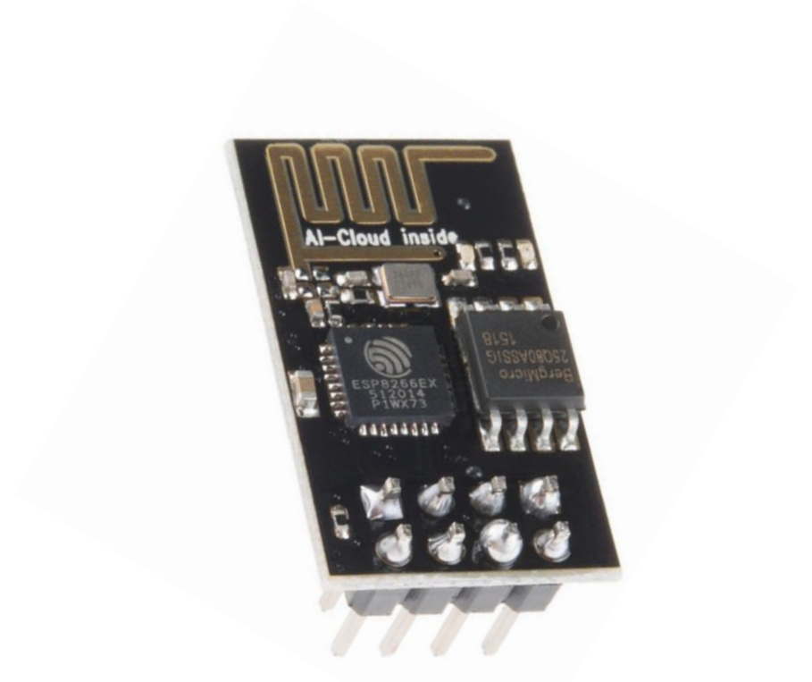
Lilota - Tasmota: Enabling Migration from Tasmota to Lilota for ESP32 and ESP8266 devices

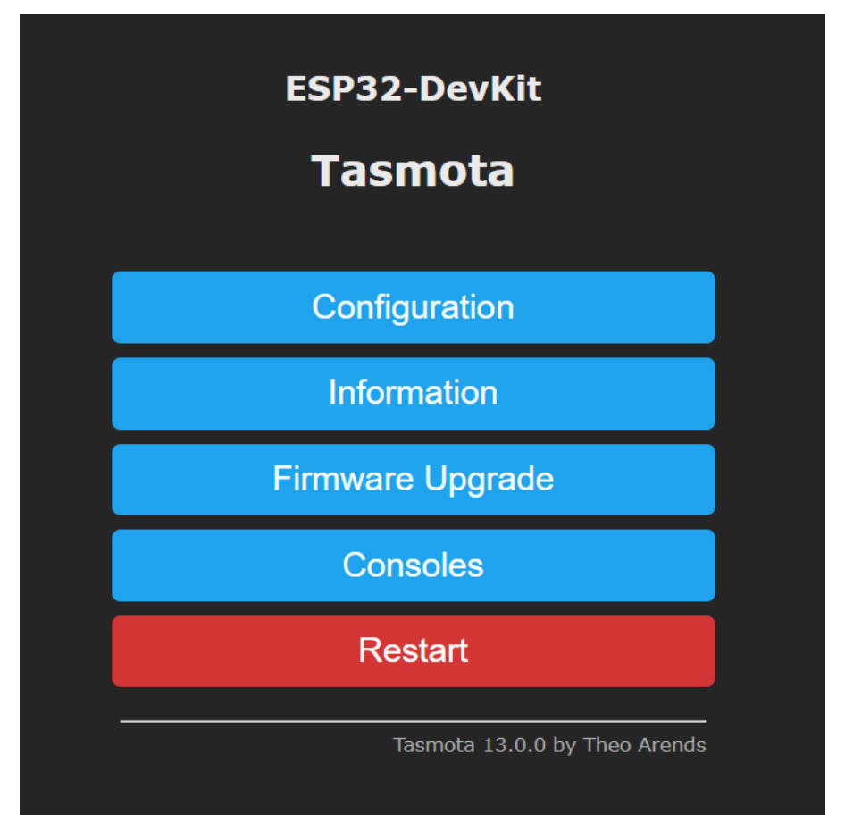
Ethan Do (Mentor: Mike Ferdman)
Newport High School, Department of Computer Science, Stony Brook University

Background

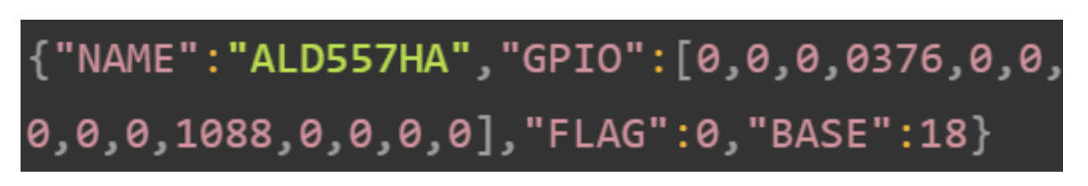
- ESP32 and ESP8266 are low-cost chips made for Internet of Things (IoT) projects. They come with GPIOs and can support other inputs/outputs such as ADC, DAC, PWM, and more.
- The ESP32 is the successor to the ESP8266, so it has some upgraded features, such as an extra CPU core, more GPIOs, and Bluetooth.
- Tasmota is a public firmware that can control ESP32 and ESP8266 devices. It allows control via MQTT, the Tasmota Web UI, HTTP, or a serial monitor.
- Tasmota Templates is a feature of Tasmota that allows users to create, modify, and import various configurations for supported devices.
- Lilota (Little Interpreted Language Over The Air) is meant to be an alternative firmware for ESP32 and ESP8266 devices that solves the problems of Tasmota and other firmware. It uses a C library called LIL that provides a compact scripting language.

ESP32 chip


ESP8266 chip (on a development board)



Tasmota Web UI for ESP32



```
{"NAME":"ALD557HA","GPIO":[0,0,0,0376,0,0,0,0,0,1088,0,0,0,0],"FLAG":0,"BASE":18}
```

Example Tasmota Template

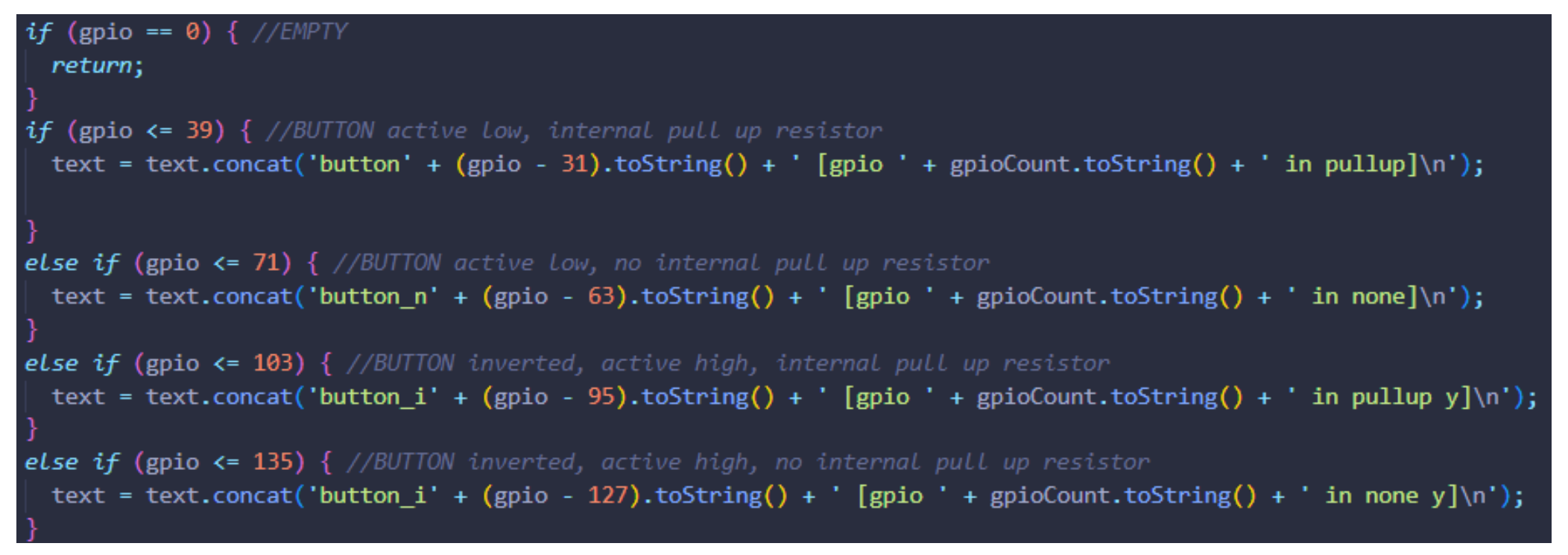
Objectives

- My primary objective is to allow any users to easily migrate from using Tasmota to Lilota.
- This includes the tasks of:
  - Taking in a Tasmota Template and "translating" it for use in Lilota
  - Implementing GPIOs and other I/O (ADC, DAC, PWM, Counters)
  - Creating LIL commands for the user.

Methods

Tasmota Template Translation

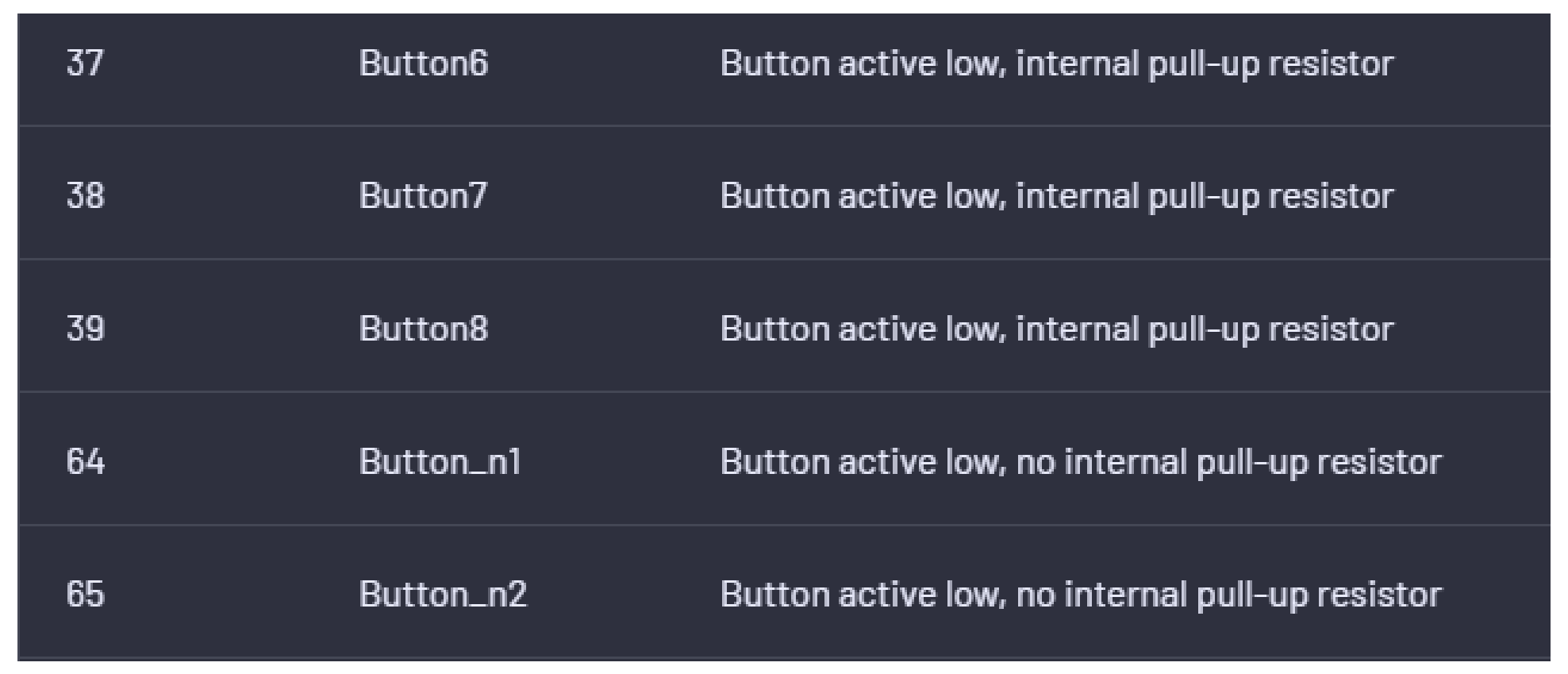
- Tasmota Templates are stored as a JSON string with four name/value pairs. The second pair, "GPIO", represents what input or output device is connected to each GPIO pin. Each input/output device is represented as a number, which is recorded in the Tasmota documentation.
- A typescript program is used to translate Tasmota Templates. The program reads each number and outputs a LIL command that "creates" an object for each GPIO.



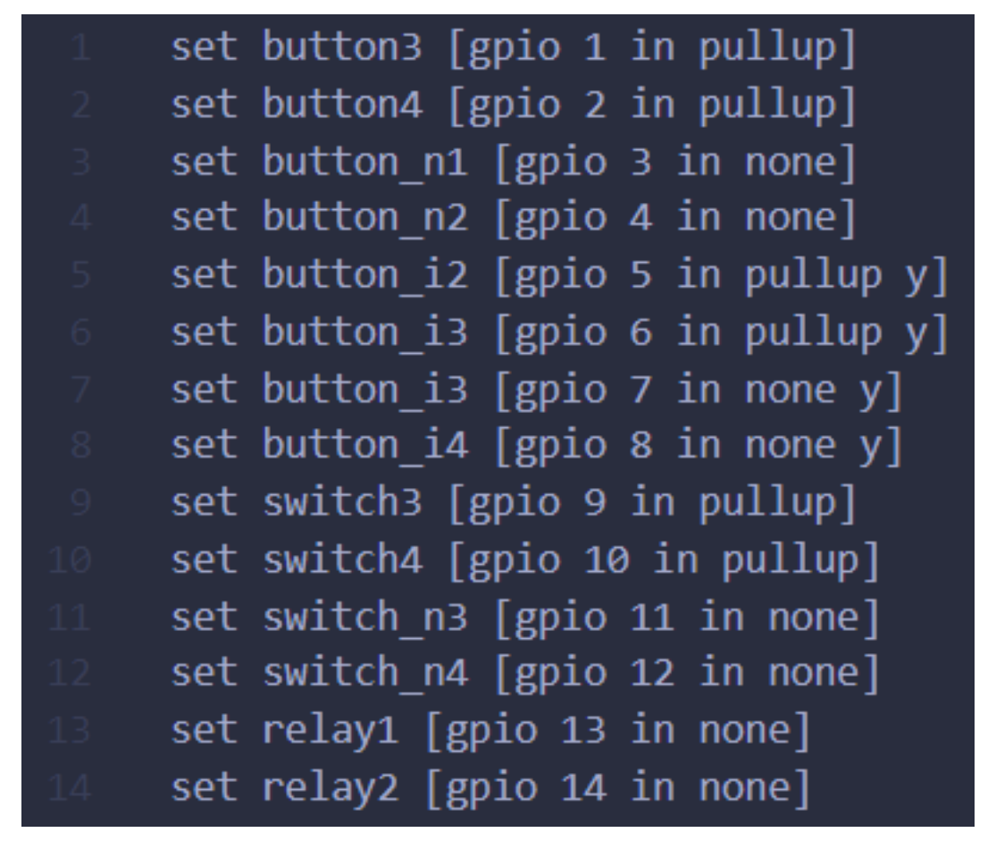
```
if (gpio == 0) { //EMPTY
  return;
}
if (gpio <= 39) { //BUTTON active low, internal pull up resistor
  text = text.concat('button' + (gpio - 31).toString() + ' [gpio ' + gpioCount.toString() + ' in pullup]\n');

}
else if (gpio <= 71) { //BUTTON active low, no internal pull up resistor
  text = text.concat('button_n' + (gpio - 63).toString() + ' [gpio ' + gpioCount.toString() + ' in none]\n');
}
else if (gpio <= 103) { //BUTTON inverted, active high, internal pull up resistor
  text = text.concat('button_i' + (gpio - 95).toString() + ' [gpio ' + gpioCount.toString() + ' in pullup y]\n');
}
else if (gpio <= 135) { //BUTTON inverted, active high, no internal pull up resistor
  text = text.concat('button_i' + (gpio - 127).toString() + ' [gpio ' + gpioCount.toString() + ' in none y]\n');
}
```

{"NAME":"Example Template","GPIO":[34, 35, 64, 65, 97, 98, 130, 131, 162, 163, 194, 195, 224, 225, 257, 258, 290, 291, 320, 321], "FLAG":0, "BASE":45}



37	Button6	Button active low, internal pull-up resistor
38	Button7	Button active low, internal pull-up resistor
39	Button8	Button active low, internal pull-up resistor
64	Button_n1	Button active low, no internal pull-up resistor
65	Button_n2	Button active low, no internal pull-up resistor



```
set button3 [gpio 1 in pullup]
set button4 [gpio 2 in pullup]
set button_n1 [gpio 3 in none]
set button_n2 [gpio 4 in none]
set button_i2 [gpio 5 in pullup y]
set button_i3 [gpio 6 in pullup y]
set button_i3 [gpio 7 in none y]
set button_i4 [gpio 8 in none y]
set switch3 [gpio 9 in pullup]
set switch4 [gpio 10 in pullup]
set switch_n3 [gpio 11 in none]
set switch_n4 [gpio 12 in none]
set relay1 [gpio 13 in none]
set relay2 [gpio 14 in none]
```

GPIOs

- Tasmota has four general types of GPIOs: buttons, switches, relays, and LEDs. Each can be inverted, and buttons and switches can have a pullup, pulldown, or no resistor. Both input and output is only represented in 1's and 0's. For more precise input and output, ADC/DAC or PWM is used.

ADC/DAC

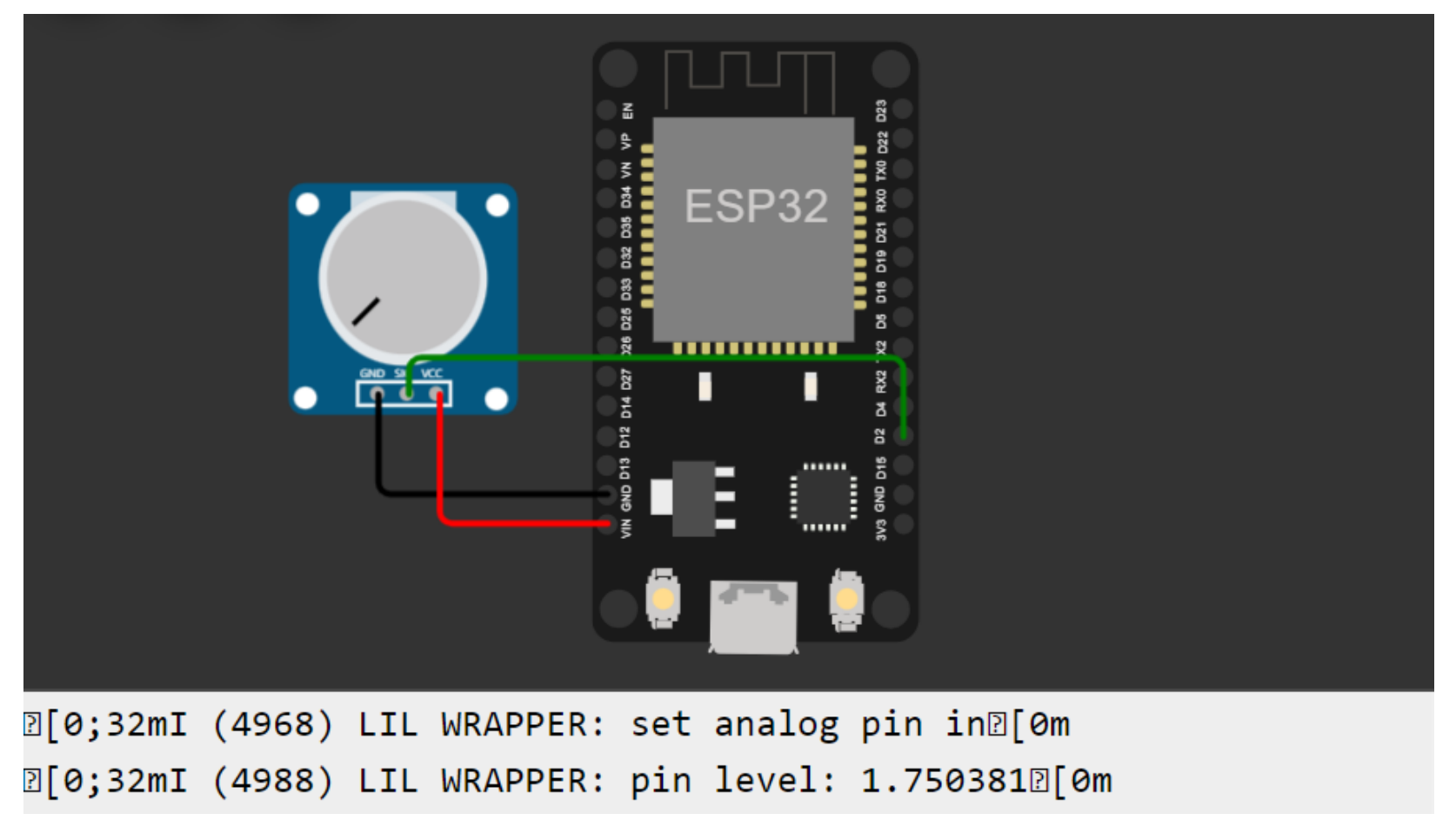
- ADC (Analog to Digital Convertor) and DAC (Digital to Analog Convertor) are devices that can take in and output specific voltages.

PWM

- PWM is a method of using digital signals. PWM alternates between "on" and "off" in a device to keep the power at a certain level. It controls the duty cycle (how often the signal is "on") and the frequency that the cycle happens at.

Functionalities

- Tasmota Template Translation
- LIL commands:
  - GPIO
    - gpio [port] [in\out] [pullup\pulldown\none] [inverted] (optional)
    - gpio get
    - gpio on
    - gpio off
    - gpio change
  - ADC/DAC
    - analog [channel] [adc/dac]
    - analog get
    - analog out
    - analog change
  - PWM
    - pwm [gpio pin] [speed mode] [sig frequency] [duty resolution] [initial duty](optional)
    - pwm set [duty]
    - pwm update
    - pwm get



```
[0;32mI (4968) LIL WRAPPER: set analog pin in[0m
[0;32mI (4988) LIL WRAPPER: pin level: 1.750381[0m
```

Setting and reading from a potentiometer on an online simulation of the ESP32

Conclusion

- Tasmota users will now have the option to migrate to using Lilota. However, there are still a couple of things that need to be completed for this transition process to be complete.
- Next Steps:
  - Adding more specific components to the Template Translator
  - Implementation of more I/O devices and more LIL commands for these I/O.