Open Source Smart Home

Zachary Kauffman’s 490H Project

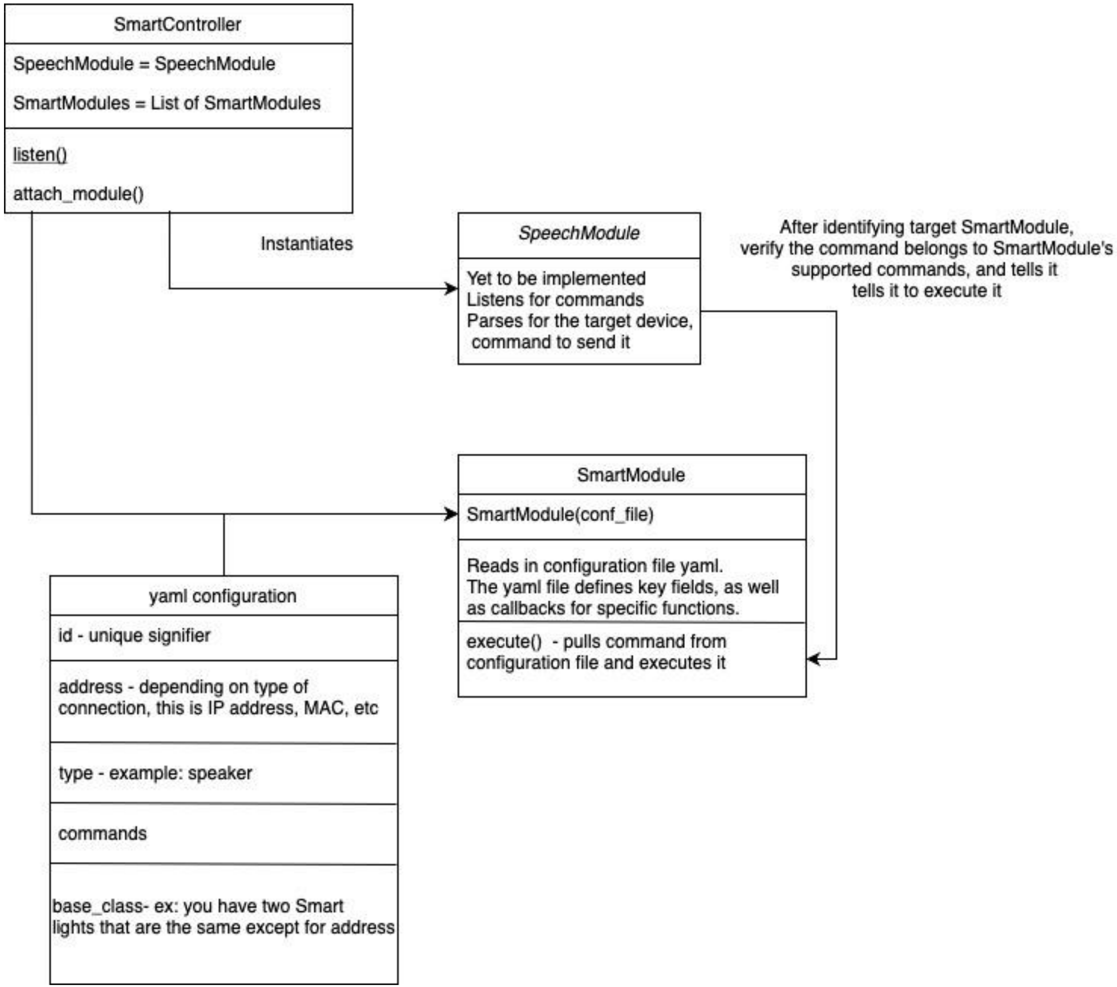
Goal

- Smart Homes are on the rise, from Google Home to Amazon Alexa.
- Using these smart devices hands over insane amounts of data over to big tech companies, regardless of what they may say. See examination by Sogeti (https://labs.sogeti.com/google-home-spying/)
- What if we could replace the spyware offered by a Google Home with an open-source, private alternative without sacrificing the simplicity?

Project

To achieve my project’s stated goal, I am going to utilize a raspberry pi connected to various IoT sensors and smart devices. The raspberry pi is going to be running some AI software (something like MyCroft) with which I am going define common actions corresponding to voice commands. This is going to involve diving deep into mycroft’s development APIs and writing Python code.

The end project will aim to make pairing new devices easy and offer the simple format of an off-the-shelf smart home device. This will involve a specified, easy to modify, configuration structure that will allow users to define their own devices.



SmartController

SpeechModule = SpeechModule

SmartModules = List of SmartModules

listen()

attach_module()

Instantiates

SpeechModule

Yet to be implemented
Listens for commands
Parses for the target device, command to send it

After identifying target SmartModule,
verify the command belongs to SmartModule's
supported commands, and tells it
tells it to execute it

SmartModule

SmartModule(conf_file)

Reads in configuration file yaml.
The yaml file defines key fields, as well as callbacks for specific functions.

execute() - pulls command from configuration file and executes it

yaml configuration

id - unique signifier

address - depending on type of connection, this is IP address, MAC, etc

type - example: speaker

commands

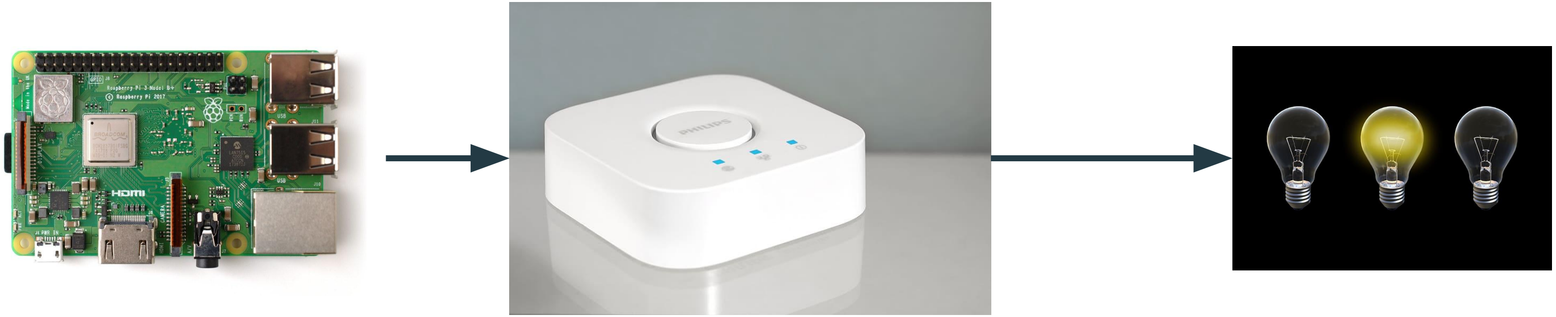
base_class- ex: you have two Smart lights that are the same except for address

Biweekly Progress - Aug 23 - Sept 3

- Basic Control Loop
  - "Listen"
  - Parse each command for a command and a target
    - At its most basic level: know that "turn on bedroom light" means "tell the bedroom light to turn on"
    - Collect valid targets from a list of activated smart modules
- Configuration yaml file structure for smart devices
- Voice Tests
  - Started testing with pre-existing speech recognition code that is NOT open source, this was for preliminary tests of the microphone quality
  - Microphone I already had for the Pi was not sufficient for recognizing commands from somewhere else in the room

Biweekly Progress - Sept 4 - Sept 18

- Enhancements to the yaml file structure for smart components
  - Tested with apartment’s smart light setup.
  - Good initial test of the control loop, nice API to work with
  - Seamlessly replaced phone in the equation with Pi



Biweekly Progress - Sept 4 - Sept 18

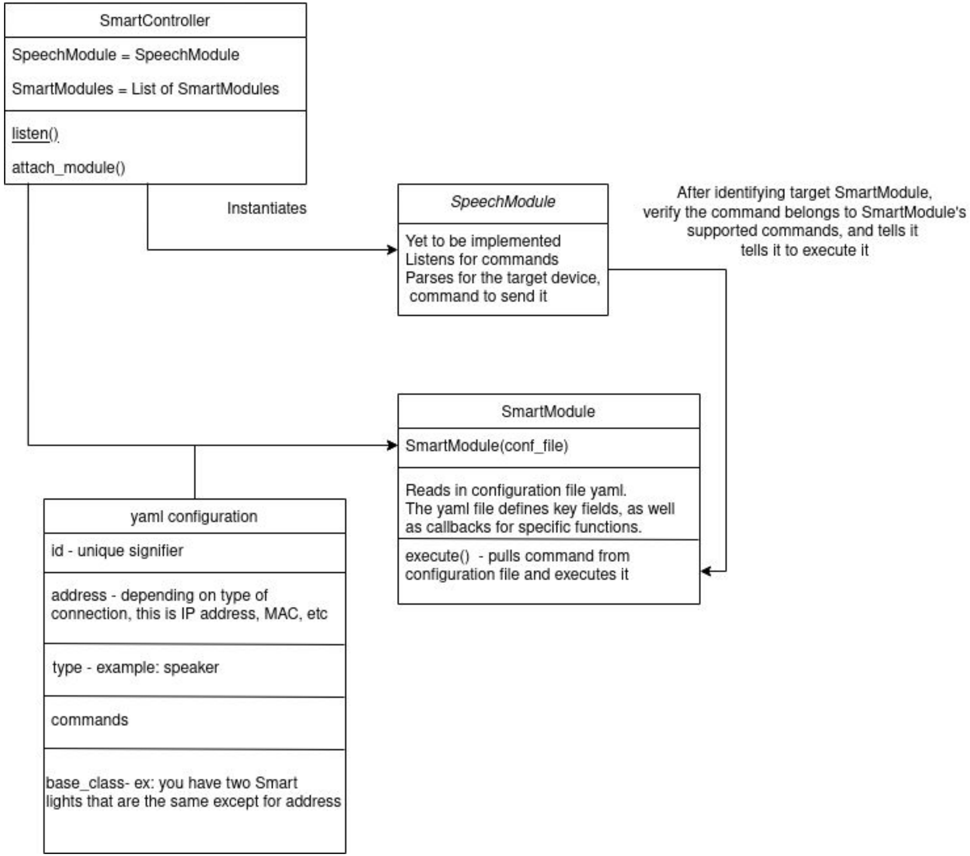
- Enhancements to the yaml file structure for smart components
- Voice tests with new microphone
    - Started testing with Google’s API to get a baseline
        - within 5 feet - “turn on the light” recognized 100% of the time
        - within the same room - “turn on the light” mistakenly reported (25% of the time) as
            - “turn on the flight”
            - “turn the light”
            - basic language processing (RiveScript) should alleviate
    - Researched open source alternatives - CMUSphinx
    - “open source” alternatives that record and

Biweekly Progress - Sept 4 - Sept 18

- Enhancements to the yaml file structure for smart components
  - Tested with apartment’s smart light setup.
- Voice tests with new microphone
  - Started testing with Google’s API to get a baseline
  - Researched open source alternatives - CMUSphinx
- Started getting CMUSphinx setup to work, having some issues with the build process.
- Outlined WiFi sensing component, examined feasibility of solely using Pi+AP.
  - Ordered ESP32s, Access Point will be difficult to get this to work with

Roadblocks

- Building/running CMUSphinx’s demo



SmartController

SpeechModule = SpeechModule

SmartModules = List of SmartModules

listen()

attach_module()

Instantiates

SpeechModule

Yet to be implemented
Listens for commands
Parses for the target device, command to send it

After identifying target SmartModule, verify the command belongs to SmartModule's supported commands, and tells it tells it to execute it

SmartModule

SmartModule(conf_file)

Reads in configuration file yaml.
The yaml file defines key fields, as well as callbacks for specific functions.

execute() - pulls command from configuration file and executes it

yaml configuration

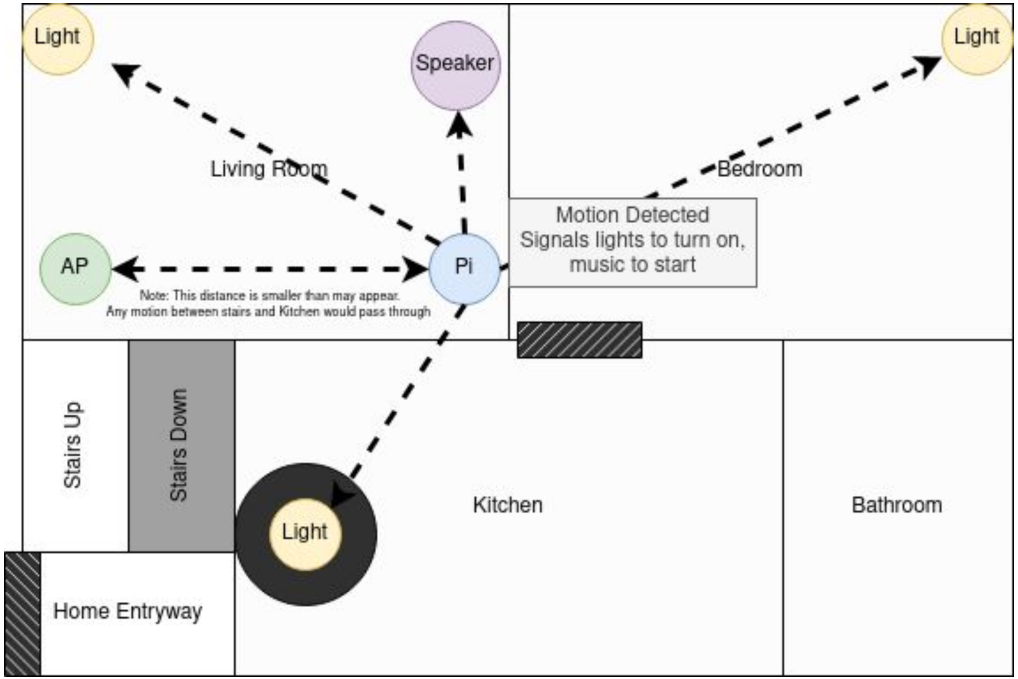
id - unique signifier

address - depending on type of connection, this is IP address, MAC, etc

type - example: speaker

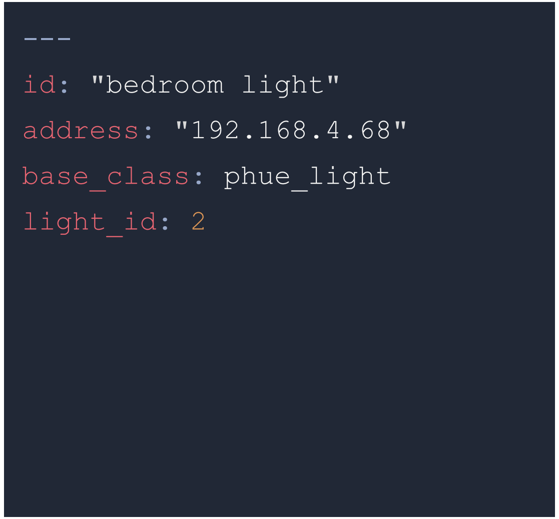
commands

base_class- ex: you have two Smart lights that are the same except for address

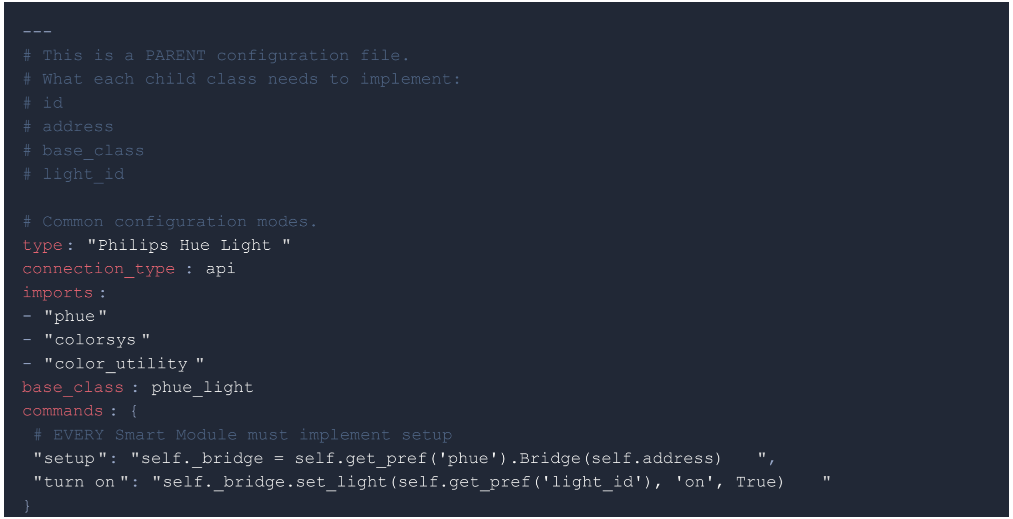

Light
Speaker
Light
Living Room
Bedroom
Motion Detected
Signals lights to turn on, music to start
AP
Pi
Note: This distance is smaller than may appear.
Any motion between stairs and Kitchen would pass through
Stairs Up
Stairs Down
Light
Kitchen
Bathroom
Home Entryway

Proposed test setup for WiFi-based motion detection

.yaml format for configuration



```yaml
---
id: "bedroom light"
address: "192.168.4.68"
base_class: phue_light
light_id: 2
```

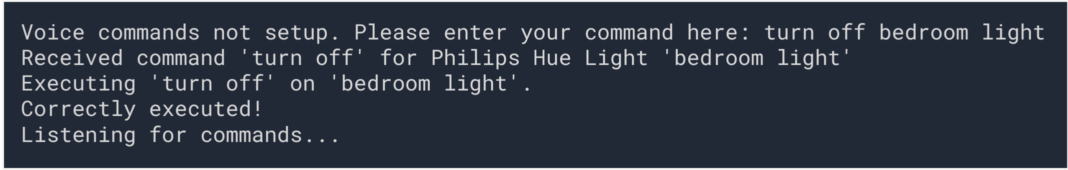


```yaml
---
# This is a PARENT configuration file.
# What each child class needs to implement:
# id
# address
# base_class
# light_id

# Common configuration modes.
type: "Philips Hue Light "
connection_type : api
imports :
- "phue"
- "colorsys "
- "color_utility "
base_class : phue_light
commands : {
 # EVERY Smart Module must implement setup
 "setup": "self._bridge = self.get_pref('phue').Bridge(self.address)    ",
 "turn on ": "self._bridge.set_light(self.get_pref('light_id'), 'on', True)    "
}
```

Goals for next period

- Solidify the open-source speech recognition solution. No guarantee that CMU Sphinx will be good enough once it's built and working...
- Integrate speech recognition into the already existing control loop (ie, replace the command line with a microphone loop)



```
Voice commands not setup. Please enter your command here: turn off bedroom light
Received command 'turn off' for Philips Hue Light 'bedroom light'
Executing 'turn off' on 'bedroom light'.
Correctly executed!
Listening for commands...
```

"Turn off the bedroom light"