Requirements:

* Create module for just handling all of ipx communication,
* Needs to open/close ports and handle all potential errors
* Vision is that its as easy as using IPX.get\_uids when coding main script
* Just code to focus on sending the commands and receiving the right responses.
* Need to get the hang of serial communication
* Automate com port detection, don’t leave selection to person using the software

Tasks:

* Mess around with serial communication and properly understand it
* Then finish coding IPX communication module

21/10/2025:

Tasks:

* Add response error checking for everything, thing is the response of things like get\_raw are a string of just numbers, same with list\_uids, so cannot check if the same thing is in there, perhaps with uid, can check with 102, but this is a janky method and will cause error if changed to a different number
* Verify that execute and verify function is working as expected, is currently returning none
* Make sure execute and verify always returns a tuple even if something fails
* Figure out how to manipulate responses in a separate file, then transfer logic to the main file

22/10/25:

* Gemini suggesting to add both a multiline reader, and a single line reader  
  Unsure as not entirely sure how IPXs respond, is it always one line?
* Fully working version of the code without the verification function, works very effectively
* ATM have to use with as for the \_\_enter\_\_ and \_\_exit\_\_ dunder methods, not sure if this is the most effective way of using the code, I would like a version where the end user wouldn’t have to utilise the with as function, and just write say ipx = IPXserialcommunicatr (etc etc). However, gemini thinks this may be fairly complicated to implement, going to leave as is for now

Next steps:

* Explore implementation of with as statements internally:
* Test if the code fully works with dan
* Add logging/.printing so each uid prints as its received
* Structured response from command to command -> higher level, return byte array or string?
* Add error catching for example if a janky loads of bytes are received, catch this, small error catching needs to be added
* Firstly focus on understanding how the ipxs communicate with the terminal across all command
* Additional argument in function for returning different things – higher priority, focus on getting hang of manipulating the data received

Above task pretty much completed

Tasks for tomorrow (23/10/2025):

* Create centralised helper function for decoding string to avoid repeating
* Ask gpt how to approach custom timeouts as different commands have different response timings.
* Make readme.md
* Ensure all set commands are fully working, maybe change from send and receive to just use send receive and listen?, need to change the logic slightly, so that everytime some newline is received etc, the code can be updated to pass some of the logs to info logging, example is configuration where loads of lines are made
* Finetune timeouts for every function
* Calibration procedure has a very large delay
* Focus on creating the configuration logic for the sensors,
* Add automatic switch for turning ipx on and off -> maybe at higher level

Maybe something to consider?:

A screenshot of a computer

AI-generated content may be incorrect.

* Original uids:  
  

Completed the IPX.py module

Potential improvements:

* Perhaps add a check to send\_receive\_listen function, so that once the last configuration line is received, stop listening as waiting 20s for that function to finish is a waste?
* Or update for one line responses to stop listening after the response is received
* Maybe change some info logging to debug, such as the statements which say the serial port has been opened successfully.
* Clearing the terminal in case a command gets stuck?, clearing the input -> ask dan is this possible?

(perhaps integrate all functions into ipx gui, not just for building/configurating but for messing around with IPXs as well)

Move on to configurator code:

* Need to add verification command that ensure the sensors are outputting exactly what they need
* Get raws and check if something is wrong, eg 0 mean, 0 std dev
* Exceptionally high value returned in get raw means something is wrong
* Need to add verification function (function which verifies response is as expected)
* Add receive function as listed before