Free your mind and your devices will follow, using ESPHome

(An introduction to ESPHome)

Paul Schulz

MawsonLakes.Org

Thursday, April 18, 2024

Welcome

We acknowledge the Bailai (By-ee-lee), Gurang, Gooreng Gooreng and Taribelang Bunda (Tara-bell-ang Bunn-duh) people who are the traditional custodians of this land.

Table of Contents

Introduction

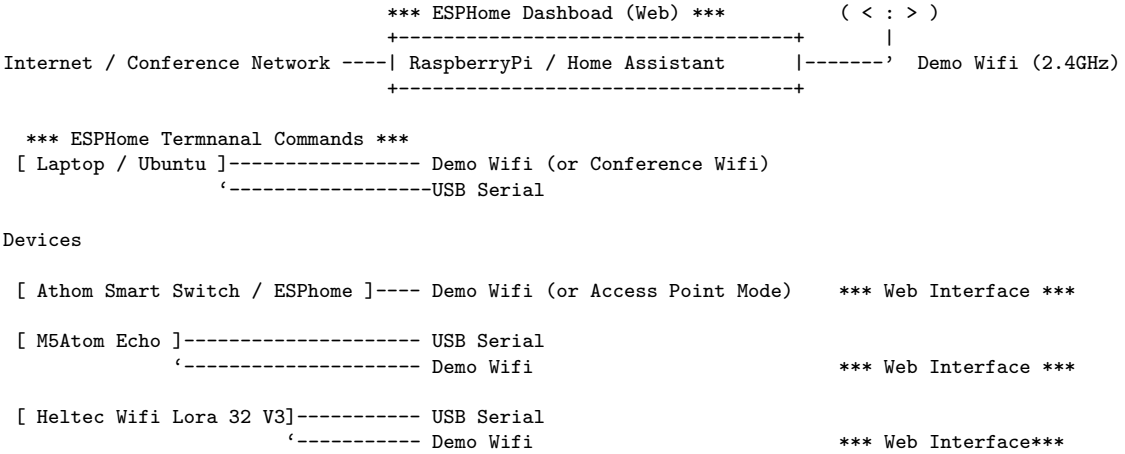
Motivation

- Device Ownership - Louis Rossmann (Strong Language Warning)
  `https://www.youtube.com/watch?v=AddtrV6UFFs&t=34`
  Used with permission.

- There is a need to support freedom loving and open source friendly companies
  - Nabu Casa - `https://www.nabucasa.com/`
  - Athom - `https://www.athom.tech/`
  - M5Stack - `https://m5stack.com/`
  - Heltech Automation - `https://heltec.org/`
  - ... and many others.

Aim: The purpose of this talk is to introduce ESPHome as one way that end users can have control over their devices

Demonstration Setup



Presentation and files are available at:
`https://github.com/PaulSchulz/everything-open-2024.git`

What is ESPHome?

ESPHome is a system for creating and installing firmware (hardware programming) for ESP6288 and ESP32 microprocessor based devices. These microprocessors have been designed for IoT applications. Both chip families have builtin Wifi, the ESP32 also has built-in Bluetooth.

ESP8266
`https://en.wikipedia.org/wiki/ESP8266`
ESP32
`https://en.wikipedia.org/wiki/ESP32`

Other programming systems available in the same space:

- Arduino
- PlatformIO
- Micropython
- Tasmota
- ...

What is ESPHome? (cont.)

Characteristics

- Written in Python
- The device configuration is stored in a YAML formatted file
- Uses 'components' in YAML to describe the required functionality
- Converts YAML into C/C++ code
- Uses either Arduino and/or PlatformIO and the ESP-IDF platform to compile the result into firmware

Benefits and Advantages

- Full control of device is available to the end user
- Ease of maintenance and upgradability.
- Configuration control is possible via standard methods (git).
- Straight forward to customise and modify.

Hardware that uses ESPHome (Examples)

Smart Devies

Athom Smart Switch AU v2

M5Stack Atom Echo Smart Speaker

Development Boards

Heltec Wifi Lora 32 v3

How do I install ESPHome?

Install as Add-On in Home Assistant (SDCard Image running on Raspberry Pi)

- `https://esphome.io/guides/getting_started_hassio.html`

Install manually (python application, using pip)

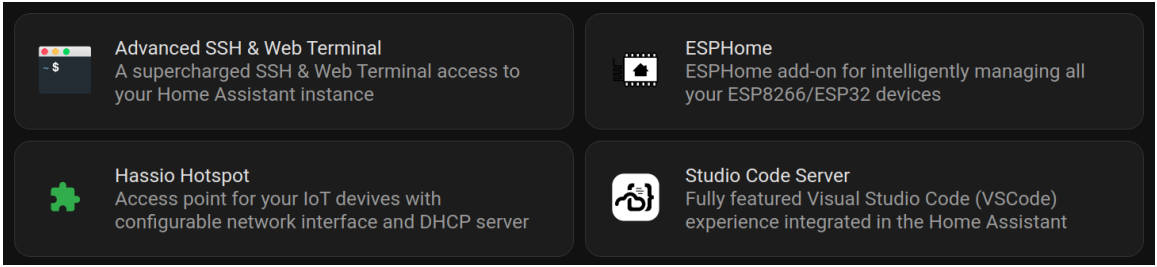
- See: `https://esphome.io/guides/installing_esphome`
- Windows, Mac, Linux

Install manually for development

- Use the manual installation instructions but checkout the code directly from the github ESPHome repository.
- See `https://github.com/esphome/esphome`

Install in Home Assistant

Home Assistant Add-ons



Also useful:

- ZeroTier (VPN)

HA link: `http://energy-monitor-e11230.local:8123/hassio/dashboard`

Install and run from Linux Command Line

Download and install from GitHub

```
git clone https://github.com/esphome/esphome.git
cd esphome
./script/setup
```

To setup python environment:

```
source venv/bin/activate
```

Create a working directory for your files:

```
cd ..
mkdir esphome-eo2024
cd esphome-eo2024
```

Run the ESPHome dashboard:

```
esphome dashboard .
```

Demonstrations

Using ESPHome on Home Assistant

Links:
`http://energy-monitor-e11230.local:8123`
`http://energy-monitor-e11230.local:8123/5c53de3b_esphome/ingress`

Getting Started

From the command line:

```
esphome wizard demo.yaml
```

Step 1 name: demo
Step 2 esp: ESP32
Step 3 board: m5stack-atom
Step 4 wifi:

File Layout

- demo.yaml
- secrets.yaml

Installation

```
esphome run m5stack-atom-echo.yaml
```

What does the YAML configuration look like?

Website and documentation: `https://esphome.io`

```yaml
esphome:
  ...

web_server:
  local: true

# Enable logging
logger:
  hardware_uart: UART0
  logs:
    sensor: DEBUG
    binary_sensor: DEBUG
    wifi: DEBUG
```

```yaml
binary_sensor:
  - platform: gpio
    pin: GPIO39
    name: Button

light:
  - platform: esp32_rmt_led_strip
    id: led
    name: None
    pin: GPIO27
    chipset: SK6812
    num_leds: 1
    rgb_order: grb
    rmt_channel: 0
```

For Developers

Individual Devices

- Encapsulation
- Improved Error Checking
- Code Re-use
- Easy Distribution
- Easy Maintenance and Upgrades

Other things you can do / Advanced Topics

- Creating persistent data
- Creating custom components
- Porting an existing Arduino sketch to ESPHome.
- Integrating with Home Assistant
- Customising the “web_server” component

Porting an existing Arduino Sketch

digitalBlinkWithoutDelay.ino ⟶

```
...
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis
      >= interval) {
    previousMillis = currentMillis;
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }
    digitalWrite(ledPin, ledState);
  }
}
```

my_blink/my_blink.h

```
...
#include  "esphome.h"
static cont char *const TAG = "my_blink";


namespace esphome {
  namespace my_blink {

    class MyBlink : public Component {
      public:
        ...
        void setup() override { ... };
        void loop() override { ... };

    }
  }
}
```

Porting and existing Arduino Sketch (cont.)

my_blink/my_blink.h

```
...
#include  "esphome.h"
static cont char *const TAG = "my_blink";

namespace esphome {
  namespace my_blink {

    class MyBlink : public Component {
      public:
        ...
        void setup() override { ... };
        void loop() override { ... };

    }
  }
}
```

my_blink/__init__.py

```
import esphome.config_validation as cv
import esphome.codegen as cg
from esphome.const import CONF_ID

my_blink_ns =
  cg.esphome_ns.namespace("my_blink")
MyBlink =
  my_blink_ns.class_("MyBlink", cg.Component)

CONFIG_SCHEMA = cv.Schema(
    cv.GenerateID(): cv.declare_id(MyBlink),
).extend(cv.COMPONENT_SCHEMA)

def to_code(config):
    var = cg.new_Pvariable(config[CONF_ID])
    yield cg.register_component(var,config)
```

Porting and existing Arduino Sketch (cont.)

Adding external libraries
In YAML

```
esphome:
  ...
  libraries:
    - EEPROM
```

or in __init__.py (need to confirm)

```
def to_code(config):
  ...
  cg.add_library("EEPROM", None)
```

Directory structure

```
<CONFIG_DIR>
  node1.yaml
  node2.yaml
  my_components/
    my_blink/
      __init__.py
      my_blink.cpp
      my_blink.h
```

YAML File

```
# use all components from a local folder
e xternal_components:
  - source:
      type: local
      path: my_components
```

Resources

Documentation and Support

- Website: `https://www.esphome.io`
- Discord Server: ESPHome #general-support (#devs)

Getting Started

- Browser Based: `https://web.esphome.io`
- Github: `https://github.com/esphome/esphome.git`
- Home Assistant Add-on
  `https://esphome.io/guides/getting_started_hassio.html`

Thankyous, Conference Surprise, Any Questions?

Thanks goes to:

- Everything Open 2024
- ESPHome Developers - @jesserockz
- Home Assistant & Nabu Casa
- IoT Experimenters Group, Port Adelaide-Enfield Library - Robert Hart
- Open Energy - Ewan Parson

and

Special Conference Offer from Athom

If you wish to purchase a Athom Smartplug AU ESPHome, for the next week and for the first 100 orders, get 10% off wth the code - **EO2024**
Product Link: `https://www.athom.tech/blank-1/esphome-au-plug`

— Questions? —

About the Speaker

Paul Schulz is a long time Linux and Open Source Developer, and has worked as a Linux System Administrator for Software Engineering Teams, in a variety of industries. He is happiest when he can submit a patch to fix a bug, particularly if it has been bothering him for a while.

Experience Paul has presented several times at Linux.Conf.Au on a variety of topics, and supports and contributes to several free and open source projects.

Contact Paul Schulz - `mailto:paul@mawsonlakes.org`