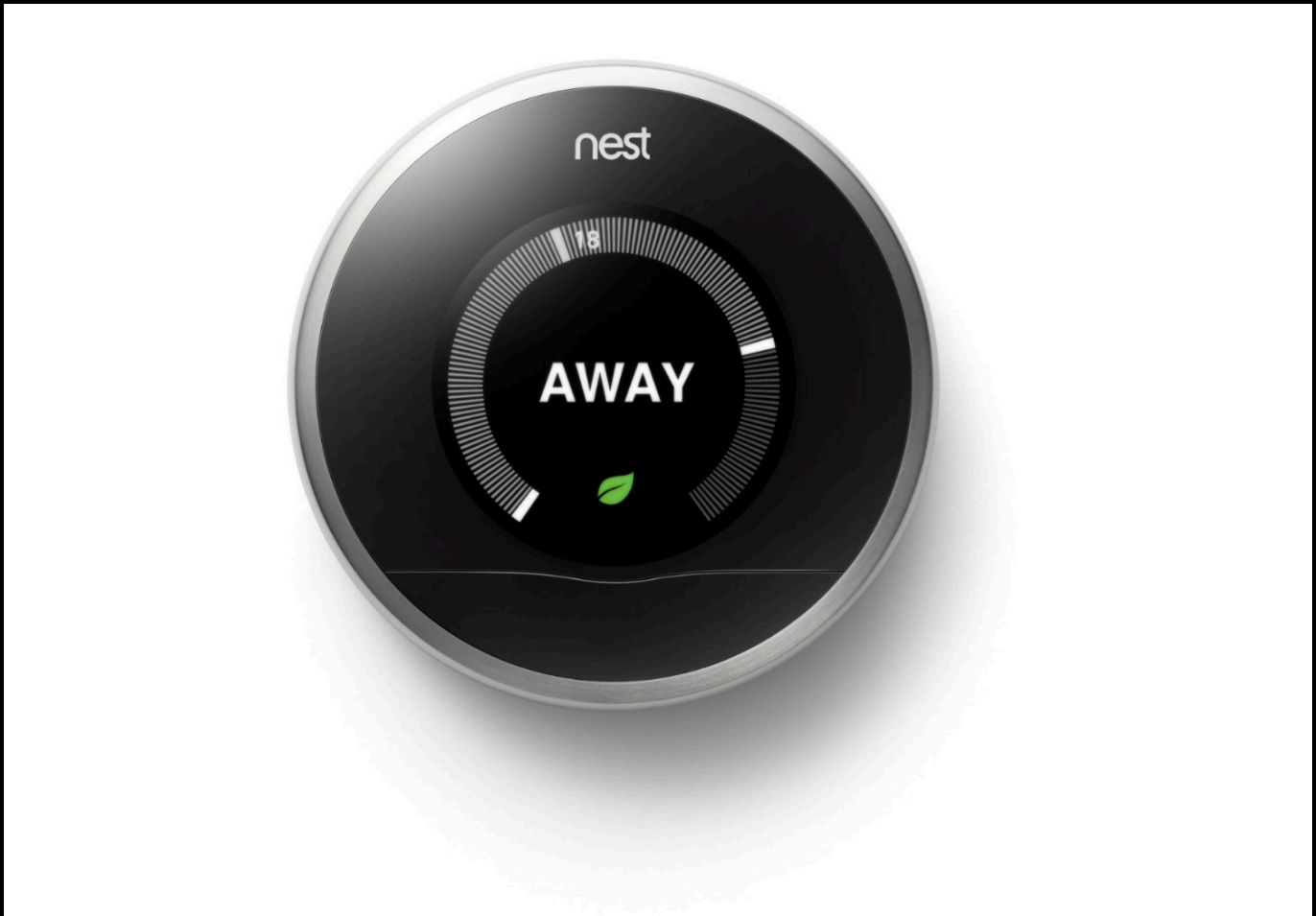
Schlage Z-wave Lock

KEVO – Bluetooth LE (Not Z-Wave)

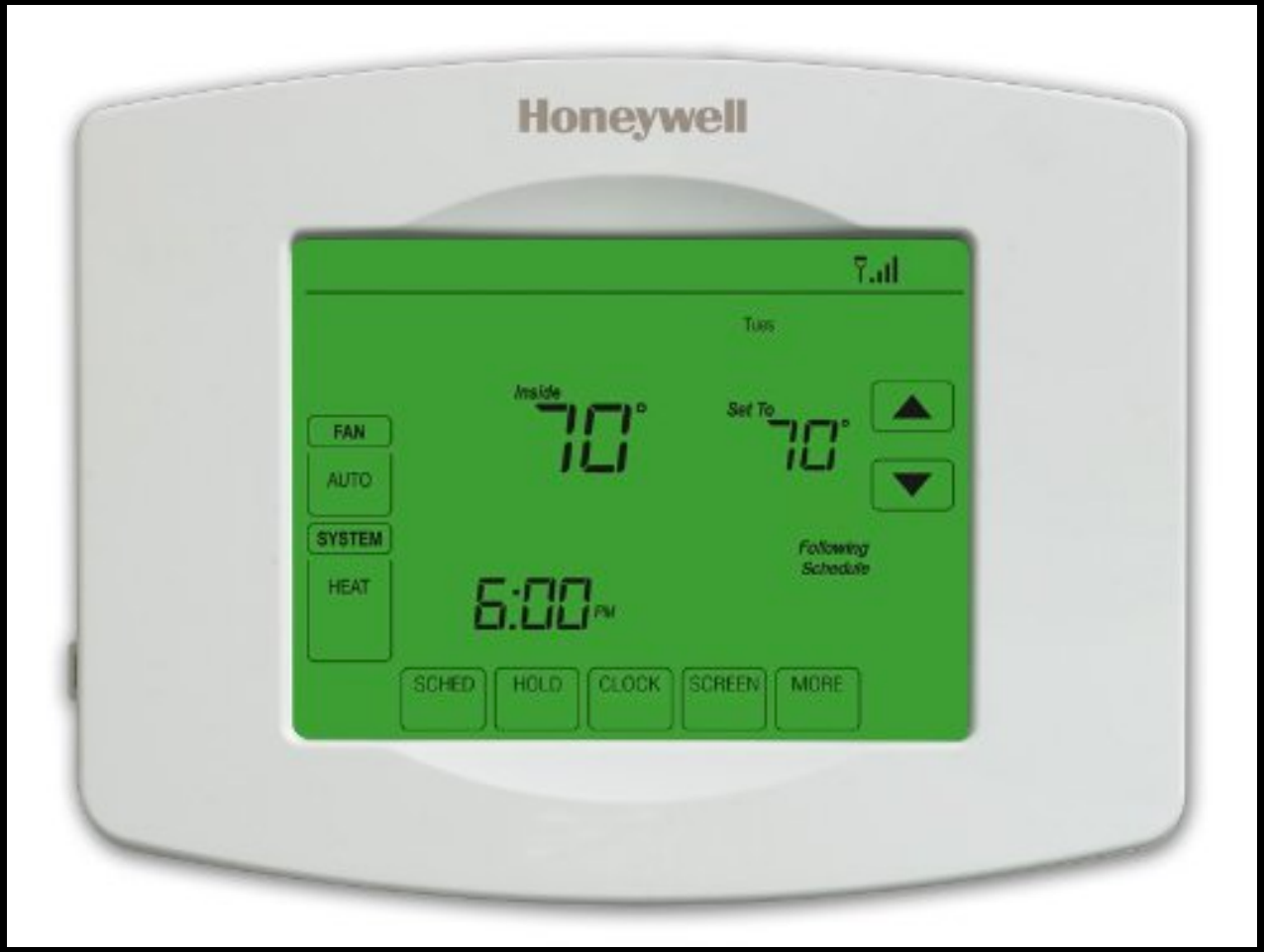
Kwickset – Z-Wave

nest Thermostat ($250)

REST API available from nest and full SDK.



Z-Wave + WiFi ~$75 to $200



Aeon Labs – Z-Wave Switch

Phillips Hue (Zigbee & Ethernet)

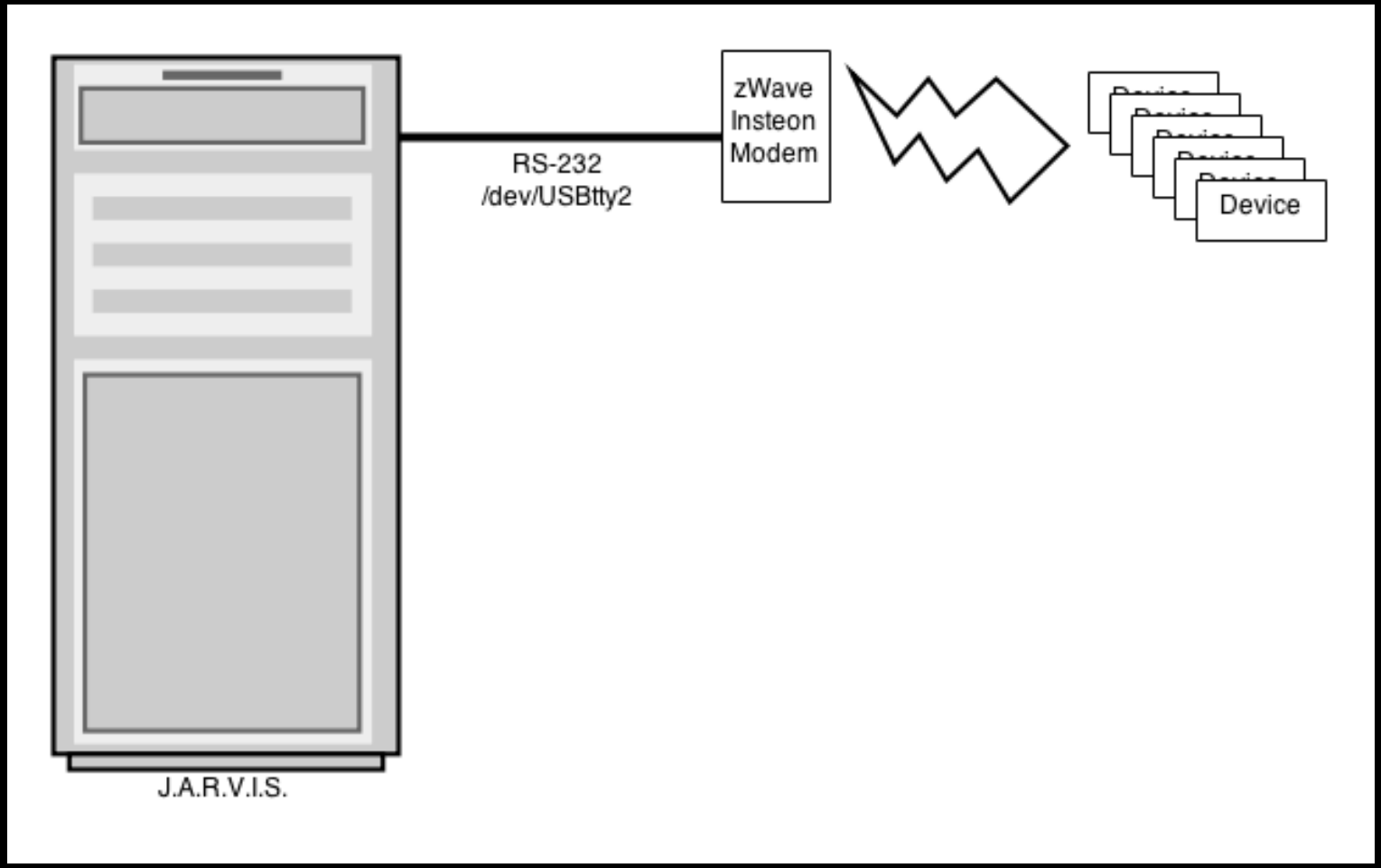
http://www.developers.meethue.com/tools-and-sdks

WeMo – WiFi Switch

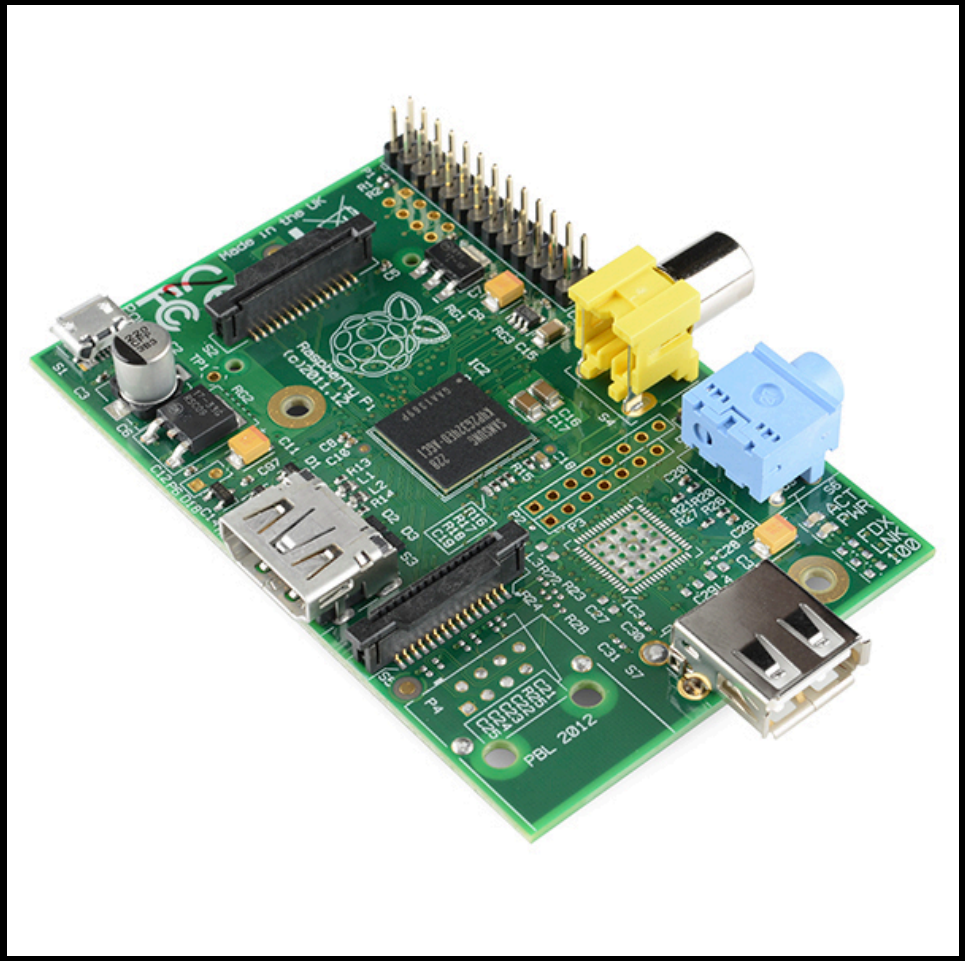
Many NPM modules available to use.

Z-Wave Flood Sensor

Z-wave Interfacing

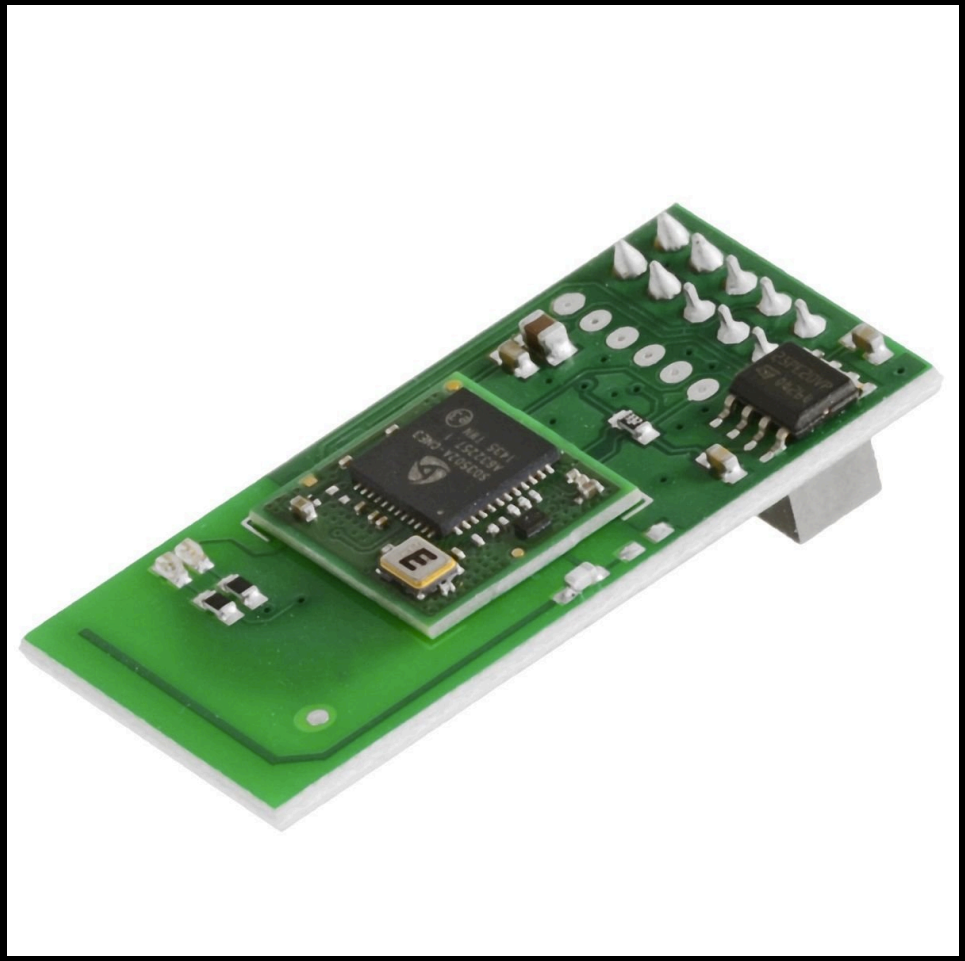


Raspberry Pi ~ $30



Z-Wave GPIO Daughter Card - $60

http://razberry.z-wave.me/



Z-Wave USB ~$40

Google ‘openzwave’ for npm modules.

How close are we to J.A.R.V.I.S.?

J.A.R.V.I.S.

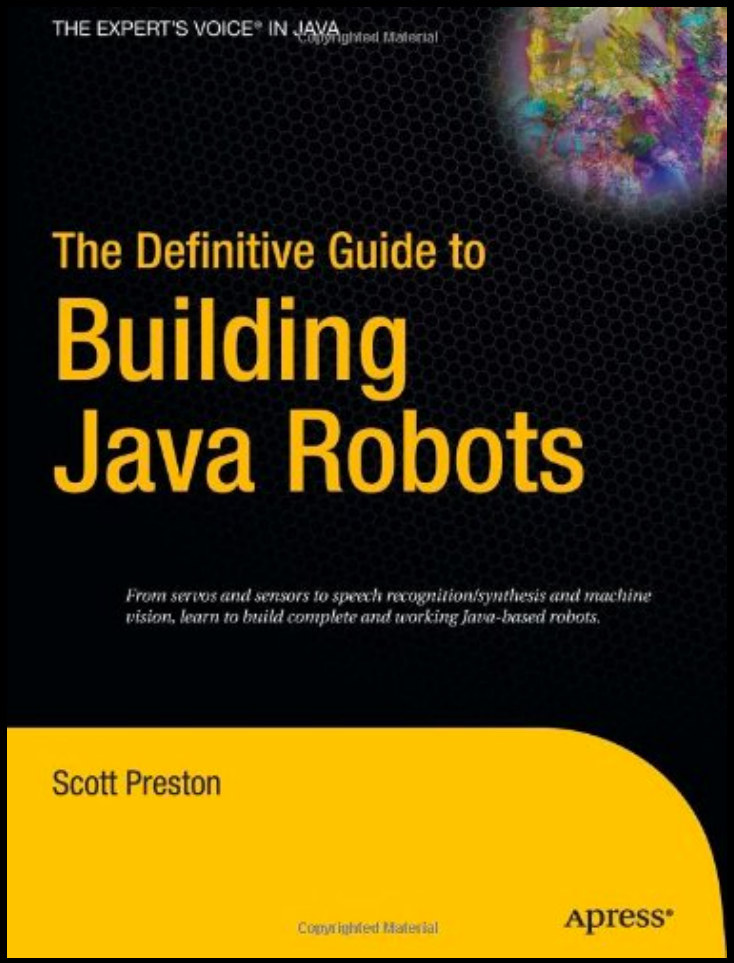
- **Home Sensors**
- **Home Control (Lights/Blinds/Etc.)**
- **Speech Recognition / Text-To-Speech**
- Computer Vision
- TV / Stereo Control
- Event Based System & Robot Control
- ~~3D Metal Printers~~

Speech Recognition

w/Sphinx4

Open Source Java Speech Recognition

Wrote this book in 2005.



SR – Part 1 (Chapter 6)

Gets “best” result

```
while (true) {
    Result result = recognizer.recognize();
    if (result != null) {
        String resultText = result.getBestFinalResultNoFiller();
        JarvisUtils.logger("I heard: " + resultText);
        String recognizerURL = "http://192.168.1.60/voice/
respond/";

ApacheHttp.postSingleString(recognizerURL,"txt",resultText);
        JarvisUtils.pause(2000); // wait a second before listening
again
    }
}
```

SR – Part 1 (Chapter 6)

```
while (true) {
    Result result = recognizer.recognize();
    if (result != null) {
        String resultText = result.getBestFinalResultNoFiller();
        JarvisUtils.logger("I heard: " + resultText);
        String recognizerURL = "http://192.168.1.60/voice/
respond/";

ApacheHttp.postSingleString(recognizerURL,"txt",resultText);
        JarvisUtils.pause(2000); // wait a second before listening
again
    }
}
```

Passes to service for response & TTS

Before 2014

- Each Call To Specific Words → Map to specific URLs
- Each time I wanted to update, I had to re-compile, and updated.
- I wanted something more dynamic.
- if command1, call service1, if command2, call service2

Early 2014

Sent words to service.

Service mapped words to new service endpoints.

Late 2014

Changed from service mappings to event publish via RabbitMQ

Text To Speech

- 3rd Party Voice (Cepstral TTS)
- Festival TTS (Free)
- Linux command I'm using:

/usr/bin/padsp swift "words to speak"

Remote Text To Speech

```
swift "$1" -o tmp.wav // create wav
lame tmp.wav tmp.mp3 // convert to mp3
scp tmp.mp3 pi@10.10.10.32:tmp.mp3 // send
ssh pi@10.10.10.32 'mpg123 tmp.mp3' // play
```

Remote Text To Speech

```
swift "$1" -o tmp.wav // create wav
lame tmp.wav tmp.mp3 // convert to mp3
scp tmp.mp3 pi@10.10.10.32:tmp.mp3 // send
ssh pi@10.10.10.32 'mpg123 tmp.mp3' // play
```

Remote Text To Speech

```
swift "$1" -o tmp.wav // create wav
lame tmp.wav tmp.mp3 // convert to mp3
scp tmp.mp3 pi@10.10.10.32:tmp.mp3 // send
ssh pi@10.10.10.32 'mpg123 tmp.mp3' // play
```

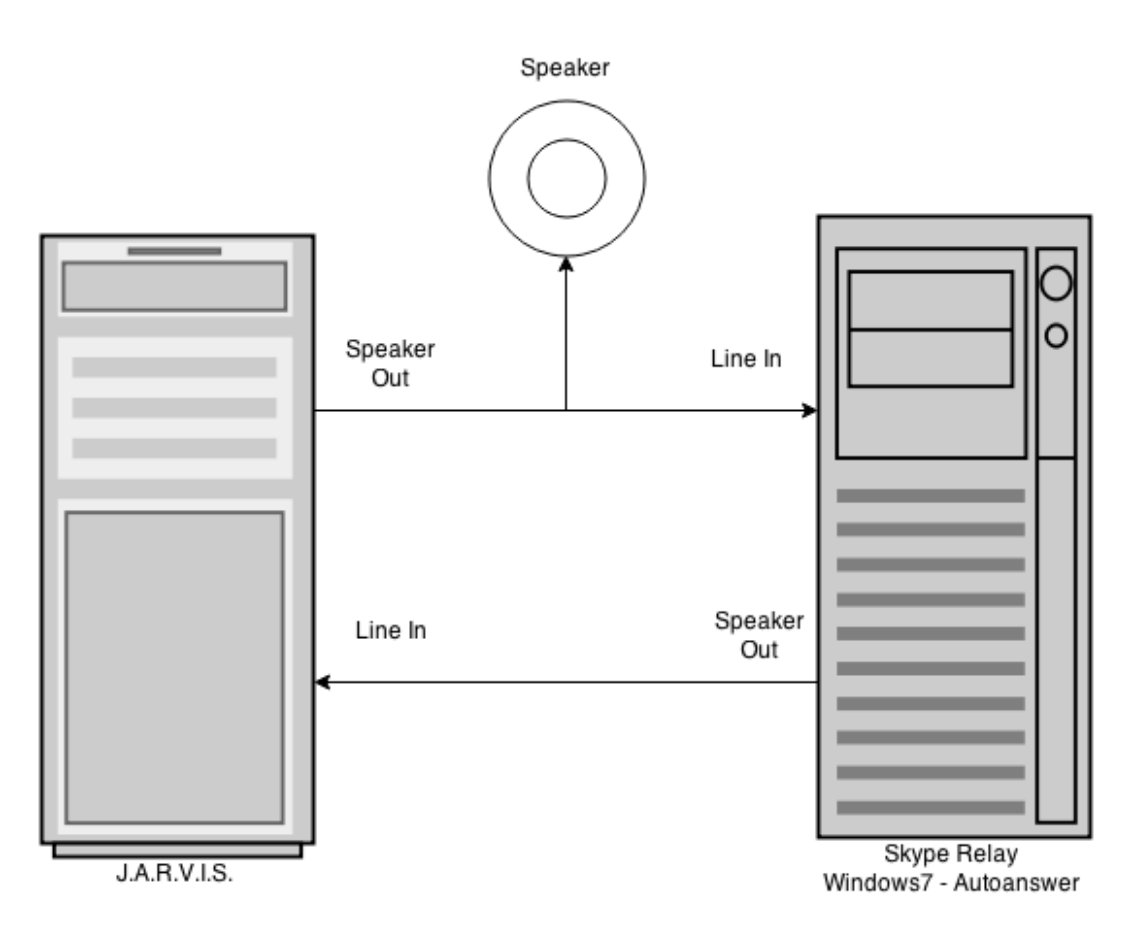
Remote Text To Speech

```
swift "$1" -o tmp.wav // create wav
lame tmp.wav tmp.mp3 // convert to mp3
scp tmp.mp3 pi@10.10.10.32:tmp.mp3 // send
ssh pi@10.10.10.32 'mpg123 tmp.mp3' // play
```

How do I talk to J.A.R.V.I.S.?

- Microphone
- Wireless Headset
- Bluetooth
- Skype via API?
- Skype as Services??

Skype As Service (low-tech)



Putting It all Together

- Speech Recognition – Publishing Events
- Text To Speech either via Speaker, Skype (Phone) or Remote Speaker
- Tell J.A.R.V.I.S. to turn on lights and turn off lights

J.A.R.V.I.S Voice Demo

What Next?

Computer Vision

- 9 Network Cameras in Total
- To Report Motion Detection Events
- Each capable of doing face recognition (still debugging)
- J.A.R.V.I.S. – Using Java based motion detection in multiple threads for each camera defined via UI
- Table of Camera Stored in J.A.R.V.I.S Database

Foscam Pan/Tilt

Resolution 640x480 and 1280x720

Camera Parameters

- Name
- Snapshot_url for taking JPG image
- Sample Frequency in milliseconds
- Sensitivity
- Top, Left, Height, Width
- Notify URL / Event To Publish

Vision Summary

- 9 Cams Mapped To Location
- J.A.R.V.I.S. monitors “motion” events and “face” events
- Checks For Motion every 500ms to 2000ms
- Face checks only if motion

TV / Stereo Control

Device Might Have an API

LIRC w/Raspberry PI

DirecTV - Power On

http://192.168.1.99:8080/remote/processKey?key=poweron&hold=keyPress

DirecTV – Power Off

http://192.168.1.99:8080/remote/processKey?key=poweroff&hold=keyPress

DirecTV – Change Channel

http://192.168.1.99:8080/tv/tune?major=10

Sparkfun – Max Power LED $2.95

LIRC

- Linux Infrared Remote Control - http://www.lirc.org/
- apt-get install lirc
- update /etc/modules
- update /etc/lirc/hardware.conf
- update /etc/lirc/lircd.conf

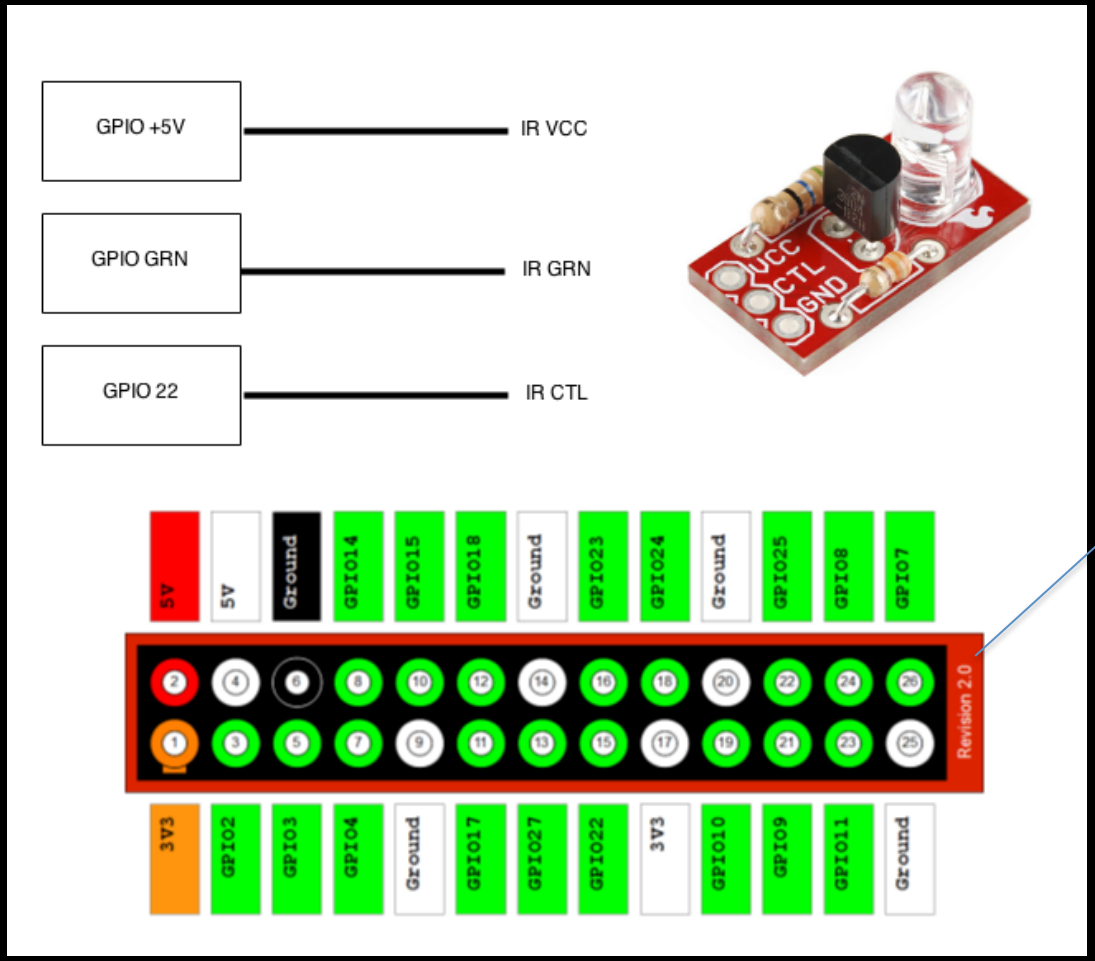
Invoking From Service

Use php exec(garage-power.sh)

```
#!/bin/bash
ssh pi@10.10.10.32 'irsend
SEND_ONCE yam POWER'
```

** Subscribe to STEREO_ON event, calls shell

LIRC Diagram

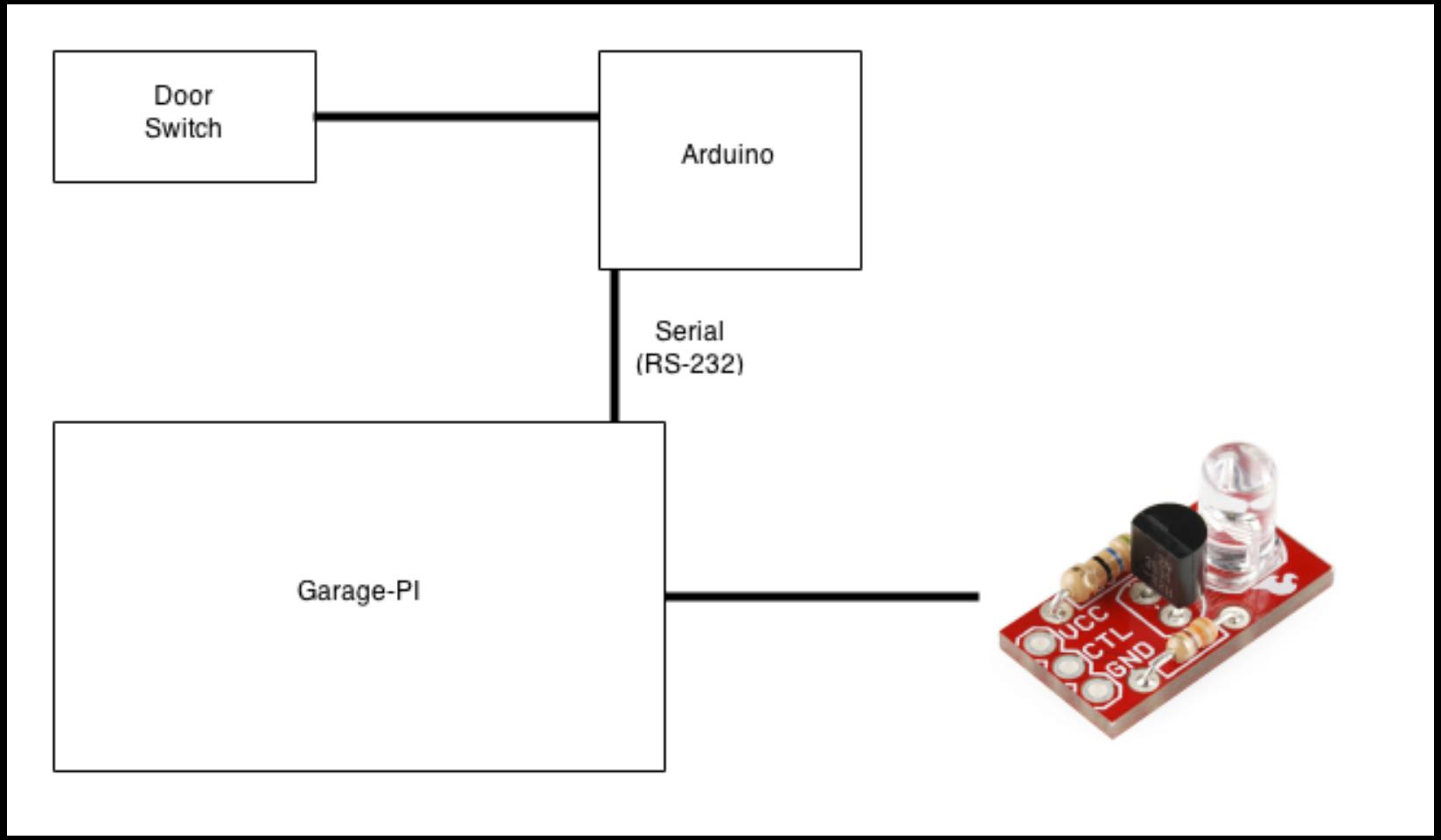


Raspberry PI GPIO

lircd.conf

```
begin remote

  name  yam
  bits          32
  flags SPACE_ENC
  eps           20
  aeps          200

  header       8800  4400
  one           550  1650
  zero          550   550
  ptrail        550
  repeat       8800  2200
  gap         38500
  toggle_bit      0

  frequency    38000

      begin codes
          POWER                0x000000005EA1F807
```

Garage Wiring Diagram



Garage Behavior

- **PUBLISHERS (garage-pi)**
- If (open) publish (‘garage-door-open’)
- **CONSUMERS (garage-pi, jarvis, feynman)**
- turn on PC
- greet via tts
- turn on garage lights,
- turn on garage TV or Stereo (different rules)

Final Step

Event Based Architecture

Why Events?

- Everything that's happening is some kind of event.
- Multiple subscribers to events - jarvis, tvroom-pi, garage-pi, basement-pi, feynman-jr, feynman (in garage)
- One places to control event processing vs. in each app

Before Events

- Speech Recognition – Mapped to services, words were hard coded, each update required new java compile.
- Hard coded URLS everywhere – 4 Pis, new functionality, required going everywhere and updating service definition.
- Chains of Services (IF-THIS-THEN-THAT) IFTTT required going lots of places

NodeJS Installation

npm install amqp

RabbitMQ - Installation

apt-get install rabbitmq-server

Dynamically Load Event Rules

- {
  "name": "test",
  "events": [{
  "name": "broadcast",
  "data": {
  "endpoints": [
  "http://www.google.com",
  "http://www.scottpreston.com"]
  }
  }, {
  "name": "notify",
  "data": {
  "message": "tested google and scottpreston"
  }
  }
  ]}

Dynamically Load Event Rules

Event Name
(What's Published)

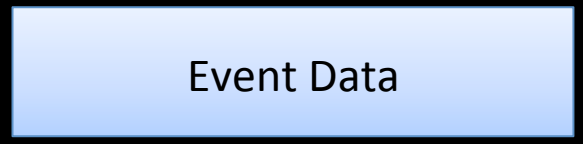
```
{
    "name": "test",
    "events": [ {
          "name": "broadcast",
          "data": {
            "endpoints": [
              "http://www.google.com",
              "http://www.scottpreston.com"]
          }
       }, {
          "name": "notify",
          "data": {
            "message": "tested google and scottpreston"
          }
       }
    ]}
```

Dynamically Load Event Rules

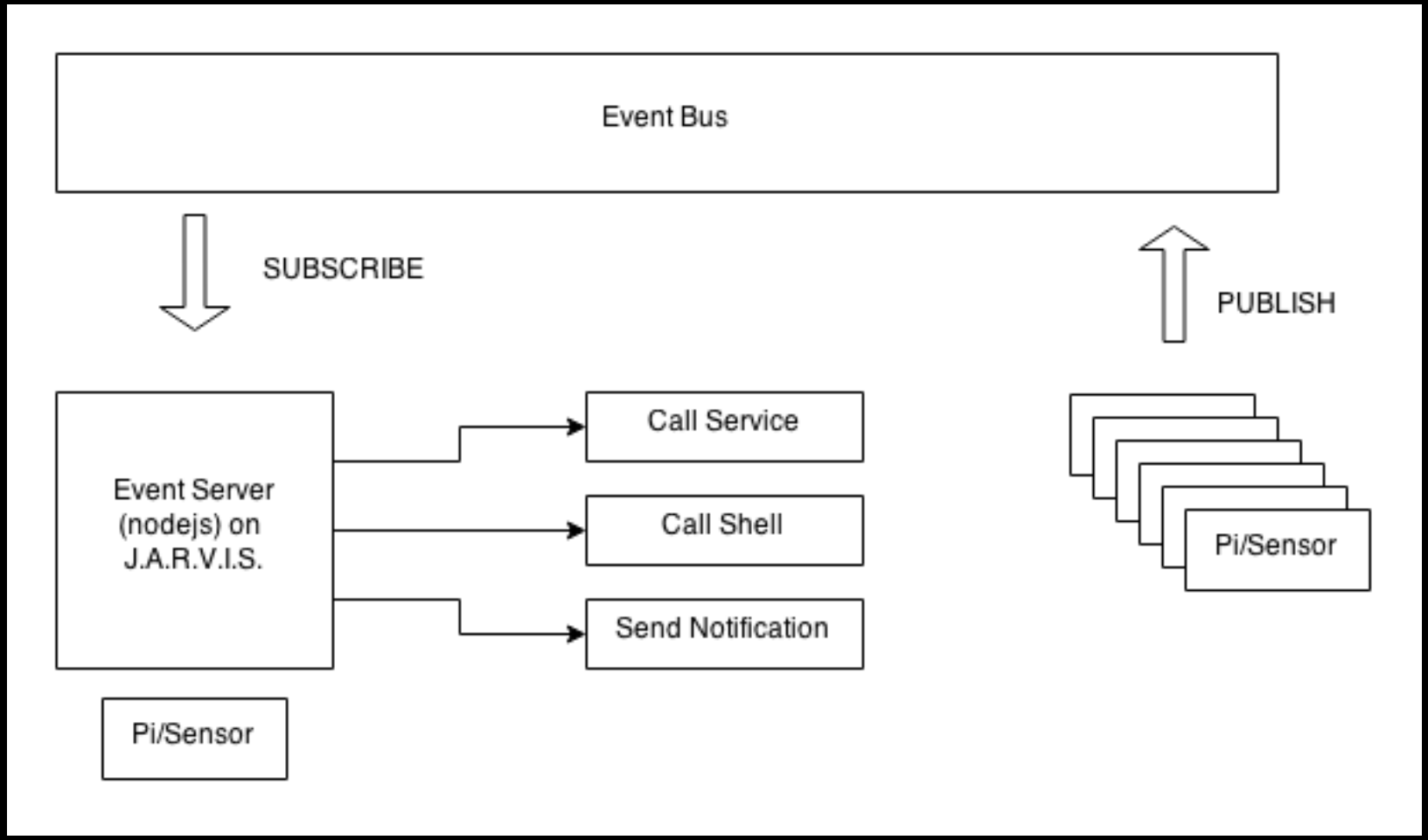
- {
  "name": "test",
  "events": [{
  "name": "broadcast",
  "data": {
  "endpoints": [
  "http://www.google.com",
  "http://www.scottpreston.com"]
  }
  }, {
  "name": "notify",
  "data": {
  "message": "tested google and scottpreston"
  }
  }
  ]}

Dynamically Load Event Rules

- {
  "name": "test",
  "events": [{
  "name": "broadcast",
  "data": {
  "endpoints": [
  "http://www.google.com",
  "http://www.scottpreston.com"]
  }
  }, {
  "name": "notify",
  "data": {
  "message": "tested google and scottpreston"
  }
  }
  ]}



Putting It All Together



Event-server.js

Create Connection

```
function createConnection() {
   var connection = amqp.createConnection({host: 'localhost'});
   connection.on('ready', function () { connection.exchange('house', {
         type: 'fanout',
         autoDelete: false
      }, function (exchange) {
         connection.queue('tmp-' + Math.random(), {exclusive: true},
            function (queue) {
               queue.bind('house', '');
               console.log(' [*] Waiting for house events. To exit press CTRL+C')
               queue.subscribe(function (msg) {
                  var message = msg.data.toString('utf-8');
                  console.log("message received = " + message);
                  var eventDetail = JSON.parse(message);
                  processEvent(eventDetail.name, eventDetail.data);
               });
            })
      });
   });
}
```

Event-server.js

```
function createConnection() {
    var connection = amqp.createConnection({host: 'localhost'});
    connection.on('ready', function () { connection.exchange('house', {
            type: 'fanout',
            autoDelete: false
        }, function (exchange) {
            connection.queue('tmp-' + Math.random(), {exclusive: true},
                function (queue) {
                    queue.bind('house', '');
                    console.log(' [*] Waiting for house events. To exit press CTRL+C')
                    queue.subscribe(function (msg) {
                        var message = msg.data.toString('utf-8');
                        console.log("message received = " + message);
                        var eventDetail = JSON.parse(message);
                        processEvent(eventDetail.name, eventDetail.data);
                    });
                })
        });
    });
}
```

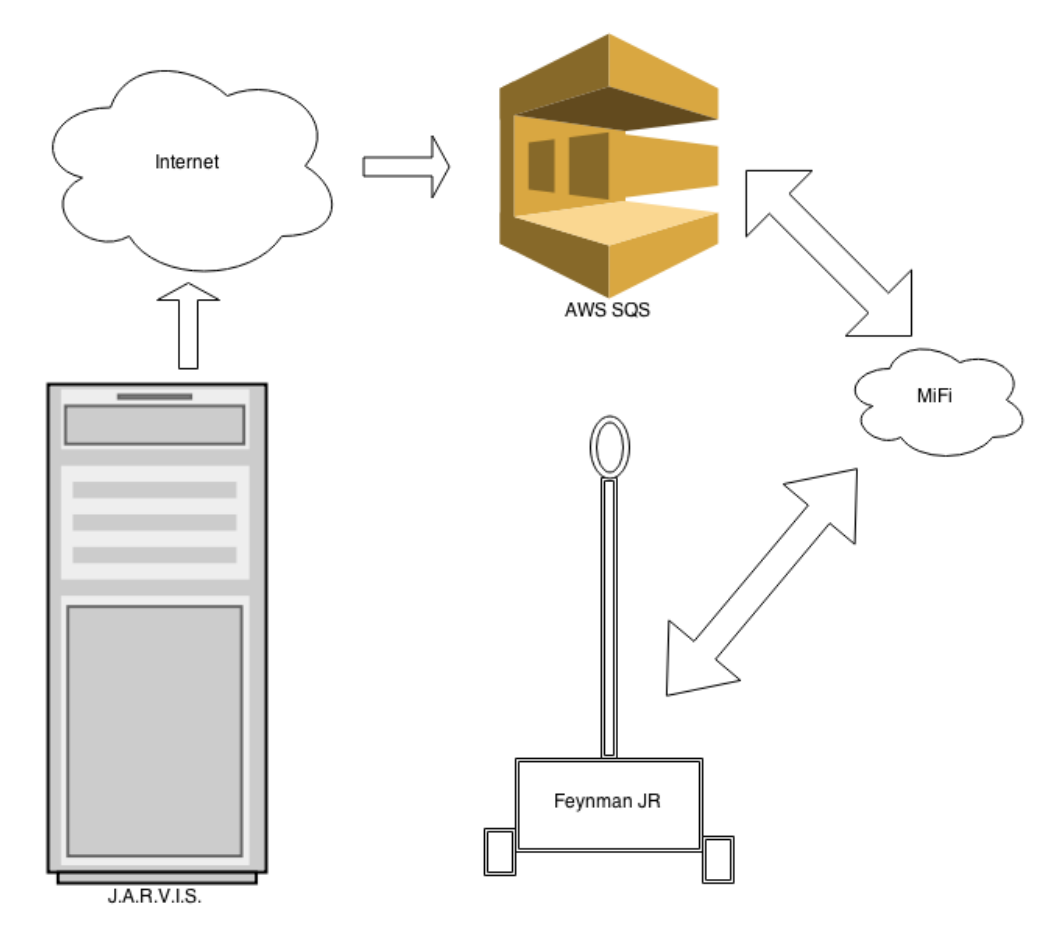
Process Event

Event-server.js - Continued

```
handledEvents.broadcast = function (endpoint) {
    if (endpoint) {
        request(endpoint, function (error,
response, body) {
            …
        });
    }
};
```

Event Handler
(to make http call to service)

Non-Local Events - SQS



J.A.R.V.I.S & Feynman JR.
DEMO

Thank You

@scottpreston

Http://github.com/scottpreston