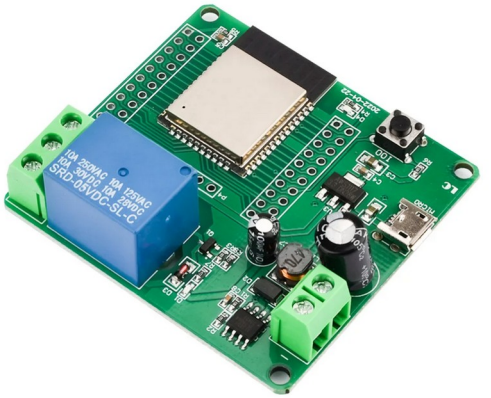

Tasmota + Berry

Not just for home automation

Darryl Bond

Everything Open 2024

ESP32 SOC

- ESP8266 released in 2014 by Expressif
- The ESP32 series of microcontroller are a substantial evolution on the ESP8266
- Faster, Dual Core CPU with more memory and many more GPIO
- CPU architecture using the Xtensa LX6 ISA like the ESP8266.
- Some models use the RISC-V ISA
- Addition of Bluetooth on most models
- Many more ADC in addition to DAC and hardware counters

ESP Family

	ESP8266	ESP32	ESP32-S2	ESP32-S3	ESP32-C3	ESP32-C6
CPU	Xtensa32Bit	Xtensa32Bit	Xtensa 32 Bit	Xtensa 32bit +	RISC-V 32bit	RISC-V 32bit +
Cores	1	2	1	2	1	1
Clock	80-160MHz	160-240MHz	160-240MHz	160-240MHz	120MHz	160MHz
RAM	160kB	520KB	320KB	512KB	300KB	512KB
Flash	External SPI Flash					
Wifi 802.11 b/g/n	✔	✔	✔	✔	✔	+ax +Zigbee
Bluetooth	✘	Classic & BLE	✘	BLE	BLE	Bluetooth5 + LE
GPIO	18	34	34	45	22	22
DAC	✘	2	2	2	✘	✘
ADC	1 10bit	18 12bit	20	20	2	7
Interfaces	SPI-I2C-I2S	✔ +CAN	✔ +CAN	✔ +CAN	✔	✔

ESP32 Software Support

- Expressif IDF is the official Development Framework
- Arduino support early for ESP8266
- Arduino support continued with ESP32
- PlatformIO with VSCode
- Supports most Arduino libraries

Tasmota

- Tasmota was first called Sonoff-MQTT-OTA, first commit in 2016 by Theo Arends.
- Tasmota grew into a fully fledged ecosystem for virtually any ESP8266/ESP32 based device.
- Tasmota was not designed for Home Assistant, although it eventually grew to interface to it. It is not really comparable with ESPHome.
- A Tasmota device can be stand alone

ESP8266 Tasmota Design

- Peripheral Device drivers and configuration to assign GPIO
- Wifi with Over the Air upgrade capability.
- MQTT support
- Comprehensive command set
- Rule Engine
- Capacity limits of ESP8266 meant that there were many different images to support the required drivers or functions

Commands and Rules

- Command

  POWER1 ON

- websend [192.23.1.20] power1 0

- Rule

  ON POWER1#STATE=1 DO RULETIMER1 600 ENDON

  ON RULES#TIMER=1 DO POWER1 0 ENDON

- There are other triggers depending on the peripherals configured

  ON ANALOG#A0div10 DO DIMMER %value% ENDON

Road to ESP32

- A scripting engine that worked with Tasmota was needed:
  - General purpose programs
  - Ability to write device drivers without recompiling
- Review of options
  - Micropython - took over the machine
  - LUA - heavyweight and obscure syntax
  - https://github.com/cesanta/elk – Javascript Syntax
  - Berry – Python Syntax

Berry

- Lightweight: A well-optimized interpreter using very few resources.
- Fast: optimized one-pass bytecode compiler and register-based virtual machine.
- Powerful: supports imperative programming, object-oriented programming, functional programming.
- RAM saving: RAM usage starts at ~10kb
- Additional libraries provided, with deep hooks into Tasmota

Berry Code

- Tasmota incorporates a Web based UI to Berry: Berry Scripting console

```
 print("Hello World")
Hello World
 import string
 print(string.format("Hello Everything Open 20%d",24))
Hello Everything Open 2024
```

Program Execution

- Berry programs in the file-system can be executed:
  - by the load() function from the Berry console
  - Run at startup if named ‘autoexec.be’ which can then load() other files.
  - From the Tasmota console using the ‘br’ command
  - From a tasmota rule

```
ON Time#Minute=0 DO br load(“midnight.be”) ENDON
ON DS18B20#Temperature<10
    DO WEBSEND[192.168.10.3] br load(“TooCold.be”)
```

Support Libraries

- Berry has a number of modules that can be imported as required:
  - String
  - JSON
  - MATH
  - MQTT
  - Tasmota
  - GPIO

Tasmota module

- Send a Tasmota command from Berry

```
tasmota.cmd(“dimmer 60”)
```

- Create a Tasmota command in Berry

```
def hello_from_berry()
    print(“Hello from Berry”)
end
tasmota.add_cmd(“hello”,hello_from_berry)
```

- Timers

```
tasmota.set_timer(600,hello_from_berry)
tasmota.add_cron("*/10 * * * * *",hello_from_berry)
```

Tasmota module

- add_rule()
- read_sensors()
- get_power() / set_power() / get_switches()
- url_get()
- rtc()/time_str()
- Web interface functions
- ...

Tasmota Module (Driver)

- Berry can be used to add device drivers

```
class MyDriver
  def every_second()
    # do something
  end
end
d1 = MyDriver()
tasmota.add_driver(d1)
```

- Override driver class methods as required
- Add fast_loop() to Tasmota main loop
  - fast_loop function called 200 times/sec

- Driver with fast_loop

```
class my_driver
def every_100ms()
   # called every 100ms via normal way
  end

  def fast_loop()
    # called at each iteration, and needs to be registered separately and explicitly
  end

  def init()
    # register fast_loop method
    tasmota.add_fast_loop(/-> self.fast_loop())

  end
end

tasmota.add_driver(my_driver())                # register driver
```

Other Modules

- MQTT
  - Subscribe and publish MQTT messages
- GPIO
  - Perform operations on GPIOs
  - Read and write hardware counters
- WIRE
  - I2C bus operations
- PERSIST
  - Manage persistent data

LVGL Graphics

- Inexpensive ESP32 hardware with touch graphics displays
- Uses LVGL Tasmota firmware image.
- Berry calls into LVGL library

Matter

- Matter protocol implemented in Berry
- UI implemented in Berry
- ESP8266 does not support Berry and Matter
- ESP32 can be configured as a Matter bridge to ESP8266 Tasmota device

Home Automation?

Everything Open 2024

Links

- Tasmota Documentation https://tasmota.github.io/docs
- Tasmota Berry Doc https://tasmota.github.io/docs/Berry/
- Berry Introduction (in 20 minutes of less) https://tasmota.github.io/docs/Berry-Introduction/
- The Berry Script Language Reference Manual https://berry.readthedocs.io/en/latest/