**Building the super-node on I2C**

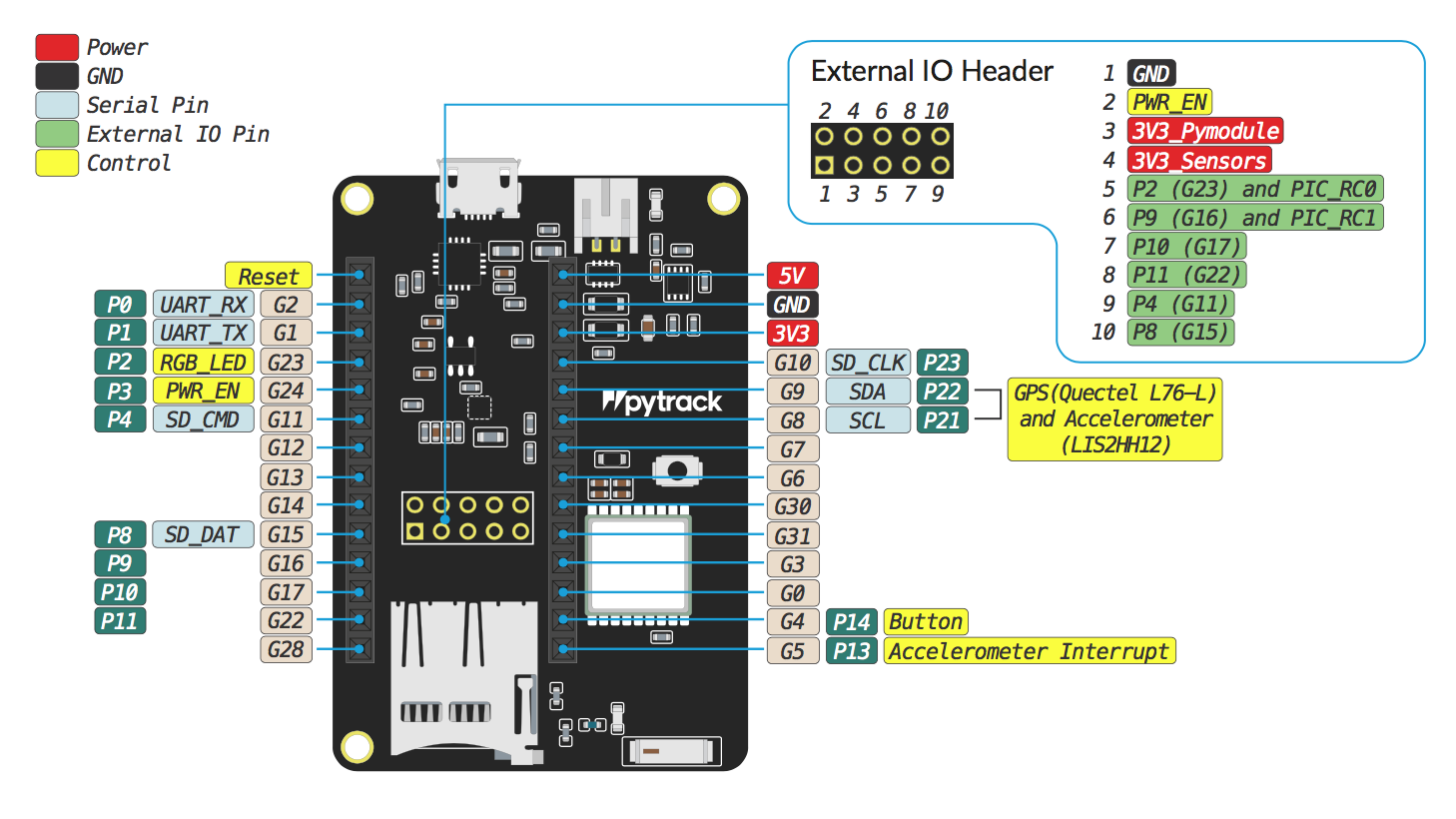
Pytrack or Pysense?

For our project, the super-node will be the end-device that will send weather data and GPS coordinates if the super-node is moved from one location to another. 4 parameters are measured: temperature, barometric pressure, relative humidity and GPS coordinates.

The Pysense expansion board seems to be a good choice for the weather measurements, because it implements all 3 parameters we need to measure. However, it would imply that a GPS sensor must be connected to the board if the super-node is going to be moved.

GPS measurement could either be dropped (which would imply that the super-node will be a static node) or the super-node could be connected to the Pytrack expansion board. If this way is chosen, this would mean that the GPS coordinates will come from the built-in GPS chip on the expansion board and weather measurements will come from added sensor(s).

The Pytrack board has 10 external IO headers, were the sensor(s) inputs and outputs could be soldered. It is easier to deal with only one sensor (like the BME280 which can measure all 3 weather parameters) than two. The last situation is a bit more complex to manage, especially if we use I2C communications.



**Fig.1: Pytrack pinout schematic [1]**

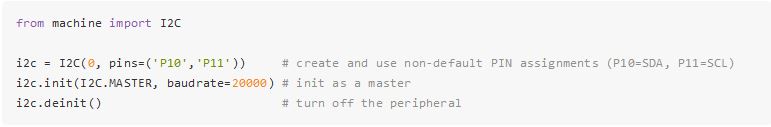
The figure above (Fig1) shows that several pins on the external IO header fulfill a specific role in the functioning of the board. For example, pin 1 to pin 4 are used for electrical supply and pin 2 is used by the RGB\_LED. This means that not all pins can be used to connect the sensor to the board (2 components on the same pin could destroy one another). A deeper look at this schematic suggests that pin 10 and pin 11 are free and could be used by the sensor to communicate with the lopy.

I2C communication

Now comes the choice of the transmission bus. SPI and I2C are the main ones, but for this project, we will focus more on I2C. It is a serial bus that is synchronous and bi-directional. It allows only half-duplex communications (one device at a time speaks, not both at the same time). The devices connected to this bus work in a Master-Slave relation. Connection is made through SDA (Serial Data Line) and SCL (Serial Clock Line). SDA line is used to transfer the data in both directions and SCL line is the synchronization clock [2].

Pycom products (such as the lopy or the pytrack expansion board) support I2C communications. The connected sensor (=slave) must have I2C compatibility as well to communicate with the lopy (=master). Also, the pins that are going to be used must allow for I2C communication.

By default, the lopy module uses pin 9 and pin 10 for I2C[3] [4]. However, it is possible to change pin assignments and set our own I2C communication pins [4]. This is shown in the example below:



**Fig.2: example code of I2C using non-default pins (here pin 10 and pin 11) [5]**

The example in the figure above (Fig.2) is interesting, because it provides a simple example of how to initiate an I2C communication on non-default pins and it also shows us that pin 10 and pin 11 are usable for this purpose.

Bibliography

[1] Pytrack pinout: <https://docs.pycom.io/datasheets/boards/pytrack> (viewed the 30.03.2019 at 21h00)

[2] Wikipédia – I2C: Topologie: <https://fr.wikipedia.org/wiki/I2C#Topologie> (viewed the 30.03.2019 at 21h10)

[3] Lopy pinout (image): <https://docs.pycom.io/.gitbook/assets/lopy-pinout.png> (viewed the 30.03.2019 at 21h20)

[4] Lopy pinout: <https://docs.pycom.io/datasheets/development/lopy.html#pinout> (viewed the 30.03.2019 at 21h20)

[5] Example of I2C communication using non-default pins: <https://docs.pycom.io/firmwareapi/pycom/machine/i2c.html#example-using-non-default-pins> (viewed the 30.03.2019 at 21h20)