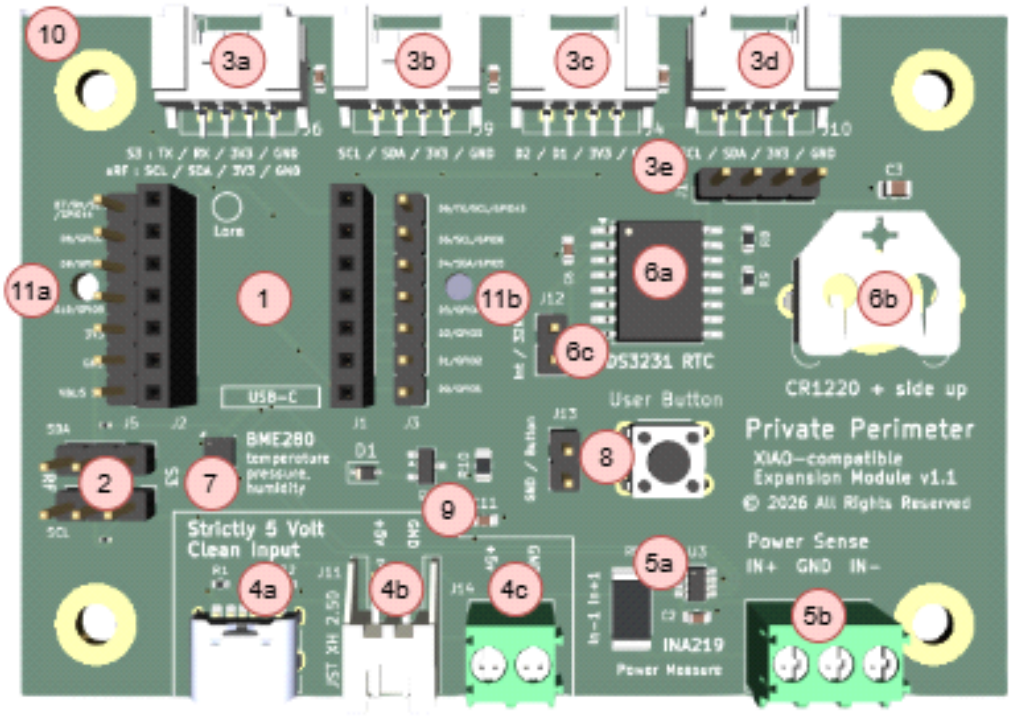
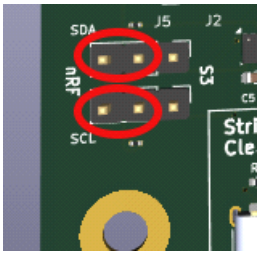
XIAO Expansion Module Installation Notes

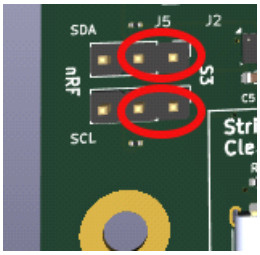
This board assumes that your system has a Battery Management System (BMS) that supplies a clean 5 volt output. Your batteries and solar panel should be managed by the BMS.



1. Decide which processor you intend to use: The Seeed Studio nRF52840 or the Seeed Studio WIO ESP32S3. Once you decide, use the provided jumpers to set the jumper pins on the board (2) to match the selection so that the I2C and Tx/Rx signals will be routed correctly.
For the nRF52840, jumper the two rows of pins so that the center pin and the leftmost pin are jumpered and the rightmost pin is left open. For the WIO ESP32S3, jumper the two rows of pins so that the center pin and the rightmost pin are jumpered and the leftmost pin is left open.



nRF jumper setting -



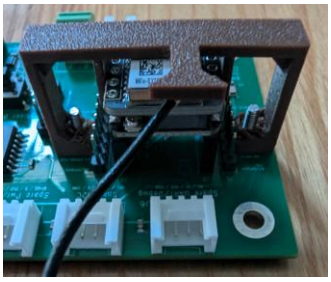
ESP32S3 jumper setting

2. Prepare the nRF52840/ESP32 processor module (not provided) with the antenna and wifi antenna connections (ESP32-only).
3. For the nRF52840, If you have the antenna retention clip, install it between the two processor boards to protect the U.fl antenna connection. Plug the processor module into the provided 2x7 socket (1) **with the USB pointing downward** to match the silkscreen. Check carefully to make sure all 7 pins on both sides are inserted into the sockets.

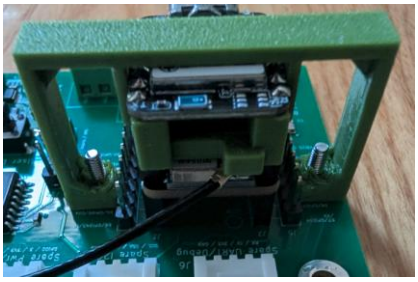


ESP32 Mounting Bracket - nRF Antenna Retainer - nRF Mounting Bracket

4. If you have the mounting bracket, install it above the processor module on either side of the sockets (11a, 11b), centering the holes over the PCB holes. Then insert the machine screws into the PCB holes from underneath so that the threads come up within the bracket space. Note that there are different brackets for the nRF52840 and the WIO ESP32S3 modules because they are different heights. For the ESP32, make sure that the antenna retainer points back over the u.FL antenna connector.



Mounted ESP32S3



Mounted nRF

5. Decide what you are going to monitor the power on since there is a single channel INA219 on the board. Use the terminal block (5b) to wire it where the input pair go to IN+ and GND and the output goes to GND and IN-. For example, to monitor the battery, if the battery uses a JST PH 2.0 connector which is common, you can use a JST PH 2.0 pigtail pair wired to the terminal block as follows:
Wire the female pigtail between IN+ and GND, then plug the battery wires into it.
Wire the mael pigtail wires between IN- and GND and route it to your BMS.
Be careful to check polarities because black and red are commonly criss crossed between the battery holder and the pigtails. There is no standard that they follow, so you might find yourself using red and black in a non-standard way.
6. Provide ***clean 5 volt power*** from your BMS into the Expansion Board using one of the three connectors (41, 4b, 4c) as you see fit. There is a reverse polarity detection circuit that will attempt to reject wires that are crossed. There is also a diode in the path so that when you are powering and possibly flashing firmware directly on the processor using its USB-C, power back to the BMS is disabled to avoid a conflict. Many BMS modules have a 5 volt USB output which is a very nice way to connect to the the Expansion Module. But you can also use the JST XH or the terminal block.
7. Provide a CR1220 battery for the Real Time Clock. We would provide it but it greatly complicates shipping and they are readily available via Amazon and sometimes at Walmart on the battery rack. Positive side up. This battery is designed to last 10 years as long as the main battery is powered most of the time. There are also pins (6c) for tapping into the RTC alarm and 32KHz signals for special features in the code.
8. Use M3 machine screws to mount this board as you see fit (10).

Optional Connections and Features

The Expansion Module has several very useful expansion connectors for various uses for both developers and special projects.
9. User Button - there is a user button (8) that is available for general use by connecting the Dupont pins next to it to one of the unused I/O lines on the processor. GND is implied, so you can simply add a Dupont jumper to whichever I/O line you want to use and then provide the code modification to monitor it as you see fit.
10. There are pins that are exported from the DS3231 RTC that can be used to setup an alarm interrupt so that you can wake up the processor at certain times. You will need to jumper over to one of the processor pins and provide the code to set up the interrupt for it.
11. There are two I2C Grove connectors available for expansion (3b, 3d). For example, you can plug a Seeed Studio OLED display into one of them and enable the display in the Meshcore Repeater project and it will begin using the display automatically.
12. There is an additional I2C interface available (3e) via Dupont pins for jumpering to other sensor modules.
13. There is a general purpose Grove connecter (3c) available for exporting two of the I/O lines and power and ground for custom uses.
14. There is a Grove connector (3a) that has two different purposes depending on the jumpers on the board. The nRF52840 has this

as another I2C port since it does not have enough pins to support a serial port. The ESP32S3, on the other hand, has this as an addition serial port. You can create a Grove cable that will route Tx,Rx,GND to a serial to USB adapter and plug it into a PC for development. This is immensely useful when doing power measurements where the normal USB would affect the test. You can also use it for debug logging to a PC.
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("INA219 calibration value %u\r\n",INA219.getCalibration());
Serial0.printf("INA219 bus voltage %f\r\n",INA219.getBusVoltage_V());
Serial0.printf("INA219 shunt voltage %f\r\n",INA219.getShuntVoltage_mV());
Serial0.printf("INA219 current %f\r\n",INA219.getCurrent_mA());
Serial0.printf("INA219 power %f\r\n",INA219.getPower_mW());
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");
```