**CAN Bus Home Sensor Network**

**2024-08-03**

**Jeff Mangum**

Hardware:

* Home Node (Node 0):
  + [Adafruit RP2040 CAN Feather with MCP2515 CAN Controller](https://www.adafruit.com/product/5724)
  + [Adafruit Featherwing 128x64 OLED](https://www.adafruit.com/product/4650)
* Remote Nodes (Nodes 1 through 3):
  + Nodes 1 and 2: [Adafruit RP2040 CAN Feather with MCP2515 CAN Controller](https://www.adafruit.com/product/5724)
  + Node 3 (Send Node): [Adafruit ESP32-S3 Feather with 4MB Flash 2MB PSRAM](https://www.adafruit.com/product/5477) with [Adafruit CAN Bus Featherwing MCP2515](https://www.adafruit.com/product/5709)
* Sensors:
  + 3 – [Adafruit Sensirion SHT41 Temperature and Humidity Sensor](https://www.adafruit.com/product/5776)
  + 1 – [Adafruit MS8607 Pressure Humidity Temperature PHT Sensor](https://www.adafruit.com/product/4716)
* CAN network wiring: Home telephone wiring (4 strand with 2 twisted pair; I believe that the standard is RJ-11)
* All nodes connected to power using AC-to-UBS converters, and all nodes equipped with LiPo batteries. Test of remote node with [Adafruit 3.7v 2500Ah LiPo battery](https://www.adafruit.com/product/328) indicated that remote node can send measurements every second for about 1.5 days on battery power.

This configuration results in needing to have two “listen” nodes: one for the real-time display (Home Node) and a second for sending measurements to Adafruit IO (Send Node). This seemed to work fine for my application. One could possibly reduce this to one listen node (by stacking the OLED on the ESP32-S3 Feather). On the other hand, separating the send functionality from the real-time display meant that I could put the send node at one of the remote locations (which could be chosen to be more accessible to wifi).

No special wiring or pin assignments needed (all plug-and-play). Word of warning, though, when using RJ-11 patch cables. Some cables are “swapped” (black-red-green-yellow on one end, yellow-green-red-black on the other end), while others are “straight”. Since most of the connectors are see-through plastic, you can simply look at the colors of the wires in the connectors to see if you are using a swapped or straight patch cable. I used all three wires (H, L, and ground) for the CAN bus connections. See connection assignments used in Table 1. Did not notice any problems (i.e. dropped packets) with this four CAN node setup.

The Home Node (node 0), with is the one with the OLED display, allows one to display real-time measurements from all nodes using the three buttons on the OLED Featherwing. The OLED button reader library I used allowed for complex short-, long-, or multi-click input to select each of the four nodes to display. I found that in order to display the fourth node (node 3 in my setup) that I had to use a long-click as I could not get the multi-click capability to work.

|  |  |  |
| --- | --- | --- |
| Wire Color | RJ11 Pin Number | CAN Bus Wire Assignment |
| Straight RJ11 CAN Bus Node Wiring | | |
| Red | 3 | H |
| Black | 2 | Ground |
| Green | 4 | L |
| Crossed RJ11 CAN Bus Node Wiring | | |
| Red | 4 | H |
| Black | 5 | Ground |
| Green | 3 | L |

Software:

* CircuitPython 9.1.1
* All libraries from CircuitPython 9.x standard and community bundle (OLED button reader library was from community bundle)
* Used [cooperative multitasking (asyncio)](https://learn.adafruit.com/cooperative-multitasking-in-circuitpython-with-asyncio) to manage sensor read, CAN bus send, OLED display, and Adafruit IO upload tasks.

A small rectangular device with wires and a green light

Description automatically generatedA device connected to a cord

Description automatically generated

Left: Home node (node 0) with 128x64 OLED Featherwing. Right: Home node setup showing telephone jack connection (upper-left), battery backup power (lower-right), USB power connection (upper-middle), and environmental sensor (middle).

A small black and yellow circuit board with wires

Description automatically generatedA close-up of a device

Description automatically generated

Left: Send node (node 3) with ESP32-S3 (bottom) and CAN Bus Featherwing MCP2515 CAN controller (top). Right: Send node setup with RJ11 breakout connecting CAN Bus Featherwing to telephone jack (top), battery backup power (center), USB power connection (left, off of photo), and environmental sensor (bottom).

A circuit board with wires

Description automatically generatedA group of electrical components on a table

Description automatically generated

Left: Standard remote node (node 1) RP2040 CAN Feather with MCP2515 CAN controller. Right: Standard remote node setup with RP2040 CAN Feather controller (left), battery backup power (bottom), environmental sensor (bottom-right), and RJ11 breakout connecting CAN controller to telephone jack (right).