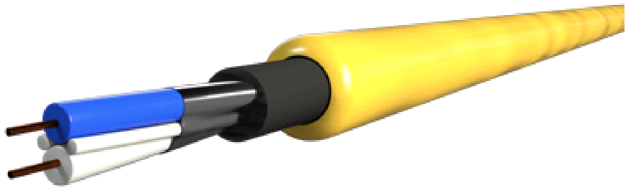
SPEED UP THE FUTURE!

Single Pair Ethernet (SPE) Design Challenge

Build an original application for 10BASE-T1L SPE for a chance to **win fantastic prizes**!



Evaluation of Single Pair Ethernet (SPE) application for CubeSats

Osiel Montoya, University of Hawaii at Mānoa
Hawaii Space Flight Laboratory
Team Name: Ke Ao Māhoe CubeSat

The Ke Ao CubeSat.

Table of Contents

1. Overview

The Ke Ao (“The Cloud” in Hawaiian) CubeSat is a low-cost educational satellite platform developed with engineering students at the University of Hawaii to advance Aerospace education in Hawaii. Ke Ao consists of a “stack” of PCBs arranged in a cube shape. In order to send signals between PCBs in Ke Ao, we use a PC/104 bus. Though this is a simple solution, the bus has some problems:

1. The header for this bus takes up a large amount of space on the PCB (52x8mm on a 100x100mm board).
2. Boards that don't use a signal must still have a header to pass it on.
3. We don't use every pin on the bus, which is a waste of valuable space and weight.

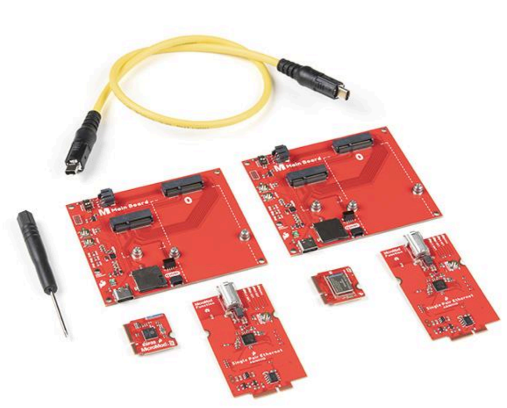
A solution to this problem is to reduce the number of pins in the bus. Ideally, we should use the absolute minimum of pins in this bus. However, we still need to pass signals of various types across the satellite. The 10BASE-T1L Single Pair Ethernet (SPE) standard presents an opportunity to reduce the number of pins down to just two. Appropriately, this project’s name is Māhoe, “twins” in Hawaiian.

This project is an initial experiment, an evaluation of the SPE standard for our purposes. In this test, the SPE standard is used to transmit sensor readings from a “sensor” board to a simulation of Ke Ao’s On-Board Computer (OBC) board.

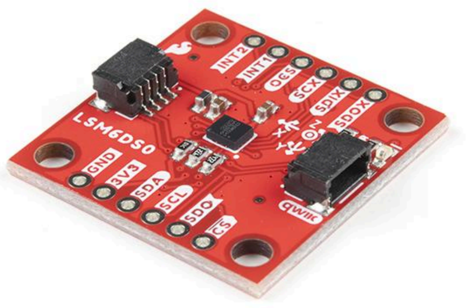
2. Setup

2.1. Hardware

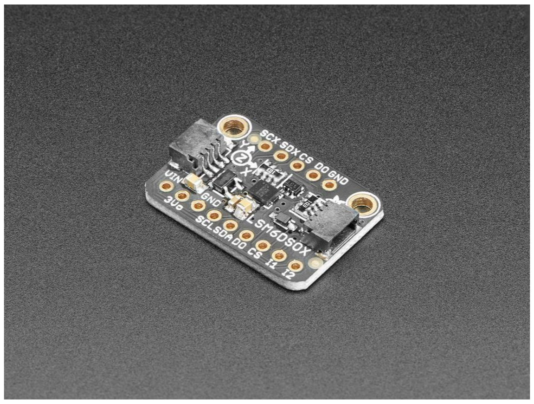
The hardware used in this project consisted of the following:

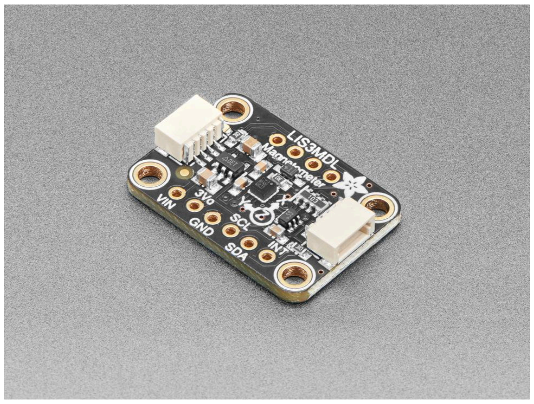

SparkFun MicroMod Single Pair Ethernet Kit

SparkFun Qwiic Cable Kit


SparkFun 6 Degrees of Freedom Breakout - LSM6DSO (Qwiic)

SparkFun Micro Magnetometer - MMC5983MA (Qwiic)


Adafruit LSM6DSOX 6 DoF Accelerometer and Gyroscope - STEMMA QT / Qwiic

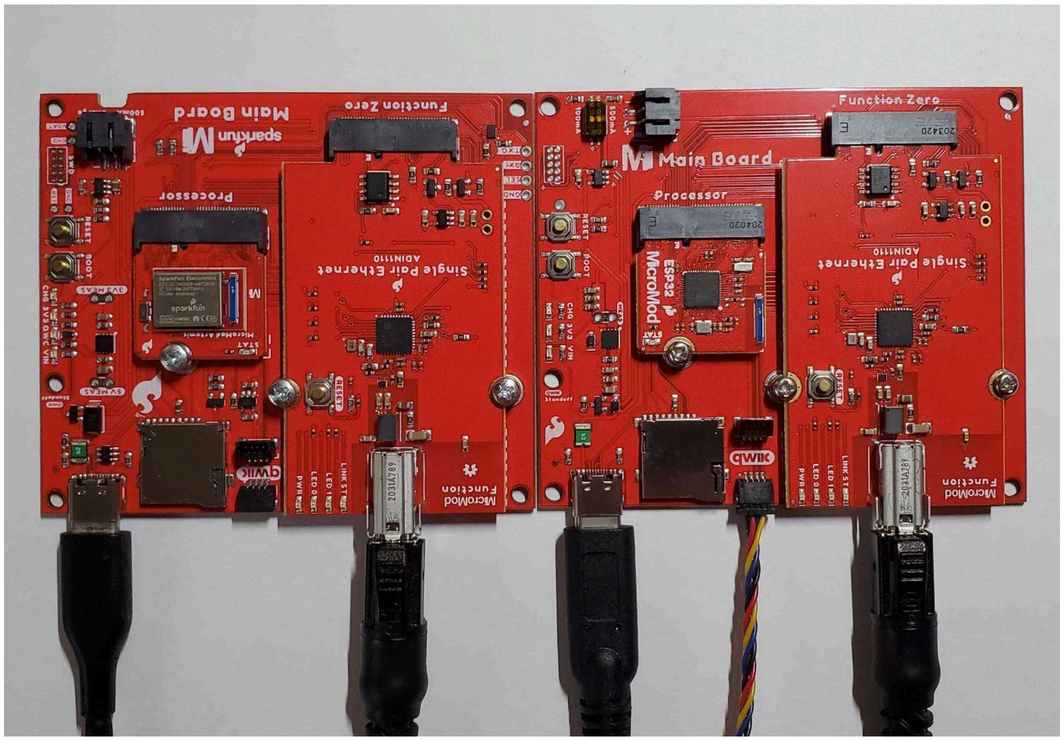

Adafruit Triple-axis Magnetometer - LIS3MDL - STEMMA QT / Qwiic

The Adafruit sensors were chosen for this evaluation as they are currently used in Ke Ao. The SparkFun sensors are functional equivalents. Additionally, the STEMMA QT and Qwiic connectors and cables are compatible.

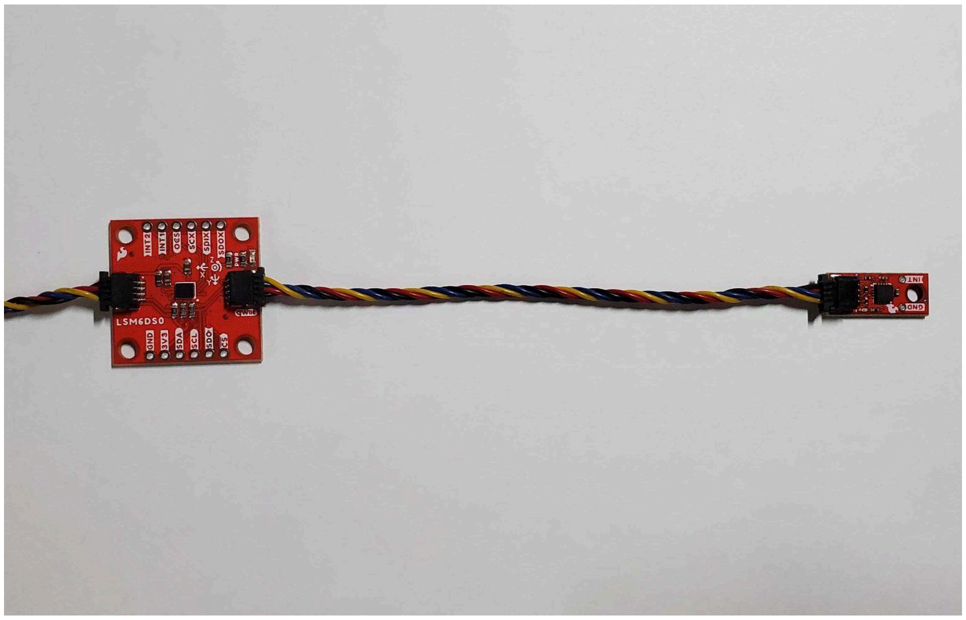
Either microcontroller could take the role of OBC or sensor board. In this project, the boards were set up as follows:

- The On-Board Computer (OBC) board containing
  - 1 SparkFun MicroMod Main Board - Single
  - 1 SparkFun MicroMod Artemis Processor
  - 1 SparkFun MicroMod Single Pair Ethernet Function Board - ADIN1110
- The Sensor Board containing
  - 1 SparkFun MicroMod Main Board - Single
  - 1 SparkFun MicroMod ESP32 Processor
  - 1 SparkFun MicroMod Single Pair Ethernet Function Board - ADIN1110
  - 1 SparkFun 6 Degrees of Freedom Breakout - LSM6DSO (Qwiic)
  - 1 SparkFun Micro Magnetometer - MMC5983MA (Qwiic)

The MicroMod Main Boards were connected to each other using the 10BASE-T1L SPE cable, and both MicroMod Main Boards were connected to a PC with USB-C cables.


The MicroMod boards and their connections. The OBC board is on the left, and the sensor board is on the right.

The sensors were connected to the MicroMod Main Board using Qwiic cables in a “daisy chain” configuration.

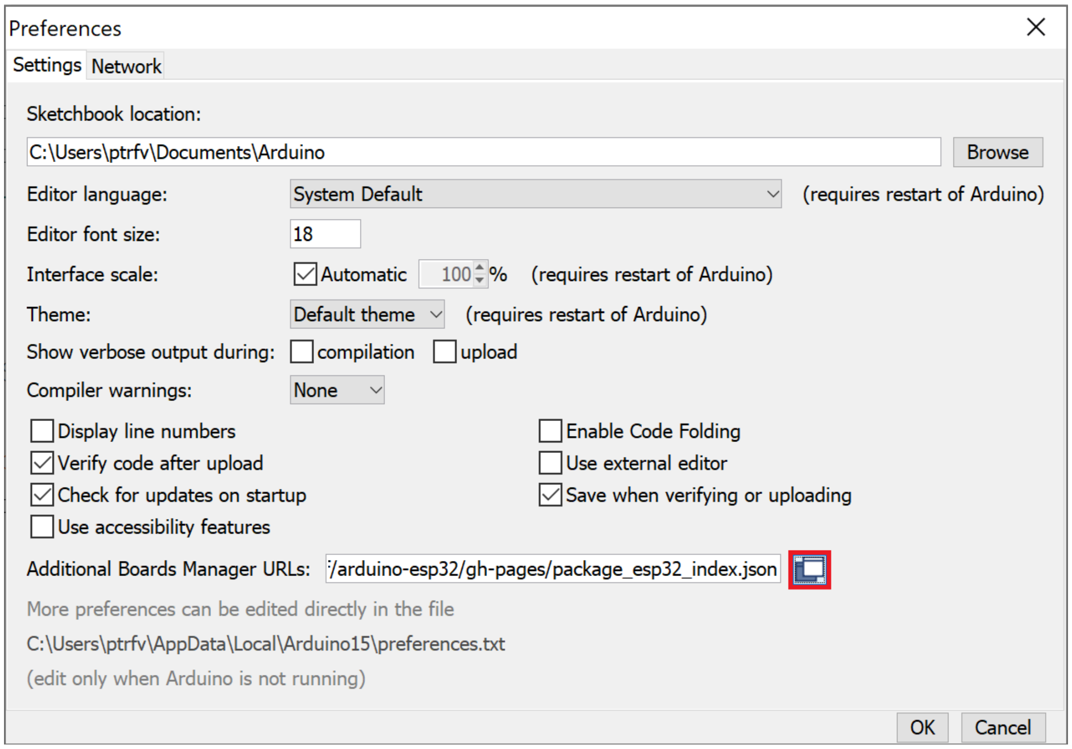

The “daisy chained” sensors.

2.2. Software

I used Arduino IDE v.1.8.19 to program the microcontrollers. This section describes how to set up the IDE with the proper board definitions and libraries.

2.2.1. Board Definitions

In the File > Preferences dialog, click the popout window button next to the Additional Boards Manager URLs textbox:



Sketchbook location: C:\Users\ptrfv\Documents\Arduino
Editor language: System Default (requires restart of Arduino)
Editor font size: 18
Interface scale: [✓ Automatic] 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None
☐ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
✓ Save when verifying or uploading
Additional Boards Manager URLs: /arduino-esp32/gh-pages/package_esp32_index.json
More preferences can be edited directly in the file
C:\Users\ptrfv\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

And add the following lines to the list of URLs:

```
Unset
https://raw.githubusercontent.com/sparkfun/Arduino_Apollo3/main/package_sparkfun_apollo3_index.json
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json
```

Next, go to Tools > Boards > Board Manager.. And search for and install the board definitions. Install the “esp32” definition by Espressif Systems and the “SparkFun Apollo3 Boards” definition by SparkFun Electronics. Install the latest version of the Apollo3 boards definition, but install version 2.0.4 of the ESP32 definition. Later versions of the board definitions cause bootlooping.

After restarting Arduino IDE, verify that the following boards are available in the Tools > Board menu:

- ESP32 Arduino > SparkFun ESP32 MicroMod
- SparkFun Apollo3 > Artemis MicroMod Processor

2.2.2. Arduino Libraries

Install the following Arduino libraries using the Tools > Manage Libraries… dialog:

- SparkFun ADIN1110 Arduino Library
- Adafruit LSM6DS
- Adafruit LIS3MDL
- Adafruit BusIO
- Adafruit Unified Sensor
- SparkFun Qwiic 6Dof - LSM6DS0
- SparkFun MMC5983MA Magnetometer Arduino Library

Note that the name of the SparkFun LSM6DS0 library in the Arduino Library Manager ends with a zero (0).

If you try to compile the code as is, you will receive an error because there is a name conflict for the STATUS_REG constant between the SparkFun LSM6DSO and MMC5983MA libraries.

To resolve this, rename the constant “STATUS_REG” to “LSM6DSO_STATUS_REG” in the LSM6DSO library source. It can be found at the library installation location (...\Documents\Arduino\libraries\SparkFun_Qwiic_6Dof_-_LSM6DS0\src for Windows users). Here, edit two files to rename this constant:

- SparkFunLSM6DSO.cpp: line 565
- SparkFunLSM6DSO.h: lines 282, 1549, 1560, 1571

2.2.3. GitHub Repository

The code used in this project is hosted in a GitHub repository named cubesat-spe. This code consists of two Arduino sketches:

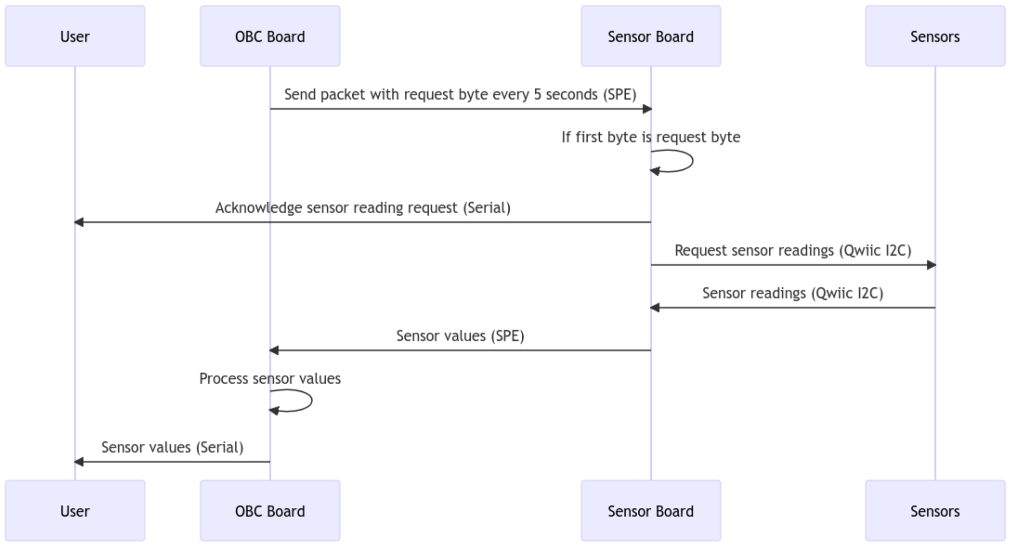
- obc.ino, the sketch for the OBC board.
- sensor.ino, the sketch for the sensor board.

The macros SPARKFUN and ADAFRUIT are defined at the top of each sketch. These are used to direct compilation for the SparkFun or Adafruit sensors respectively. The sketches are written to work with either the SparkFun or Adafruit sensors exclusively. Ensure that the correct macro is enabled for the sensors being used and that the same macro is defined across both sketches.

Here’s what the code does:

1. Every 5 seconds, the OBC board sends a packet with a request byte (0x0A) over the SPE connection to the sensor board. Every other byte in the packet’s payload is 0x00.

2. Upon receiving the packet, the sensor board checks whether the first byte of the payload is the request byte. If it is, and the sensor board is not already sending a sensor reading, it sets a flag to send a sensor reading when it can.
3. In the sensor board's main loop, if a sensor reading has been requested and there is an active link to the OBC board, the sensors attached to the sensor board are polled for their readings. The readings are packed into the payload of a new packet and sent back to the OBC board. The reading request flag is reset to open the sensor board up to another reading request..
4. The OBC board, upon receiving the packet, copies the packet's payload to a local buffer for later processing. It sets a flag indicating that new data has been received from the sensor board.
5. In the OBC board's main loop, if the new data has been received, the local buffer has its contents copied into the local sensor reading variables. These variables are printed to the Serial Console, and the new data received flag is reset.


User
OBC Board
Sensor Board
Sensors
Send packet with request byte every 5 seconds (SPE)
If first byte is request byte
Acknowledge sensor reading request (Serial)
Request sensor readings (Qwiic I2C)
Sensor readings (Qwiic I2C)
Sensor values (SPE)
Process sensor values
Sensor values (Serial)
User
OBC Board
Sensor Board
Sensors
A sequence diagram showing the order of communications in the program.

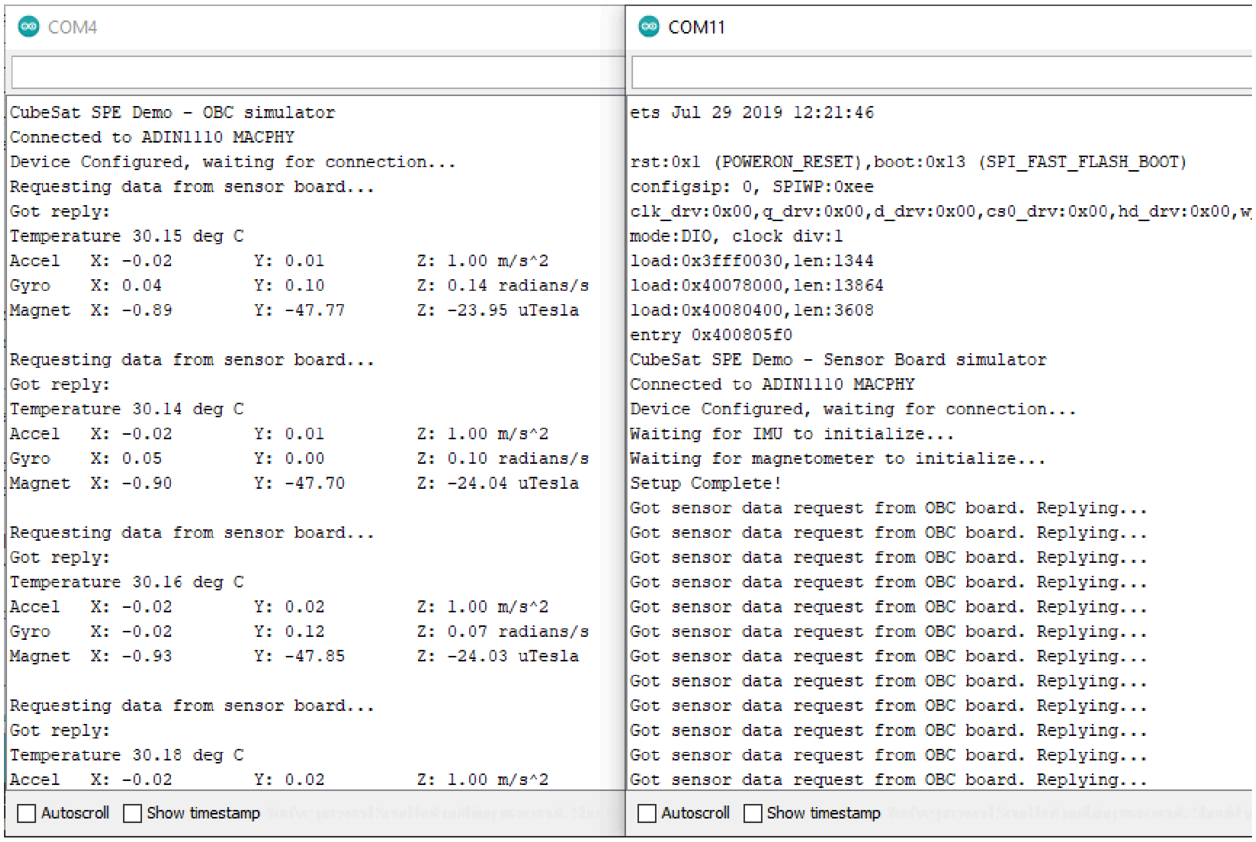
3. Deployment

After ensuring the hardware is set up as described in the hardware setup section, deployment of the code is straightforward:

1. Clone the cubesat-spe repository to your local machine.
2. Upload the obc.ino sketch to the OBC board.
   - In this test, the OBC board was the Artemis microcontroller.
   - Ensure the Artemis MicroMod board is selected in the Tools > Board menu.
3. Upload the sensor.ino sketch to the sensor board.
   - In this test, the sensor board was the ESP32 microcontroller.

    - Ensure the SparkFun ESP32 MicroMod Processor board is selected in the Tools > Board menu.
    - For the ESP32 board, press and hold the BOOT button on the MicroMod Main Board while the Terminal output displays “Connecting…”. Release the button when the code is uploaded (the Terminal displays “Writing at…”).
4. Open Serial Monitor windows for the OBC and sensor boards.
5. Press the RESET buttons on the two MicroMod Main boards simultaneously.
    - This is an optional step to synchronize the two boards.

The Serial Monitors should display output similar to that in the image below:



COM4

```
CubeSat SPE Demo - OBC simulator
Connected to ADIN1110 MACPHY
Device Configured, waiting for connection...
Requesting data from sensor board...
Got reply:
Temperature 30.15 deg C
Accel   X: -0.02        Y: 0.01         Z: 1.00 m/s^2
Gyro    X: 0.04         Y: 0.10         Z: 0.14 radians/s
Magnet  X: -0.89        Y: -47.77       Z: -23.95 uTesla

Requesting data from sensor board...
Got reply:
Temperature 30.14 deg C
Accel   X: -0.02        Y: 0.01         Z: 1.00 m/s^2
Gyro    X: 0.05         Y: 0.00         Z: 0.10 radians/s
Magnet  X: -0.90        Y: -47.70       Z: -24.04 uTesla

Requesting data from sensor board...
Got reply:
Temperature 30.16 deg C
Accel   X: -0.02        Y: 0.02         Z: 1.00 m/s^2
Gyro    X: -0.02        Y: 0.12         Z: 0.07 radians/s
Magnet  X: -0.93        Y: -47.85       Z: -24.03 uTesla

Requesting data from sensor board...
Got reply:
Temperature 30.18 deg C
Accel   X: -0.02        Y: 0.02         Z: 1.00 m/s^2
```

COM11

```
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,w
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13864
load:0x40080400,len:3608
entry 0x400805f0
CubeSat SPE Demo - Sensor Board simulator
Connected to ADIN1110 MACPHY
Device Configured, waiting for connection...
Waiting for IMU to initialize...
Waiting for magnetometer to initialize...
Setup Complete!
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
Got sensor data request from OBC board. Replying...
```

The Serial Monitor output of the sketches.
The OBC board is on the left, and the sensor board is on the right.

4. Evaluation

This evaluation aimed to determine the suitability of the 10BASE-T1L Single Pair Ethernet (SPE) standard for internal communications. While initially focused on CubeSats, this evaluation can be extended to other satellites such as microsats and other spacecraft and lunar surface systems.

This practical demonstration showed that sending sensor data over the SPE interface between an OBC and sensor board is possible. This demonstrated the basic principle of inter-board communication using a single-pair interface. This principle could transfer data between interfaces such as I2C, SPI, UART, and direct digital pins using only a single cable. This is in contrast to the current method of inter-board communication in CubeSats, with a 104-pin PC104 header bus connecting all boards in a single unified bus.

There are tradeoffs between implementing the SPE and PC104 interconnection standards. Though SPE uses fewer conductors, the 10BASE-T1L connector is relatively large (the connector protrudes ~44 mm beyond the end of the socket), and the thick cable has a large bend radius (~2.5cm). Additionally, each board would require an ADIN1110 SPE transceiver and a microcontroller to convert the Ethernet payloads into usable signals for each sensor. This would increase the cost of each board and the system's power consumption overall.

However, there are some specific cases where this system could work well in a CubeSat. For instance, in a 3U (10x10x30cm) CubeSat, a pair of OBC boards on opposite sides of the CubeSat could communicate with each other using the SPE standard. This would allow for redundancy, as the OBC boards could “failover” to each other by sending an error message or failing to send a periodic “all is well” message. Alternatively, the high throughput of the SPE standard (10Mb/s) could be used to interface high-quality sensors (say, high-resolution imaging sensors) to a dedicated “compute board,” bypassing the OBC entirely.

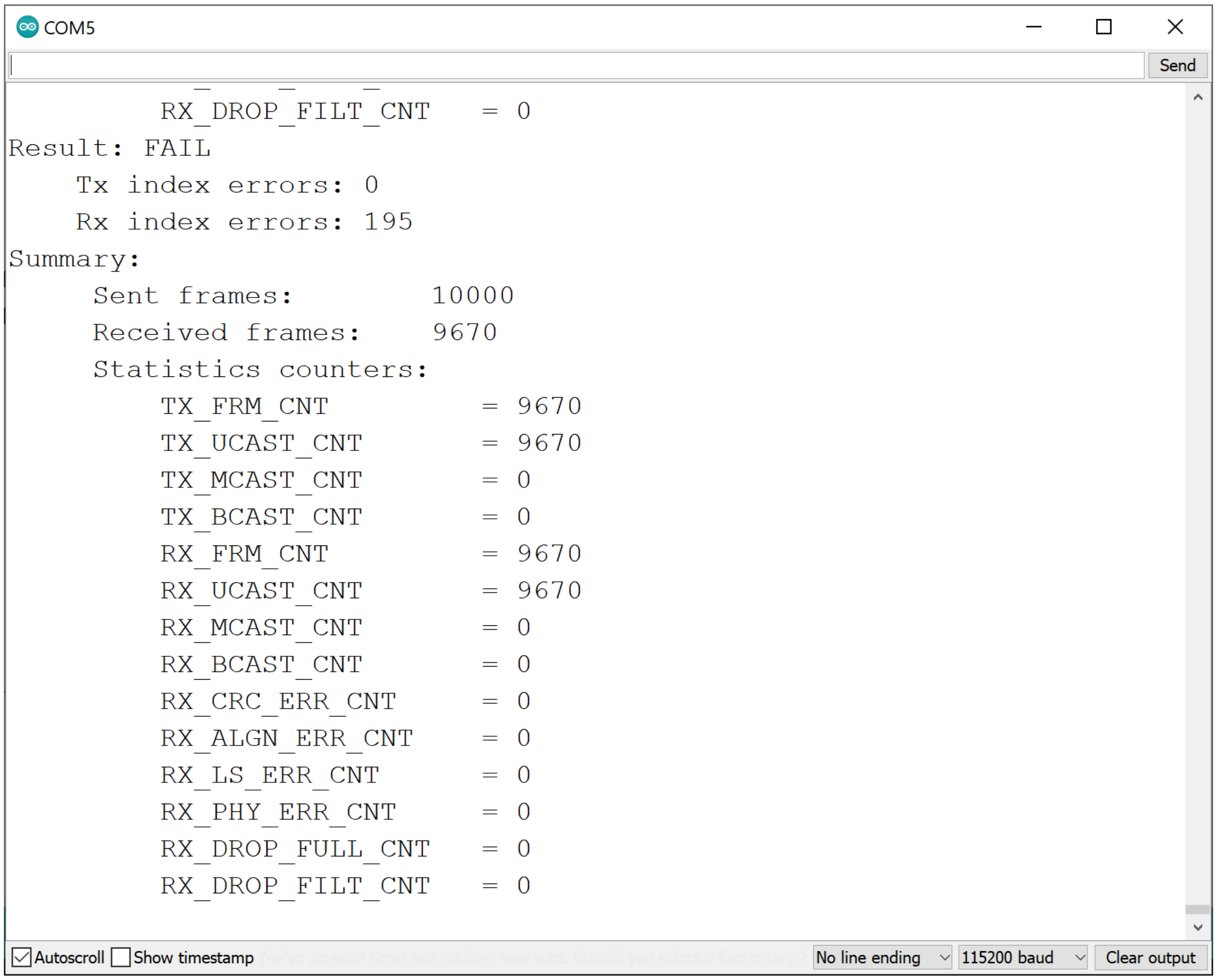
To implement this in a CubeSat, a 90-degree connector would be greatly useful and eliminate the bend radius. With proper strain relief to avoid damage due to vibration, a ~25cm cable could connect distant ends of a 3U CubeSat while remaining within the bounds of the CubeSat’s dimensions.

4.1 Transfer Characteristics

The advantage that SPE provides comes from serialization: instead of having many pins to transfer a signal for a unique purpose, the data that is to be transmitted is sent across one pair of conductors and decoded at the receiving end. To test how well SPE can transmit data, I used the example Arduino sketches Example11a_FrameTrx.ino and Example11b_HardwareLoopback.ino. Combined, these sketches place the SPE system into loopback, with one processor sending Ethernet frames that are sent back to itself. These sketches use the more advanced sfe_spe_advanced class, providing lower-level access to the ADIN1110 and its registers.

After sending 10,000 Ethernet frames, 9,670 were received in reply, indicating 330 frames were lost in transit. In addition, the sketch reported 195 receive index errors, where the received frame’s index was out of sync with the expected index. This indicates synchronization issues, either a dropped or out-of-order frame.

Though 330 dropped frames does sound concerning, this code was run as fast as possible, attempting to overfill the buffers the ADIN1110 uses to store received frames. This represents the worst-case scenario, where one board is flooded with data. Each frame was 1518 bytes in size, far more than typical sensor data frame, in the tens of bytes.



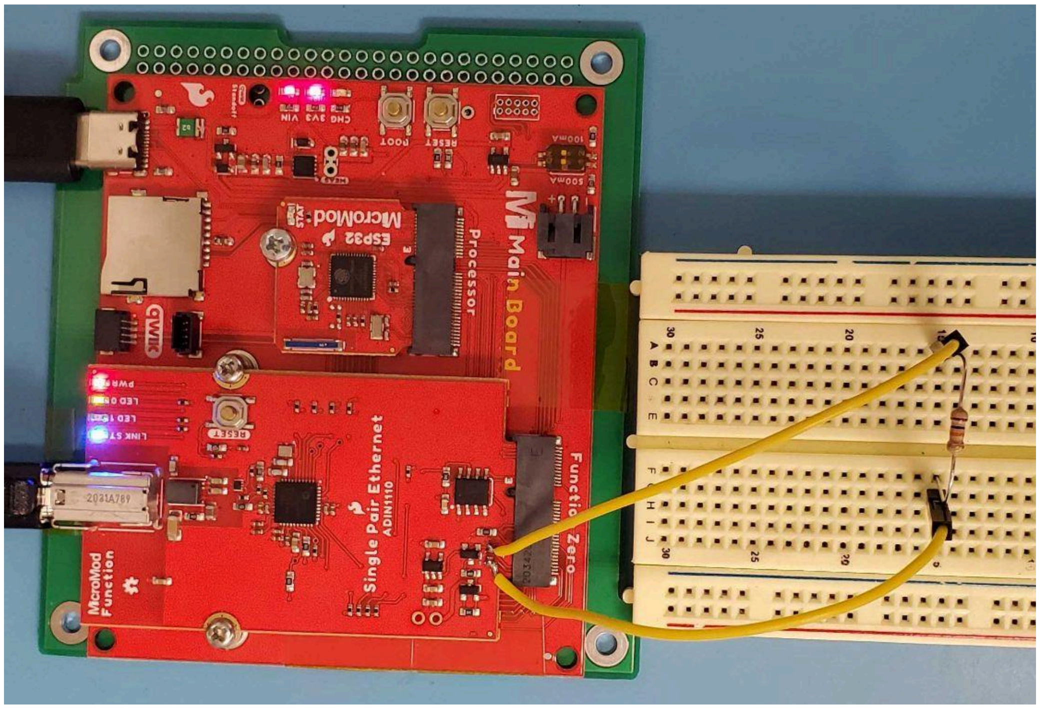
```
          RX_DROP_FILT_CNT    = 0
Result: FAIL
    Tx index errors: 0
    Rx index errors: 195
Summary:
     Sent frames:          10000
     Received frames:      9670
     Statistics counters:
          TX_FRM_CNT            = 9670
          TX_UCAST_CNT          = 9670
          TX_MCAST_CNT          = 0
          TX_BCAST_CNT          = 0
          RX_FRM_CNT            = 9670
          RX_UCAST_CNT          = 9670
          RX_MCAST_CNT          = 0
          RX_BCAST_CNT          = 0
          RX_CRC_ERR_CNT        = 0
          RX_ALGN_ERR_CNT       = 0
          RX_LS_ERR_CNT         = 0
          RX_PHY_ERR_CNT        = 0
          RX_DROP_FULL_CNT      = 0
          RX_DROP_FILT_CNT      = 0
```

☑ Autoscroll ☐ Show timestamp — No line ending — 115200 baud — Clear output

The output of the Example11a_FramTrx.ino sketch.

4.2 Resiliency

Resilience in the event of power loss is critical for space systems. To test how the ADIN1110 would fare in the event of unexpected power loss, I replaced R22 with a pair of Dupont wire leads in line with a through-hole 10kΩ resistor. This allowed me to switch power to the ADIN1110 as needed.



The ADIN1110 Function Board with switched power.

To simulate a loss of power, I removed the resistor, disabling power to the ADIN1110. The Link and PWR LEDs went out, indicating that the link between the boards was interrupted. Replacing the resistor did not restore the link. I then tried different ways of restoring the link:

- Pressing the RESET button on the ADIN1110 Function board sent a reset signal to the ADIN1110. This did not restore the link.
- Pressing the RESET button on the MicroMod Main Board did restore the ADIN1110 link.
- Checking for the link being down and sending a software reset command using reset() did not restore the link.
- Checking for the link being down and reestablishing connection to the ADIN1110 using begin() did not restore the link.

In the event of power loss, resetting the processor attached to the ADIN1110 would appear to be necessary. Further testing is needed to determine the best way to recover from unexpected power loss.

5. Challenges

This project posed some interesting challenges that I had to overcome. These are listed below, along with the solution I found.

- I couldn't upload any code to the ESP32 at first. I searched around and found that the issue has been documented on the GitHub repository for the ESP32 MicroMod processor. The comments of the Hookup Guide described a workaround that I used to successfully upload the sketches to the ESP32: holding down the RESET button when connecting.

- I initially tested the SPE kit using the built-in examples Example01aBasicSendRecieve.ino and Example01b_BasicEcho.ino. Although the example worked fine with the Artemis running Example01a and the ESP32 running Example01b, the opposite did not work. The ESP32 would bootloop, constantly resetting itself. This was due to the interrupt watchdog triggering while the ESP32 was printing the received reply in the rxCallback() function. This is expected behavior for this particular example, as callback functions should execute quickly, and Serial.println() is very slow. I made sure that the callback functions I wrote in obc.ino and sensor.ino would execute as quickly as possible, only setting flags and copying data into buffers for later processing.
- Even when avoiding writing long callback functions, the ESP32 would still bootloop. I searched further through the forums and found that downgrading the ESP32 board definitions from 2.0.11 to 2.0.4 would avoid this bootlooping issue. I made sure to note this in the software setup instructions.
- The SparkFun Qwiic 6Dof - LSM6DSO library ends with a zero (0) rather than a letter 'O'. It seems this had been fixed in the library.properties file of the repo, but the changes haven't propagated to the Arduino Library Manager. While the name of the library was changed correctly, the URL still has the zero at the end. The Arduino Library Manager list requires manual intervention to update the URL of a library already in the list, so even if the URL is corrected on the library's end, the list also needs to be updated to reflect this change. I was able to catch this error and install the library through the Arduino Library Manager.
- The SparkFun MMC5983MA Magnetometer Arduino Library and SparkFun Qwiic 6Dof - LSM6DSO library both define a constant, STATUS_REG. I'm not sure what the best policy would be to avoid name conflicts on a generic name. I solved it by prepending the sensor name to SENSOR_REG for only the LSM6DSO library, as it had the fewest references to STATUS_REG.
- Ke Ao uses a Teensy 4.1 as part of its OBC. I also ordered and received a pair of SparkFun MicroMod Teensy Processors, to evaluate the ADIN1110 on Teensy. However, no code using the ADIN1110, including the BasicEcho examples, would compile for Teensy. I found through the forums this is because the pins for the Teensy MicroMod were not defined in the ADIN1110 library. I chose to stick with the Artemis and ESP32 boards for this evaluation as a result. I did find a pull request adding STM32 MicroMod support that provides a good example for adding board compatibility to the ADIN1110 library at a later date.
- The initial version of the OBC.ino sketch would send a request packet to the sensor board with only one byte in its payload. This would fail, and through further testing I found that the sendData() function had a minimum payload size of 46 bytes. Looking at the source, the sendData() function should automatically pad any payload that is less than 46 bytes with zeroes. I'm not sure why this isn't happening, but I worked around this by manually padding the payload before passing it to the sendData() function.
- I noticed that, after sending the first sensor data, the sensor board would not send any further data. I traced the problem to the sendData() function. After it executes, the sensor

board's MAC is overwritten to be the OBC board's MAC. I also don't know why this is happening, but I corrected this by manually restoring the sensor board's MAC after calling the sendData() function.

6. Future Work

There is much more work to be done with this communications system. I believe this could be adapted to work within the limited size of CubeSats, and provide high-speed internal communications.

- The transfer characteristics and resiliency tests done in this initial project were an initial test of the ADIN1110's capabilities. Further testing in throughput, buffer capacity, and power loss recovery are necessary.
- The 10BASE-T1L connector and footprint are rugged, but not sufficiently compact for our needs. A next step would be to evaluate using headers as the communications medium, rather than a shielded twisted-pair wire. The risk of interference due to lack of shielding is increased, but the short distance between boards may mitigate this.
- The MicroMod boards we currently have do not support Power over Ethernet, though the 10BASE-T1L standard does. A further avenue of study would be to implement power and data over a single two-conductor bus. This would keep the connections between boards to the absolute minimum.
- A breakout board for the ADIN1110, along with a low-power microcontroller, would be the next step in testing the SPE standard on Ke Ao. This could be a module, connected to two pins on the current PC104 bus, that would act as an Ethernet interface.

7. Conclusion

In conclusion, we successfully demonstrated using 10BASE-T1L Single Pair Ethernet (SPE) networking for transmitting sensor data across boards in CubeSats. We evaluated the practicality of implementing this standard in CubeSats and found that it has great potential for weight and size savings.

Looking forward, the benefits of using SPE for single-string architectures common in small (<12U) CubeSats don't outweigh the power and cost drawbacks, particularly when using Commercial Off-The-Shelf (COTS) CubeSat hardware. However, the SPE standard could be more beneficial in the dual- or triple-string architectures possible in 16U microsats and larger space systems. For example, microsats typically use 4-wire point-to-point UART interfaces, making sharing the data between multiple OBCs difficult.

Using routable Ethernet packets that can be replicated or rerouted, SPE could enable more robust architectures by using automotive Ethernet switches as the data/control plane. To support these future architectural benefits, flying SPE in a small CubeSat as a technology pathfinder could demonstrate the readiness of compatible hardware and software.

8. Acknowledgments

I would like to thank Dr. Miguel Nunes for discovering this opportunity and Amber Imai-Hong for logistical support on this project. Additionally, Douglas Hill provided invaluable feedback and advice on this report.