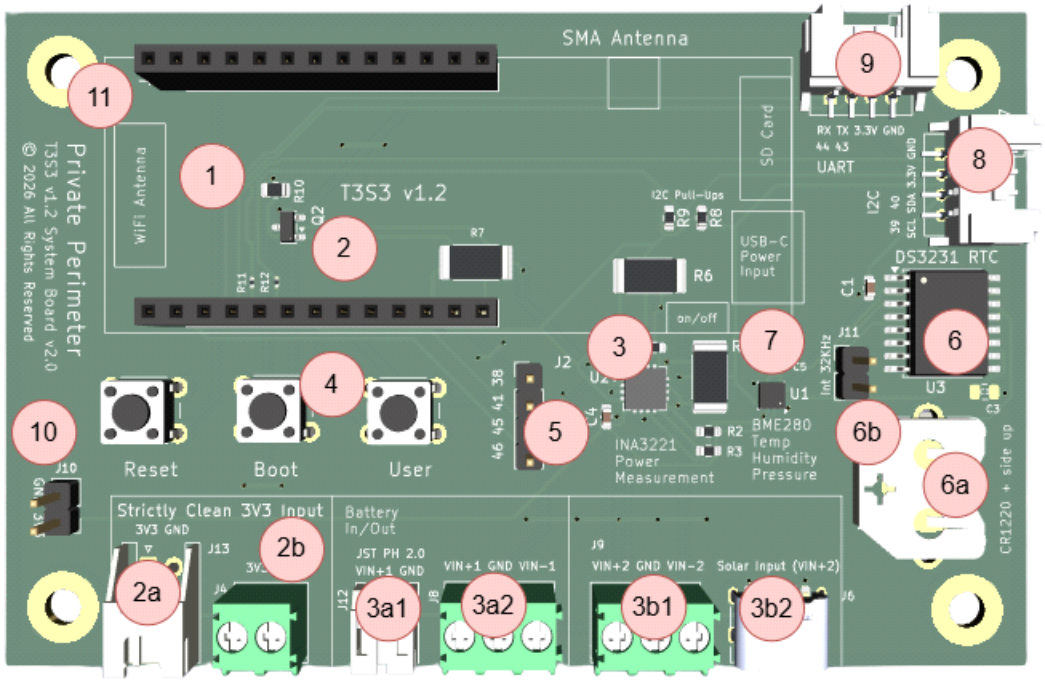
T3S3 Expansion Module Installation Notes

This board assumes that your system has a Battery Management System (BMS) that supplies a clean 3.3 volt output. Your batteries and solar panel should be managed by the BMS.



1. Plug a T3S3 v1.2 board into the expansion module with the USB facing right and the SMA connector facing up (1). You may want to reflash the processor either before this step or after you are done wiring it up using the USB-C that is on the T3S3. It should be okay to use that power even when you have the module powered from the BMS since there is circuitry to detect and protect it.

The MeshCore T3S3 variant requires the firmware to be recompiled to enable the sensors on this board. That is because the I2C on the T3S3 is only connected to the OLED Display and is not available for use.
To recognize the sensors, you will need to recompile the firmware with these additonal settings in the file :

variants\lilygo_t3s3\platformio.ini

-D ENV_PIN_SDA=40

-D ENV_PIN_SCL=39

Then reflash the target with this firmware. This binary will also be available on our web site for download.
There are also many extra options on this board that you may want to enable in the firmware to take advantage of additional options at the same time including the user button and compiling for very low power operation.

2. Connect the battery holder to the Battery input. Use either the standard JST PH 2.0 that is common for many 18650 battery holders (3a1) or use the terminal block to wire into the VIN+1 and GND ports (3a2). Make sure to pay attention to polarity since there is no universal standard.
If using the terminal block, Private Perimeter provides optional JST PH 2.0 pigtail cables that can be used to wire a female connector to VIN+1 and GND and then plug the battery holder into that. Then plug the male JST PH 2.0 cable into GND and VIN-1 and plug the other end into your BMS. Alternatively, you can hard wire it as you see fit. If you wish to bypass battery monitoring, you can just plug your battery holder directly into your BMS.

3. Plug in the solar panel if you wish to track the input power. You can plug it into the USB-C connector (3b1) or the terminal block (3b2). For either one, you will need to route the output from VIN-2 and GND to your BMS. If you wish to bypass this, you can just plug your solar panel directly into your BMS.

4. Insert a CR1220 battery into the battery compartment for the Real Time Clock (6a). You can obtain a battery from Amazon or they are available at some Walmart stores. We would provide it except that it greatly complicates shipping. The Real Time Clock

will operate without it, but it will lose time when your batteries run out. With the CR1220 installed, the RTC should keep time for up to 10 years if your system normally has battery power.

Optional Connections and Features

The Expansion Module has several very useful expansion connectors for various uses for both developers and special projects. Private Perimeter will be writing notes on how to code and enable some of these options in the near future.

5. The User Button is available for use in your code as you see fit (4). It is attached to GPIO 42.

6. GPIO pins 38, 41, 45, 46 are brought out to Dupont pins so that you can jumper to extra functionality as you see fit (5). For example, you could power external equipment through a MOSFET module that is controlled by a GPIO pin. You can also jumper from the RTC alarm interrupt to one of these pins to manage a clock alarm interrupt.

7. The DS3231 RTC alarm interrupt and 32KHz pins are brought out to Dupont pins (6b). You can jumper these as you see fit. For example, you could jumper the alarm output to one of the GPIO pins and add some alarm code to wake up the processor at certain intervals.

8. There is a Grove-compatible I2C connector on the board to plug in various modules from Seeed Studio (8). Then enable them in the code or else write your own interface to it.

9. There is a Grove-compatible UART port on the board that you can use as an alternate serial port (9). This is useful fore debug and especially useful when you want to work on power saving code and test it without using the USB port which will affect the power.

When using it as an addition serial port, you can enable it in the MeshCore C code like this:

In the examples\simple_repeater\main.cpp code, for the primary setup() and loop() code:

```
void setup() {
    Serial.begin(115200);          // normal serial port on the USB already in the code
    Serial0.begin(115200);         // add this for our additional serial port at the connector

void loop()
// add this code for example after the normal command processing to accept commands over this new serial port
    // now also accept commands over Serial0
    while (Serial0.available() && len < sizeof(command)-1) {
        char c = Serial0.read();
        if (c != '\n') {
            command[len++] = c;
            command[len] = 0;
            Serial0.print(c);
        }
        if (c == '\r') break;
    }
```

And then in the src\helpers\sensors\EnvironmentSensorManager.cpp code you can add new code to display sensor values:

```
Serial0.printf(" \r\n");
Serial0.printf("BME280 temperature %f\r\n", BME280.readTemperature());
Serial0.printf("BME280 humidity %f\r\n", BME280.readHumidity());
Serial0.printf("BME280 pressure %f\r\n", BME280.readPressure() / 100);
Serial0.printf(" \r\n");
```