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.
* Lack of pin headers to use external power sources.
* USB connector is not a Micro USB
* Total cost of components is $56,76 which might be a little too close to the allowed budget when board manufacture and assembly costs are accounted for.
* Extended component cost of $3 per part does not appear to be budgeted for. This will also exacerbate the issue above.
  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.

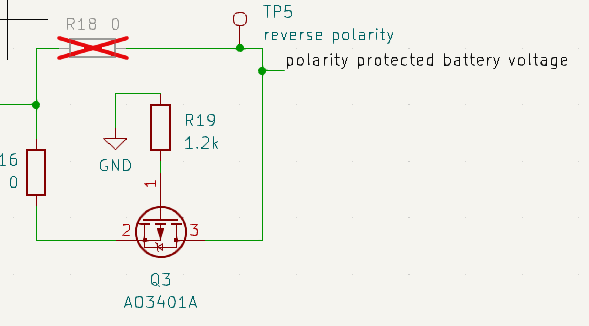
The Git repository is poorly documented and is missing a lot of files. We have chatted about this in person, the comment is just for completeness’ sake. I would advise uploading all KiCad files as well as a folder for datasheets of the components you use, which will make it more friendly for other engineers 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**

It was suggested that an external connection point this puts across the battery so that we can easily supply 3.3 volts without needing to do difficult modifications to our circuit board. I suggest adding two pin headers one connected to the 3.3 Volt positive line and one connected to ground. do not populate these pin headers as we can do this in the lab and these pin headers are expensive.

I suggest selecting do not populate for components such as the battery that will not be populated. This can be done by clicking edit or tapping E while having your mouse hovered over the symbol on the schematic in kicad and then under attribute selecting the checkbox which says do not populate.

I also suggest creating a method which can be used to bypass components with zero ohm resistors on your circuit for extra safety in case a component breaks. This is a viable way of doing this:



The circuit section can be permanently removed from the circuit by de-soldering R16 and soldering R18. this will remove Q3 from the circuit and create a short-circuit to the next section of the circuit.

* + 1. **Microcontroller**

The USB we are required to use is a micro-USB (JLC part no. C404969). It functions almost identically to the currently selected USB, so shouldn’t be difficult to change.

* + 1. **Sensing**

High level requirements have been met. Adding more sensors could be considered; they add more functionality to the board and act as a good failsafe if the sensors stop working for whatever reason. Additionally, with some research it’s not too difficult to find sensors that fit within budget.

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

* Many of the debugger requirements are met by test points; but the actual 10 pin debugger is missing.
* FTDI test lines
* Plug and detect.
* Must provide an easy 5 Volt power connection to the debugger.
  1. **Comments from**
     1. **Power**

A pin to manually input 5V power is missing.

* + 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**

Test points for the digital communication lines and for the analogue sensor are present, but there is no connection directly to the 10 pin debugger. Additionally, jumpers could be added to disconnect the sensors in the instance that you want to push data directly into the microcontroller without risking damage to the sensing circuits.

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

There are some global labels on the schematic; convention is that it’s generally better to use net labels and hierarchical labels.

On the schematic on the gitlab, there is no root sheet that links the various components together – thus; the digital sensors currently don’t appear to actually be connected to the microcontroller (and the analogue sensors are connected using global labels rather than hierachical labels)

The individual schematics were well laid out, and the text explanations made understanding the functionality easy to understand. Some comments on the actual circuitry can be seen in the next few points:

* + 1. **Power**

I am worried that the Schottky diode will produce a lot of heat and could break. I suggest doing the calculations to see what the Max current that your circuit will draw is roughly and then calculating the amount of heat that will run through your Schottky diode using the formula power is equal to voltage times current (P =VI). From the data sheet for the Scott key diet that you are using you can only have 400 mW through it.

I also suggest using bigger resistance for the LED's connected to the battery charger to help prevent them from breaking. The same calculation can be done to work out the current that should be drawn through those resistors. I suggest just using a 1K or even bigger resistors - you will have dimmer LED's but it is less likely to break.

I also don't understand why capacitor C8 is 100 nanofarads. I would suggest using a bigger value capacitor such as 10 microfarads. Smaller value capacitors are used to decrease noise but as this is a power line we are more interested in stabilizing the power produced; therefore we should use a bigger valued capacitor. The data sheet for this component suggests using a 10 microfarad capacitor.

I am also unsure what resistor R6 is for; it may just be an unnecessary power draw.

I was very impressed with your circuits and was very interested in your DC to DC converter as this seem to be a good solution to the problem.

* + 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 stricrly 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.

I2C pullup resistors are shown on the sensing schematic. Given that the pullups are necessary for EEPROM as well, it may be clearer to move the pullup resistors to the microcontroller schematic, or at least to include a note that the resistors appear on a different schematic.

* + 1. **Sensing**

The resistors used in the analogue sensor voltage divider have low values that are likely to draw unnecessarily high amounts of power. It may work better to use higher resistance values (and possibly a higher resistance thermister as well).

The resistor in parallel with the thermistor seems unecessary; it should be possible to get a analogue reading simply by putting the thermistor in a voltage divider with a single other resistor. The non-linear nature of the thermistor can be accounted for in software.

The pullup resistor values on the I2C lines are in the upper range of what is useable. Given that both the sensor and the EEPROM are being connected to the same I2C bus, the overall capacitance connected to the bus is greater than it would be if only the sensor were connected; thus the pullup resistor values might be too large. Decreasing the resistor values will draw slightly more power, but the I2C communication will be more likely to work.

It could be useful to include MOSFET based power switches for the sensors so that they can be turned off by software to save power.

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 valid.

There are areas on the front copper ground plane that are not filled. It is generally good practise to ensure that these areas can be filled by placing vias to the back copper ground plane.

* + 1. **Power**

It is difficult to see if you have use the correct width tracks – but in general I will suggest that all power tracks are much thicker than any other tracks.

* + 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**

No particular issues.

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**

If you have space I suggest adding labels to the different modular sections of your component E.G. the DC to DC converter and the battery charger.

* + 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**

Labelling the different sensor modules could make it clearer which components are the actual sensors.

1. **Low voltage protection circuit**
   1. **Rating:** 4/5
   2. **Comments from Power**

I'm very impressed that you found this DC to DC converter. I have read through the data sheet and I’m a little skeptical about the undervoltage lockout as it says it only locks out at 1.5 volts, but you indicate that this component only operates from 2.5 to 5 volts. However, I may be misreading something and suggest that you just double check this.

If your component does not have an undervoltage lockout here are two website links that I used to build my undervoltage lockout circuits. <https://www.analog.com/en/technical-articles/battery-protection-circuit.html>

[https://web.eecs.utk.edu/~afoshie/projects/UVLO/#design](https://web.eecs.utk.edu/~afoshie/projects/UVLO/" \l "design)

1. **Physical Board Design**
   1. **Rating: 4**/5
   2. **Comments from General**

Layout is mostly appropriate. TP1 will be difficult to access once the microcontroller has been put in place and should probably be moved. The board is moderately small, which is appropriate for the use case (the board is likely to be left somewhere for extended periods of time and taking up less space is better). That being said, there is a lot of unused space on the board which indicatest that with careful board design it could be made even smaller.

It might be nice to have rounded corners on the board, and to include mounting holes so that the board can be put in a protective case (to protect it from the elements while it is left gathering data)

1. **Test point and recovery approaches**
   1. **Rating: 3**/5
   2. **Comments from**
      1. **General**

As mentioned previously, through hole test points could be useful as they allow you to solder wires to your test points later on if you are in a pinch and want to manually wire the test points.

* + 1. **Power**

Test points are free to make, and I suggest using as many as possible. I think you have all the important test points but could put one on in between your PMOSFET and your DC to DC converter.

* + 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 not necessary but are advisable in the event of trace failure.

* + 1. **Sensing**

There are sufficient test points. However, there is no redundant routing, no jumpers to allow the sensors to be disconnected in the event that data needs to be fed directly to the microcontroller, and no backup in case the submodules do not work. As mentioned previously, it is possible to add addiotional sensors within budget which will add additional functionality to the board and act as a good failsafe in case the “main” sensors don’t work; for whatever reason.

The fuses seem like an unnecessary expense, as the risk of the sensors short circuiting is low. It may be better to instead have 0 ohm resistors connected to the sensor power so that the sensor modules can be disconnected entirely if for some reason they negatively affect the rest of the circuity; this will be a lot cheaper than the fuse.