Roger Clark

Freelance IT Consultant & Developer, Melbourne Australia

TTGO T5 eInk display – update

10 OCT 2018

by Roger Clark | posted in: Uncategorized | 2

In the last few days I've been attempting to resolve a major design fault on the TTGo T5 eInk display board, where the battery management chip (IP5306) was completely turning off the power to the ESP32, and the other chips, when I put the ESP32 into deep sleep mode.

The problem is a design fault, and is primarily because the battery management chip (IP5306) is designed for "power bank" devices, where a LiPo cell is used to power a 5V USB device, rather than powering a microcontoller or any other 3.3V devices.

The IP5306 only has a datasheet written in Chinese, but I noticed that it had an option to attach a button to pin 5, and grounding this pin, via a resistor, caused the chip to wake up and put 5V on its pin 8.
Also momentarily pressing the button, while the chip was awake, cause it to stay awake even when there ESP32 was in deep sleep and and consuming hardly any current.

So my first idea was to make the the ESP32 "press the button" by connecting one of its GPIO pins to the button pin (5) via a 1k resistor, then make the ESP32 wake up from deep sleep to metaphorically press the button, every 30 seconds, before immediately returning to deep sleep.

Search

Top Posts & Pages

- DMR MMDVM duplex hotspot
- Investigating a RCWL 9196 / RCWL-0516 "Radar" motion detector module
- STM32F103 and Maple / Maple Mini with Arduino 1.5.x IDE
- Updating firmware on USBASP bought from eBay
- GD-77 Firmware V3.2.1 released with frequency range limiting for FCC compliance
- Radioddity GD-77 firmware 3.1.8 released
- Radioddity GD-77 Community CPS for firmware 3.1x
- Arduino STM32 - USB Serial and DFU
- Arduino on the nRF51822 Bluetooth Low Energy microcontroller
- List of Radioddity GD-77 bugs

Categories

Select Category

This was never going to be a perfect solution, as the IP5306 (battery manager) seemed to need a button press pulse of at least 100 milliseconds, hence the ESP32 would need to be consuming its normal current for 0.1/30 of the time. But this would only be 0.3% of the total time, so I thought if this method worked, it would be adequate.

Unfortunately, although initially this strategy appeared to work OK, I found that if I left the board running from a battery overnight; when I checked in the morning, the IP5306 had turned off.

I think that the IP5306 probably has a maximum operation time, or perhaps a maximum count of the number of button presses which it allows before turning of when there is virtually no load.

But whatever the reason, it became clear that the IP5306 is totally unsuitable for the application to which its been put, and I took the radical decision to remove the chip from the PCB completely and use an external battery charger / protection module instead.

I have a hot air reflow tool, but because I didn’t want to damage any components near the IP5306, I decided to cut its legs, to remove it. However I found that even after cutting all 8 visible pins the chip seemed to be stuck to the board.

What I didn’t realise was that its a 9 pin package with a large Ground terminal on the base of the chip, which probably also acts as a heat sink. Luckily they had not used that much solder paste under the chip and I it came away without too much force and the board was not damaged.

Once the chip had been removed, I was able to solder a wire to the pad for pin 1, which receives 5V when the board is connected to USB (or I presumed powered via the 5v pin on the header), and a wire to pin 8, which was the output (5V which I presume feeds into the EA3036 multi output buck converter), and a third wire to the ground pad in the middle of the IP5306 footprint.

I already have a stock of TP4055 based battery charger modules with built in battery protection. So I soldered the module to these 3 wires, and connected the same 2000mAH LiPo cell to the new battery charger module.

Once everything was connected, I was surprised to find that the battery charger module was only outputting 1V, even though the battery was almost fully charged and measured as 4.15V, and initially I thought perhaps the charger module was faulty; however I found that if I supplied power via USB, not only did the whole TTGO board start working, but when I removed the USB power that it continued to be powered via the LiPo battery.

So it appears that these battery charger / undervoltage protection modules need to be initially primed with 5V on their input before the undervoltage protection is removed.

Running from battery power, of around 4.1V, the TTGO 5T board seemed to work fine, and checking the spec of the EA3036, in theory it will operate with an input voltage as low as 2.7V, which is less than the minimum operating voltage on the LiPo cell, so I think the TTGO board will work fine from battery, however I will leave it running a Wifi server (drawing around 100mA), until the battery voltage gets close to its minimum value, to confirm my supposition.

However before leaving the battery drain program running, I also tested the current being drawn from the battery when the ESP32 was put into deep sleep mode.

Unsurprisingly I found that the TTGO board as a whole was taking around 7mA, but I was not surprised because the red “power” LED was still illuminated and I thought perhaps that accounted for the majority of the current.

But I was wrong…

I removed the LED and found virtually no difference in the current being drawn from the battery. So the next step was to check whether the current was actually being drawn by the TTGO board or by the new battery charger / protection board.

So after a small amount of rewiring and my multi-meter connected between the new battery charger module and the TTGO board, I measured virtually the same current.

By now I wasn't particularly surprised by this, because the NS4148 class D audio amplifier chip has a typical standby current of 3.5mA, and could potentially be taking more, plus I am not sure how much current will be taken by the EA3036 buck converter even when it has virtually no load on one of its outputs.

So the next set of tests I need to perform are to around the NS4138, and I'll probably have to find the pullup resistor for its "CTRL" pin, which needs to be pulled low in order for the NS4148 to be put into its low power standby mode.

However I think if putting the NS4148 into standby does not radically reduce the power that is consumed, I will not have much scope to improve things, as the circuit may also include multiple wasteful pullup/ pulldown combinations and these could be hard to track down, and the EA3036 would need to be replaced with something which had very low quiescent current properties, and I'm not quite sure what devices would fit the bill.

BTW. It looks like disabling the outputs on the EA3036 that drive the NS4148 is not practical as the "enable" pins are connected together and only have a short length of track on the PCB before they go to a via, and it will be hard to track down where the via goes.

Also using a 2000mA battery, a 5mA load would in theory last around 16 days, and this is probably adequate for any use I can think of for the whole module.

Anyway, I'll post another followup article when I have had chance to do a discharge test on the battery and also experimented with the CTRL pin on the audio amplifier.

Update. 11th Oct 2018

I've managed to get the current drain down to 1.5mA but I don't think its practical to get it any power.

I removed the pullup resistor on the NS4148 audio amplifier CTRL (Pin 1) and soldered a 10k resistor to ground instead.

I then connected the IO19 header pin to Pin1 on the NS4148.

This keeps the NS4148 in low power mode, until I drive IO19 high. i.e before I can play any audio, I have to do this, and afterwards set IO19 as high impedance (aka input)

I've had a look at the EA3036 multi output buck converter, to see if I could use its ENable pins to shut down part of the chip that was not driving the ESP32, but this would be hard to do, because all the enables seem to be directly connected together and share the same (10k) pullup resistor. So it would take some extremely delicate surgery on the board to cut the track(s) right next to the chip and control the outputs separately.

And I'm not entirely sure whether that would actually help reduce the current, as 2/3 of the EA3036 would still be running

So without a schematic for this board, its going to be very difficult to know what else could be taking any current.

Consequentially, I think I'll need to live with 1.5mA when the ESP32 is in deep sleep.

PS. I'm still doing battery voltage testing, but since I have a 2000mAH battery attached, its taking ages to flatten it just using the current drain from the ESP32 with its Wifi turned on.

So I'll either need to leave it on for a few days, or find a way to drain the battery a lot faster.

Previous Post Next Post

2 Responses

ken 11/10/2018 |

wow impressive

Roger Clark 11/10/2018 |

Hi Ken

Since I posted this last night, I have left the board running, drawing around 100mA with the Wifi on, and it was still working this morning.

I've also modified the board again, by removing the 10k pullup on pin 1 of the audio amplifier and replacing with a non SMD 10k resistor to GND, and also connected this pin to GPIO19 on the header.

This has reduced the current drawn, to 1.5mA in deep sleep mode, as the audio amplifier is now also in low power standby mode unless its enabled

Unfortunately I don't think its going to be possible to improve this, and get the current drain levels down to the sub 100 uA values that the ESP32 is capable of

Leave a Reply

Enter your comment here...

Archives

Select Month

Categories

Select Category

Recent Posts