**General**

Missing files on git

Global labels?

Root sheet? What is your layout.

No folder of datasheets.

**Power**

Schottky diode taking the entire board’s current through it – make sure is rated correctly.

Cannot turn power on/off to sensors or to hard reset them.

C8 should be bigger.

LED resistors too small.

What is R6 for.

Battery is not DNP.

Add external connectors (01x02) for battery and Voltage out.

**Sensing**

Small resistors are going to drawer a lot of power.

Fuse is good form of safety – might be expensive.

Check if pullups are correct for both sensor and EEPROM as they are both on the same line.

Maybe consider redundant.

Is the parallel combination actually necessary.

1. **Adherence to Mr Pead’s basic requirements**
   1. **Rating:** 3/5
   2. **Missing requirements** (all other requirements are satisfied):

* 18650 battery holder is not DNP.
* It does not look like power components can be easily bypassed.
* USB connector is not a Micro USB
* Total cost of components is $56,76 which is just above the recommended budget.
* NB!! Extended component cost of $3 per part does not appear to be budgeted for.
  1. **Comments from**
     1. **General**

The market is currently favourable, putting our budget at around $80, however I can see that a few parts have been chosen that are quite expensive as well as I cannot see any extended component costs include – which are $3 a per component type as a reminder. For example, the FT231XS-R is $3.10 and is an extended part, bring the total cost to $18,50 while the total cost on your BOM is $15,50 for the same component.

The Git repository is poorly documented and is missing a lot of files. We have chatted about this in person, the comment is just for completeness’ sake. I would advise uploading all KiCad files as well as a folder for datasheets of the components you use, which will make it more friendly for other engineers to read. To bulk add files to a git, you can just add the folder they are in using >git add [folder name].

* + 1. **Power**

-

* + 1. **Microcontroller**

The USB we are required to use is a micro USB part no. C404969, they work pretty much the same so it should not be too hard to put in.

* + 1. **Sensing**

-

1. **Adherence to Mr Pead’s debugger requirements**
   1. **Rating:** 4/5
   2. **Missing requirements**

* FTDI test lines
* Plug and detect.
  1. **Comments from**
     1. **Power**

-

* + 1. **Microcontroller**

You do not have test points for the signal coming though the differential pair lines from the USB (D+ and D-) nor the USART signal going to the board. These might be helpful if your USB or FTDI fails so you are able to read/input a signal to these lines.

There are no test points for your EEPROM I2C lines. I would advise adding them so that another board may communication with your EEPROM and get its data, in case of a failure.

There is no system to plug and detect the USB. I would advise taking the 5V input from the USB and running it over a voltage divider to drop it down to 3V3, and then read that signal using a GPIO pin. The voltage divider resistors should be big to prevent power losses., i.e., 3.3MΩ and 1.7MΩ.

* + 1. **Sensing**

-

1. **Schematic**
   1. **Rating:** 2/5
   2. **Comments from**
      1. **Power**

-

* + 1. **Microcontroller**

This is going to be long – I do apologise. In general, I would advise reading through the STM32F051C6 datasheet and looking at the STM32 dev board circuit diagram. Both can be found on the EEE2046F Vula site.

**Pins**

The BOOT0 and NSRT pins on the microcontroller are floating.

The BOOT0 pin, along with the BOOT1 bit, tells the microcontroller what mode to boot in. 0 for use current program, 1 for wipe program and look to BOOT1 for further action, you are welcome to do some research on these – they are not very clear in the datasheet.

The NSRT pin is dedicated to resetting your board and telling it to start again (based on the BOOT0 pin and BOOT1 bit). I would highly recommend putting a button to pull this pin low (resetting the board) for debugging purposes and restarting your program. I would advise additional buttons as well for further debugging and user input.

Only two of the seven power connection have been made. There are four voltage pins and three ground pins that are each linked to different functionalities on the microcontroller such as the ADC and GPIO clocks. I would advise connecting them all to their respective power and ground lines. You can find all of their functions on the STM32 datasheet.

The are no connections points for the STM link debugger. I assume this means you are planning on moving the STM32 board between the dev board and your HAT to debug it. I would advise against this as moving it between the two board over 40 times throughout the process of writing the code may get tedious and damage the connectors. The STM link debugger communicated through three channels: NSRT, SWDIO (PA13) and SWCLK (PA14). I would advise making pin headers on your board to receive these three lines at least and link them to your board.

There are a lot of floating pins that may cause ERC errors. A simple fix is to use ‘No-Connection Flags’ (press Q) which places an X on the pin which will tell the board that these are not floating.

**Wiring**

The are no redundant channels/wires to your microcontroller for any commination or sensor lines. These are not necessary, but I would advise them in case of a PCB or STM32 failure where one channel is not accessible.

The EEPROM chip is connect to 5V and from what I can see from your power schematic, you are unable to produce 5V from the battery. Thus, the EEPROM will only work when a USB is connected, and you will be unable to store data from your sensors in the field or remote location. I would advise changing it to be powered by 3V3 and confirm that that EEPROM can run on 3V3 power.

The UART connections are backwards. For UART/USART, there are transmission lines (Tx) and receiver (Rx) pins. When connecting between the FTDI and STM32, the transmission pin (Tx) should connect to a receiver pin (Rx) and vice versa.

I am unsure whether the Ferrite bead is necessary for the USB to FTDI connections so I will leave it up to your discretion. If it is not highly necessary, I would advise using a capacitor rather to save costs.

**Components**

The FTDI connect is a valid chip and will do the job, however it has a lot of unused functionality and is very expensive. I am using an HT42B534 series chip. They make a version purely dedicated to differential pair lines to UART which is only 8 pins and a lot cheaper. The exact model I am using is a HT42B534-2.

There are no pull-up resistors for your I2C lines between the EEPROM and microcontroller. I2C lines are recommended to have pull-up resistors attached to the lines, the EEPROM datasheet will give you some recommended values.

* + 1. **Sensing**

-

1. **PCB Layout**
   1. **Rating:** 5/5
   2. **Comments from**
      1. **General**

We are using through hole test points; it is a different approach will allow us to solder wire to the board instead of just holding them there. Your approach is also completely valid.

* + 1. **Power**

-

* + 1. **Microcontroller**

Board layout looks clean, the USB could maybe be closer to the edge of the board. I would also advise against running traces between the pins of the microcontroller headers, as they might be misprinted and short circuit. Rather run them around the sides of it. Additionally, TP1 will be quite hard to access once the microcontroller is in place.

* + 1. **Sensing**

-

1. **Silk Screen**
   1. **Rating:** 4/5
   2. **Comments from**
      1. **General**

Some of the component names are covered by the components themselves.

The team number, version, and team members are missing from the silk screen.

It would be worth labelling your test points to make it easier to debug.

* + 1. **Power**

-

* + 1. **Microcontroller**

It might be worth labelling the pins on the microcontroller in silk screen: PA0, PA1, NSRT, etc. to make for easier debugging and reading of the circuit.

* + 1. **Sensing**

-

1. **Low voltage protection circuit**
   1. **Rating:** /5
   2. **Comments from Power**

-

1. **Physical Board Design**
   1. **Rating:** /5
   2. **Comments from General**

no idea what to say here.

1. **Test point and recovery approaches**
   1. **Rating:** 4/5
   2. **Comments from**
      1. **Power**

-

* + 1. **Microcontroller**

There are a few more test points that can be added for the I2C lines and the USB differential pair and UART lines. Otherwise, there are sufficient test points.

There is no redundant routing for communication nor sensing lines which are no necessary but are advisable in the event of trace failure.

* + 1. **Sensing**

-