AUTODESK
Instructables

PY32Duino Anemometer A.k.a. Making a Wind Speed Meter From a Disposable Vape.

By petercd in CircuitsArduino
Unpublished

Introduction: PY32Duino Anemometer A.k.a. Making a Wind Speed Meter From a Disposable Vape.

I pulled apart a bunch of disposable vapes to see if there were any useful electronics inside, they had been tossed into a recycling bin and were free for the taking.

Two of the 15 had a tiny QFN32 ic (quad flat no leads, 32 pins) that upon further research revealed that it was a PY32F030EK28 made by PUYASemi, a tiny micro controller in a 5mm² package that had twice the performance of an Arduino Uno.

About 3 years ago, I'd made an anemometer using a Mega2560 and 128*64 oled which was a bit overkill seeing as I was only using 1 of the 55 inputs.

Thus the idea was born to reverse engineer the pcb and see if I could get the display to show speed, after all it had pretty much everything needed, battery, usb-c onboard charging, display, housing and mcu.

Be aware that these PY32 mcu's out of a vape might not be suitable for anything more complex than what Im doing as I ran into an issue that suggests that the vape manufacturer (Nasty Juice Co based in Malaysia) might have ordered a custom port map from the supplier, PUYA SEMI.

More about that in the coding section.

Supplies

In order to complete this project, you're going to need either of the 2 vapes in the first pic, Nasty Bar 20000 or BLVK Bar 20000.

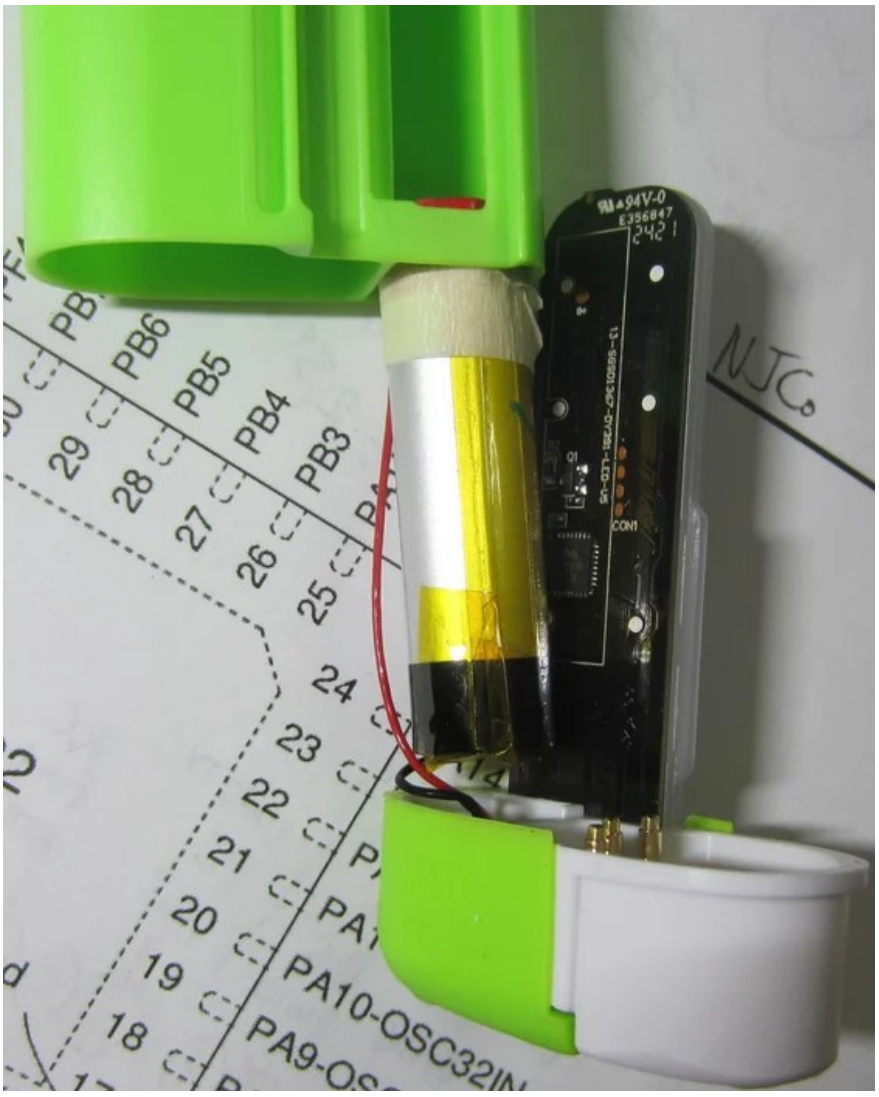
1. I chose the Nasty Bar, so all my findings are based on that.
2. Stl's available from https://www.thingiverse.com/thing:2559929/files
3. On the software side, Arduino IDE, I used V2.3.4, the rest covered in the coding section.
4. Soldering iron and possibly a DMM to verify the pinouts are the same as my discovery.

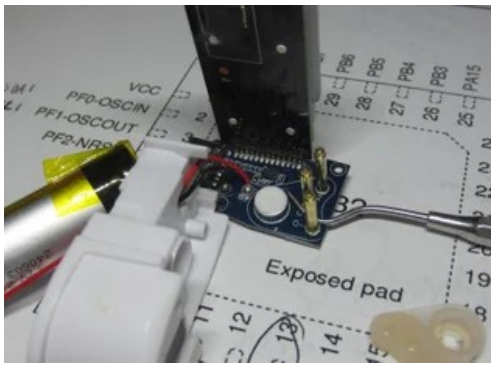
The BLVK version has a similar pcb, but the front led layout is very different. It is the better pcb to work with as the via's are large enough to slide a thin copper wire through to act as an attachment for wiring.

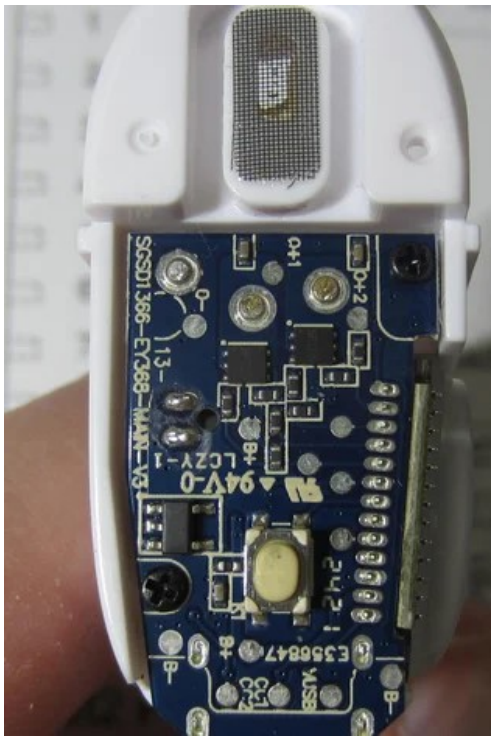
I've junked quite a few of the Nasty pcbs due to lifted pads and tracks etc.

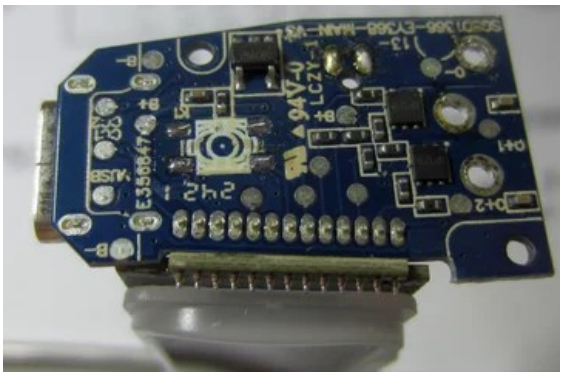
The easiest part was printing out the anemometer cups and mounting them up on the roof.

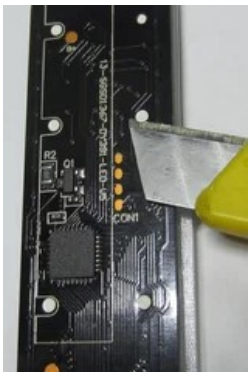
Step 1: Hammer Time.











Once the 2 halves are pried apart, the first and most important thing to do is to desolder the battery. Even when "flat" at 2.5v, it has enough power to turn a wire into a glowing orange filament.

Naturally you pull it out of the plastic housing before disconnecting the battery.

I take off the red wire on the pos terminal before poking around on the pcb.

Theres 2 small screws to be removed on the bottom usb pcb and that bit of plastic can be recycled.

I usually pull out the juice sponge and toss it into the bin along the the silicone end caps.

Plastic parts, into recycle bin.

I removed the clicky switch when the pcb didnt want to lie flat in my wooden box. A small cavity was cut out for the SOT23-5 which connects to the vac switch and switches the mosfets on.

The tabs holding the screen on have to be cut so we can access test pads for BOOT0 and a UART Rx connection.

Step 2: Making Connections Prior to Coding

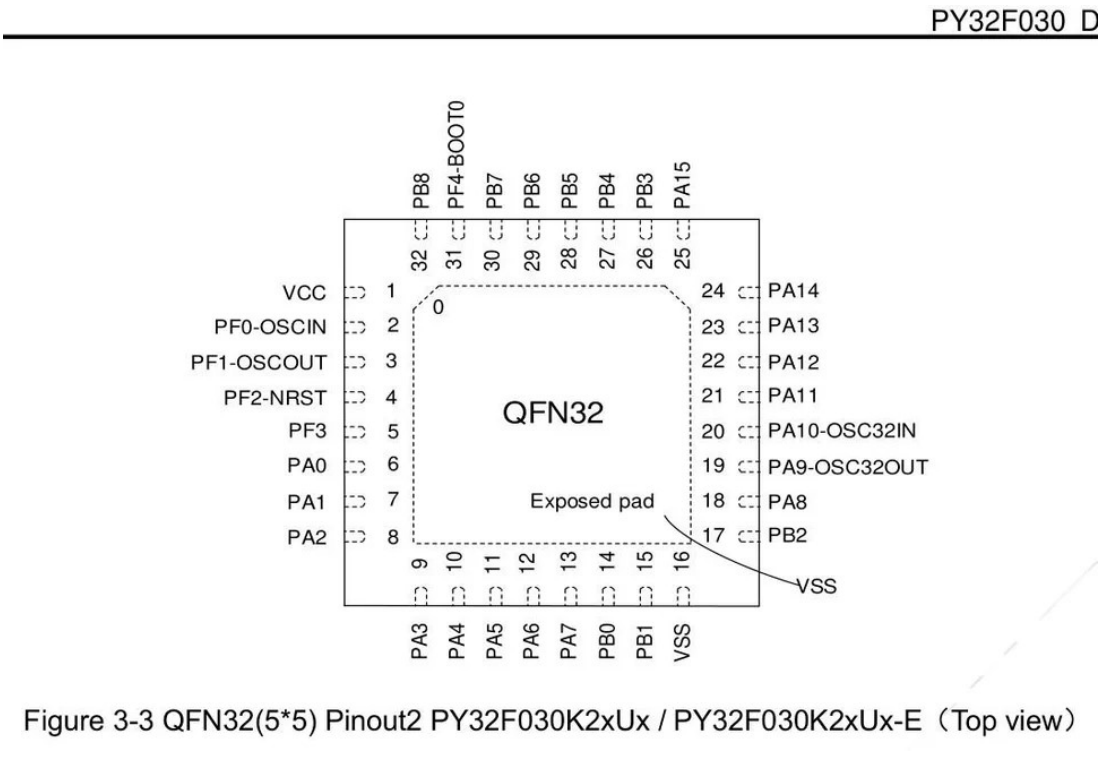
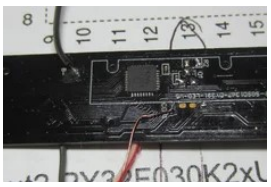
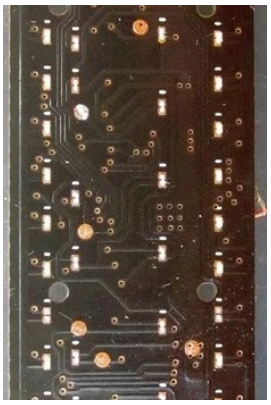
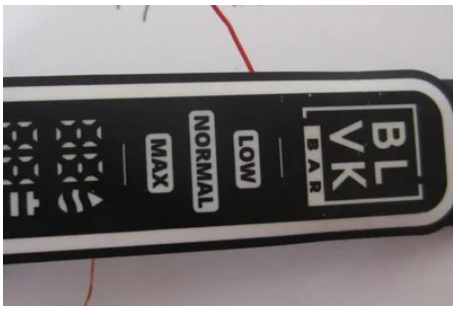
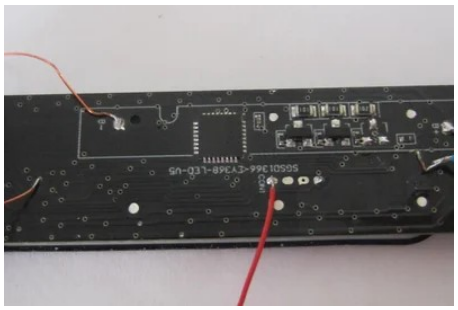


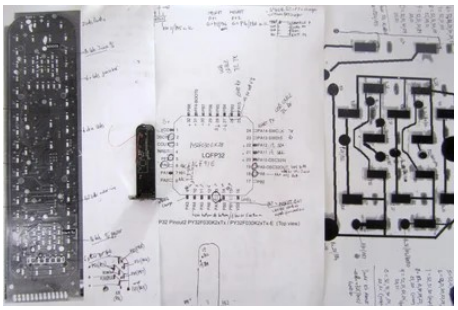
Figure 3-3 QFN32(5*5) Pinout2 PY32F030K2xUx / PY32F030K2xUx-E（Top view）











In order to upload to the mcu, you need to pull BOOT0 high, I used a slide switch from the +V rail to the BOOT0 (board pin 31) on the leds side of the pcb.

Then the TX PA2(pin 8) connects to the RX of your TTL cable.

I used a FT232 style cable, I had a lot of problems with Prolific blocking my old PL2303 TTL cable. A CH341A also worked.

Next is the RX uart PA3 (pin9) which connects to the TX of the TTL cable.

Initially I used the UART pins on PA14 & PA15, but many times the mcu wouldnt respond, so I used PA2 & PA3 as some of the PUYA Semi docs seemed to indicate that as a default in their example code.

So with the +, -, tx and rx connected, as well as BOOT0 pulled high, you can start uploading code to the mcu.

I traced out pertinent pins on the small pcb with the usb connector so I might as well include it here.

USB-C pcb MCU (board pins in brackets)

1 - SWCLK PA14 (PIN 24) CLK FOR SWD, serial wire debug

2 - ~~PB1 (pin 15) not used~~

3 - ~~PB0 (pin14) not used~~

4 - PA7 (pin 13) input from outside pulses

5 - ~~PA6 (pin 12) not used~~

6 - ~~PA5 (pin 11) not used~~

7 - ~~PA4 (pin 10) not used~~

8 - PA3 UART RX (pin9) goes to TX TTL cable

9 - ~~PA1 (pin 7) not used~~

10 - PA2 UART TX (pin 8) goes to RX TTL cable

11 - VSS (pin 16) batt negative, GND

12 - VCC (pin 1) batt positive

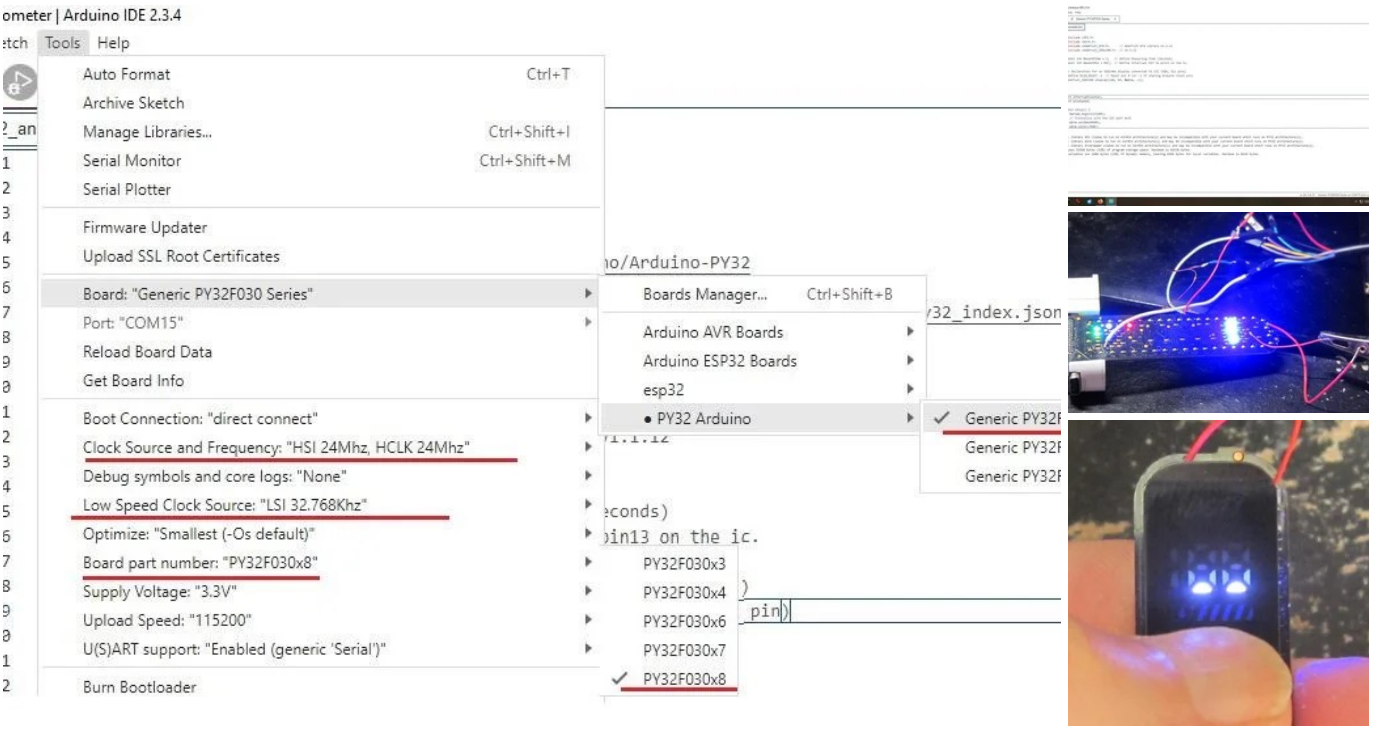
13 - SWDIO PA13 (pin 23) SWD data line.

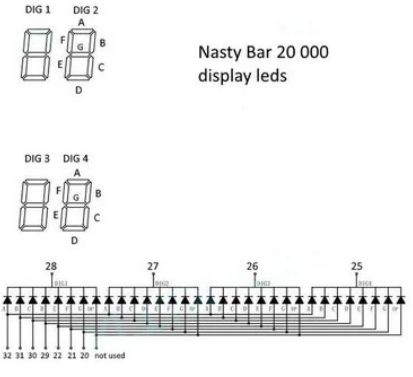
Take note of pin 1 on the silkscreen, its on the side farthest from the usb-C socket.

I traced the SWD lines to the usb-C connector and in theory they could be used with a modified USB C cable and a JLink debugger to flash code to the chip.

So thats why I left them connected in the final build.

Step 3: Coding





Port	AF0	AF1	AF2	AF3
PA0	SPI2_SCK	USART1_CTS	-	LED_DATA_B
	AF8	AF9	AF10	AF11
	-	USART2_TX	SPI1_MISO	-
PA1	AF0	AF1	AF2	AF3
	SPI1_SCK	USART1_RTS	-	LED_DATA_C
	AF8	AF9	AF10	AF11
	-	USART2_RX	SPI1_MOSI	-
PA2	AF0	AF1	AF2	AF3
	SPI1_MOSI	USART1_TX	-	LED_DATA_D
	AF8	AF9	AF10	AF11
	-	-	SPI1_SCK	-
PA3	AF0	AF1	AF2	AF3
	SPI2_MISO	USART1_RX	-	LED_DATA_E
	AF8	AF9	AF10	AF11
	-	-	SPI1_MOSI	-
PA4	AF0	AF1	AF2	AF3
	SPI1_NSS	USART1_CK	SPI2_MOSI	LED_DATA_F
	AF8	AF9	AF10	AF11
	-	USART2_TX	-	-
PA5	AF0	AF1	AF2	AF3
	SPI1_SCK	-	-	LED_DATA_G
	AF8	AF9	AF10	AF11
	-	USART2_RX	-	-

F multiplexing function mapping

AF0	AF1	AF2	AF3	AF4	AF5
-	-	TIM14_CH1	SPI2_SCK	USART2_RX	-
AF8	AF9	AF10	AF11	AF12	AF13
USART1_RX	USART2_TX	-	-	I²C_SDA	-
AF0	AF1	AF2	AF3	AF4	AF5
-	-	-	SPI2_MISO	USART2_TX	-
AF8	AF9	AF10	AF11	AF12	AF13
USART1_TX	USART2_RX	SPI1_NSS	-	I²C_SCL	TIM14_CH1
AF0	AF1	AF2	AF3	AF4	AF5
-	-	-	SPI2_MOSI	USART2_RX	-
AF0	AF1	AF2	AF3	AF4	AF5
USART1_TX	-	-	SPI2_MISO	USART2_TX	-
AF8	AF9	AF10	AF11	AF12	AF13
-	-	SPI1_NSS	-	-	TIM3_CH3
AF0	AF1	AF2	AF3	AF4	AF5
-	-	-	-	-	-

The Arduino core for the PUYA mcu was made possible by HalfSweet on Github. https://github.com/py32duino/Arduino-PY32

Before you begin, you need to add an additional board in the boards manager url in "preferences".

https://github.com/PY32Duino/Arduino-pack-json-ci/releases/download/Nightly/package_py32_index.json

Once that is done, make sure you have the relevant items underlined in the first pic. If you have the F030x3 selected instead of the F030x8 its going to fail to compile due to lack of memory.

At compile time it seems to bring up an incompatible warning which can be safely ignored.

What led me to think that Nasty ordered a custom port map, is that I couldnt get any of the generic py32 examples to work.

So I tried the generic sevseg for AVR on my Mega2560 and it worked fine.

Same sketch on the py32 always lit the wrong segments.

Going through the data and reference sheets showed correct digit tube control pins, but only the A segment was as PUYA advised. The rest were split up between port A, port B and port F.

In the official PUYA example the leds are multiplexed with an "alternate function 3" in the code for the segments and AF6 for the com0 to com3 digit tubes.

Checking the port map for port F aka BOOT0 or pin 31 shows no AF3, so even if you declare it in code, nothing happens.

Yet...Nasty is using pin31 aka BOOT0 for segment B of all 4 digit tubes.

This was copied from official docs.

> 2. The COM0/COM1/COM2/COM3 of the digital tube are connected to the PA15/ PB3/PB4/PB5 of the stk board respectively
>
> 3. The SEG A B C D E F G DP of the digital tube are connected to the PB8/PA00/ PA01/PA02/PA03/PA04/PA05/PA06
>
> of the stk board respectively
>
> 4. Please confirm the selected chip model during use and whether the above IO ports are all led out

It was at this point that I tried to desolder a 5mm² ic in the hope that I'd flip it over and bridge the proper pins to the leds on the main pcb.

It didnt work so well as can be seen by the missing pins on the ic in the pic at the start.

This leads me to think that Nasty ordered a custom port map from the factory, well that and the fact that the flash memory wasnt read protected.

So... more surprises may lurk.

Next attempt to get things working was to use the i2c oled, struggled in the beginning as I wasnt seeing any output on the serial monitor, changed to UART PA2/PA3 and it worked fine.

Then it was a matter of tinkering in my old sketch to declare the proper i2c pins which became the win.

Remember to change the constant in line 67 of the ino sketch, mine is 2.396 for the radius of 106mm on my cup rotor.

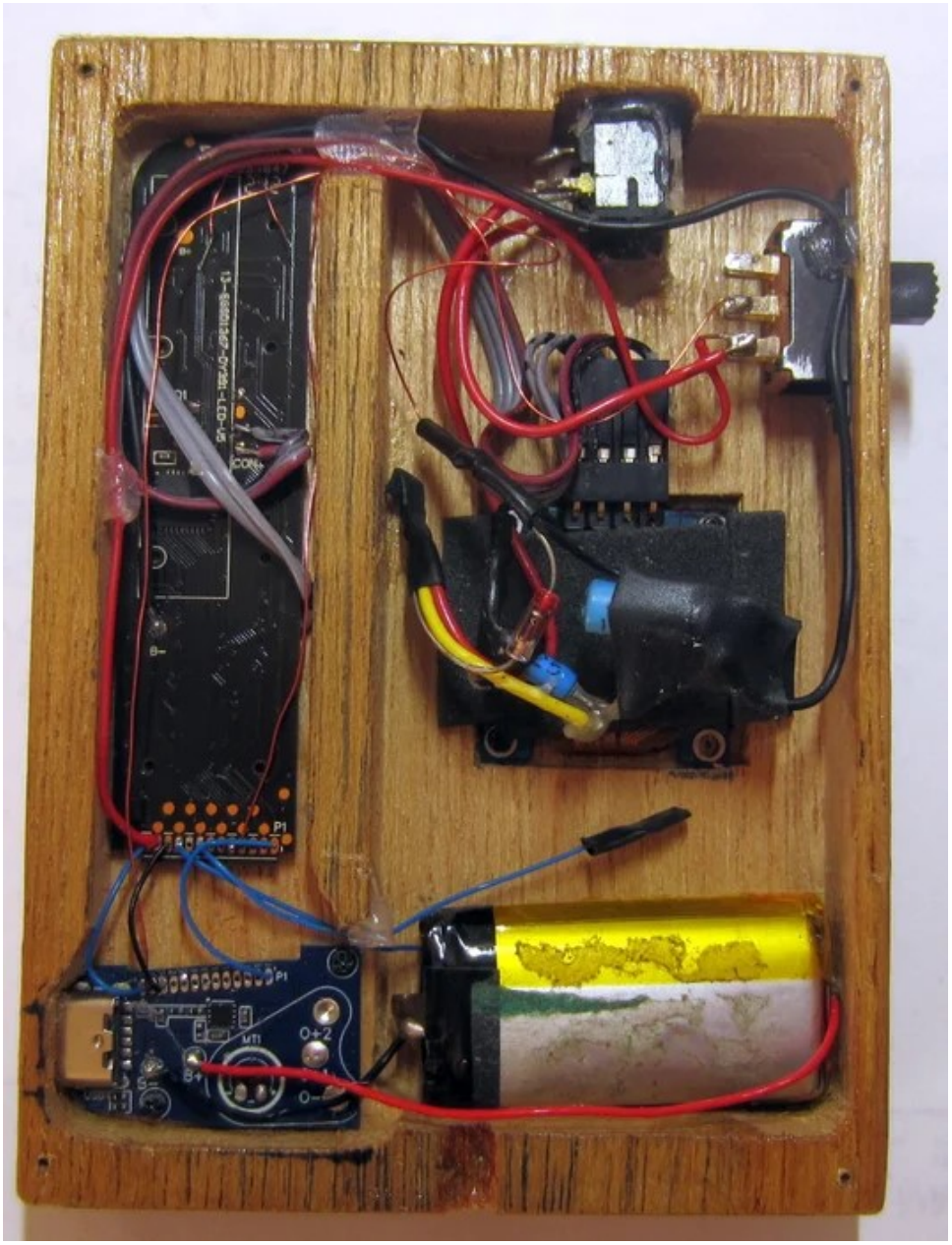
You also need to declare the i2c pins or you'll get no output to the display.This must be done before wire begin.

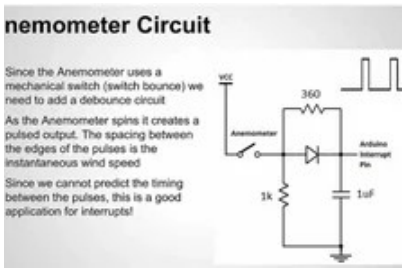
```
Wire.setSDA(PA10);

Wire.setSCL(PA9);

Wire.begin();
```

Sketch attached and also available in my Github repo.

Feel free to fork and make pull requests if you are able to get the leds working on the vape's display.

Step 4: Final Hardware Build.





I have 2 wires running up to the roof, one takes 4.2v up to the reed switch and the other end of the reed switch connects to the debounce circuit.

The other side of the debounce goes to PA7 of the PY32 mcu.

Any small signal diode should work, I used a glass AAZ15 as it was in my parts tub.

The debounce circuit is optional and it works without it, you could also code a debounce wait function in the sketch, but I found it easier to solder components at hand.

Some cavities were cut out of a meranti plank for the parts to fit in and closed off with a rear panel from an ice cream tub.

A big thanks to TonySC, a software dev on a local vape forum, familiar with the Chinese language who helped and guided with the predominately Chinese documentation.