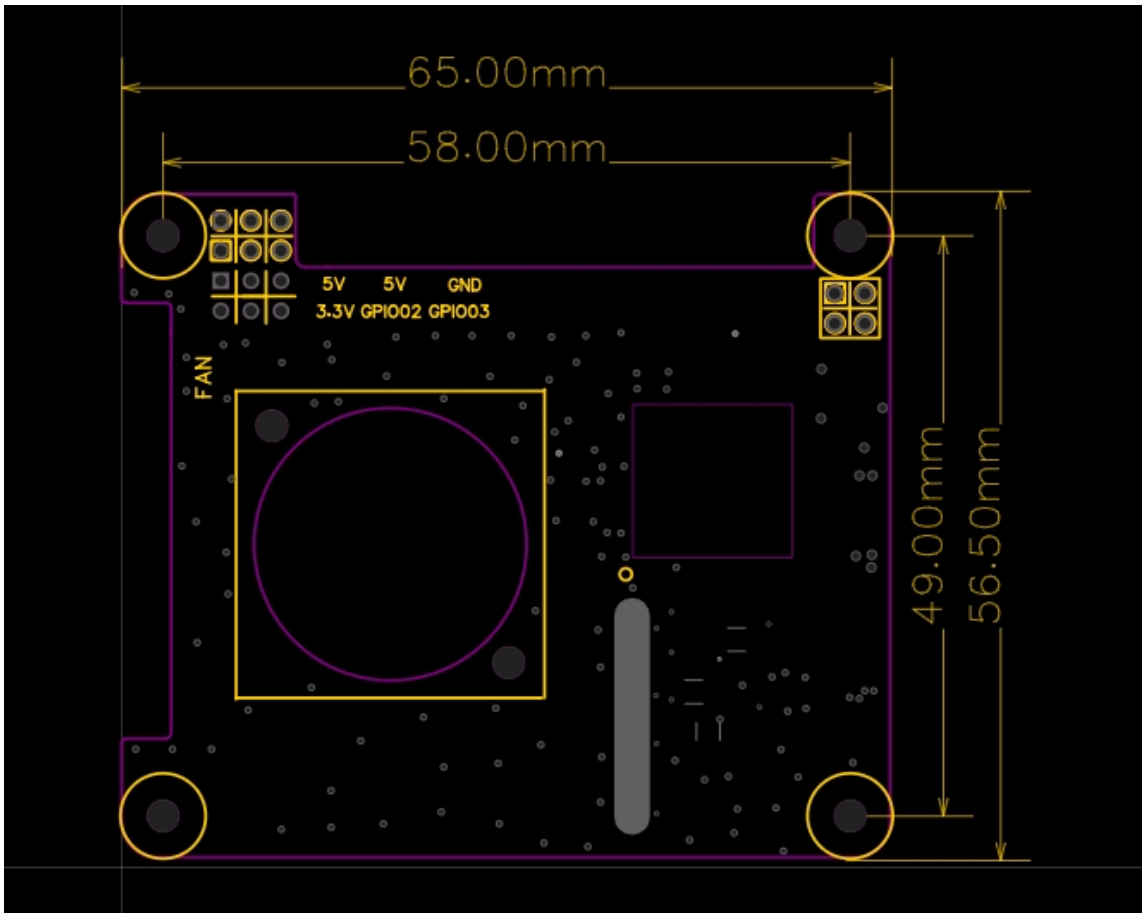
Raspberry Pi PoE HAT

DESCRIPTION

5V, 2.5A PD(Powered Device) Integrated Module (Isolation Type)

FEATURES

- Fully supports IEEE802.3af
- Input Voltage Range 37V to 57V
- Support PoE applications in both of Fast / Gigabit Ethernet environments.
- Short Circuit Protection
- Over-temperature Protection
- Programmable Classification (Default:Class 0)
- High Efficiency
- Isolation level 1.5KVrms.
- Easy Installation and Low Cost (Isolation Type, Minimum External Devices required)
- Low Output Ripple and Noise
- 1500Vrms Isolation (Input-Output)



Model	Input Voltage (after bridge)	Input Current	Output Voltage	Output Current	Efficiency	Ripple Output Noise
	V	mA	V	A	%	mV
Raspberry Pi PoE HAT	50.56	298	5.136	2.5	85.22%	140
	50.6	236	5.136	2	86.02%	110
	50.64	177	5.136	1.5	85.95%	95
	50.66	121	5.136	1	83.79%	75
	50.72	64	5.136	0.5	79.11%	55
	50.9	8	5.136	0	0.00%	

No.	Parameter	Symbol	Min	Typical values	Max	Unit
1	Input Voltage	Vin	37	48	57	V
2	output voltage	+VDC	5	5.1	5.3	V
3	output Current	PWR	0.05	2	2.5	A
4	Isolation Voltage	VISO		1.5		KV
5	Ripple Output Noise	VRN		150		mVp-p
6	Storage Temperature	Tj	-40	25	85	℃
7	Operating Temperature	Tstg	-40	25	60	℃

风扇 FAN 接口，由 PIN03 ,GPIO02 控制，树莓派默认高电平，通电风扇会转动。
可以设置高温启动，低温关闭，参考代码如下：

```
import RPi.GPIO as GPIO
import time

fan_on_temp = 50.0
fan_off_temp = 45.0
fan_ctrl_pin = 26
ctrl_interval = 3
fan_is_running = False
```

```
def get_cpu_temp():
    temp_file = open("/sys/class/thermal/thermal_zone0/temp")
    cpu_temp = temp_file.read()
    temp_file.close()
    return float(cpu_temp)/1000


GPIO.setmode(GPIO.BCM)
GPIO.setup(fan_ctrl_pin,GPIO.OUT)
GPIO.output(fan_ctrl_pin,GPIO.LOW)
#pwm = GPIO.PWM(fan_ctrl_pin,10000)


while 1:
    time.sleep(ctrl_interval)
    cur_temp = get_cpu_temp()
    print("cur temperature: %.2f" %cur_temp)
    if(cur_temp < fan_off_temp):
        if fan_is_running == True:
            print("fan off")
            fan_is_running = False
#            pwm.stop()
            GPIO.output(fan_ctrl_pin,GPIO.LOW)
    elif(cur_temp > fan_on_temp):
        if fan_is_running == False:
            print("fan on")
            fan_is_running = True
#            pwm.start(100)
            GPIO.output(fan_ctrl_pin,GPIO.HIGH)

    if(fan_is_running == True):
        fan_speed_duty = int((cur_temp -
fan_off_temp)/(fan_on_temp-fan_off_temp)*100)
        if fan_speed_duty > 100:
            fan_speed_duty = 100
        print("set fan speed(0-100): %d" %fan_speed_duty)
#        pwm.start(fan_speed_duty)

pwm.stop
GPIO.cleanup()
```