2nd Generation Low Power Environmental Sensor Suite

ECE 413 Capstone Project Final Report

Spring 2022

Industry Sponsor: ECE Department - Dr. David Burnett

Team 13: Wei Yan, Eyal Eynis, Travis Johnson, Wenyu Bi

Faculty Advisor: Dr. John Acken

June 10, 2022

Version 2.0

Table of Contents

Executive summary

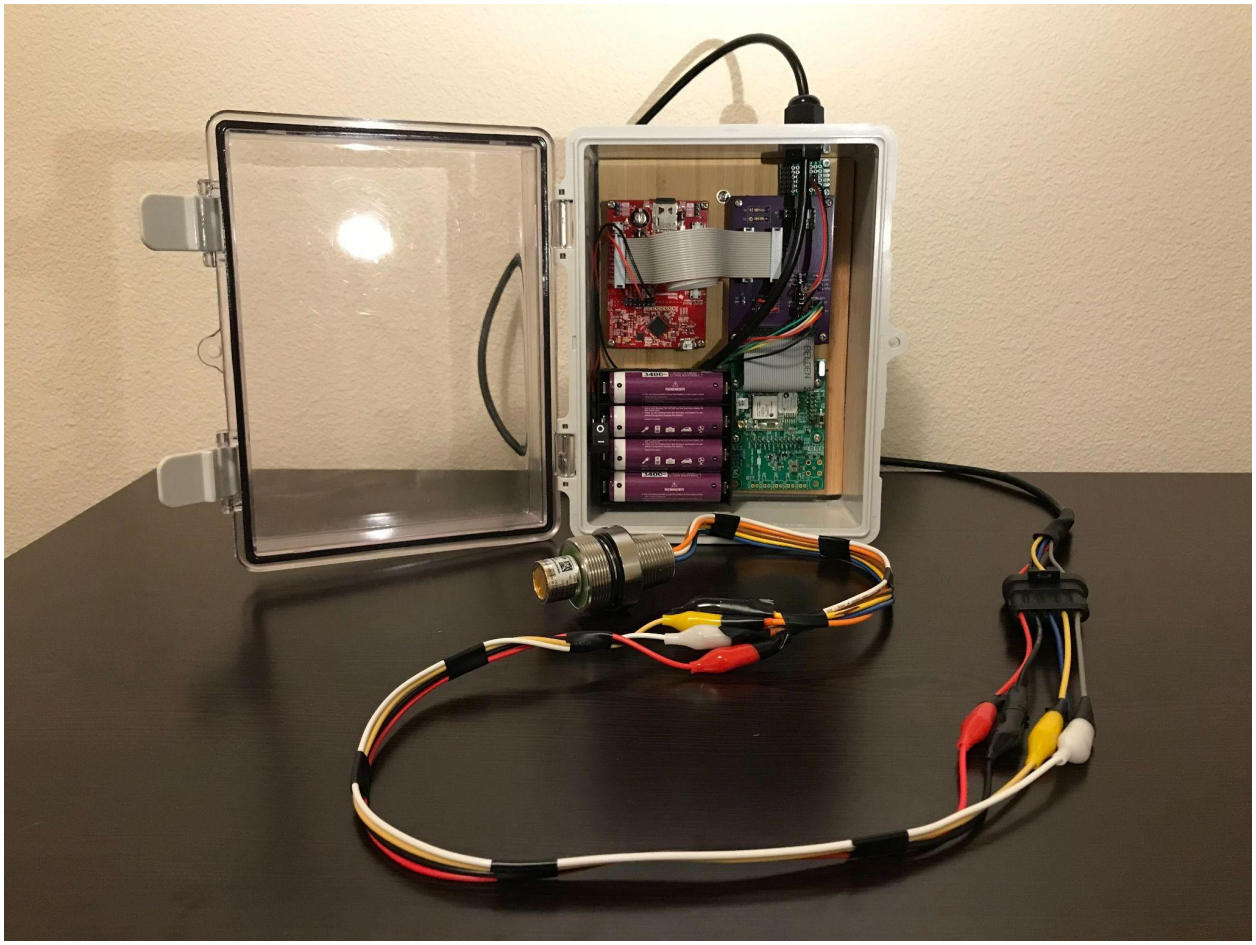


Image: Low Power Environmental Sensor Suite hardware and enclosure:

The Low Power Sensor Suite is a network of low power, wireless motes that measure the environmental data within the network, and displays this data to the user in a meaningful way. Our team was responsible for building a single sensor mote, that is fully functioning, and field deployable enclosed in a weather resistant IP67 enclosure, pictured above. The sensor suite will collect sensor data from

temperature, humidity, and N2O sensors, and wirelessly transmit this data to a remote laptop (base station), as well as save sensor data on an on-board SD Card.

Using SmartMesh IP technology, potentially hundreds or thousands of motes can be built and deployed in a mesh network covering a large geographical region, collecting relaying sensor data. Each mote is solely powered by batteries, and in the current configuration can run for approximately 2.5 months from 4 fully charged batteries. This system has the potential to be reconfigured with less power intensive sensors, and could run for over a year or more.

The main purpose of this device is to measure the N2O emission caused by the out-gassing of radioactive decay to be deployed by the Department of Energy. The end user will manage the Sensor Suite through setup, maintenance, and data interpretation. Management entails the deployment of new motes and a base station, exchanging and recharging of spent batteries, and responding to environmental changes based on data presented to the user via a visual dashboard.

This report details the background of this project, and what our team did to improve on last year's design. It includes full instructions and design files on how to built the mote, where to purchase all the components, set up all the required software, and how to program the firmware.

Problem Background

The main problem faced currently is that gas sensors consume a lot of power. This makes it really difficult to deploy a device that can measure N2O for a long period of time that is powered by batteries. N2O or nitrous oxide can damage the ozone layer, which humans rely on to prevent most of the sun's ultraviolet radiation from reaching earth's surface. N2O is 300 times more potent than CO2, and it has a shorter life span. Identifying N2O emissions early on could help us take measures to prevent it and somehow prevent it from going into the ozone layer. The emission of greenhouse gasses has increased year over year since 1850. The demand for equipment that is able to identify these greenhouse gasses emissions and be power friendly in order to be deployable is increasing. This is because greenhouse gasses emissions into our atmosphere are the main cause for global warming.

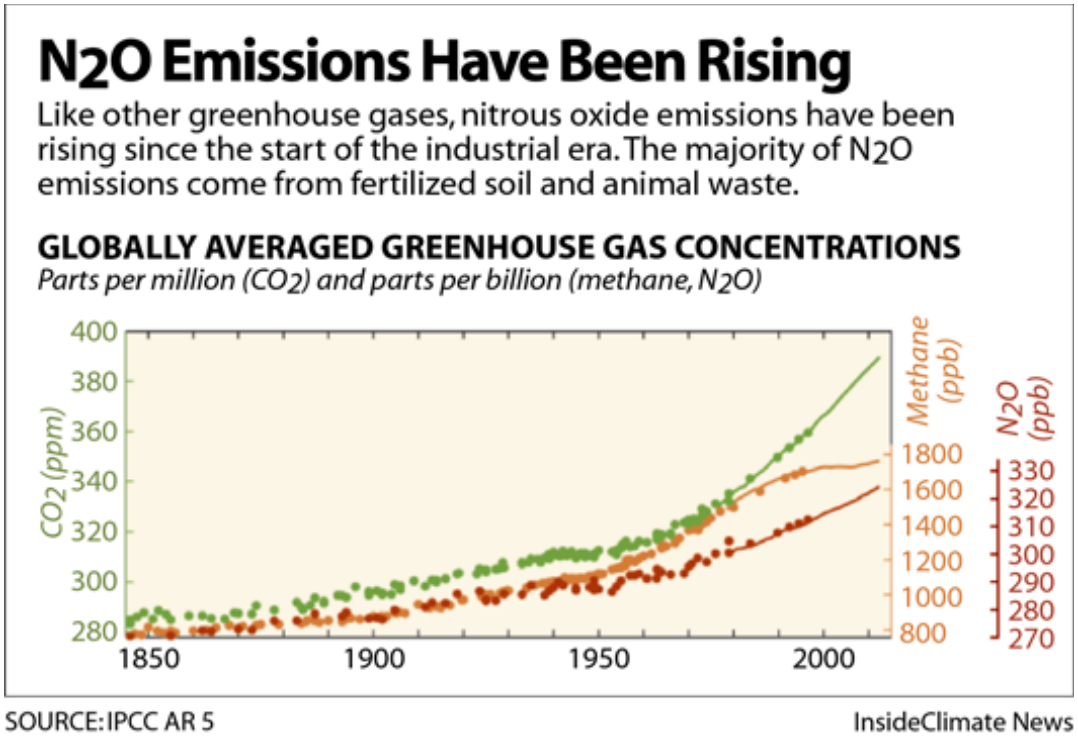


Figure 1: Greenhouse gas emissions over the years

Need statement

There is a need for low power systems that are able to support these high power gas sensors. N20 sensors are not power friendly sensors and therefore not easily deployable out in an agricultural field or nuclear waste storage facility, where an external source power is not available. In order for this system to be deployable in the field it needs to have a wireless communication system and the ability to enter low power modes of operation. These low power states are critical because we want to regulate the power consumed of the entire system. The power of the

entire system is regulated by the MSP-EXP430FR5994 and this power needs to remain the lowest while the sensors are not active.

Objective statement

Creating a low power, low cost data sampling system that allows large geographical areas to be measured with low maintenance and low deployment cost is our objective. Smart mesh IP is an automated mesh networking stack that uses the LTP modules to create a data transmission network with no need for specialized infrastructure. The lack of need for networking infrastructure and the low power performance of the LTP modules drives down the deployment and maintenance cost and creates a compelling system that can address the agricultural industries needs, and makes detection of greenhouse gas emissions possible.

Deliverables

Hardware:

- Prototype Mote with sensors, wireless transmission, microcontroller with on board SD card storage, and PCB junction board.
- Weather resistant IP67 housing

Software:

- Frontend software and GUI: software that will manage as well as receive and store data from a mesh network. Moreover, a user interface that will display collected data as well as allowing the user to observe mote status.

Documentation:

- Project Proposal
- Final Report
- Compiled Weekly Progress Reports
- Bill of Materials
- System schematic & functional diagram
- Operations Guide
- Design files for any custom mechanical components or PCBs
- ECE Capstone Poster Session poster

Requirements

Must

- The microcontroller must be in a low power state when not in use.
- The entire system must consume less average power compared to last year's system.
- The system must be converted into a PCB assembly.
- The system must include at least 3 field monitoring sensors.
- The system must be placed in an outdoor-fieldable enclosure.

Should

- The environmental sensor suite should use rechargeable batteries.
- The environmental sensor suite should have a greater lifetime than last year's system.

May

- The environmental sensor suite may have forms of power scavenging such as solar power and/or wind power.
- The outdoor-fieldable enclosure for the system may be waterproof.
- Every sensor may have a programmable sampling rate by the user.

Design

The system architecture has three core modules: sensor mote - the collection of data via sensors, mesh network - the transportation of the data via mesh network, and basestation - the presentation & storage of this data via the front-end software. To achieve this we are using a variety of off-the-shelf sensors combined with the MSP-EXP430FR5994 microcontroller, the mesh network is implemented using off-the shelf smart mesh IP evaluation motes that we integrate to work with our microcontroller. For the Base station we built on last year capstone's Python applications which are used for storing sensor data and for displaying stored data on a GUI.

The sensor motes and the mesh network have to balance power consumption and compatibility. We have two possible architectures for the mote, one is where the sensors feed directly into the wireless module (master mode) and the other is that the sensors feed into a separate microcontroller that then feeds into the wireless module (slave mode). This architecture is largely possible because of the Smartmesh networking system which automates and simplifies the setup and maintenance of a mesh network, using their low power LTP wireless modules. We decided to go with the slave mode option because the MSP-EXP430FR5994 gives us more low power options that we'll need to minimize power consumption of the system. More details about master and slaves modes can be found here: https://www.analog.com/media/en/technical-documentation/user-guides/SmartMesh_IP_Tools_Guide.pdf

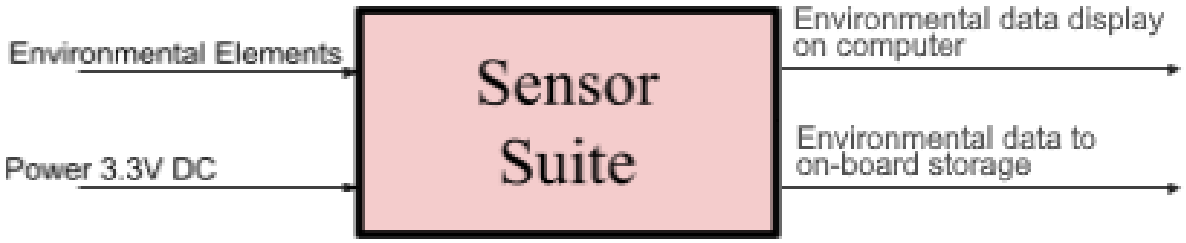


Figure 2: Level 0 Block Diagram

Module	Sensor Suite
Inputs	Power: 3.3V DC Environmental Elements: Measured by various sensors. Temperature, -40˚C to 85˚ C Humidity, 10% to 90% RH N20, 0 - 1000 ppm
Outputs	Environmental Data Display: Graphical representation, SDCard data file
Functionality	Displays environmental data based on measurements sampled from sensors.

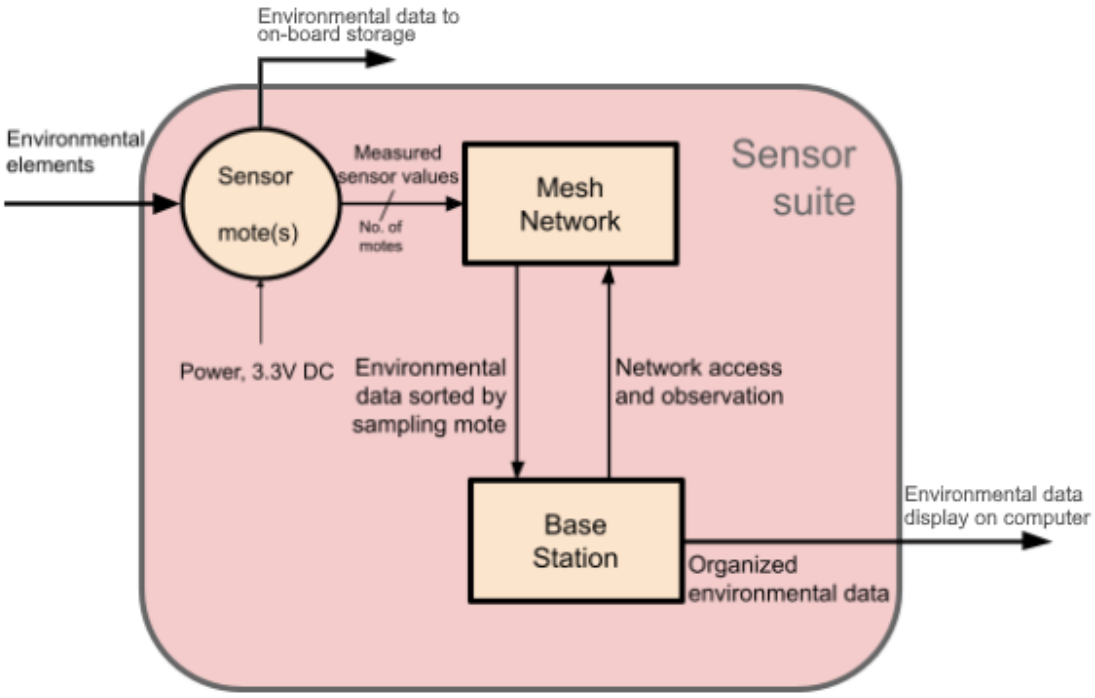


Figure 3: Level 1 Block Diagram

Approach

This section details what our team received from the previous capstone team, what our contribution to the project, and how we approached the project and each section. In each section, we include possible developments future teams can make.

Background

The previous Capstone team that created the First Generation Low Power Environmental Sensor Suite was able to produce a workable prototype built on a breadboard and PCB shield to connect the various components. They wrote the code for numerous sensors to work on a TI CC3200 microcontroller, the code to connect the SmartMesh systems, and created a GUI.

They recommended using a different microcontroller since the CC3200 was unable to output the correct voltage to power certain sensors. Their prototype was built on a breadboard, but when we received it, we were unable to get it working. The GUI code had very few comments, and had many bugs that needed to be fixed.

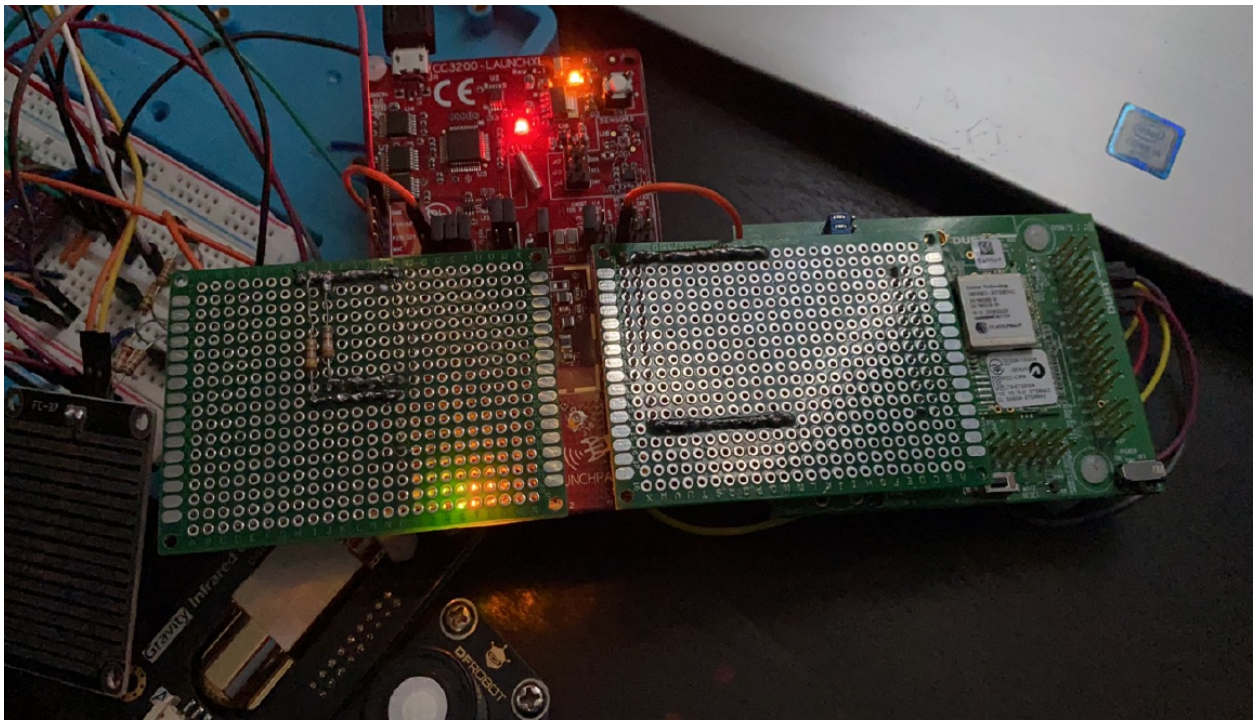


Image: Previous Capstone Team's Sensor Suite prototype

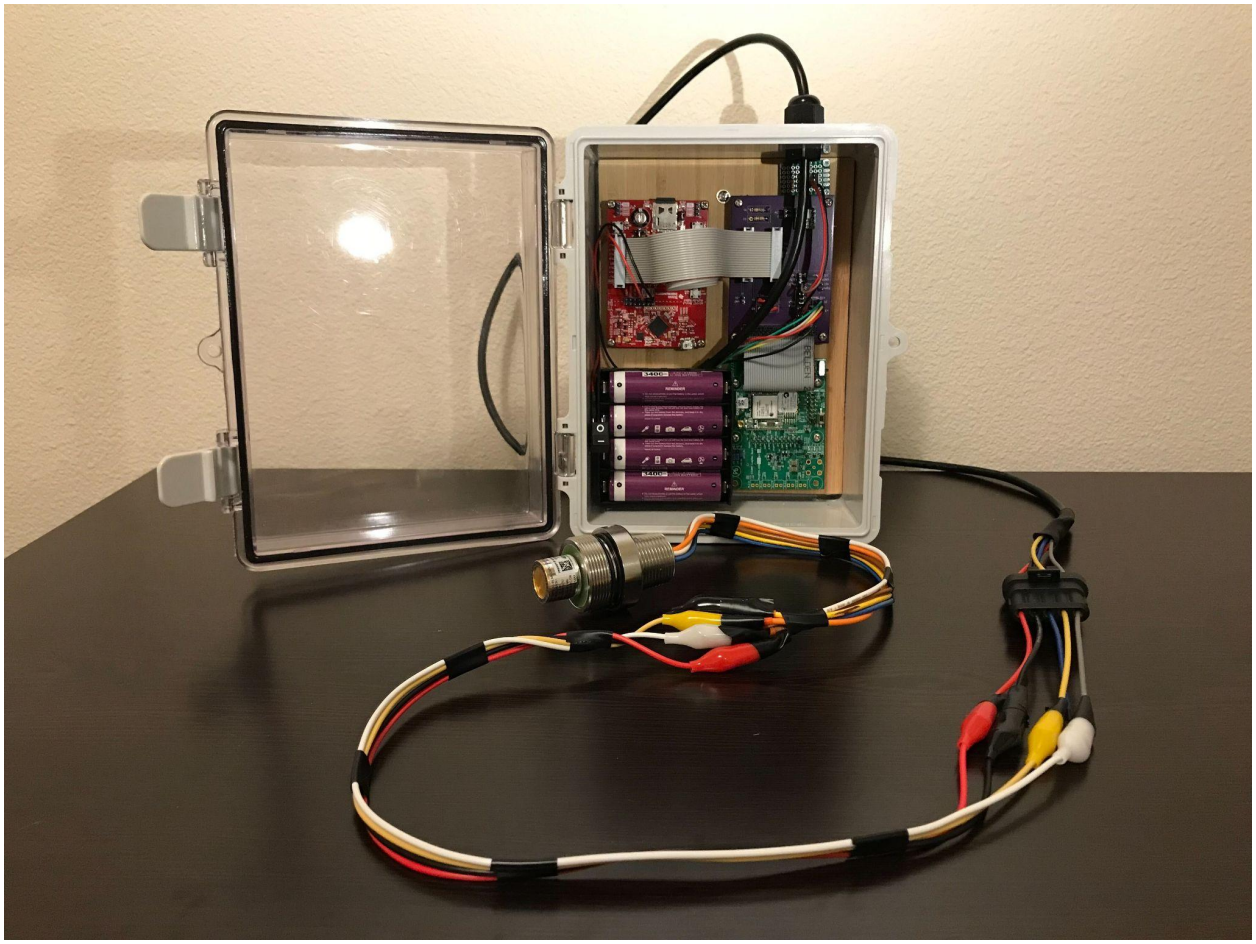


Image: Result of this year's Capstone project: Fully functioning and deployable sensor suite.

Microcontroller

Our first major decision was whether to continue with the CC3200 or choose a different microcontroller. We decided to take the previous team's recommendation, and decided to not use the CC3200, and chose the MSPEXP430FR5994. This microcontroller is designed to enter into very low power modes, uses a similar code base as the CC3200, and had a built-in SD Card reader.

We created new code for the network of sensors and to connect to the SmartMesh, which all worked very well. This was a good decision to use the MSP430.

The MSP430 can interface with other MSP430 boards to use the EnergyTrace function to measure the energy usage of the system.

Energy Trace measurements

We found out it is possible to measure power consumption of the entire system using a MSPEXP430FR2355. We used the interaction from this microcontroller to the one we used in our system (MSPEXP430FR5994). This allowed us to look at the entire power consumption of the system.

The thing we didn't explore as much is using MSPEXP430FR5994 to run energy trace on MSPEXP430FR5994. This will allow us to use EnergyTrace++ instead of regular EnergyTrace. EnergyTrace++ gives in high detail the power consumption breakdown of each peripheral and much more. This would allowed us to provide better power consumption measurements.

SmartMesh Code

For the base station we used a python application called SensorDataReceiver.py which collects incoming sensor data from mote and stores the sensor data in a PC log file. Last year's capstone developed this application but several syntax errors

had to be fixed and some of the code had to be adjusted to meet the needs of our project. Python 3.10.1 was used for developing this code.

SD Card

Since we were using Energia IDE, we had to find an SD Card library that would work with this IDE. Although TI provides SD Card code in their "Out of the Box Experience" that works with Code Composer Studio, this code is jumbled inside a larger program making it difficult to parse out. Also, they do not provide any library compatible with Energia IDE. This makes utilizing the SD Card very difficult to use on this launchpad.

We eventually found an SD Card library on the 43oh.com forums. This library needed modification before it functioned properly. Please see Appendix B for more details how we got the SD Card to function.

Our SD Card only logged the millis() return value, which returns that time since the system was powered on. Further, the SD Card files do not have correct time stamps when saved. Future teams may want to obtain precise date and time stamps for the logs, and for the files created on the SD Card.

Front-End Software

Last year's capstone was able to develop a GUI that allowed data to be viewed in a meaningful way, through a table and a graph. However, launching the GUI in the beginning of the project did not work. Syntax errors had to be fixed and additional

libraries had to be imported into the code in order for the GUI to work on Python 3.10.1. After these implementations, the GUI was able to be launched but there was plenty of work that needed to be done. First, the table in the GUI did not update as new data arrived. Secondly, the graph did not display data correctly. Thirdly, some buttons in the GUI did not function at all. These are some of the issues that we encountered with the previous code. The team was able to simplify the GUI to show the important details (table showing latest 100 pieces of data, graph showing latest 100 pieces of data), reduce non-functional buttons, and most importantly get the GUI to a stable state during operation.

Junction Board

The previous Capstone team was not able to produce a PCB to connect the components, but had used a breadboard and a shield. Our team designed and built up a junction board as the main interface between the MSP430, Smartmesh 9018B and the sensors.

The design of this junction board is to allow full connection to all of the pins of the MSP430, so that many types of sensors can be added. Future teams can easily add traces and pin headers to the junction board so connect sensors such as O2, CO2, light intensity, etc.

The 2 row pin headers, for example, labelled J70/J71, are debug headers. The purpose is to easily add circuitry if needed. This occurred when we needed to add a transistor to the N2O sensor. If no added circuitry is needed, just place a jumper

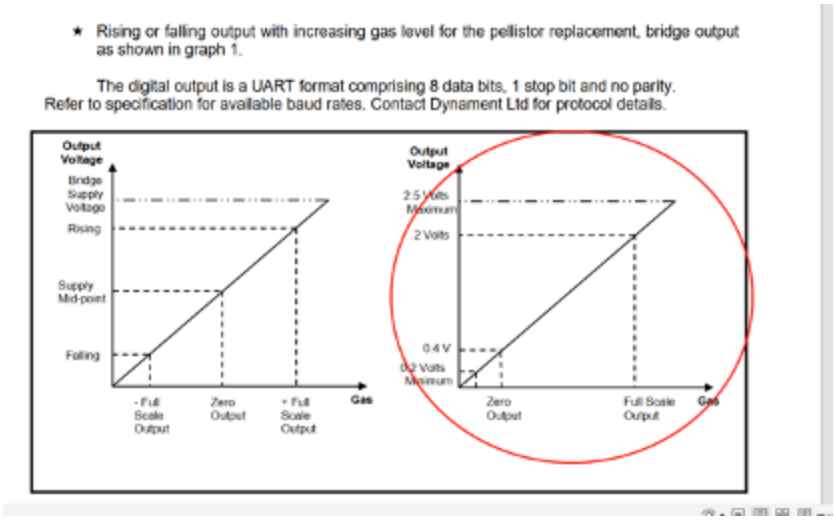
shunt to connect the traces. These debug headers could be eliminated from the design of the PCB once a proven design is finalized.

Sensors

We needed to include temperature, humidity and N2O sensors in this system. The temperature and humidity sensor code worked fairly simply and did not post much difficulty. The code worked for both the HIH8120 and the HIH6120. The HIH6120 was used because most suppliers were out of stock for the HIH8120.

The N2O sensor code was more difficult to program because the default baud rate of N2O is 38400, so the hardware serial port must be used. However, the hardware serial port of MSP430FR5994 is occupied by smartmesh. So the root of the problem is to solve the problem of serial communication.

- **The basic working principle of N2O sensor:**
    - In the N2O datasheet, we can see that when the voltage is lower than 0.4V, it is the preheating stage, and the 2V is capped. The linear change in the voltage represents the change in the concentration of the gas. The warm-up time is shown in the picture, about 45 seconds.



Note: a zero calibration must always be carried out before a span calibration.

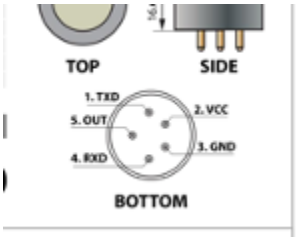
Sensor warm-up time

When power is first applied to the sensor, the voltage at the output pin is held at a pre-determined level. The default setting for this start-up value is the "zero gas" value. This condition is maintained for a default "warm-up" time of 45 seconds, after this time the output voltage represents the calculated gas value. Sensors can take up to 1 minute to indicate the correct gas reading.

Note: the sensor can calculate any reading from -100% FSD to +200% FSD in the first minute. The output value that is read using the communications pins is always held at -250% FSD during the "warm-up" time.

Both the voltage at the output pin during the "warm-up" time, and the duration of the "warm-up" time can be pre-programmed to alternative values at the time of ordering sensors.

- The Dynament TDS0045 N2O sensor comes in two versions, a 3-pin version and a 5-pin version, we are using the 5-pin positive version. Its pins are TXD, RXD, VCC, GND, OUT. We did not use OUT because we are going to use serial communication to obtain sensor data.



- We need to send a command to the sensor requesting data so that the sensor can send gas data back. And, according to the content of the document below, the data sent by the sensor is a 15-bit floating point

number, and we only need the 8-11 bits of data because they represent the N2O gas content level.

Send the following bytes:

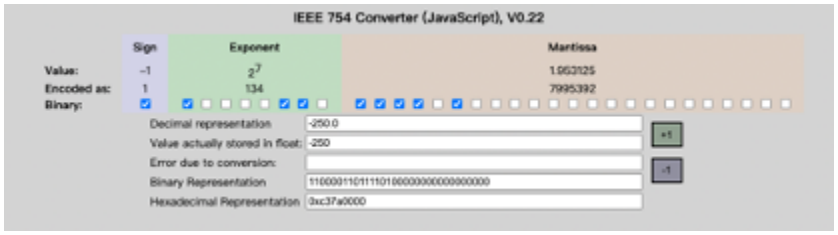
DLE, RD, Variable ID, DLE, EOF, Checksum High byte, Checksum low byte or CRC High byte, CRC low byte, i.e. 0x10, 0x13, 0x06, 0x10, 0x1F, 0x00, 0x58 0x9B, 0xBF

Packet	Data	Command
1	10	DLE
2	1A	DAT
3	8	Data Length
4	4	Version
5	0	Version
6	4	Status Flags
7	0	Status Flags
8	0	Gas Reading
9	0	Gas Reading
10	7A	Gas Reading
11	C3	Gas Reading
12	10	DLE
13	1F	EOF
14	1	Checksum High
15	A6	Checksum Low

Figure 11: Incoming Packet Description

Please note the Gas reading is a decimal not an integer. This decimal is in IEEE-754 format, you can use an online converter like this to convert it. The gas value in this case shows -250 (as it was in error mode at the time).

- The data we get is in IEEE-754 format, and we can convert the result to gas data through https://www.h-schmidt.net/FloatConverter/IEEE754.html (note the input data from #11 to #8) .



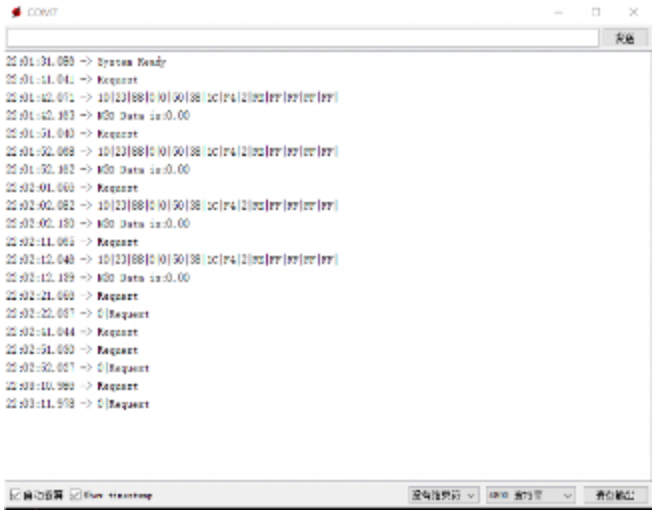
- **Problems encountered during testing:** Since the hardware serial port is occupied, try to use other serial ports to make the sensor working.
  - We connect the software serial port to RX and TX, VCC to 3.3V, and GND to GND. But after running the code, we don't get any data from

the sensor. This is because the baud rate of the sensor's default transmission data is 38400, but the software serial port does not support such a high baud rate for communication, so it fails.

- In order to check whether there is a code error, we performed the same test on the native hardware serial port (P6.0&P6.1) of the MSP430, and the result was that the sensor was working normally, so the above reason is indeed caused by the inapplicability of the software serial port.
- By reading the datasheet of the N2O sensor, we found that the sensor supports a lower baud rate to transmit data. This time we provide another idea, that is, to change the baud rate of the sensor so that the software serial port can be used to obtain data.

Temperature performance over the range -20°C to +50°C (-4°F to 122°F):	± 10% of reading up to 50% full scale and ± 15% of reading from 50% to 100% full scale
Storage temperature range:	-20°C to +50°C (-4°F to 122°F)
Humidity range:	0 to 95% RH non-condensing.
Digital signal format:	8 data bits, 1 stop bit, no parity, 2.6V logic level
Standard baud rates:	38,400, 19,200, 9600, 4800
User configurable parameters and functions:	Zero output voltage Full-scale output voltage Positive or negative going output Sensor 'zero' function Sensor 'span' function Over range value
MTBF:	> 5 years
Weight:	15 grams

After using the configuration software of the N2O sensor to change its baud rate (to 4800), the sensor can communicate through the software serial port, but it is very unstable. As shown in the screenshot below, the data reception is not stable. Several sets of data after the test are displayed as FF, and the data acquisition will fail later.



This is due to the unstable data transmission caused by the baud rate being too low. However, the software serial port usually does not support baud rates over 9600, so this attempt also failed.

- After reading the datasheet of MSP430FR5994, we found that there are other hardware serial ports in the chip, so we want to try to configure the software serial port as a hardware serial port for the N2O sensor.

We try to configure P2.5 and P2.6 as hardware serial ports in the MSP430 library (see refer to **User's Manual (Programming) 3.** for details).

- **Result:** After successfully configuring these two serial ports, the sensor can finally work normally.

N2O Cable

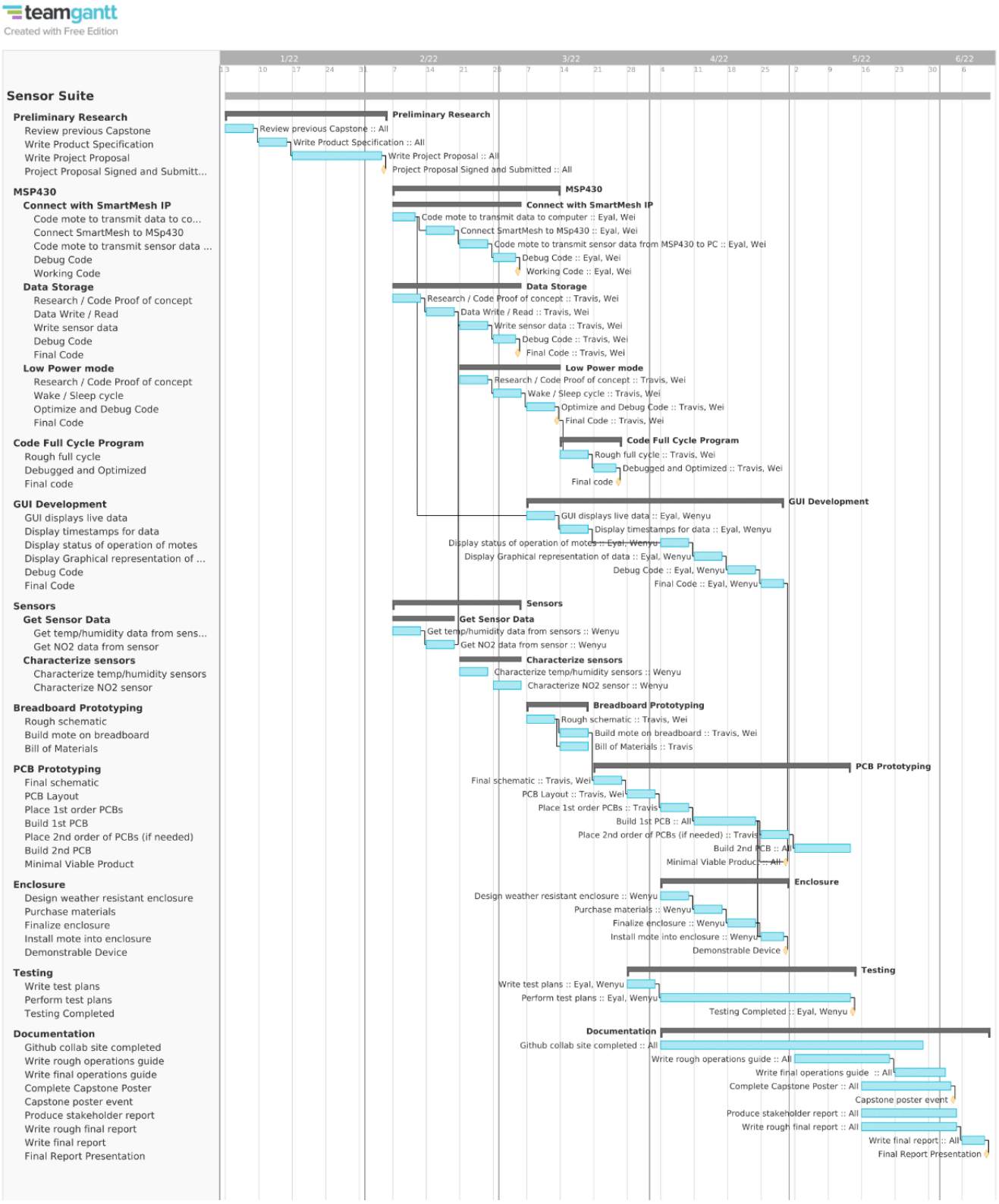
The N2O Cable allows the N2O sensor to take measurements external to the enclosure. The cable passed through the enclosure via a 0.25 inch waterproof cable gland. The cable has an outer diameter of 0.25 inch. A waterproof automotive connector was placed inline to easily connect and disconnect the sensor.

The terminal end inside the enclosure was soldered to common jumper wires. This was not the first approach, but a last minute necessity. The initial design idea was to connect the cable to the junction board using dupont style crimp connections. I was unable to get a good crimp using the tools on hand, and due to time constraints, this option was scrapped for something easy but crude.

Future teams have many options to construct a waterproof cable with strong and stable connectors on both ends, whether it is dupont or molex connectors, or something else. One idea could be to have a waterproof connection mounted to the enclosure itself.

GANTT Chart

Below is a copy of the GANTT chart that was created at the beginning of the Capstone project. Every aspect of the project was broken down into section-based milestones, which were further broken down into weekly objectives. Although team members were initially assigned to certain tasks, as shown in the GANTT chart, in most cases, each section was worked on by a different team member. These changes occurred organically as we began to understand each other's strengths and by what was needed at the time.


teamgantt
Created with Free Edition
1/22
2/22
3/22
4/22
5/22
6/22
Sensor Suite
Preliminary Research
Review previous Capstone
Write Product Specification
Write Project Proposal
Project Proposal Signed and Submitt...
MSP430
Connect with SmartMesh IP
Code mote to transmit data to co...
Connect SmartMesh to MSp430
Code mote to transmit sensor data ...
Debug Code
Working Code
Data Storage
Research / Code Proof of concept
Data Write / Read
Write sensor data
Debug Code
Final Code
Low Power mode
Research / Code Proof of concept
Wake / Sleep cycle
Optimize and Debug Code
Final Code
Code Full Cycle Program
Rough full cycle
Debugged and Optimized
Final code
GUI Development
GUI displays live data
Display timestamps for data
Display status of operation of motes
Display Graphical representation of ...
Debug Code
Final Code
Sensors
Get Sensor Data
Get temp/humidity data from sens...
Get NO2 data from sensor
Characterize sensors
Characterize temp/humidity sensors
Characterize NO2 sensor
Breadboard Prototyping
Rough schematic
Build mote on breadboard
Bill of Materials
PCB Prototyping
Final schematic
PCB Layout
Place 1st order PCBs
Build 1st PCB
Place 2nd order of PCBs (if needed)
Build 2nd PCB
Minimal Viable Product
Enclosure
Design weather resistant enclosure
Purchase materials
Finalize enclosure
Install mote into enclosure
Demonstrable Device
Testing
Write test plans
Perform test plans
Testing Completed
Documentation
Github collab site completed
Write rough operations guide
Write final operations guide
Complete Capstone Poster
Capstone poster event
Produce stakeholder report
Write rough final report
Write final report
Final Report Presentation
Preliminary Research
Review previous Capstone :: All
Write Product Specification :: All
Write Project Proposal :: All
Project Proposal Signed and Submitted :: All
MSP430
Connect with SmartMesh IP
Code mote to transmit data to computer :: Eyal, Wei
Connect SmartMesh to MSp430 :: Eyal, Wei
Code mote to transmit sensor data from MSP430 to PC :: Eyal, Wei
Debug Code :: Eyal, Wei
Working Code :: Eyal, Wei
Data Storage
Research / Code Proof of concept :: Travis, Wei
Data Write / Read :: Travis, Wei
Write sensor data :: Travis, Wei
Debug Code :: Travis, Wei
Final Code :: Travis, Wei
Low Power mode
Research / Code Proof of concept :: Travis, Wei
Wake / Sleep cycle :: Travis, Wei
Optimize and Debug Code :: Travis, Wei
Final Code :: Travis, Wei
Code Full Cycle Program
Rough full cycle :: Travis, Wei
Debugged and Optimized :: Travis, Wei
Final code
GUI Development
GUI displays live data :: Eyal, Wenyu
Display timestamps for data :: Eyal, Wenyu
Display status of operation of motes :: Eyal, Wenyu
Display Graphical representation of data :: Eyal, Wenyu
Debug Code :: Eyal, Wenyu
Final Code :: Eyal, Wenyu
Sensors
Get Sensor Data
Get temp/humidity data from sensors :: Wenyu
Get NO2 data from sensor :: Wenyu
Characterize sensors
Characterize temp/humidity sensors :: Wenyu
Characterize NO2 sensor :: Wenyu
Breadboard Prototyping
Rough schematic :: Travis, Wei
Build mote on breadboard :: Travis, Wei
Bill of Materials :: Travis
PCB Prototyping
Final schematic :: Travis, Wei
PCB Layout :: Travis, Wei
Place 1st order PCBs :: Travis
Build 1st PCB :: All
Place 2nd order of PCBs (if needed) :: Travis
Build 2nd PCB :: All
Minimal Viable Product :: All
Enclosure
Design weather resistant enclosure :: Wenyu
Purchase materials :: Wenyu
Finalize enclosure :: Wenyu
Install mote into enclosure :: Wenyu
Demonstrable Device
Testing
Write test plans :: Eyal, Wenyu
Perform test plans :: Eyal, Wenyu
Testing Completed :: Eyal, Wenyu
Documentation
Github collab site completed :: All
Write rough operations guide :: All
Write final operations guide :: All
Complete Capstone Poster :: All
Capstone poster event
Produce stakeholder report :: All
Write rough final report :: All
Write final report :: All
Final Report Presentation

Bill of Materials

This section contains the bills of materials needed to build a mote. The first bill of materials includes every component included in the mote. The second bill of materials only contains the components on the PCB itself.

The total cost of a single mote is $543.70. This price includes the cost of the PCB and soldered components, which amounts to $35.50. This does not include the cost of the N2O sensor, because Dr. Burnett purchased this, and the cost was not factored in.

These bills of materials shown below are summaries of the bill of materials and are formatted for ease of reading. See Appendix C for the complete bill of materials.

Bill of Materials - Total Build

Qty	Mfg	Mfg PN	Description	Cost
1	Oshpark	N/A	Sensor Suite Junction Board	$35.50
1	Texas Instruments	MSP-EXP430FR5994	MSP430FR5994 Launchpad	$20.39
1	Samsung	N/A	SD Card, 32 GB	$17.99
1	Analog Devices	DC9018B-B	DC9018B-B Evaluation Mote	$360.00
1	Honeywell	HIH6120	SENS HUMI/TEMP 3.3V I2C 2% 4SIP	$12.90

1	Dynament	DS0002	NO2 Sensor	$0.00
2	Yageo	RT1206BRD075KL	RES SMD 5K OHM 0.1% 1/4W 1206	$1.30
1	Otdorpatio	N/A	Enclosure, IP67 Waterproof	$27.99
1	LeMotech	N/A	1/4" Cable Gland	$0.55
1	Striveday	N/A	5-wire 1/4" cable, 3 ft.	$2.01
1	ESUPPORT	N/A	5-wire automotive connector	$2.46
10	Wirefy	N/A	heat shrink tubing, 1/8"	$0.70
1	Ltvystore	N/A	Battery Holder, 4 slot, 18650	$2.75
1	MELIFE	N/A	USB breakout board	$0.84
1	Spater	N/A	Micro USB cable	$1.60
4	SHENMZ	N/A	18650 Battery, Lithium-ion, 3.7V	$39.78
2	Assmann WSW Components	H3CCH-2006G	Ribbon Cable, 2x10	$2.70
1	Assmann WSW Components	IDSD-16-D-02.00-T-G-RW-R	Ribbon Cable 2x16	$13.40
15	RuiLing	N/A	Jumper Shunt, 2.54mm	$0.52
1	Uxcell	N/A	8 pin adapter boards	$0.32

			Total Cost:	$543.700

Bill of Materials - PCB

Qty	Mfg	Mfg PN	Description	Cost
1	Oshpark	N/A	Sensor Suite Junction Board	$20.00
1	Honeywell	HIH6120	SENS HUMI/TEMP 3.3V I2C 2% 4SIP	$12.90
1	Dynament	DS0002	NO2 Sensor	$0.00
105	MCIGICM	N/A	Pin Headers	$1.31
2	Yageo	RT1206BRD075KL	RES SMD 5K OHM 0.1% 1/4W 1206	$1.30
			Total:	$35.50

Construction of Mote

This section details how to purchase, obtain materials and construct the mote. The full bill of materials follows at the end of this section.

Base Construction

The core of the mote consist of 7 main components as follows:

1. Waterproof Enclosure and Base Board
2. Texas Instruments MSP430FR5994 Launchpad (MSP430)
3. Analog Devices SmartMesh IP DC9018B-B (DC9018B-B)
4. Sensor Suite Junction Board (Junction Board)
5. Battery pack
6. N2O Sensor
7. Temperature/Humidity Sensor

The MSP430, DC9018B-B, Temp/Humid Sensor, and N2O sensor communicate with each other via the Junction Board. The schematic and PCB layout design files for the Junction Board are included in this report and can be found in the GitHub repository.

- Connect the DC9018B-B to the Junction Board via a 2x16 ribbon cable from DC9018B-B P1 to Junction Board J8.
- Connect the MSP430 J1/J3 pin headers to the junction board J1/J3 via a 2x10 ribbon cable.
- Connect the MSP430 J2/J4 pin headers to the junction board J2/J4 via a 2x10 ribbon cable.
- Connect the N2O sensor to Junction board J7.

- Connect the Temperature / Humidity sensor to Junction board U6.
- Connect the battery pack to the MSP430 (more details in Battery Pack section). Install 4 fully-charged 18650 3.7V Batteries.

Warning: Ensure 18650 batteries are fully-charged when installed. Failure to do so may cause overheating of batteries or fire. When swapping batteries, always replace all 4 batteries with 4 fully-charged batteries.

Enclosure and Base Board

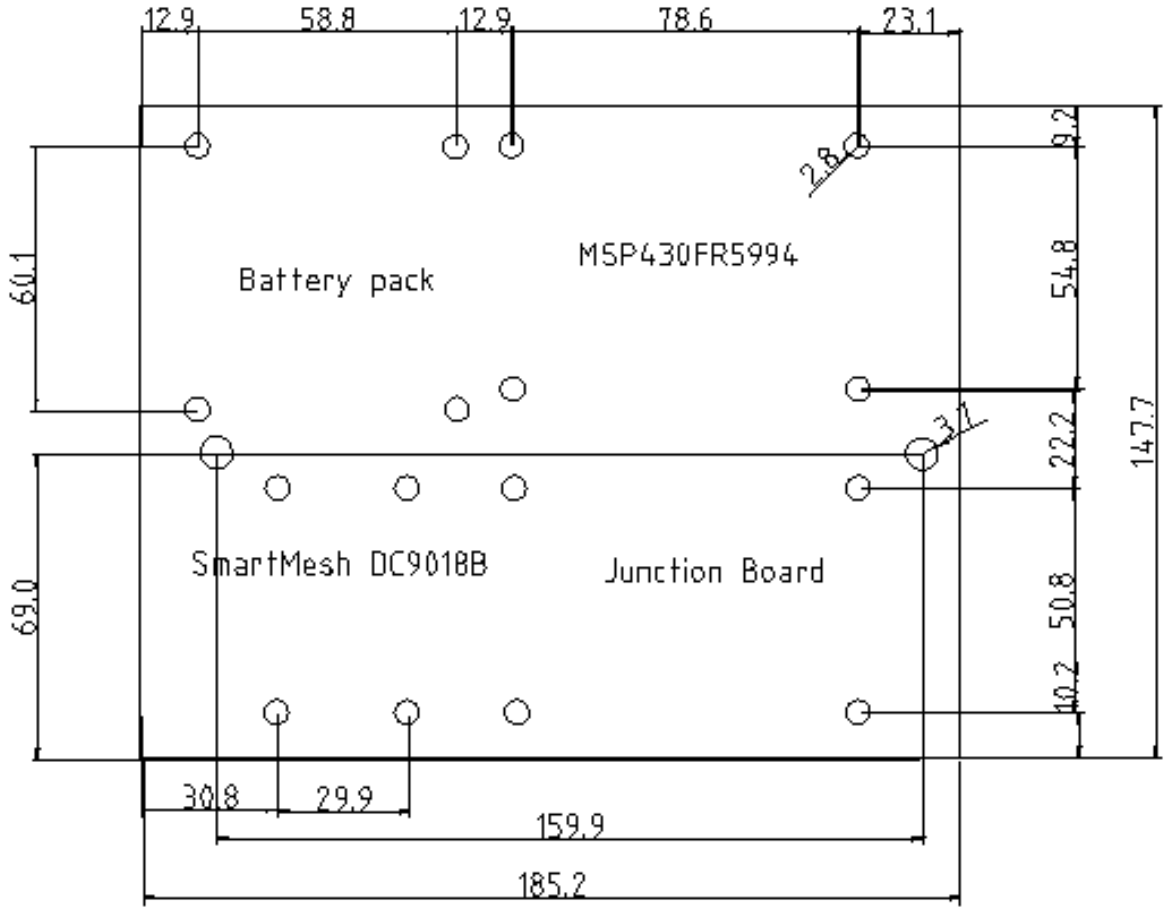
The mote is housed in a waterproof enclosure box rated IP67. Drill a 1/2" hole on the bottom of the enclosure and install a 1/4" cable gland. Feed the N2O sensor external cable through the cable gland, and tighten the cable gland to secure the cable and ensure the cable hole is waterproof.

Note: The "bottom" of the mote is relative to the desired position when the mote will be installed. For our build, the bottom was the short side of the enclosure adjacent to the junction board and MSP430. See images.

Base Board

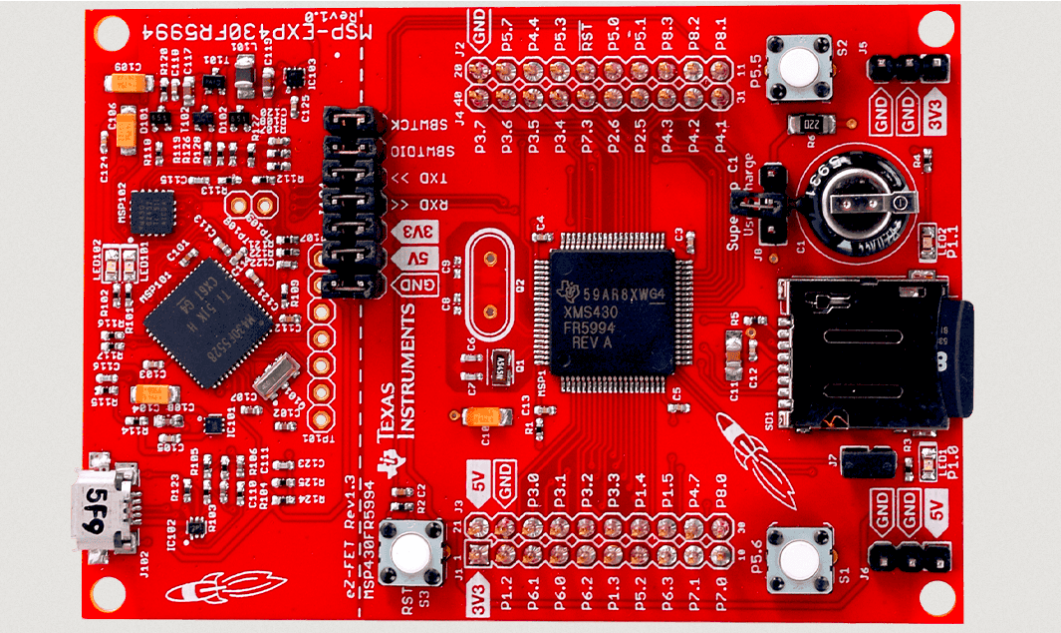
The base board is a platform for mounting the various components. It is constructed out of 1/8" thin bamboo board, purchased at the Electronic Prototyping Lab (EPL) at Maseeh College of Engineering, Portland State University. The EPL has a laser cutter to cut the bamboo board to size.

Bore holes into the base board per spec. Mount all components to the base board as detailed below. Mount the base board to the enclosure using the 2 screws provided by enclosure box.



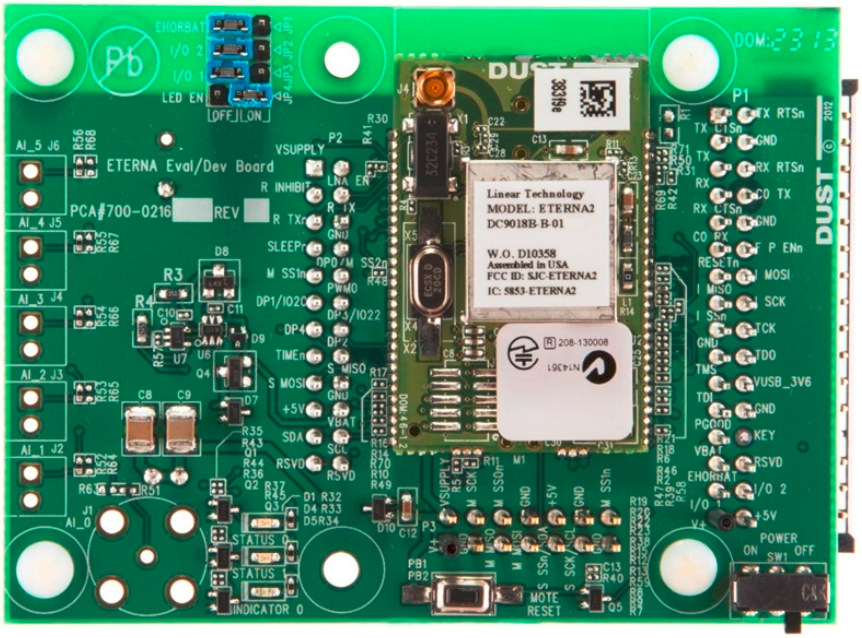
MSP430FR5994

Before plugging in the MSP430FR5994 (MSP430), upload the firmware via the Energia IDE. Then install a 32 GB SD Card to store sensor logs. Mount the MSP430 to the base board using 4 10mm standoffs.



SmartMesh DC9018B-B

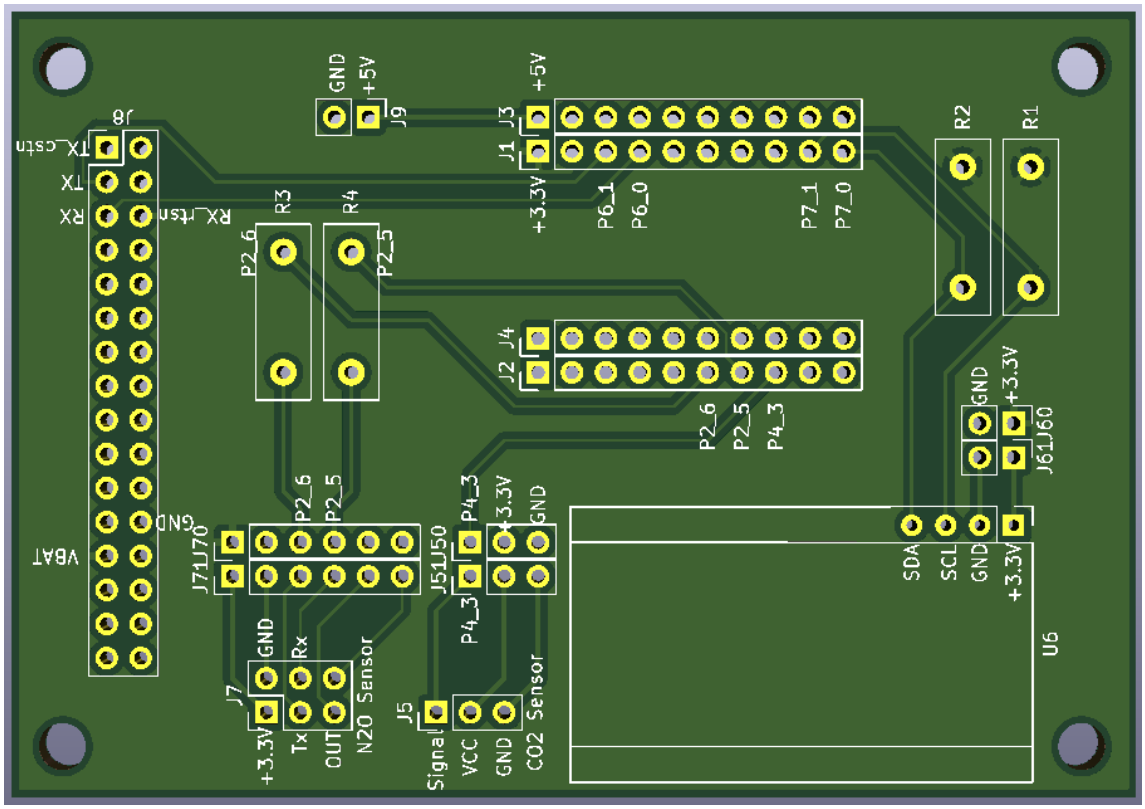
Remove the 3V lithium battery from the underside of the DC9018B-B. This is not required since the DC9018B-B will be powered via the 3.3V trace from the MSP430. Turn the power switch to ON. Mount the DC9018B-b to the baseboard using 4 10mm standoffs.



Sensor Suite Junction Board

The Sensor Suite Junction Board was designed and built by the Capstone team using KiCAD. Future motes will require this board to be fabricated and built. Details on building the junction board are included in this report.

The junction board was designed with future development in mind. Since all of the MSP430 pins are connected to the junction board via the ribbon cables, more sensors can be implemented on these pins. The junction board has empty space to implement new pin headers to connect additional sensors.

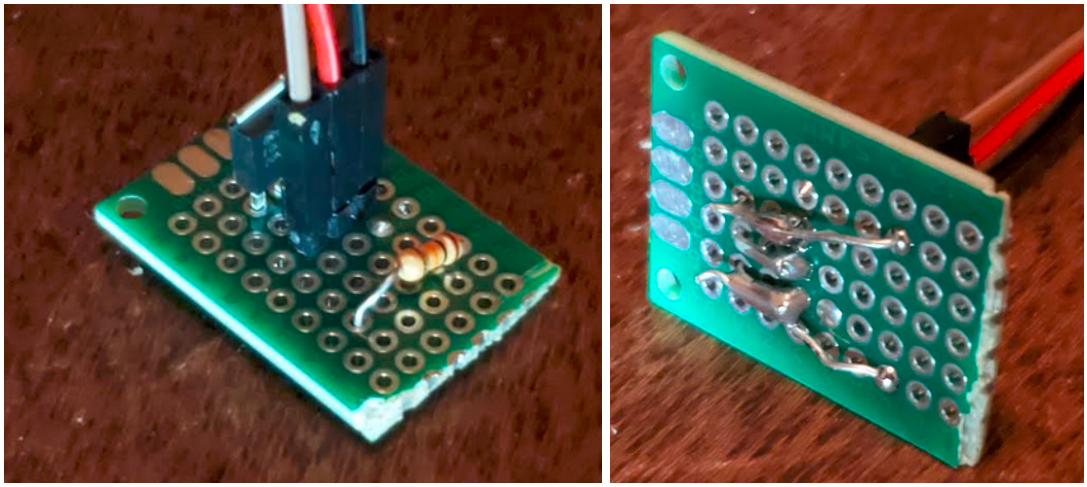


Connect the Transistor Switch Breakout Board as detailed in the following section. Add jumper shunts to pin headers J50/J51, J60/J61 and J70/J71. Solder 0Ω resistors

at R3 and R4. Solder in the temperature sensor breakout board to U6. Mount the junction board to the base board using 4 10mm standoffs. Connect the N2O sensor to the junction board via the external cable at J7.

Transistor Switch Breakout Board

Pin headers labeled J70/J71, J60/J61 and J50/J51 are debug pin headers. Connect these pins using jumper shunts. The purpose of these pins was provide backup flexibility if additional components were needed. This proved to be the case as a transistor breakout board was designed and implemented late in the project in order to turn off the N2O sensor to conserve power when it is not in use. To connect the transistor switch breakout board, remove the jumper shunt at the GND pins for J70/J71, and the P4_3 pins for J50/J51.



The transistor switch breakout board has 3 pins connected to a FQU20N06LTU transistor's gate, drain and source. Connect these pins via jumper wires to the junction board as follows:

- Gate to J50 P4_3

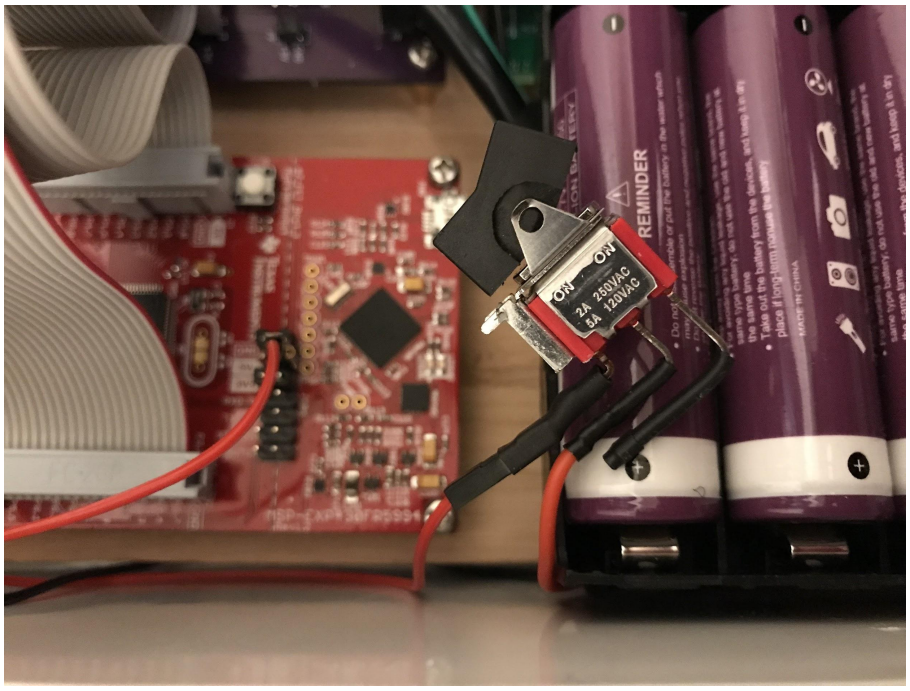
- Drain to J71 GND
- Source to J70 GND

Battery Pack

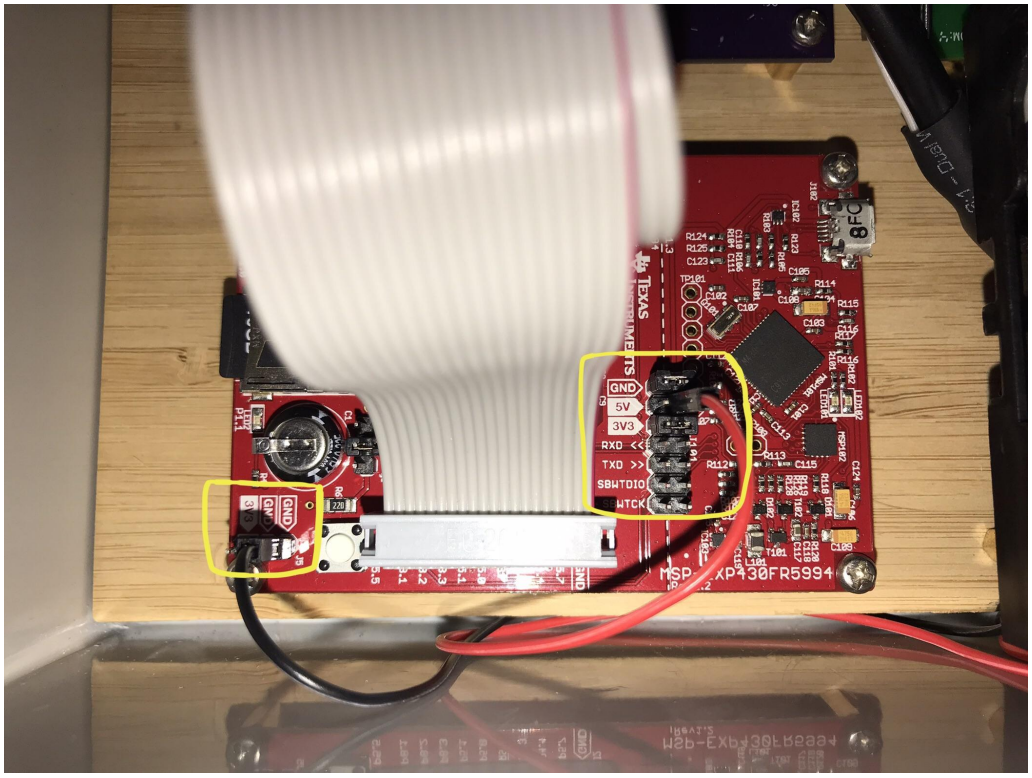
The mote is powered by 4 3.7V 3400mAh 18650 lithium-ion batteries. The battery pack was designed and built by the Capstone team. It consists of the following components:

- Battery holder with 4 slots
- 28 AWG hook -up wire
- Single pole double throw switch
- 2 breadboard jumper wires (one red, one black)

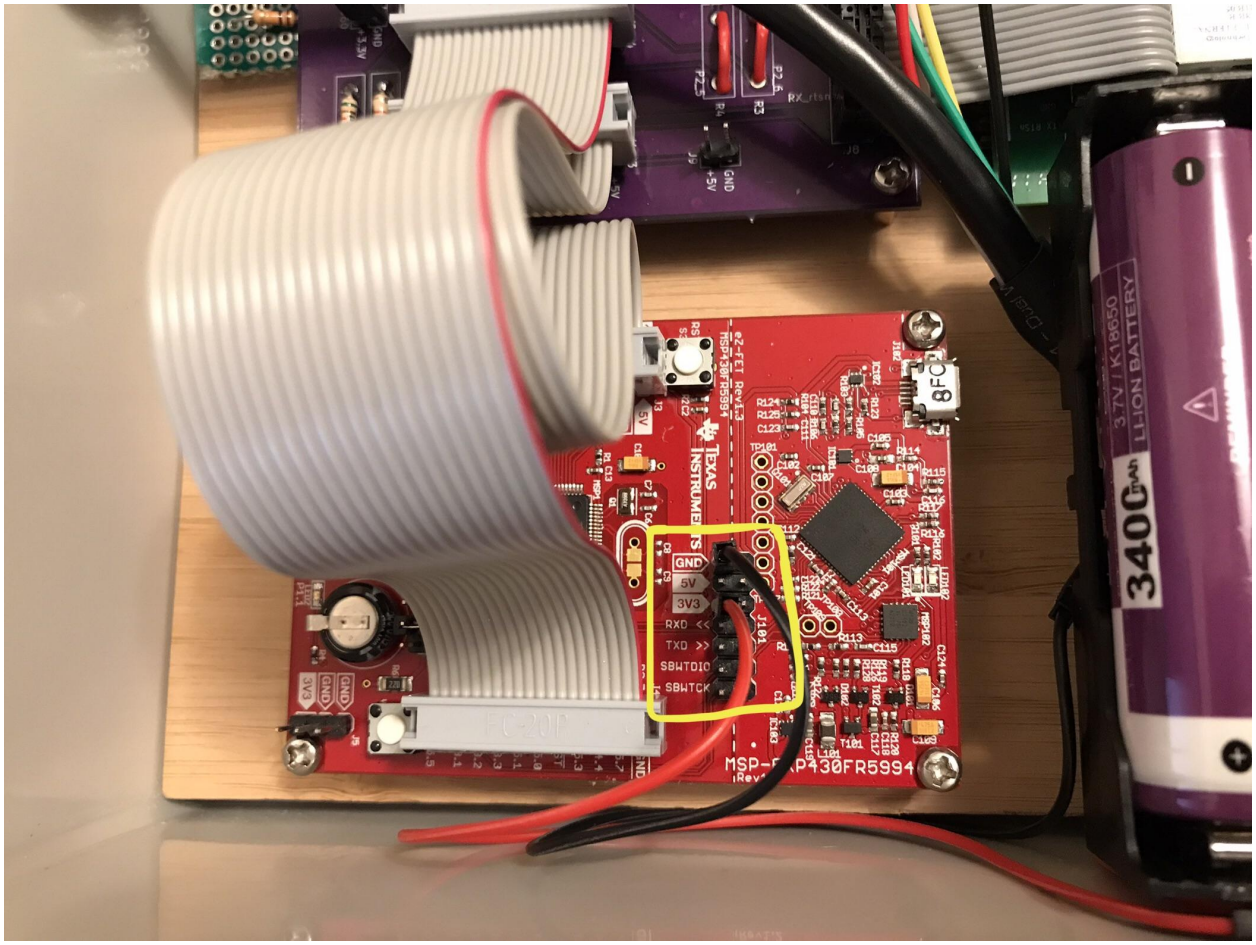
The batteries are connected in parallel. Solder the 4 positive leads together using hook-up wire. Solder another piece of hook-up wire to the grouped wires so that only wire will act as the positive lead to the entire battery pack. Then solder this one wire to the middle pin of the single pole double throw switch (SPDT) as shown in the picture below. Solder a red breadboard jumper wire to the left of the middle pin on the switch. Leave the rightmost pin on the switch unsoldered. This setup allows the user to connect or disconnect the positive side of the battery pack from the system. Solder the 4 negative leads together using hook-up wire. To the hook-up wire solder a black breadboard jumper wire. Mount the battery holder to the base board using 4 15mm standoffs.



Connect the red and black breadboard jumper wires of the battery pack to the MSP430 as shown in the picture below. Make sure to place a shunt jumper on the GND pins and the 3.3V pins located near the red jumper wire (see picture below).



Another way to connect the battery pack to the system is shown below.



N2O Sensor

Caution: The N2O sensor is ESD sensitive. Always use proper ESD procedures when handling the N2O sensor. Use ESD mat and wrist strap when handling N2O sensor.

Install the N2O sensor into the housing. Connect the housing's pigtails to the external cable.

External cable

The external cable connects the N2O sensor to the mote via the hole, which allows the N2O sensor to reside outside of mote enclosure for data collection.

The external cable is a 5-wire 0.25” OD cable that connects the N2O sensor to the junction board. It enters the enclosure via the cable gland installed on the enclosure. On the external end of the cable, solder an in-line 5-wire waterproof automotive connector between the cable and the N2O housing. Use heat shrink wrap at each solder joint, and a large heat shrink wrap over the entire splicing. The purpose of this connector is to allow easy disconnect of the N2O housing from the enclosure when needed.

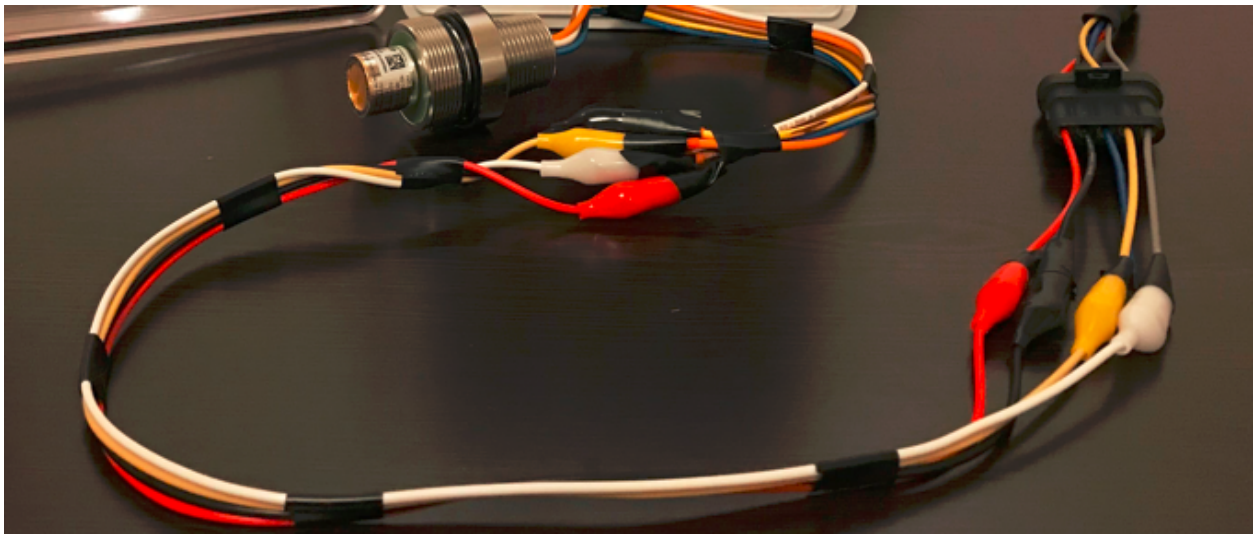
On the internal end of the cable, solder 5 male-to-female jumper wires. Use heat shrink wrap at each solder joint and a large heat shrink wrap over the entire splicing.

Wires Colors

Match up the correct colors of the cable with the jumpers and automotive connector. See below:

- VCC - Red
- GND - Black
- TX - Yellow
- RX - White / Grey
- OUT - Green / Blue

It should be noted that this cable build method should be improved. The first attempt was to use Dupont-style connectors, however, difficulty in crimping the connections prevented the use of these connectors.

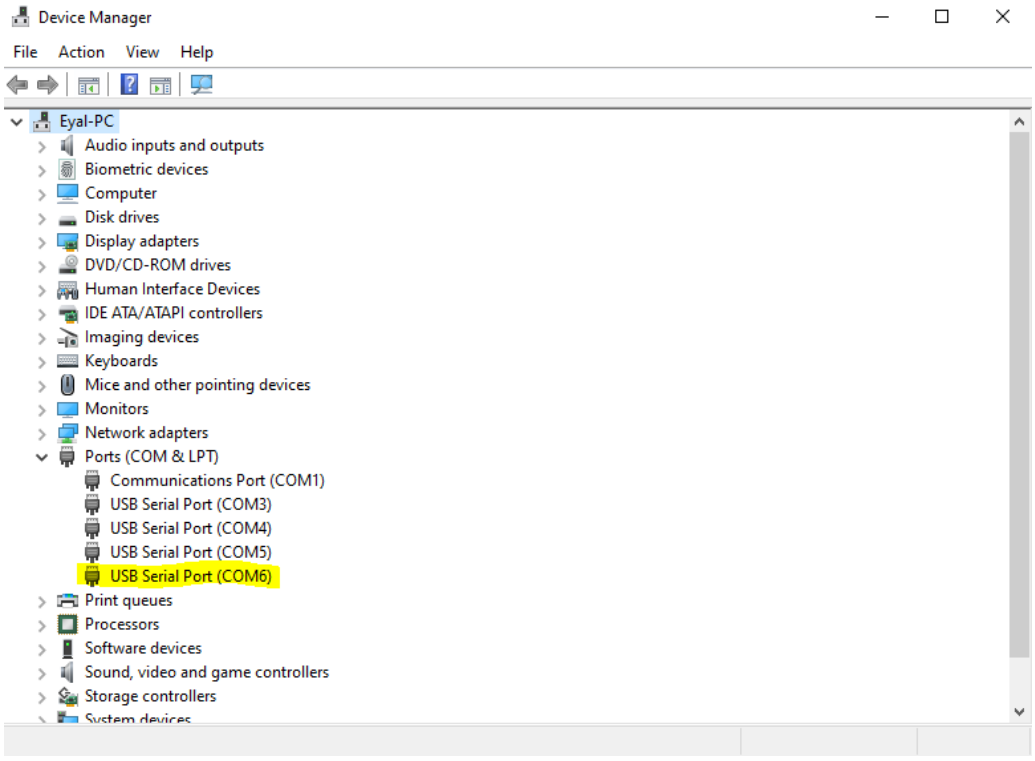


Test Plan

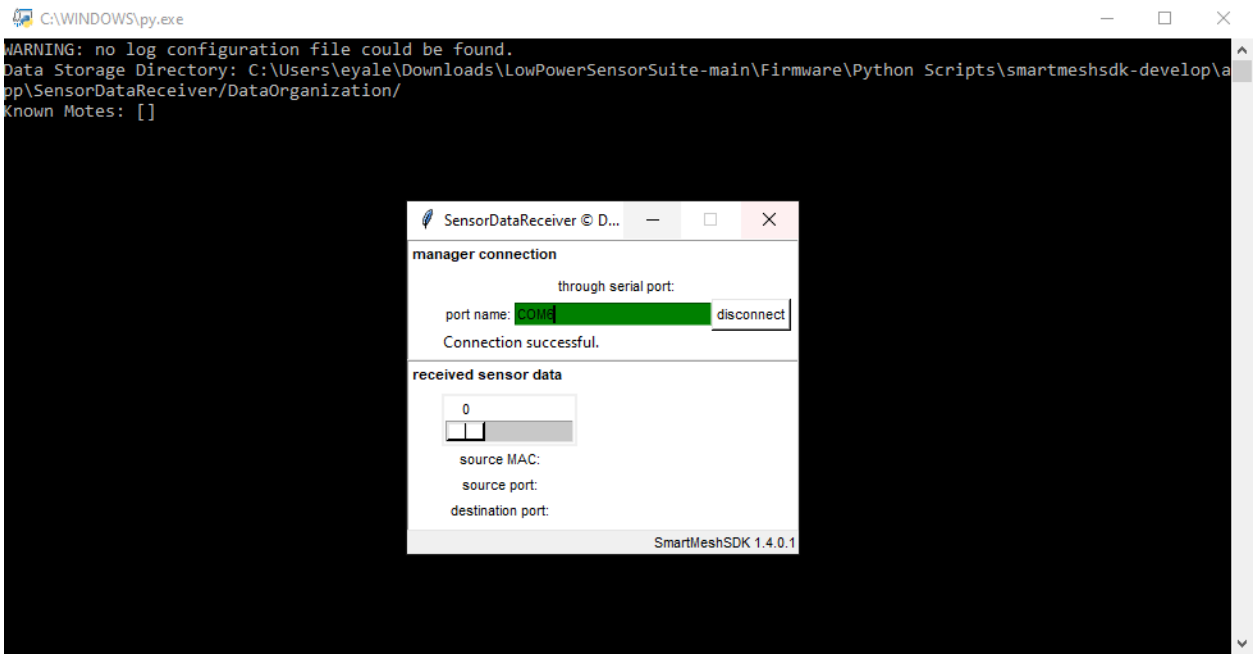
Test 1 - DC2274A-A (blue receiver, also called a manager) to Main PC connection

- Connect manager to PC
- Go to device manager to check which port the blue manager is using.
    - Blue manager will always be the highest number COM#.
        - For example if port shows COM1, COM2, COM3, and COM4.

          Blue manager is using COM4.



- Open “SensorDataReceiver” python application



This will pop up and you will need to enter the COM that the blue manager is using.

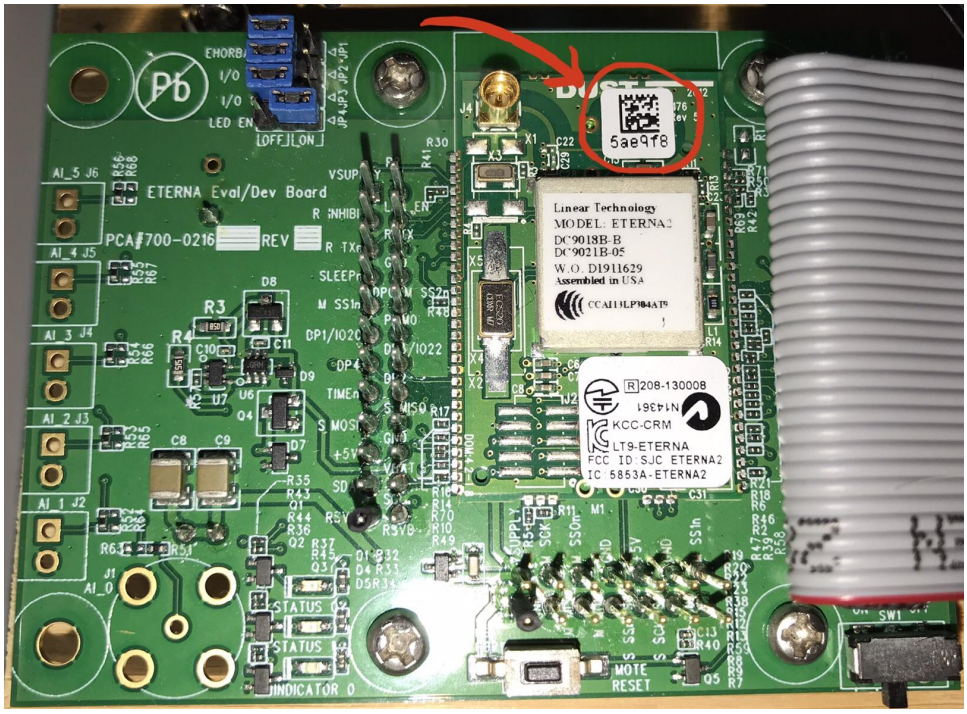
Results from Test 1

If the connection is successful then the port name field will turn green and connection between manager and PC is successful.

Test 2 - Smartmesh mote to Manager connection

- While the manager to PC connection is still going we want to see if the mote and the manager are communicating with each other.
    - Turn the power switch of the system on.
        - You will see a green LED on the MSP430 as an indicator that the batteries are properly powering up the system. You will need to wait 1-2min for next step.

- Go to this directory “smartmeshsdk-develop\app\SensorDataReceiver” and check a folder called “DataOrganization”. Inside this folder you should find a data log that is named after the mote that you are using.
  - Name of the mote will be found here:

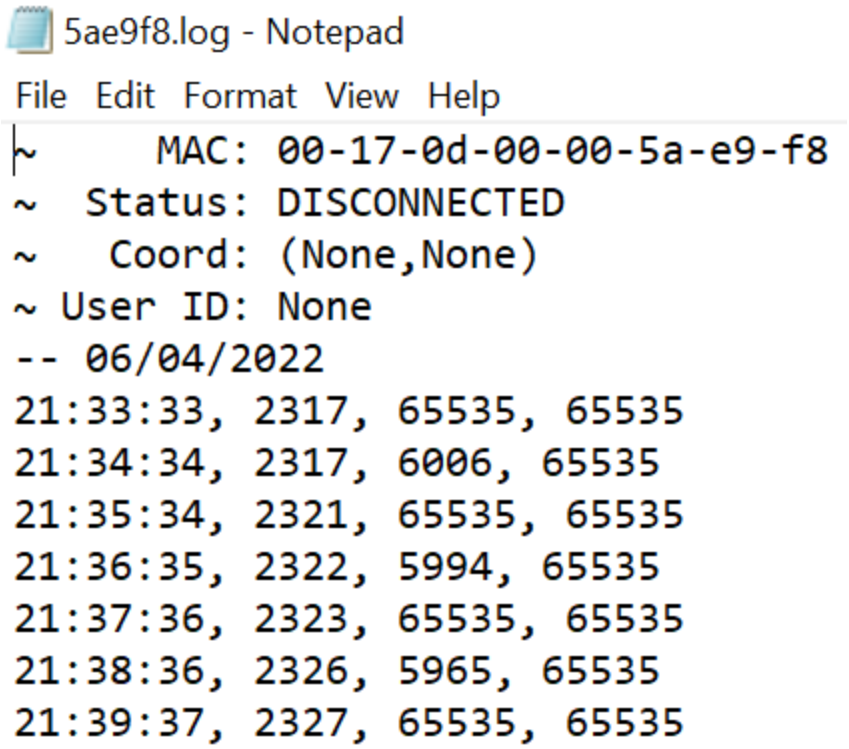


Results from Test 2

If you find this data log with mote name then the connection between Smartmesh mote and manager is successful.

Test 3 - MSP430 microcontroller to each individual sensor connection

- Open data log with Smartmesh mote name in “DataOrganization” folder inside this data log you will see the following:



- The first column corresponds to time stamp. Second column corresponds to temperature data. Third column corresponds to Humidity data. The fourth column indicates N20 data. The sensor value 65535 corresponds to a NULL value, which means sensor didn't measure any value at that time.
    - Temperature sensor has a sample rate of 1 minutes.
    - Humidity sensor has a sample rate of 2 minutes.
    - N20 has a sample rate of 10 minutes.
- Wait 10 minutes and you will see time stamp that has all three valid sensor data. This indicates that all sensors are working correctly.

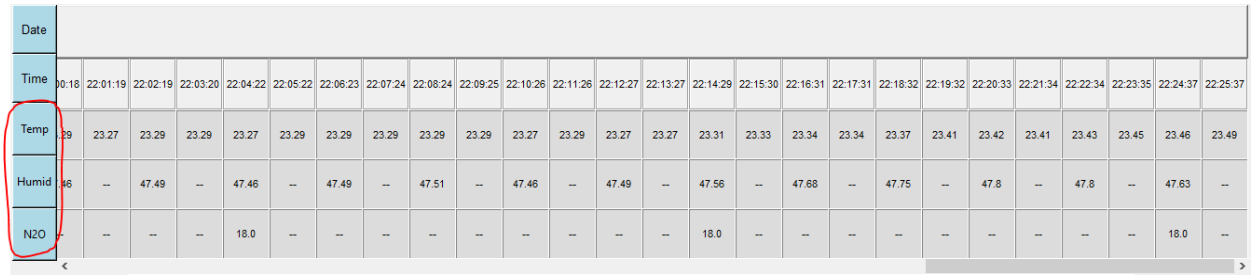
Results from Test 3

By receiving data values that are non NULL this means all sensors are correctly measuring their corresponding metric. You can apply heat to the temperature

sensor or breathe on the humidity sensor and you will see the data values change accordingly.

Test 4 - GUI functionality

- Open “SensorSuiteGUI” python application.
- Check if data on table of GUI matches the data on the datalog in “DataOrganization” folder.
- Click on these buttons and check if each graph is functional.



Date																										
Time	00:18	22:01:19	22:02:19	22:03:20	22:04:22	22:05:22	22:06:23	22:07:24	22:08:24	22:09:25	22:10:26	22:11:26	22:12:27	22:13:27	22:14:29	22:15:30	22:16:31	22:17:31	22:18:32	22:19:32	22:20:33	22:21:34	22:22:34	22:23:35	22:24:37	22:25:37
Temp	39	23.27	23.29	23.29	23.27	23.29	23.29	23.29	23.29	23.29	23.27	23.29	23.27	23.27	23.31	23.33	23.34	23.34	23.37	23.41	23.42	23.41	23.43	23.45	23.46	23.49
Humid	46	--	47.49	--	47.46	--	47.49	--	47.51	--	47.46	--	47.49	--	47.56	--	47.68	--	47.75	--	47.8	--	47.8	--	47.63	--
N2O	--	--	--	--	18.0	--	--	--	--	--	--	--	--	--	18.0	--	--	--	--	--	--	--	--	--	18.0	--

- Wait a few minutes to see if GUI updates tables and graph as new data comes in.

Results from Test 4

By accomplishing each step this will ensure all GUI features work as intended and GUI is completely functional.

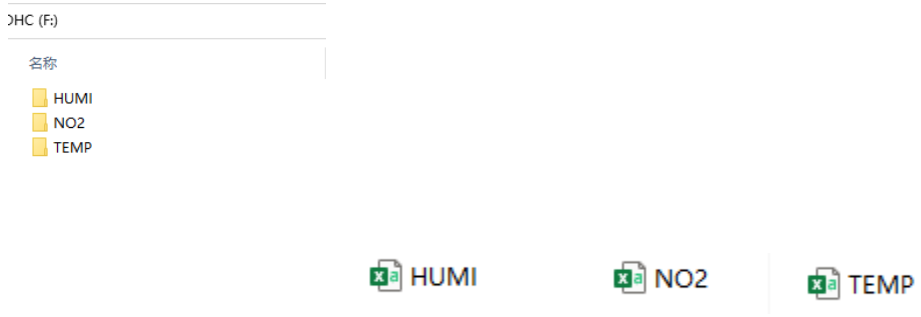
Test 5 - SD Card data logging

- We need an SD card less than or equal to 32GB, and connect it to the PC to format it to make sure it is empty. Then insert the blank SD card into the SD card slot of the MSP430, and connect them to the PC.
- Replace the library in the *LowPowerSensorSuite/SDCardNew* folder to *Document/Energia/libraries/SDCard,* then run

LowPowerSensorSuite/Firmware/Energia Code/SensorMote_ V0.1/SensorMote_ V0.1, and keep running for a longer time.

Results from Test *5*

- Reconnect the SD card to the computer.
    - We can see that a folder corresponding to each sensor is created in the SD card, and there are corresponding CSV files.



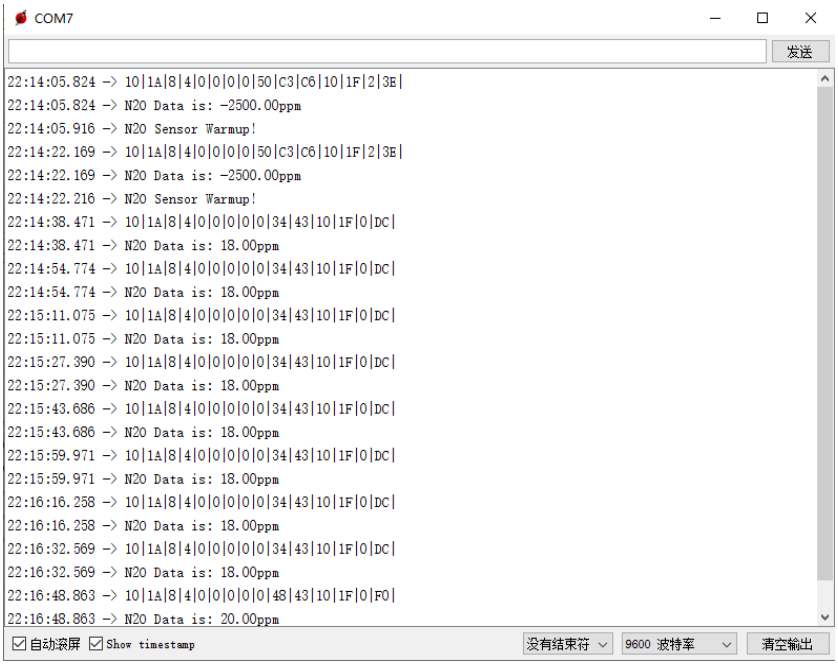
    - We can use Excel to open the CSV file, the first column is the sensor data, and the second column is millis indicating the time when the data was acquired.

Test 6 - N2O sensor

- Configure P2.5&P2.6 as hardware serial ports (refer to User's Manual (Programming) 3.)
- After wearing the ESD device correctly, connect the N2O sensor to the MSP430, and then connect the MSP430 to the computer
    - Run *LowPowerSensorSuite/N2O/N2OSerial2/N2OSerial2.ino*

Results from Test 6

- Open Serial monitor.

- The first three sets of data should be 250% (data is obtained every 15 seconds, we can know from the datasheet that the warm-up time of the N2O sensor is about 45 seconds), and there is a prompt that the sensor is warming up
    - After the warm-up time, you can see the normal gas data sent by the sensor. (The data in the screenshot below is not expressed as a percentage, since the device is already installed, we use the previous screenshot)
    - 

Results

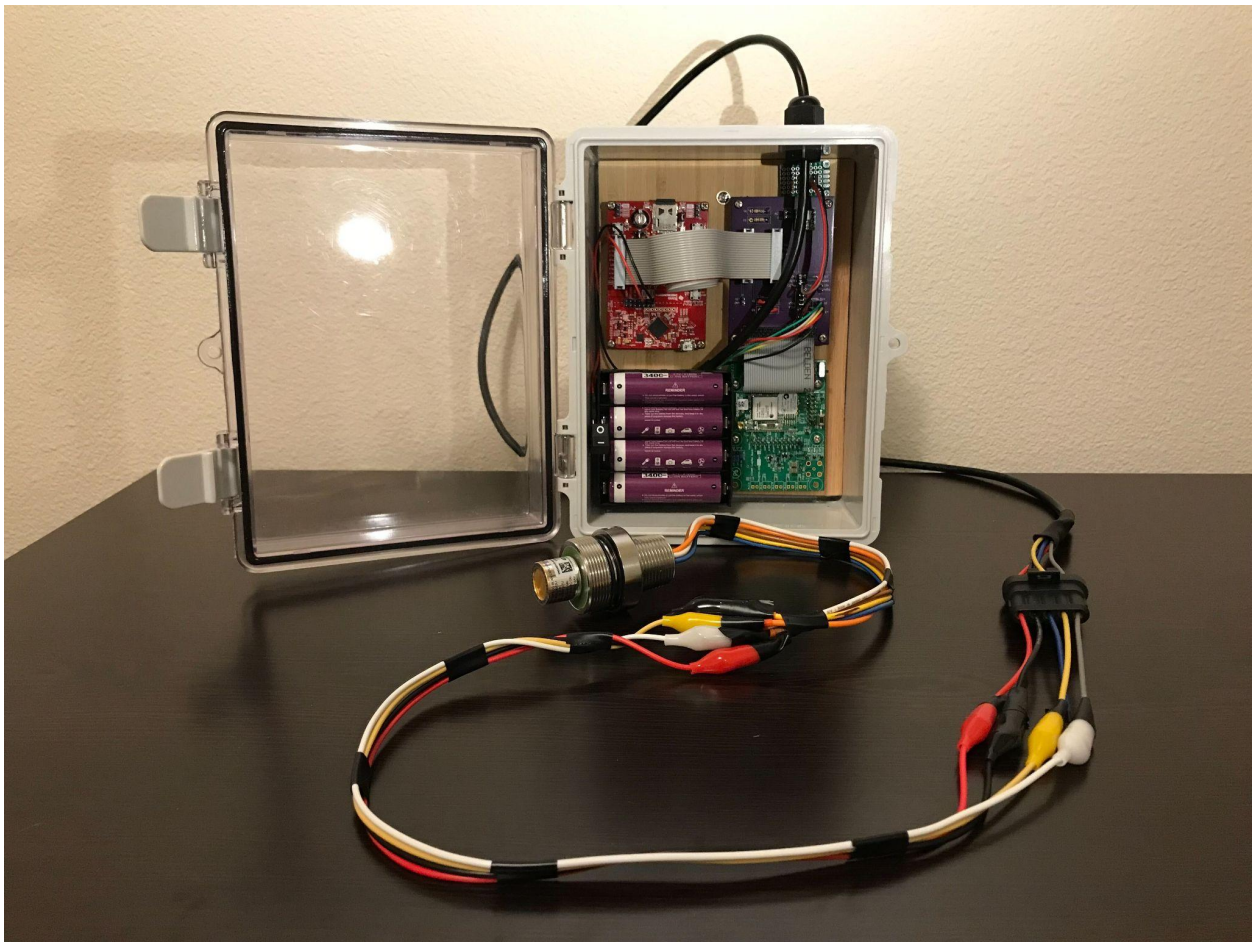
Overall this project was a complete success. We were able to create a system that is able to communicate wirelessly through Smartmesh network. This system includes three sensors: Temperature, Humidity, and N2O. These sensors are controlled through a microcontroller, MSP-EXP430FR5994. We created a clean path for communication between sensors to microcontroller, microcontroller to Smartmesh mote, Smartmesh mote to Smartmesh manager, Smartmesh manager to any computer, and then finally any computer to a Graphical User Interface. We were able to design this system in various power modes to extend battery life to the fullest. We brought the battery life from less than a day to around 18-19 days with one 3400mAh 3.7 Volts battery. Our system includes four of these batteries so we managed to get a battery life time of over 2.5 months. Lastly, we were able to use the SD card slot on MSP430-EXPFR5994 to store all the data that the sensors collect. This is for extra robustness to the system in case the Smartmesh connections between mote and manager fail.

Performance

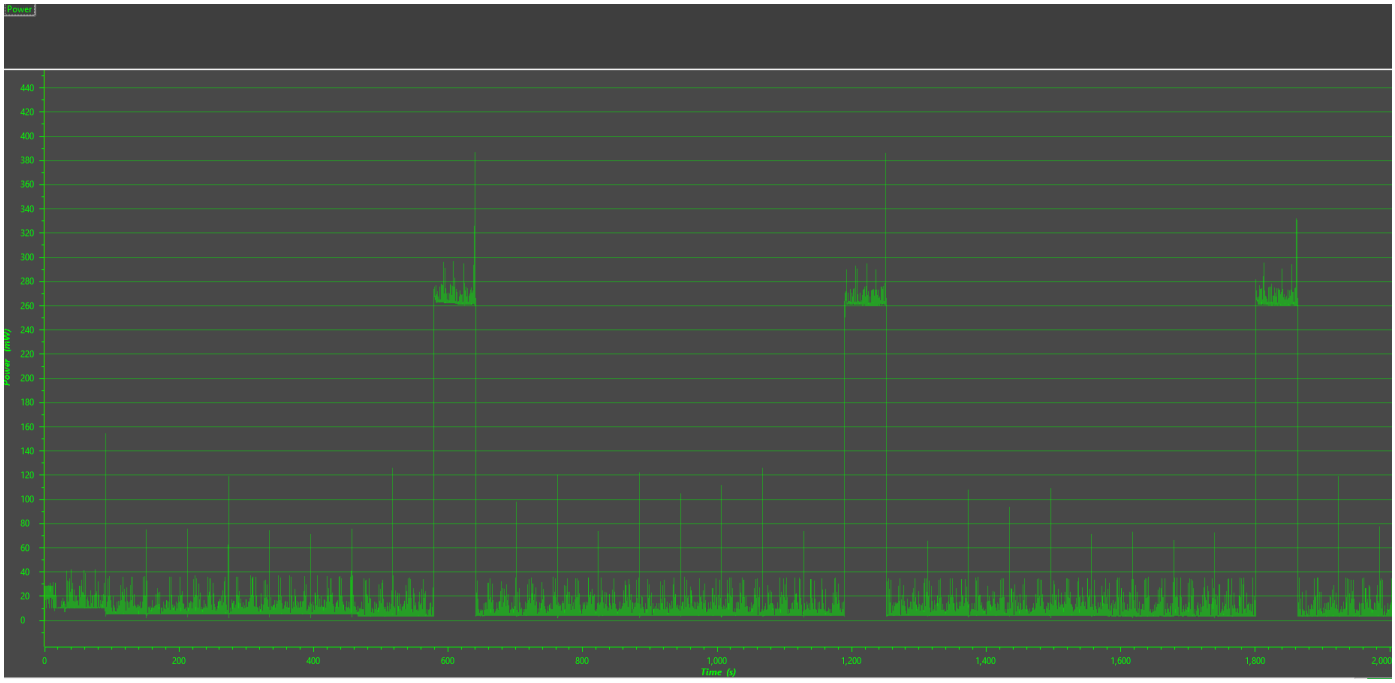
Metrics	**Value**
Active mode Current (mA)	72
Sleep mode Current (mA)	3.3
Active mode Overall Power consumption (mW)	270
Sleep mode Overall Power consumption (mW)	12
Average Power Consumption (mW)	28.32
Voltage (V)	3.7

N2O average power consumption (mW)	17.07
N2O total active time per 10 min (min)	1
Temp and humidity average power consumption (mW)	3.36
MSP430 + LED average power consumption (mW)	4.45
Smartmesh Mote average power consumption (mW)	3.44
N20 Sample rate (minutes)	10
Temperature Sample rate (minutes)	1
Humidity Sample rate (minutes)	2
Overall battery life time with 4 batteries (days)	76

Picture of Low Power Environmental Sensor Suite hardware and enclosure:



Entire system power consumption graph using Energy Trace (Detailed Energy Trace data can be found here: https://github.com/travis26pdx/LowPowerSensorSuite/tree/main/EnergyTraceData)

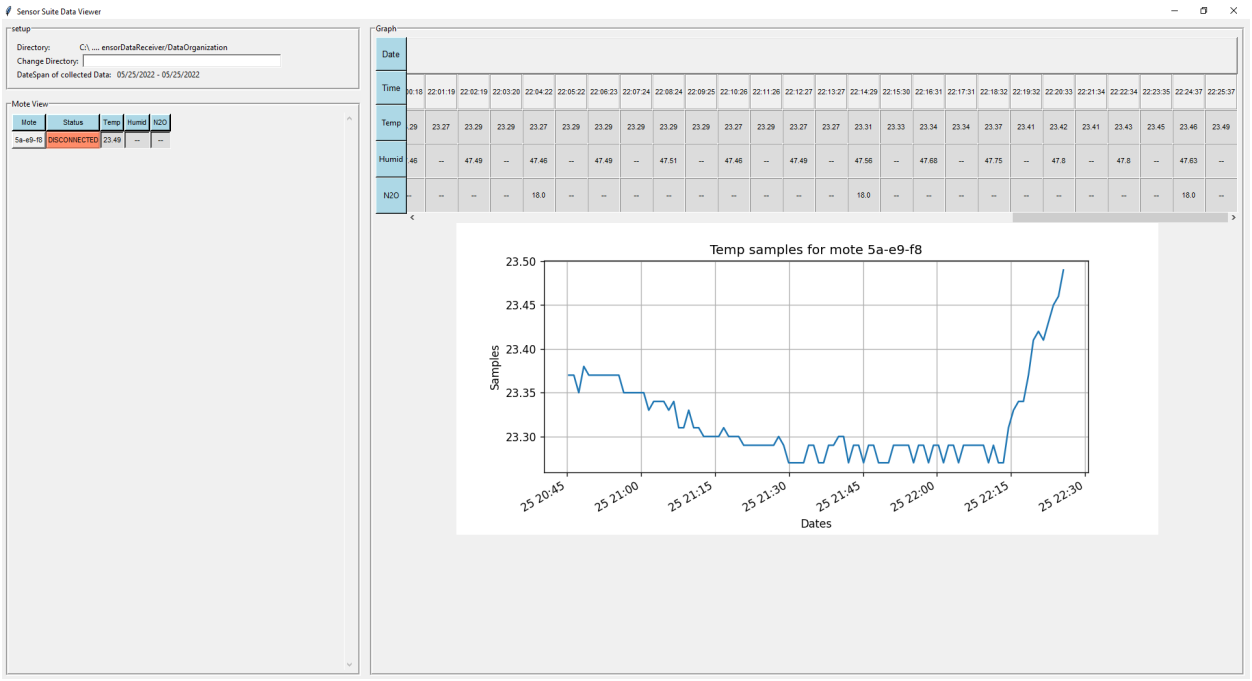


EnergyTrace™ Profile

Name	Live
System	
Time	2052 sec
Energy	58061.989 mJ
Power	
Mean	28.3248 mW
Min	0.0000 mW
Max	393.6869 mW
Voltage	
Mean	3.2915 V
Current	
Mean	8.6134 mA
Min	0.0000 mA
Max	119.8438 mA
Battery Life	Custom: 18 day 12 hour (est.)

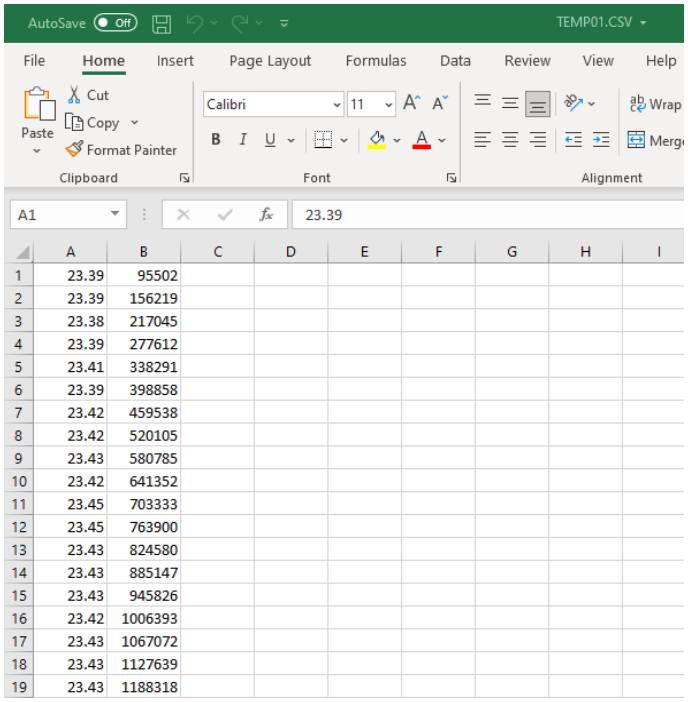
*NOTE: Custom battery life is for one 3.7V 3400mAh battery.

GUI showing data points:



SD card data log:

Timestamp for specific piece of data = Time system was turned on + millis value (second column in file). First column is the sensor data value.



	A	B	C	D	E	F	G	H	I
1	23.39	95502							
2	23.39	156219							
3	23.38	217045							
4	23.39	277612							
5	23.41	338291							
6	23.39	398858							
7	23.42	459538							
8	23.42	520105							
9	23.43	580785							
10	23.42	641352							
11	23.45	703333							
12	23.45	763900							
13	23.43	824580							
14	23.43	885147							
15	23.43	945826							
16	23.42	1006393							
17	23.43	1067072							
18	23.43	1127639							
19	23.43	1188318							

Post Mortem

Team Work

Our team worked very well together. Each member was able to organically assume ownership of each necessary section of the project, with nothing left behind. Utilizing the GANTT chart helped keep us on track, and to make sure that each team member knew what they needed to do each week, and everything that needed to be accomplished had a team member taking responsibility for it.

It was especially helpful that our Faculty Advisor wanted us to explain in the weekly reports specifically whether each portion of the project was on schedule, behind schedule and exactly by how much.

Although we worked remotely, we had weekly Zoom meetings with the Faculty Advisor and Industry Sponsor. Also, the student team members had a separate weekly Zoom meeting. We utilized a shared locker in the Fourth Avenue Building to transfer equipment. Not having a shared workspace made things difficult at certain times, but not enough to set us back.

Our Industry Sponsor attended every weekly meeting, and was very involved and very interested in our progress, providing helpful hints, and requesting various tasks and data collections. Having him available each week helped to keep us focused on the project, and he was also able to answered technical questions that we had.

N2O Sensor Problem

The following is an example of how our team was able to respond to an emergency, project-killing problem.

We received the N2O sensor in mid-April which was very late in the project. The code was quickly written and implemented. While we always knew that the sensor would be a major draw of energy, we did not know that the N2O continued to use a large amount of energy even when the MSP430 was in deep sleep mode. This destroyed our power calculations. The batteries would only have last 2 days. This was a project killer.

We found this out on May 13. Our team called together an emergency ad hoc meeting to brainstorm a solution. We knew the previous capstone team had already designed using transistors to cut the power of sensors, we looked through their work. Luckily, they also had the transistors provided with their prototype. We quickly designed the circuit to implement the transistor, soldered it to a breakout board. The next day, Wei drove to Travis' work to pick up the circuit, and he drove it to Eyal's to drop it off and implement it. By the next evening, we took EnergyTrace data, which showed that the circuit worked, and when the N2O sensor was turned off, the batteries would last 75 days! The project was saved.

Individual Post-Mortems:

Wenyu Bi

- Document problems and solutions that arise in the project in a timely manner.
- Only adequate cooperation and communication can make the team work smoothly.
- Don't hesitate to ask others for advice when encountering problems that you think can't be solved, so that we can learn new knowledge in the project.
- Good comments are as important (or even more important) as well-working code.

Eyal Eynis

- Document often and document early.
- Comment code.

- Communicate with the team often about project progress.

Travis Johnson

- I was team leader for this Capstone. I organized team meetings and emailed out meeting agendas to the team the day before to keep the discussion focused.
- Know your team members' strengths. I often delegated out executive decisions to my team members who knew more about the certain technical aspects than I did, and I was always happy with their decisions.
- Make a GANTT chart and use it to track the progress of the project. Break up large milestones into weekly goals. During meetings, start by discussing monthly goals and milestones as the context when determining weekly goals.

Wei yan

- Always prepare for worst case scenarios
- ALWAYS comment code.
- If you start your project with undocumented code from someone else, starting from scratch is your best choice.

Collaboration Site and Repository

This year's Github site (2nd gen):

https://github.com/travis26pdx/LowPowerSensorSuite

Previous Year's Github site (1st gen):

https://github.com/ECE-412-Capstone-Sensor-Suite/Team-20-Sensor-Suite

References

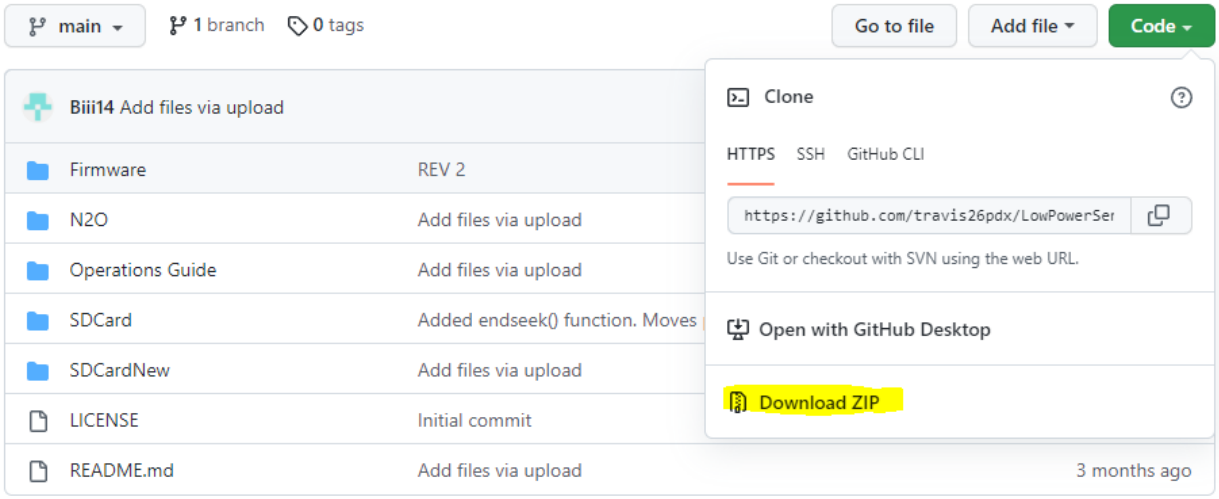
Figure 1: Greenhouse gas emissions over the years

https://insideclimatenews.org/news/11092019/nitrous-oxide-climate-pollutant-explainer-greenhouse-gas-agriculture-livestock/

Appendices

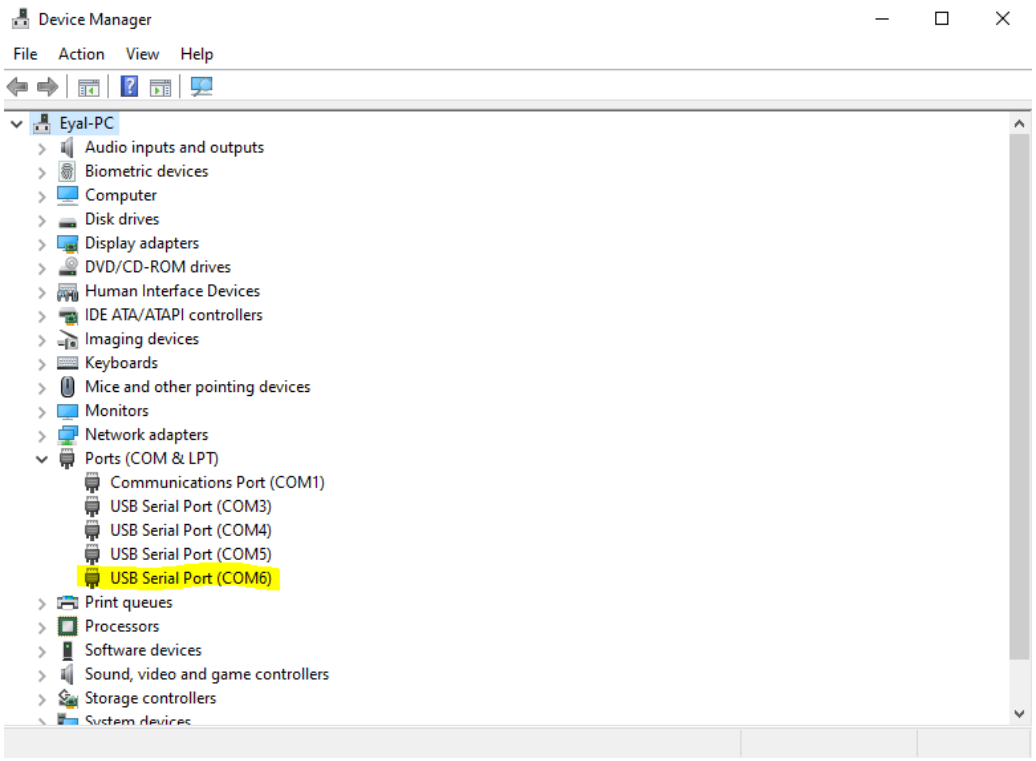
Appendix A: User's Manual (Operation)

1. Install Python 3.10.1 (https://www.python.org/downloads/)
2. On the capstone team's GitHub (https://github.com/travis26pdx/LowPowerSensorSuite) download the entire GitHub repository (see picture below). Unzip it and the folder should be named "LowPowerSensorSuite-main".



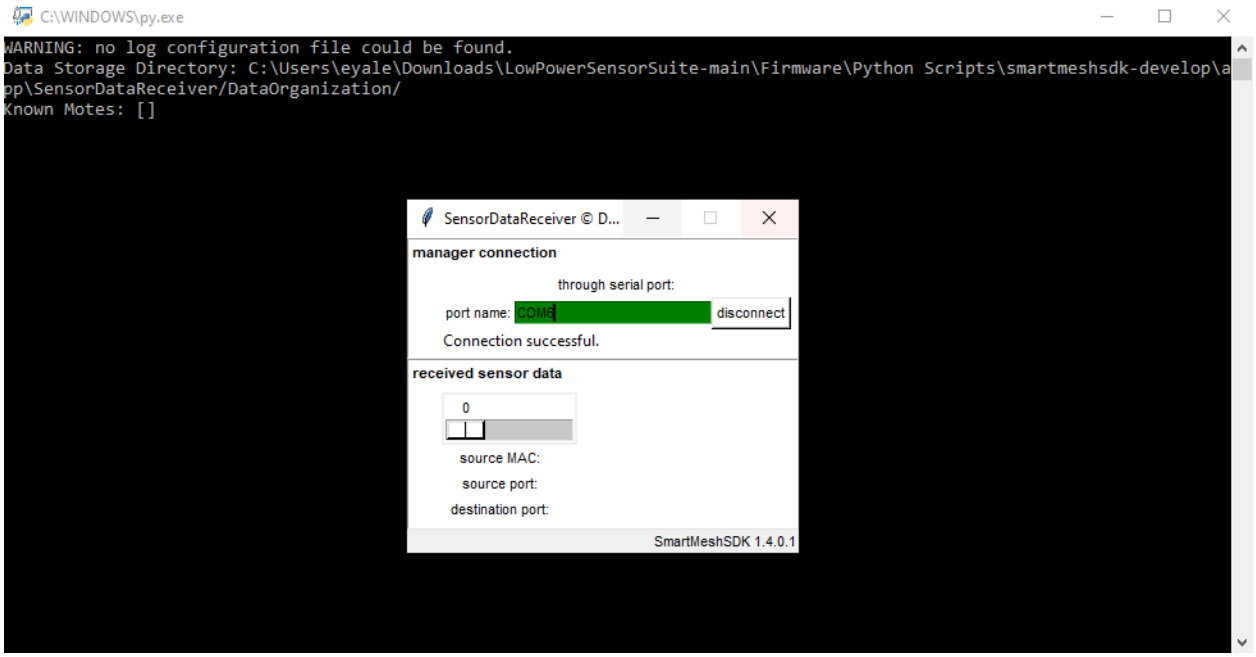
3. Open the "LowPowerSensorSuite-main" folder and go to *Firmware → Python Scripts → smartmeshsdk-develop → app → SensorDataReceiver*. In this location create a new folder called "DataOrganization" (There is NO space between Data and Organization).
4. In the Windows search bar type "Device Manager" and open the application. Then open "Ports (COM & LPT)" in the device manager. Plug in the DC2274A-A (blue receiver, also called a manager) into a USB port on your

computer. When you plug in the manager into a USB port on your computer, four new COM ports should appear in the device manager (see image below). Write down the **fourth** COM port that appears, in this case it is “COM6” but your COM port number may differ.



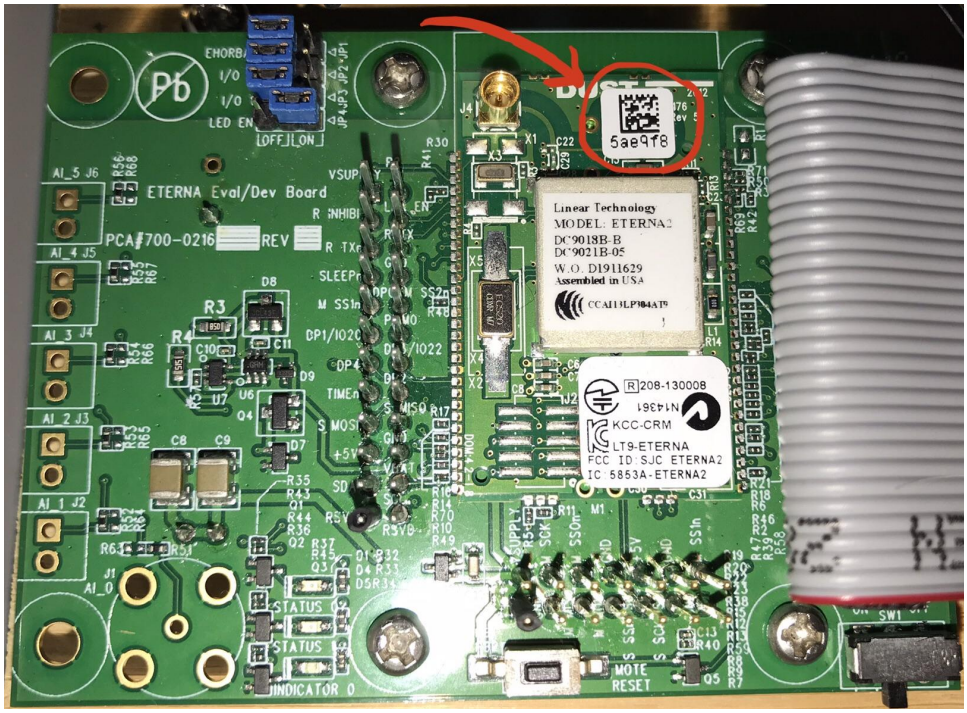
5. Now open the “SensorDataReceiver.py” application. For the port name put in the fourth COM port that appeared when you plugged in the manager and click connect (see image below). The text box will be shaded green if the

procedure was done correctly.



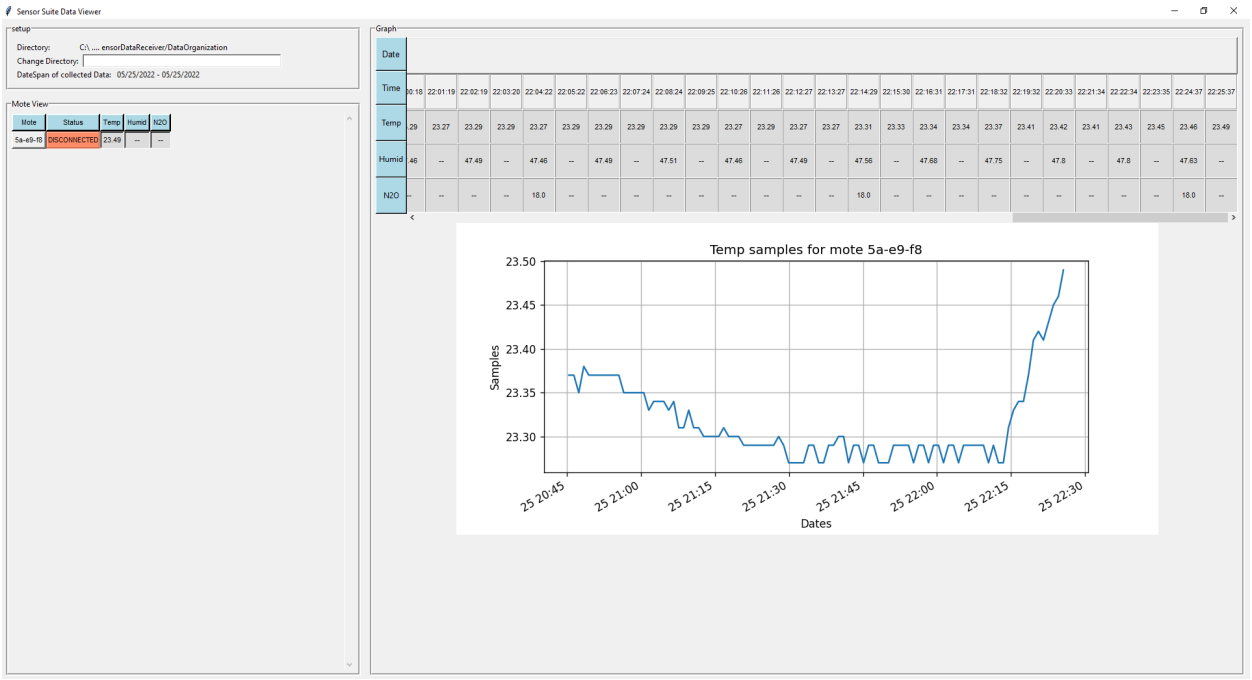
6. Now you can turn on the mote system by pressing on the switch located right next to the battery pack. After a minute and a half of the system being on, sensor data will start to be collected. Temperature will be collected 1 minute, humidity every 2 minutes, and N2O every 10 minutes. Each mote system will have a unique MAC ID. If you look at the smartmesh mote you are currently using you can see the MAC ID (see image below). If you go to the “DataOrganization” folder you will see a log file that was created with the following name “5ae9f8”. This information tells you which data log file

belongs to which smartmesh mote.

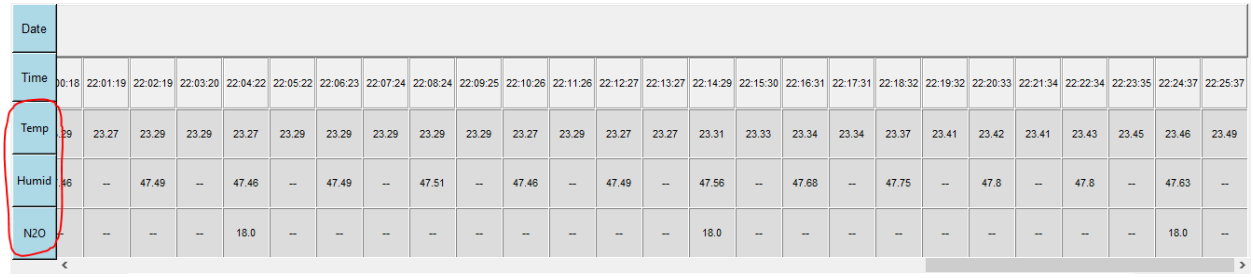


7. Now open the “SensorSuiteGUI.py” application. This application takes the data log file and represents the data in a readable manner. The top right table of the GUI will always display the latest 100 pieces of data from the log file. In the top right table “--” means that no data was collected for that sensor. This is because the sensors have different sampling rates as mentioned in Step 6. The graph in the GUI will display the same information as seen in the top right table but in a visual representation. The left table will list the MAC ID of the mote along with the status of operation of the mote, and the latest temperature, humidity, and N2O data. A full

picture of the GUI is shown below.



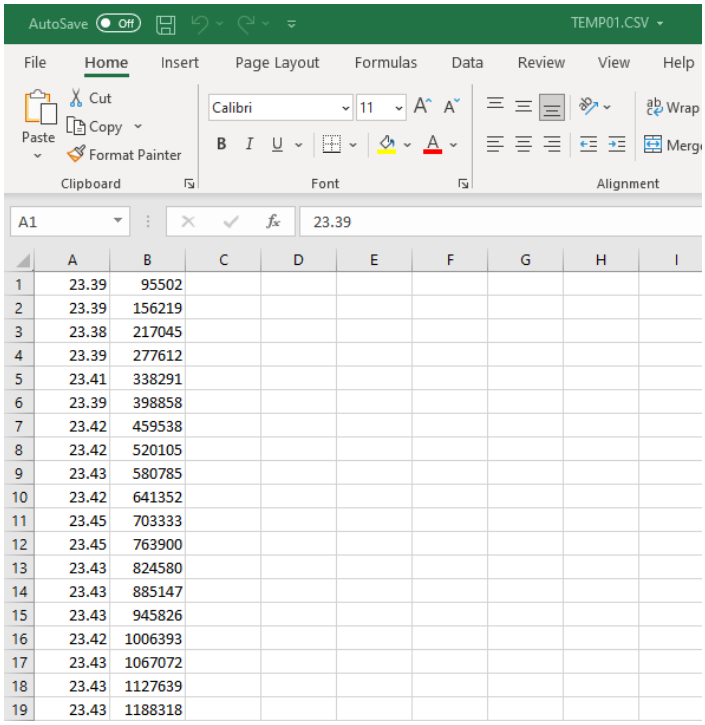
In the GUI, the following buttons are interactable (circled in red below). The “Temp” button will change the graph to display temperature data. The “Humid” button will change the graph to humidity data, and the “N2O” button will change the graph to N2O data.



8. The mote system is equipped with an SD card for backup storage of data. To access the data, remove the SD card from the MSP430 and insert it into an SD card reader on your computer. Each sensor will have its own folder for a total of three folders (TEMP, HUMI, N2O). Each SD card data log is in an

excel file format. If you open the file you will see two columns of data (see below). The first column corresponds to the sensor data value, in this case temperature sensor data. The second column corresponds to the “millis” value. The “millis” value is the amount of milliseconds that have passed since the system was turned on. This way each sensor data value can be associated with a corresponding timestamp. The user will need to write down the exact time that the system was turned on to be able to create a timestamp for a piece of data. An equation for this calculation is the following:

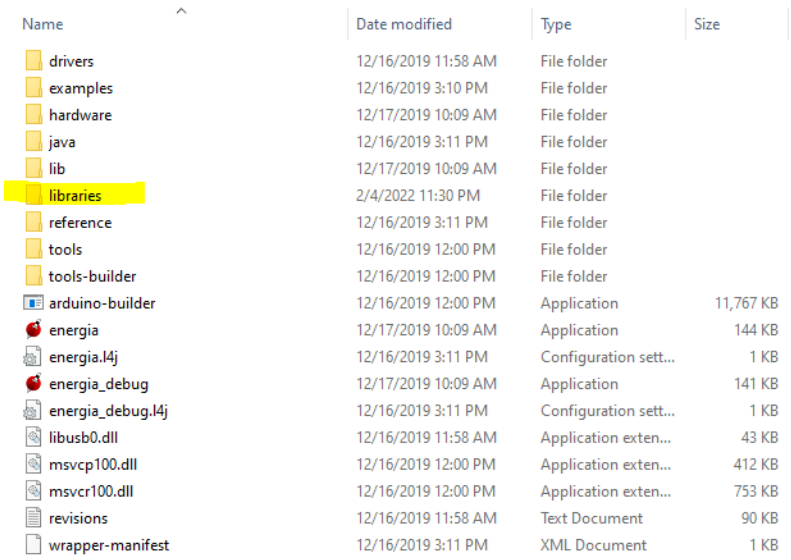
Timestamp for specific piece of data = Time system was turned on + millis value.



	A	B	C	D	E	F	G	H	I
1	23.39	95502							
2	23.39	156219							
3	23.38	217045							
4	23.39	277612							
5	23.41	338291							
6	23.39	398858							
7	23.42	459538							
8	23.42	520105							
9	23.43	580785							
10	23.42	641352							
11	23.45	703333							
12	23.45	763900							
13	23.43	824580							
14	23.43	885147							
15	23.43	945826							
16	23.42	1006393							
17	23.43	1067072							
18	23.43	1127639							
19	23.43	1188318							

Appendix B: User's Manual (Programming)

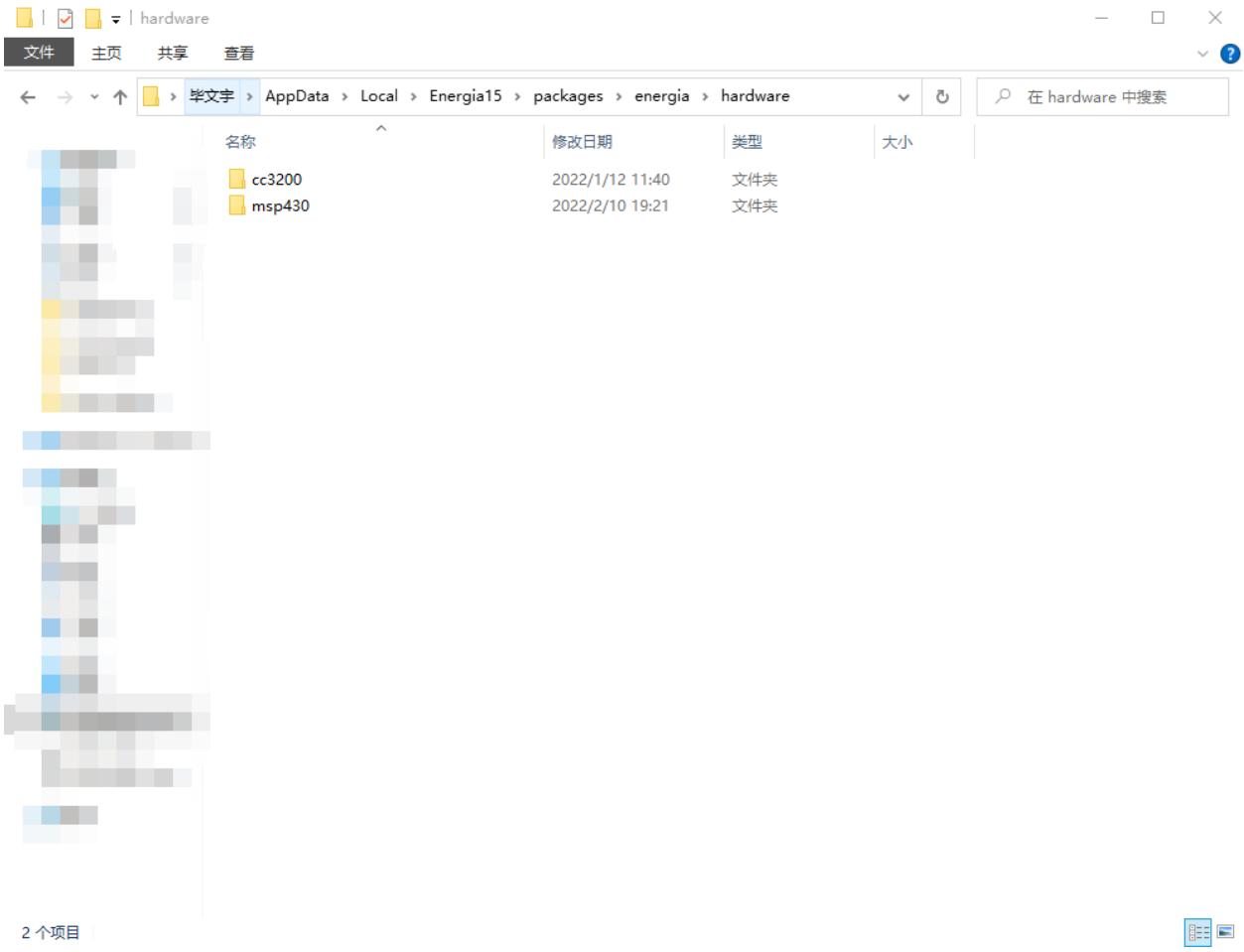
1. Install Energia IDE version 1.8.10E23 (https://energia.nu/download/)
2. Open the "LowPowerSensorSuite-main" folder and go to *Firmware → Energia Libraries*. Inside this folder should be two additional folders titled "IpMtWrapper" and "sm_clib". Both of these libraries are Energia smartmesh libraries. Make a copy of "IpMtWrapper" and "sm_clib" and place them in the "libraries" folder for Energia (see image below). Open the "LowPowerSensorSuite-main" folder again, make a copy of the folder titled "SDCardNew" and place it in the "libraries" folder for Energia as well.



Name	Date modified	Type	Size
drivers	12/16/2019 11:58 AM	File folder	
examples	12/16/2019 3:10 PM	File folder	
hardware	12/17/2019 10:09 AM	File folder	
java	12/16/2019 3:11 PM	File folder	
lib	12/17/2019 10:09 AM	File folder	
libraries	2/4/2022 11:30 PM	File folder	
reference	12/16/2019 3:11 PM	File folder	
tools	12/16/2019 12:00 PM	File folder	
tools-builder	12/16/2019 12:00 PM	File folder	
arduino-builder	12/16/2019 12:00 PM	Application	11,767 KB
energia	12/17/2019 10:09 AM	Application	144 KB
energia.l4j	12/16/2019 3:11 PM	Configuration sett...	1 KB
energia_debug	12/17/2019 10:09 AM	Application	141 KB
energia_debug.l4j	12/16/2019 3:11 PM	Configuration sett...	1 KB
libusb0.dll	12/16/2019 11:58 AM	Application exten...	43 KB
msvcp100.dll	12/16/2019 12:00 PM	Application exten...	412 KB
msvcr100.dll	12/16/2019 12:00 PM	Application exten...	753 KB
revisions	12/16/2019 11:58 AM	Text Document	90 KB
wrapper-manifest	12/16/2019 3:11 PM	XML Document	1 KB

3. Open the "LowPowerSensorSuite-main" folder and go to *N2O*, then unzip "msp430.zip" and replace it to the location below:

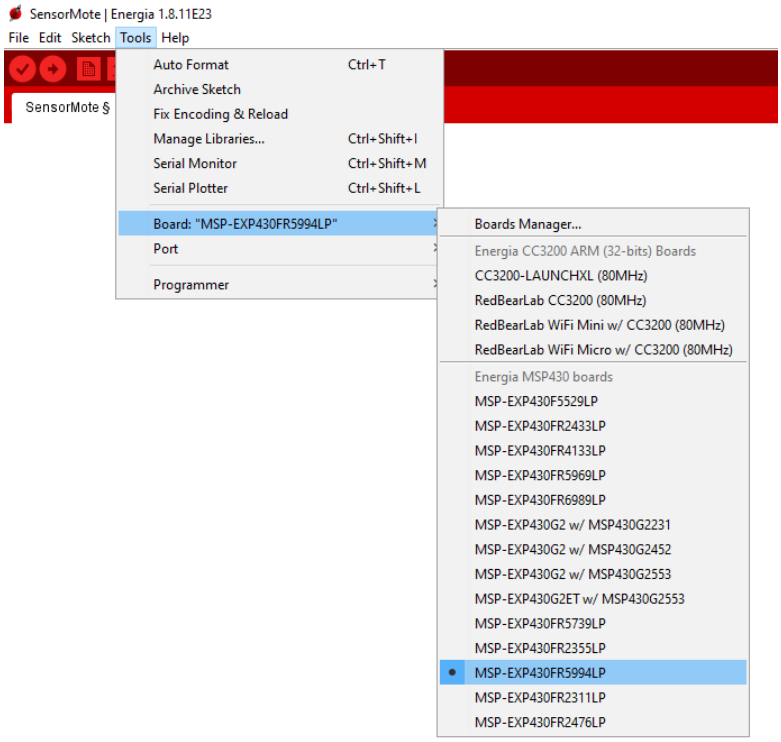
C:\Users\(User name)\AppData\Local\Energia15\packages\energia\hardware



Now you have successfully enabled P2.5 and P2.6 on the MSP430 to work as hardware serial ports. Complete configuration steps can be found at *N2O → P2.5&P2.6 serial port configuration document.pdf*

P2.5&P2.6 serial port configuration document.pdf

4. Launch Energia and make sure that in Energia the “MSP-EXP430FR5994LP” board is selected (see image below).



5. Upload the code (SensorMote_V0.1.ino) to the board using a USB cable. This code is located in *LowPowerSensorSuite-main → Firmware → Energia Code → SensorMote_ V0.1.*
6. Key notes about the Energia code (MSP430):
    a. SensorMote_V0.1.ino contains the main system code. All of this code is uploaded to the MSP430. Within the code you will find sections of code that are for collecting temperature and humidity data, N2O data, writing data to an SD card, transmitting data values through smartmesh, and making the system go into sleep mode.

b. Setting the sample rate for the various sensors:

```
#include <Wire.h>
#include <IpMtWrapper.h>
#include "sdcard.h"
#define N2OSerial Serial2
#define HIH8120Addr         0x27                    //I2C Addr
#define HIH8120Cmd          0xFF                    //HandShake Cmd
#define BytesToRead         4                       //Lens of Read Bytes
#define SampleNums          1                       //Total Sample Nums
#define SampleInterval      100                     //Ms
#define MeasureIntervalTemp     60000               //Measure every 60sec (60000ms) 1 minute
#define MeasureIntervalHumi     120000              //Measure every 120sec (120000ms) 2 minutes
#define MeasureIntervalN2O      600000              //Measure every 600sec (600000ms) 10 minutes
unsigned long MeasureTimeTemp = 0;                  //Interval Between Last SampleNums And Now SampleNums
unsigned long MeasureTimeHumi = 0;
unsigned long MeasureTimeN2O = 0;
unsigned long MeasureTimeN2OMinus1Minute = 0;
unsigned long SampleTime = 0;                       //Sample Interval
int SampleCounts = 0;                               //Current Sample Counter
/* Humiture */
float HumiVals[SampleNums];                         //Store History Humi Data
float TempVals[SampleNums];
float HumiSum = 0;                                  //Store History Humi Sum
float TempSum = 0;
float HumiAvg = 0;                                  //Calculate Average According to Humi Sum
float TempAvg = 0;
float Humi,Temp;                                    //Humi (%) and Temp (C)
bool dataSent = false;                              //Used to determine if MCU should go into sleep mode
byte N2OCmd[7] = {0x10,0x13,0x06,0x10,0x1F,0x00,0x58};
byte N2ORece[15];
float N2OData;
int minuteNine = 1;

IpMtWrapper       ipmtwrapper;
```

c. Check to see if 1 minute has passed (temperature sample rate), 2 minutes have passed (humidity sample rate), and 10 minutes have passed (N2O sample rate), get the appropriate sensor values and store

to SD card and transmit via smartmesh:

```
//============================== data generator ==================================
void generateData(uint16_t* returnVal) {
    returnVal[0] = 65535;                                   //Missing data filled with 65535 instead of 0 because 0 is valid data for N2O.
    returnVal[1] = 65535;                                   //Tried using -1 before but uint16_t data type only goes from 0 to 65535.
    returnVal[2] = 65535;
    if (millis() - MeasureTimeTemp >= MeasureIntervalTemp) {    //Measure temperature every 1 minute
        MeasureTimeTemp = millis();                         //Reset Timer
        GetHIHSensor();
        returnVal[0] = (int)(Temp * 100);                   //Values multiplied by 100 because only integers can be sent via SmartMesh
        SDwrite(SENSOR_TEMP, Temp, millis());
    }
    if (millis() - MeasureTimeHumi >= MeasureIntervalHumi) {    //Measure humidity every 2 minutes
        MeasureTimeHumi = millis();                         //Reset Timer
        returnVal[1] = (int)(Humi * 100);
        SDwrite(SENSOR_HUMI, Humi, millis());
    }
    if (millis() - MeasureTimeN2OMinus1Minute >= (MeasureIntervalN2O - 60000) && minuteNine) { //Sample rate of N2O minus 1 minute
        MeasureTimeN2OMinus1Minute = millis();              //Reset Timer
        digitalWrite(P4_3, HIGH);                           //Apply 3.3V to gate of transistor
        minuteNine = 0;
    }
    if (millis() - MeasureTimeN2OMinus1Minute >= MeasureIntervalN2O) {
        MeasureTimeN2OMinus1Minute = millis();              //Reset Timer
        digitalWrite(P4_3, HIGH);                           //Apply 3.3V to gate of transistor
    }
    if (millis() - MeasureTimeN2O >= MeasureIntervalN2O) {      //Measure N2O every 10 minutes
        MeasureTimeN2O = millis();                          //Reset Timer
        N2OGet();
        returnVal[2] = (int)(N2OData * 100);
        digitalWrite(P4_3, LOW);                            //Apply 0V to gate of transistor
        SDwrite(SENSOR_N2O, N2OData, millis());
    }
    dataSent = true;
}
```

d. Initialization of various sensors, SD card, and smartmesh:

```
void setup() {                              //put your setup code here, to run once:
  ipmtwrapper.setup(
      60000,                                // srcPort
      (uint8_t*)ipv6Addr_manager,           // destAddr
      61000,                                // destPort
      100,                                  // dataPeriod (ms)
      generateData                          // dataGenerator
  );
  Serial.begin(9600);                       //Init Serial
  Wire.begin();                             //I2C Begin
  initSDCard();                             //Init SD Card
  N2OSerial.begin(38400);
  pinMode(P4_3, OUTPUT);                    //Set P4_3 to be an output
  digitalWrite(P4_3, LOW);                  //Must set P4_3 to low (0V) when defining pin mode for P4_3 because it will output logic level high (3.3V) by default.
}
```

e. Sleep mode:

```
void loop() {
  // put your main code here, to run repeatedly:
  if (dataSent == true) {
      sleep(60000);                                 //Have MCU sleep for 60 seconds after data has been sent
  }
  dataSent = false;
  ipmtwrapper.loop();
}
```

f. Temperature and humidity collection function:

```
void GetHIHSensor(){
  byte HIHdata[4];                                  //Store Data Read
  byte HIHstatus = 0;                               //Sensor Status
  int Reading;                                      //Convert Data Read
  //float Humi,Temp;                                //Humi (%) and Temp (C)
  Wire.beginTransmission(HIH8120Addr);              //I2C HandShake
  Wire.write(HIH8120Cmd);
  Wire.endTransmission();
  delay(50);                                        //Wait for Sensor Prepare Data (50 ms Delay(~36.65 ms))
  Wire.requestFrom(HIH8120Addr, BytesToRead);       //Read 4 Bytes Data
  HIHdata[0] = Wire.read();
  HIHdata[1] = Wire.read();
  HIHdata[2] = Wire.read();
  HIHdata[3] = Wire.read();
  HIHstatus = HIHdata[0] & 0b11000000;              //Get Sensor Status
  if (HIHstatus != 0) {
    Serial.println("Status Bits Error!");
    //return 1;
  }
  /* Humidity is located in first two bytes */
  Reading = 0;
  Reading = HIHdata[0] << 8;                        //High 8 bits Data of Humi
  Reading |= HIHdata[1];                            //Combine High 8 bits and Low 8 bits Data
  Humi = float(Reading) / 16382.0 * 100.0;          //Convert it to Real Value: Reading/(2^14-2)*100
  /* Temperature is located in next two bytes, padded by two trailing bits */
  Reading = 0;
  Reading = HIHdata[2] << 6;                        //Only the first 6 bits of data are valid, and the remaining 2 bits are filled bits
  Reading |= (HIHdata[3] >> 2);
  Temp = float(Reading) / 16382.0 * 165.0 - 40;     //Convert it to Real Value: Reading/(2^14-2)*100
  /* Add Current Values To Array */
  HumiVals[SampleCounts] = Humi;
  TempVals[SampleCounts] = Temp;
}
```

g. N2O collection function:

```
void N2OGet(){
  N2OSerial.write(N2OCmd,7);
  delay(1000);
  if(N2OSerial.available()){
    byte recebyte = N2OSerial.read();
    Serial.print(recebyte, HEX);
    Serial.print("|");
    if(recebyte == 0x10){
      N2ORece[0] = 0x10;
      int i;
      int m = 0;
      char ReadArray[4];
      bool flag;
      while (N2OSerial.available()){
        flag = false;
        for(i = 1;i < 15;i++){
          N2ORece[i] = N2OSerial.read();
          Serial.print(N2ORece[i], HEX);
          Serial.print("|");
          if(i >= 7 && i <= 10){
            ReadArray[m++] = N2ORece[i];
          }
        }
        Serial.println();
        N2OData = *(float*)ReadArray;
        N2OData = N2OData / 10;
        Serial.print("N2O Data is: ");
        Serial.print(N2OData);
        Serial.println("ppm");
        if(N2OData == -2500){
          Serial.println("N2O Sensor Warmup!");
        }
      }
    }
  }
}
```

h. To change the amount of data that can be sent via smartmesh, in the Energia folder go to *libraries* → *IpMtWrapper* → *IpMtWrapper.cpp*. Open *IpMtWrapper.cpp* and find the section of code in the picture below. The current system has three sensors so that is why dataVal is dataVal[3] in the picture below. Follow the same logic flow as in the image below to add additional values to be sent via smartmesh. Only the parts outlined in red below need to be modified.

```
481  void IpMtWrapper::api_sendTo(void) {
482     dn_err_t err;
483     uint16_t dataVal[3];
484     uint8_t  payload[6];
485     uint8_t  lenWritten;
486
487     // record time
488     app_vars.fsmPreviousEvent = millis();
489
490     // log
491     Serial.println("");
492     Serial.print("INFO:     api_sendTo... returns ");
493
494     // arm callback
495     fsm_setCallback(&IpMtWrapper::api_sendTo_reply);
496
497     // create payload
498     app_vars.dataGenerator(dataVal);
499     //dn_write_uint16_t(payload, dataVal);
500     payload[0] = (dataVal[0] >> 8) & 0xff;
501     payload[1] = (dataVal[0] >> 0) & 0xff;
502     payload[2] = (dataVal[1] >> 8) & 0xff;
503     payload[3] = (dataVal[1] >> 0) & 0xff;
504     payload[4] = (dataVal[2] >> 8) & 0xff;
505     payload[5] = (dataVal[2] >> 0) & 0xff;
506
507     // issue function
508     err = dn_ipmt_sendTo(
509        app_vars.socketId,                              // socketId
510        app_vars.destAddr,                              // destIP
511        app_vars.destPort,                              // destPort
512        SERVICE_TYPE_BW,                                // serviceType
513        0,                                              // priority
514        0xffff,                                         // packetId
515        payload,                                        // payload
516        sizeof(payload),                                // payloadLen
517        (dn_ipmt_sendTo_rpt*)(app_vars.replyBuf)        // reply
518     );
519
520     // log
521     Serial.println(err);
522
523     Serial.print("INFO:     sending value: ");
524     Serial.print(dataVal[0]);
525     Serial.print(", ");
526     Serial.print(dataVal[1]);
527     Serial.print(", ");
528     Serial.println(dataVal[2]);
529
530     // schedule timeout event
531     fsm_scheduleEvent(SERIAL_RESPONSE_TIMEOUT, &IpMtWrapper::api_response_timeout);
532  }
```

7. Key notes about the Python applications for the GUI:

   a. The GUI is composed of three files which are found in *LowPowerSensorSuite-main → Firmware → Python Scripts → smartmeshsdk-develop → app → SensorDataReceiver*. Those three files are titled “SensorSuiteGUI.py”, SensorSuiteAPI.py”, and “GUI_GRAPHING.py”. SensorSuiteGUI is responsible for creating the visual elements of the GUI, SensorSuiteAPI reads the PC data log file, and GUI_GRAPHING creates a graph of the data using a MATLAB library.
   b. More information about the GUI can be found in the section titled User’s Manual (Operation) which is located above.

8. Useful resources in regards to smartmesh:
   a. https://dustcloud.atlassian.net/wiki/spaces/ALLDOC/overview?homepageId=1015815
   b. https://www.analog.com/media/en/technical-documentation/data-sheets/59012ipmfa.pdf
   c. https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc9018b-b.html#eb-overview
   d. https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc9021b.html
   e. https://www.analog.com/media/en/technical-documentation/user-guides/SmartMesh_IP_Tools_Guide.pdf
9. Software and version numbers:
   a. Python 3.10.1
   b. Energia 1.8.10E23

c. KiCAD 5.1.10

d. Code Composer Studio 11.2.0 (For energy trace)

e. Microsoft Visual Studio Code 1.67.2 (For editing python applications)

f. GitHub

Appendix C: Energy Trace Resources

Running energy trace video - https://www.youtube.com/watch?v=a1GC78RDTs4

How to use MSP430 with Code Composer Studio -

https://www.xanthium.in/msp430-launchpad-software-development-code-composer-studio-ccs-and-opensource-msp430gcc

Appendix D: SD Card Library

To use the SD Card Library

Copy the SD Card library to Energia/Libraries

How Library was Created

The library for the SD Card library is a modified library posted by user named geometrikal on the 43oh.com forum. The URL to the github for the library is: https://github.com/geometrikal/Jaffl

The library is called “Just Another Fat FS Library” or jaffl for short.

This library did not work immediately with the MSP430FR5994, specifically, the SD Card would not initialize. Based on the posts by user JKR, the following 2 modifications were made to file: jaffl.cpp.

- Line 121: Add 0 to begin(0)
- Line 42: Add SPIClass spi

After these edits, the SD Card would initialize and data could be written to the SD Card.

EndSeek function

There was a problem with the SD Card library. Each time a file was opened, all

existing data would be overwritten. It was discovered that whenever a file was opened, the file pointer started at the beginning of the file. This caused any write functions to begin writing at the beginning of the file, deleting existing data. Most FatFS libraries allow for an APPEND flag when a file is opened. An APPEND flag opens the file with the file pointer at the end of the file. This causes new data to be written at the end of the file.

Our solution was to write a function in the jaffl.cpp file called EndSeek(). This function is called after the file is opened. It causes the file pointer to move to the end of the file.

SD Card Functions

To interface with the SD Card library and send log data to the SD Card, 2 files were created:

1. SDCard.cpp
2. SDCard.h

These files include 2 functions:

1. void initSDCard(void).
a. This mounts the SD Card and is called when the system is first powered on.
2. void SDwrite(unsigned char mode, float value, unsigned long time)
a. This is the write function. It takes arguments to tell it which sensor type, the sensor value, and the millis() return value.
b. The function then writes the data to the SD Card.

For example, the following line of code writes the temperature sensor value. Look in file: SensorMote_V0.1.ino, line 44:

```
40      if (millis() - MeasureTimeTemp >= MeasureIntervalTemp) {    //Measure temperature every 1 minute
41          MeasureTimeTemp = millis();                             //Reset Timer
42          GetHIHSensor();
43          returnVal[0] = (int)(Temp * 100);                       //Values multiplied by 100 because only integers can be sent via SmartMesh
44          SDwrite(SENSOR_TEMP, Temp, millis());
```

Appendix E: Bills of Materials

Bill of Materials for the complete build.

Bill of Materials:	Team 13	Second Generation Low Power Environmental Sensor Suite							
Last Modified:	6/8/2022								
Schematic Version	1.1								
BOM revision:	1								

Quantity	Part References	Mfg	Mfg PN	Description	Dist	Dist Part Number	Cost Ea.	Cost Total	URL
1	1	Oshpark	N/A	Sensor Suite Junction Board	Osh Park	N/A	$35.50	$35.50	https://oshpark.com/
1	2	Texas Instruments	MSP-EXP430FR5994	MSP430FR5994 Launchpad	Digi-Key	296-45386-ND	$20.39	$20.39	https://www.digikey.com/en/products/de

									tail/texas-instruments/MSP-EXP430FR5994/6645208
1	3	Samsung	N/A	SD Card, 32 GB	Amazon	N/A	$17.99	$17.99	https://www.amazon.com/Samsung-MicroSDHC-Adapter-MB-ME32GA-AM/dp/B06XWN9Q99?th=1
1	4	Analog Devices	DC9018B-B	DC9018B-B Evaluation Mote	Digi-Key	DC9018B-B	$360.00	$360.00	https://www.digikey.com/en/products/detail/analog-devices-inc/DC9018B-B/4696309
1	5	Honeywell	HIH6120	SENS HUMI/TEMP 3.3V I2C 2% 4SIP	Digi-Key	480-5706-1-ND	$12.90	$12.90	https://www.digikey.com/en/products/detail/honeywell-sensing-and-productivity-solutions/HIH8120-021-001/4291627
1	6	Dynament	DS0002	NO2 Sensor	Dynament	DS0002	$0.00	$0.00	https://www.dynament.com/products/gas-sensors/
2	7	Yageo	RT1206BRD075KL	RES SMD 5K OHM 0.1% 1/4W 1206	Digi-Key	YAG5090CT-ND	$0.65	$1.30	https://www.digikey.com/en/products/de

									tail/yageo/RT1206BRD075KL/5936961
1	8	Otdorpatio	N/A	Enclosure, IP67 Waterproof	Amazon	N/A	$27.99	$27.99	https://www.amazon.com/dp/B09F8YGGHB?ref=ppx_yo2ov_dt_b_product_details&th=1
1	9	LeMotech	N/A	1/4" Cable Gland	Amazon	N/A	$0.55	$0.55	https://www.amazon.com/dp/B08ZYMDYHD?psc=1&ref=ppx_yo2ov_dt_b_product_details
1	10	Striveday	N/A	5-wire 1/4" cable, 3 ft.	Amazon	N/A	$2.01	$2.01	https://www.amazon.com/dp/B07Y32RQ2Z?ref=ppx_yo2ov_dt_b_product_details&th=1
1	11	ESUPPORT	N/A	5-wire automotive connector	Amazon	N/A	$2.46	$2.46	https://www.amazon.com/dp/B014P8XJQE?psc=1&ref=ppx_yo2ov_dt_b_product_details
10	12	Wirefy	N/A	heat shrink tubing, 1/8"	Amazon	N/A	$0.07	$0.70	https://www.amazon.com/dp/B084GDLS

									CK?psc=1&ref=ppx_yo2ov_dt_b_product_details
1	13	Ltvystore	N/A	Battery Holder, 4 slot, 18650	Amazon	N/A	$2.75	$2.75	https://www.amazon.com/dp/B07CWK81BJ?psc=1&ref=ppx_yo2ov_dt_b_product_details
1	14	MELIFE	N/A	USB breakout board	Amazon	N/A	$0.84	$0.84	https://www.amazon.com/dp/B07W7XMV3W?psc=1&ref=ppx_yo2ov_dt_b_product_details
1	15	Spater	N/A	Micro USB cable	Amazon	N/A	$1.60	$1.60	https://www.amazon.com/dp/B01FSYBQ9Q?psc=1&ref=ppx_yo2ov_dt_b_product_details
4	16	SHENMZ	N/A	18650 Battery, Lithium-ion, 3.7V	Amazon	N/A	$9.95	$39.78	https://www.amazon.com/dp/B071CHP4PL?psc=1&ref=ppx_yo2ov_dt_b_product_details
2	17	Assmann WSW	H3CCH-2006G	Ribbon Cable, 2x10	Digikey	H3CCH-2006G-N	$1.35	$2.70	https://www.digikey.

		Components				D			com/en/products/detail/assmann-wsw-components/H3CCH-2006G/1218561
1	18	Assmann WSW Components	IDSD-16-D-02.00-T-G-RW-R	Ribbon Cable 2x16	Mouser	200-IDSD16D0200TGRWR	$13.40	$13.40	https://www.mouser.com/ProductDetail/Samtec/IDSD-16-D-02.00-T-G-RW-R?qs=0lQeLiLlqyYrkofe6uZx%252BA%3D%3D
15	19	RuiLing	N/A	Jumper Shunt, 2.54mm	Amazon	N/A	$0.03	$0.52	https://www.amazon.com/dp/B07VJKCHVN?psc=1&ref=ppx_yo2ov_dt_b_product_details
1	20	Uxcell	N/A	8 pin adapter boards	Amazon	N/A	$0.32	$0.32	https://www.amazon.com/uxcell-Adapter-Board-Converter-25pcs/dp/B07FK4BQHL
							Total:	$543.70	

Bill of Materials - PCB

Last Modified:	6/8/2022
Schematic version:	1.1
PCB version:	1.00
BOM revision:	1.00

Quantity	Part References	P/NP*	Mfg	Mfg PN	Description	Dist	Dist Part Number	Cost Ea.	Cost Total	URL
1	U1	P	Oshpark	N/A	Sensor Suite Junction Board	Osh Park	N/A	$20.00	$20.00	https://oshpark.com/
1	U6	P	Honeywell	HIH6120	SENS HUMI/TEMP 3.3V I2C 2% 4SIP	Digi-Key	480-5706-1-ND	$12.90	$12.90	https://www.digikey.com/en/products/detail/honeywell-sensing-and-productivity-solutions/HIH8120-021-001/4291627
1	J7	NP	Dynament	DS0002	NO2 Sensor	Dynament	DS0002		$0.00	
105	J1-J9, J50/J51,J60/	P	MCIGICM	N/A	Pin Headers	Amazon	N/A	$0.01	$1.31	https://www.amazon.com/MCIGICM-Header-2-45mm-Arduino-Connector

	J61,J70/J71									/dp/B07PKKY8BX/ref=sr_1_3?crid=1S1YS41OR59XP&keywords=pin+headers&qid=1654734282&s=electronics&sprefix=pin+headers%2Celectronics%2C153&sr=1-3
2	R1,R2	P	Yageo	RT1206BRD075KL	RES SMD 5K OHM 0.1% 1/4W 1206	Digi-Key	YAG5090CT-ND	$0.65	$1.30	https://www.digikey.com/en/products/detail/yageo/RT1206BRD075KL/5936961
								Total:	$35.51	

Appendix F: Weekly reports

Week 1 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

1/14/2022

January Team Leader: Travis Johnson

- **This Week**
  - Team Review
    - Set up Google Drive for shared documents
      - Weekly Progress Reports
      - Rough Product Design Specification
  - Eyal Eynis
    - Added to Rough Product Design Specification
    - Researched low power solutions to wireless sensor networks (dynamic voltage scaling and reverse body biasing)
    - Read through some of the SmartMesh IP user guide and tried to create a wireless mesh network between mote and manager using Easy Start Guide.
  - Travis Johnson
    - Created Capstone Timeline document
    - Created GitHub collab site:
      - https://github.com/travis26pdx/LowPowerSensorSuite
    - Added to Rough Product Design Specification
    - Researched SmartMesh IP in journals.
      - Discovered that sensors can be connected directly to motes, and do not require a second MCU.
      - Study showed that a mote can run on 2 AA batteries for years. From 5.0 years to 16.8 years.
    - Began reading SmartMesh IP User Guide
  - Wei Yan
    - Added to Rough Product Design Specification
    - Installed proper drivers and programs to run Energia IDE
    - Successfully communicated with CC3200 as long with some hardware
    - Research on power scavenging
      - Solar and wind seem to the most viable option if we do decide to have some power scavenging.
  - Wenyu Bi

        - Set up Google Drive for shared documents
        - Added to Rough Product Design Specification
        - Research on using CC3200 to make sensors work
        - Research on the use of Smartmesh
            - Trying to find ways to make them work independently based on the article Travis found
- **Next week**
    - Team Plan
        - Find a way to use sensors directly with the motes so that a secondary MCU is not needed.
    - Eyal Eynis
        - Continue trying to create a wireless mesh network between mote and manager.
        - Once a mesh network has been established, I will try to send data from the mote to the manager.
        - Research more about the capabilities of the mote itself (processing power, sensor capability)
    - Travis Johnson
        - Configure SmartMesh IP
        - Continue reading SmartMesh IP documentation
        - Determine whether secondary MCU is needed / whether all sensors desired can be connected directly to SmartMesh IP.
    - Wei Yan
        - Use CC3200 to communicate with a sensor
        - Power characterize one or two sensors
        - Research a way to use the Smart mesh to control each sensor to complete eliminate the use of the CC3200
        - Find and understand the code that controls each sensor and dissect it to be able to run by itself and test its power characteristics
    - Wenyu Bi
        - Continue work on controlling sensor work with CC3200
        - Research on how to control sensors directly with Smartmesh
        - Measure the power consumption of various sensors when the device is operational
- **Questions / Problems**
    - Team Blocks
        - Lack of secure shared collab space on campus prevents permanent setup and active collaboration.
        - Can we buy more DC2274 so every team member can have one?
        - How to determine the battery type, voltage and expected life, without having done enough research sensors and the microcontroller.
    - Eyal Eynis
        - LED indicators on the mote should switch to a different lighting pattern when the mote has successfully joined the mesh network, but the LEDs did not do this. I need more time to resolve the issue.
    - Travis Johnson

        - Did not have DC2274 USB manager to work on SmartMesh IP at home.
        - Did not find documentation how to convert a mote into a root node.
    - Wei Yan
        - Trying to understand code, due to poor documentation. I will resolve this issue this week.
    - Wenyu Bi
        - Due to the small number of devices, everyone got a short test time.

Week 2 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

1/21/2022

January Team Leader: Travis Johnson

- **This Week**
    - Team Review
        - The focus of this week was to better understand the SmartMesh system and the microcontroller used to control the various sensors.
    - Eyal Eynis
        - Successfully created a mesh network between mote and manager.
        - Was able to ping the mote to receive temperature and battery level data (built in temperature sensor).
        - Read through some of the SmartMesh SDK documentation and the mote's datasheet (DC9018B-B / LTP5902-IPM).
    - Travis Johnson
        - Made rough calculations of each sensors power consumption.
            - High power (>100mAh/year: O2, CO2)
            - Low power (<10 mAh/year: Temp, RH, accelerometer)
            - Unclear (wind - no part number, moisture - no datasheet)
        - Created Rough Project Proposal
            - Incorporated PDS and Timeline into Proposal document
    - Wei Yan
        - Researched new microcontroller in order to improve last's year performance.
            - New microcontroller is MSP430FR2335
        - Worked on understanding the code and libraries needed for Smart mesh and microcontroller interaction
        - Researched alternative microcontrollers for back up plan in case MSP430FR2335 takes too long to ship.
            - MSP430FR5994 is readily available in Digi key
    - Wenyu Bi

        - Successfully use CC3200 to control CO2 and water contact sensor work
        - Sensor data can be obtained from the serial monitor
        - Use the MOS module to control the interval between each measurement of the sensor (not fully completed)
- **Next week**
    - Team Plan
        - Main focus will be to develop a clearly defined project proposal.
    - Eyal Eynis
        - Research how to transfer sensor data from MCU to mote and then to manager.
        - Add to project proposal document.
    - Travis Johnson
        - Complete Draft Project Proposal
    - Wei Yan
        - Add to project proposal document
        - Keep working on last year's code to be able to modify it to the new microcontroller
    - Wenyu Bi
        - Combine the codes for controlling the three sensors through the MOS module, get the device up and running
        - Add to project proposal document.
- **Questions / Problems**
    - Team Blocks
        - Could not obtain shared collab space.
    - Eyal Eynis
        - I tried connecting a SmartMesh mote to the mesh network but it would not connect. I later figured out that the mote that I was using was actually configured as a manager and verified this through software and the LEDs on the mote. The issue has been fixed.
    - Travis Johnson
        - COVID shutdowns impacting personal life. I will recruit a team member to assist in the documentation work.
    - Wei Yan
        - Understanding the flow of the code of last year's group seems to be a problem due poor documentation and little to no comments.
    - Wenyu Bi
        - When trying to get the sensor to work, the code for last year's project was so messy that I had to write the new code to make the device work.
        - When testing the CO2 sensor, due to the parameters of the CC3200, the ADC range is 0-1.5V, and the voltage signal output by the CO2 sensor is 0-3.3V, so a pair of resistors are needed to divide the voltage (Fixed)

Week 3 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

01/28/2022

January Team Leader: Travis

- **This Week**
  - Team Review
    - Decided to use Microcontroller MSP430
      - Ordered 4 dev boards
  - Eyal Eynis
    - Added to project proposal document.
    - Read documentation on how to connect smartmesh IP mote to MSP430 on SmartMeshSDK website.
    - Was able to experiment with SensorDataReceiver and Upstream python scripts available as part of the SmartMeshSDK.
      - SensorDataReceiver connects to the smartmesh manager and displays data received from the mote.
      - Upstream allows the mote to send a 16 bit value to the manager.
  - Travis Johnson
    - Continued Project Proposal
    - Began GANTT chart.
  - Wei Yan
    - Added to Project Proposal
    - Researched N20 Datasheet
      - 240mW of power consumption
      -
  - Wenyu Bi
    - Added to Project Proposal
    - Continue to use CC3200 to make three sensors work and record data at the same time, and add MOS Module to successfully control the device to work every 30 minutes (including 180s preheating time)
- **Next week**
  - Team Plan
    - Start implementing code into the MSP430 to make the sensors work with the new board. Also connect the smartmesh mote to the MSP430.
  - Eyal Eynis
    - Try to connect the smartmesh mote to the MSP430.
    - Understand the SensorDataReceiver and Upstream python scripts better in order to start coding transmission of data from MSP430 to the mote and then to the manager.

    - Travis Johnson
        - Finalize project proposal
        - Complete GANTT chart
        - Obtain FA/IS sign-off on Project Proposal, and submit to Andrew
    - Wei Yan
        - Start Programming MSP430
            - Connect to Smart mesh
            - Write code for N20 sensor
    - Wenyu Bi
        - Research on how to control sensors with MSP430
        - Try to add N2O sensor and make it work
- **Questions / Problems**
    - Team Blocks
        - Have not been able to experiment with the MSP430 since it did not arrive yet.
    - Eyal Eynis
        - We were not able to get the MSP430's this week so unfortunately we have not been able to start implementing code onto the new board yet.
    - Travis Johnson
        - What is the budget goal?
        - Need review of stakeholders in project proposal.
        - Will enclosure be weather "proof" or does "resistant" suffice?
    - Wei Yan
        - N20 sensor seem to consume a lot of power
        - N20 can be ordered with a different start up time than normal. Normal start up time is 1 min. If we can possible lower that, our power consumption would decrease significantly.
    - Wenyu Bi
        - Since the team didn't get the new microcontroller, it was only able to test with boards from last year's project

Week 4 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

02/04/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        - Completed Project Proposal, submitted for approval.
    - Eyal Eynis

        - Added to and revised project proposal document.
        - Added to project proposal presentation.
    - Travis Johnson
        - Finalized project proposal document
        - Created GANTT chart
        - Prepared presentation
    - Wei Yan
        - Revised Project Proposal
        - Prepared presentation
    - Wenyu Bi
        - Revised project proposal
        - Prepared presentation
        - Researched on NO2 sensor and MSP430
- **Next week**
    - Team Plan
        - Attempt to establish communication between the MSP430 and the smartmesh IP mote, research MSP430 on board storage, and get usable sensor data.
    - Eyal Eynis
        - Physically connect smartmesh IP mote to MSP430.
        - Start writing the code for the smartmesh to gather sensor/custom data from MSP430 and send the data to the manager/computer.
    - Travis Johnson
        - Research and preliminary code for data storage for MSP430.
    - Wei Yan
        - Connect Smart mesh IP to MSP430
        - Connect NO2 sensor to MSP430
    - Wenyu Bi
        - Trying to get some sensors working with MSP430
        - Trying to help connect MSP430 and Smartmesh
- **Questions / Problems**
    - Team Blocks
        - Importance of the Capstone website for course requirements was not made clear early in the course.
    - Eyal Eynis
        - Some criteria in evaluation form unclear such as what are “external specifications”?
    - Travis Johnson
        - Capstone's website is outdated and lacks detail of student expectations.
        - Requirements in Evaluation Form are unclear and unexplained.
        - Having an example (or boilerplate) Project Proposal document, showing how to fulfill requirements would have reduced time spent on the proposal document.
    - Wei Yan
        - MSP430 hasn’t arrived.

    - Wenyu Bi
        -

Week 5 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

02/11/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        - Obtained MSP430FR5994 boards and began focused technical work.
    - Eyal Eynis
        - Connected MSP430 to SmartMesh mote using jumper wires.
        - Coded MSP430 to send custom/test data to SmartMesh mote and then the mote sends data to the manager with the data being seen on the computer screen.
        - Wrote a user guide (v0.1) on how to connect the MSP430 to the SmartMesh mote and is available to view on the team's GitHub.
    - Travis Johnson
        - Research SPI, SD Card, and MSP430 datasheets
        - Successfully wrote to SD Card using CCS pre-packaged “out-of-box” experience.
        - Obtained FATFS drivers written for Energia IDE. Which did not work immediately upon implementation and requires more changes. Program sent an “SD Card not enabled” error when attempting to open file on SD card inside the FR5994’s built-in SD card.
    - Wei Yan
        - Researched low power modes for MSP430.
    - Wenyu Bi
        - Focused on converting CC3200 code to MSP430 code to get the sensors working
- **Next week**
    - Team Plan
        - Continue technical work.
    - Eyal Eynis
        - Start writing the code that will allow the MSP430 to send sensor data to the mote and then to the manager/computer instead of custom/test data.
    - Travis Johnson
        - Get successful code for SD Card read/write using the simpler FR2355

board, to bypass the internal configuration of the FR5994.
    - Wei Yan
        - Figure out which clocks does sensors and smart mesh interface use in order to determine lowest low power mode we can get away with.
    - Wenyu Bi
        - Continue to complete the code that controls the sensor
        - Control the power supply of the sensor using the MOS module
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - At first I was not able to upload code to the MSP430 because the computer could not recognize the device. I installed the correct drivers and the issue is now resolved.
        - Initially after successfully uploading the code to the MSP430, the MSP430 was not able to communicate with the mote. I fixed the issue by connecting some of the jumper wires to a different set of receive and transmit pins found in the MSP430's datasheet.
    - Travis Johnson
        - The SD Card driver written for Energia IDE uses the G2553 board and assumes using a breakout board. Have not found explicit instructions for FR5994 and Energia IDE.
        - Need to successfully implement configuration (pull up resistors, LMP mode) for FR5994 to enable SD card on Launchpad.
    - Wei Yan
        - I had some problems connecting to MSP430. I am not sure what CLK does the sensors and Smart mesh interface use to be able to know which low power mode we can get away with.
    - Wenyu Bi
        - I2C is very weird. When using OLED for testing, it works normally, but nothing happens when connecting HIH8120 (temperature and humidity sensor)

Week 6 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

02/18/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review

        -
    - Eyal Eynis
        - Coded MSP430 in Energia to send an array of values to the manager/computer instead of a single value however the MSP430 serial monitor does not show multiple values being sent, only one value.
        - Modified the SensorDataReceiver Python script to take the array of values and to store them into a log file but for some reason when new values get sent to the manager, the manager disconnects and the values do not get stored. Disconnection of the manager did not happen when only one value was being sent from the MSP430.
    - Travis Johnson
        - Behind schedule. I found an SD Card driver written for the MSP430FR5994 that works with Energia IDE, but it did not work, and freezes during initialization. This is a similar problem other users report online. After some debugging, I fixed the freeze, but now the SD card will not mount.
    - Wei Yan
        - Successfully communicated to MSP430.
        - I was able to test different different low power modes and research a bit more what each CLK did and what would be the best power optimization we can do.
    - Wenyu Bi
        - The control of the temperature and humidity sensor is completed, and the data of the sensor can be obtained
- **Next week**
    - Team Plan
        -
    - Eyal Eynis
        - Continue debugging the Python (using Visual Studio Code program) and Energia code to fix the array of values not being sent and stored. Also fix the manager connection when new values arrive.
        - Implement Wenyu's sensor code into the transmission of data code so that actual sensor data can be transmitted via the smartmesh.
    - Travis Johnson
        - Solve problem to mount the SD Card so that I can create/open a file and write to it.
    - Wei Yan
        - Write small code to utilize low power mode with external LED as load.
    - Wenyu Bi
        - If I can get a NO2 sensor I will try to control the new sensor and get the data
        - If not getting a new sensor, I can try to help Eyal trying to transmit sensor data via smartmesh
- **Questions / Problems**
    - Team Blocks

        - The timeline scheduled code for NO2 sensor this week. Since NO2 has not arrived yet, and not expected for many weeks, this will be delayed.
    - Eyal Eynis
        - Need more time to debug the SensorDataReceiver Python script which is responsible for storing into a log file the array of values being sent from the MSP430.
        - Also need more time to debug the Energia MSP430 code which is supposed to show an array of values being sent to the manager in the serial monitor but only one value appears in the serial monitor.
    - Travis Johnson
        - The driver did not work and needed debugging.
    - Wei Yan
        - No questions
    - Wenyu Bi
        - I2C problem solved: https://github.com/energia/msp430-lg-core/issues/62
        - This version of energia's I2C itself is flawed. There is a fix already committed into the master branch but not in the release. So the I2C content of the version we downloaded is still wrong. Download the master folder in GitHub and replace all the original folders to work properly.

Week 7 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

02/25/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        - Made progress on all fronts.
    - Eyal Eynis
        - On schedule.
        - Successfully debugged the SensorDataReceiver python script to get a piece of data from the smartmesh mote and store the data into a log file on the computer.
        - Successfully implemented Wenyu's sensor code (for temperature) into the transmission of data code so now temperature data is collected by the MSP430 and then sent to the smartmesh mote and then sent to the manager where the manager stores the temperature data in a log file with the date and the time received.

    - Travis Johnson
        - Progress made, but still behind schedule by 1 week.
        - I successfully created code that creates a folder, file and writes to file onto an SD Card, using Code Composer Studio IDE. The goal is to meet this objective using Energia IDE.
    - Wei Yan
        - Ahead of schedule.
        - Successfully put MSP430 into low power mode.(Power in low power mode ~2uW in active mode ~9.7mW)
        - Implemented Energy trace program to measure power consumption of MSP430 at all times.
    - Wenyu Bi
        - On schedule. (NO2 sensor goals on hold pending arrival)
        - Rewrite a code that only controls the HIH8120 temperature and humidity sensor (test the environment every ten seconds)
        - Hand over the temperature and humidity sensor, wiring diagram and I2C problem solving method to Eyal.
- **Next week**
    - Team Plan
        - Next week is the final week for preliminary technical development. The week after begins designing the schematic and breadboarding.
    - Eyal Eynis
        - Figure out a way to send data from multiple sensors to the manager so that for example temperature and humidity can both be sent via smartmesh. Maybe by sending a value from a sensor based on the sampling time.
    - Travis Johnson
        - Port SD Card code to Energia IDE to create and write to files onto SD Card.
    - Wei Yan
        - Connect Energy Trace program to see power consumption of MSP430 with Temp and Humidity sensor.
        - See how Smart Mesh interacts with low power modes.
    - Wenyu Bi
        - Try to use smartmesh to get the data collected by the sensors
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - No questions.
    - Wei Yan
        - Any updates on NO2 sensor?
    - Wenyu Bi
        -

Week 8 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

03/04/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    -
  - Eyal Eynis
    - On schedule.
    - Worked on setting up Demo on Wei's laptop.
    - Was able to send both temperature and humidity data via smartmesh however sample rate is the same for both sensors.
    - Uploaded recent Energia code and Python scripts to team's GitHub.
  - Travis Johnson
    - 1 week behind schedule.
    - Completed SD Card driver. It now writes to SD Card.
    - Uploaded SD Card code to github.
  - Wei Yan
    - On schedule.
    - Worked on setting up Demo
  - Wenyu Bi
    - On schedule.
    - Completed the combination of the board sleep function, and tried to write the sensor data to the SD card and back it up.
- **Next week**
  - Team Plan
    - Begin design files: Schematic
  - Eyal Eynis
    - Adjust code to allow different sampling rates for different sensors.
  - Travis Johnson
    - Work with Wenyu on SD Card writing function.
    - Design rough schematic.
  - Wei Yan
    - Help Eyal to send data array to smart mesh.
  - Wenyu Bi
    - Complete the sensor sleep wake-up function and the sd card data writing function, combining the three parts of the work
- **Questions / Problems**

- Team Blocks
  -
- Eyal Eynis
  - The “IpMtWrapper” library for Energia has a function called “setup” that has an argument which is the number of milliseconds between transmission of data. I am having a hard time understanding how to take this argument into account when trying to code different sampling rates for different sensors.
  - By inspecting the libraries that last year’s team provided on the GitHub, it seems as if they changed some of the code in the “IpMtWrapper” library in order to be able to send an array of values via smartmesh but this change of code is not documented which makes it difficult to be implemented into the current code.
- Travis Johnson
  - None.
- Wei Yan
  - Having trouble understanding how the previous group modified the library to send an array of values to smart mesh.
- Wenyu Bi
  - The SD card I have has too much memory and the board cannot be recognized properly. I bought two new SD cards with small memory and waited for them to arrive for testing.

Week 09 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

03/11/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    -
  - Eyal Eynis
    - 1 week behind schedule.
    - Successfully modified “IpMtWrapper” library to be able to send an array via SmartMesh.
    - Sensor data can now be sampled at different times. Currently I have set up temperature to be sampled every 10 seconds, humidity every 20 seconds, N2O every 30 seconds (placeholder value was used for N2O since we do not have N2O sensor yet). SmartMesh mote transmits data every 10 seconds.

        - Fixed a bug that would post the current date into the log file every time data was received. A new date will only be logged now if it is actually a new date.
    - Travis Johnson
        - Design Files On schedule
        - SD Card code 1 week behind schedule.
        - Created rough draft of schematic
        - Helped Wenyu debug SD Card code
    - Wei Yan
        - On schedule
        - Started to look into last year's group GUI program and look at other options
    - Wenyu Bi
        - Completed the function of writing sensor data to SD card. Our program can create different folders and save the data collected by different sensors as CSV files.
- **Next week**
    - Team Plan
        -
    - Eyal Eynis
        - Start developing the GUI so that live data can be displayed.
    - Travis Johnson
        - Continue debugging SD Card code.
        - Create Bill of Materials
        - Continue with schematic
    - Wei Yan
        - Choose optimum GUI program and start developing GUI.
    - Wenyu Bi
        - Research and solve how to store unlimited data in the SD card (now only ten), do research in this area and achieve the function.
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - Since we are connecting devices together, which each have their own schematics drawn on their datasheets, I want to simply refer to these schematics on our schematic, and add in the connecting lines. What is the proper way to add references to other schematics in a schematic?
    - Wei Yan
        - No questions
    - Wenyu Bi
        - No questions

Week 10 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

03/18/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    - Completed CATME surveys.
    - GUI - behind schedule 1 week.
    - Low power mode - on schedule.
    - Design files - on schedule.
    - SD Card - behind schedule due to Append function issue.
  - Eyal Eynis
    - Tried to make last year's GUI run but there has been no success.
    - Decided to write a new GUI using Python.
    - Implemented sleep mode into the current code (MCU now sleeps for 10 seconds and then sends data and then goes back to sleep).
  - Travis Johnson
    - Continued schematic rough draft
    - Created Rough Bill of Materials
    - Began rough model of PCB and final product.
    - Investigated adding APPEND function to SD card library (file: ff.c). Created an issue on Github site since this may take some time to resolve.
  - Wei Yan
    - Worked on understanding GUI from the previous year.
    - Decided that we will just write a new GUI
    - Collaborated with Eyal on how to implement sleep mode with current code.
  - Wenyu Bi
    - Worked on using sd card to store unlimited data.
- **Next week**
  - Team Plan
    - Finalize minimally (but complete) functioning breadboard prototype.
    - Finalize initial design files, in preparation for PCB design to occur on following week.
  - Eyal Eynis
    - Continue developing the GUI program using Python.
  - Travis Johnson

        - Finish Schematic
        - Finish Bill of materials
        - Continue work on adding Append function to fatfs library.
    - Wei Yan
        - Continue working on GUI program.
    - Wenyu Bi
        - Continue working on solve the data overwriting problem of sd card, and improve the combination of storage function and previous code
- **Questions / Problems**
    - Team Blocks
        - Finals week slowed down work on Capstone project.
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - Issue created on Github site: The SD CARD FATFS library does not contain an Append flag. Therefore, everytime a file is opened, the pointer is set to 0, so any subsequent writes will overwrite existing data.
            - Possible Solutions:
                - 1. Add an append function to the library.
                - 2. Use the lseek function to move the pointer to the end of the file after open.
            - Can FA or IS provide additional solution possibilities?
    - Wei Yan
        - Does the GUI have to be live data?
    - Wenyu Bi
        - No questions.

Week 11 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

03/25/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        -
    - Eyal Eynis
        - I finally managed to get last year's GUI to run by fixing more syntax errors and importing additional libraries.
            - Last year's GUI required 10 data values to be in the log file to

operate correctly, so I modified the GUI code to be able to work with 3 data values (temperature, humidity, N2O).
- I also modified last year's GUI to update the data table every time new data was stored into the log file so that live data could be read.

■ I integrated Wenyu's and Travis' SD Card code into the current code. Temperature, humidity, and N2O data get sent via SmartMesh and are also stored on the SD card, then the MSP430 goes to sleep for 10 seconds and the cycle repeats.

○ Travis Johnson
■ SD Card code completed.
- Fixed the append problem by writing a function into the library that moves the pointer to the end of the file.
- Wenyu and I completed the SD Card library so that it can be easily integrated into the main code.

■ Finished Schematic, as far as possible.
- Need NO2 sensor before adding it to schematic.

■ Completed Bill of Materials.

○ Wei Yan
■ Worked on GUI for understanding what needs to be changed to get basic GUI working.
■ Modified last year's GUI to match the tables to our actual sensors.

○ Wenyu Bi
■ Completed the improvement of the sd card function, so that it can store unlimited data instead of overwriting the data every time it is stored.
■ Completed the encapsulation of the sd card library together with Travis, which is convenient for Eyal to do the functional test of the prototype.

- **Next week**
○ Team Plan
■ Complete the Test Plan

○ Eyal Eynis
■ Continue developing the GUI for the project and attempt to fix the issues mentioned in the Questions / Problems section.

○ Travis Johnson
■ Continue finished device model.
■ Begin PCB Layout.

○ Wei Yan
■ Keep working on GUI to make the graph show our actual values.

○ Wenyu Bi
■ Complete the Test Plan.
■ Debug the function of the NO2 sensor after obtaining the NO2 sensor.

- **Questions / Problems**
○ Team Blocks
■

    - Eyal Eynis
        - The graphs in the GUI do not plot the data correctly. I need more time to understand the code used for graphing data.
        - If more than two dates are present in the log file, the GUI will display the first date and the last date instead of all of the dates. Need more time to solve this issue.
    - Travis Johnson
        - For Bill of Materials: How much does the NO2 sensor cost? Is supplier Dynament or someone else?
        - I need NO2 sensor before finalizing pinout and trace on schematic, and PCB.
    - Wei Yan
        - Do we want the GUI to be interactive at all or just for display?
    - Wenyu Bi
        - No question

Week 12 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

04/01/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        -
    - Eyal Eynis
        - Adjusted GUI code to display the seconds in the timestamps instead of just hours and minutes.
        - Adjusted GUI code to display all of the dates from the log file where each date displayed on the GUI will span an amount equal to the number of samples that was collected for that date.
    - Travis Johnson
        - Finished overall schematic.
        - Create Rough junction board schematic.
        - Finished bill of materials
        - Drew model
        - Determined enclosure: Polycase SK-18.
        - Behind schedule on PCB design.
    - Wei Yan
        - Adjusted GUI, so that all buttons now work and they switch to correct graph.

        - Adjusted sizes of tables and graph. Also, made sure both tables show the correct values.
    - Wenyu Bi
        - Debugged the code and added N2O's control code into my original code according to the information provided by N2O's datasheet. Faster testing when sensors arrive.
        - Update on test plan information: Andrew emailed that he would provide us with the detailed requirements of the test plan later, and postponed the due date
- **Next week**
    - Team Plan
        -
    - Eyal Eynis
        - Continue developing the GUI.
    - Travis Johnson
        - Finish PCB design.
        - Order PCB.
    - Wei Yan
        - Plot the correct set of data in GUI. It seems the x and y axis label are messed up
    - Wenyu Bi
        - Work on the test plan based on the information provided by Andrew
        - After getting the N2O sensor, test the overall function of the code (sensors and SD card part)
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - Is there a preferred connector type for external components?
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - No questions.

Week 13 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

04/08/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    -
  - Eyal Eynis
    - Graphs on the GUI display the last 10 pieces of data from the log file.
    - Graphs on the GUI are updated when new data arrives so live data can be seen.
  - Travis Johnson
    - Finished PCB.
    - Need to order PCB. (Behind schedule)
  - Wei Yan
    - Removed date drop down menu because it didn't work.
    - Removed span and day scroll bar because they didn't work.
    - Managed to intercept 0 from data in the table to change to a "--"
  - Wenyu Bi
    - Drafting the test plan
- **Next week**
  - Team Plan
    -
  - Eyal Eynis
    - Work with Wei to set up energy trace with full system.
    - Further develop GUI
      - Fix bugs
        - When in fullscreen, the data table won't update.
      - Other tasks
        - Making the size of the right table bigger so that data can be easily read.
  - Travis Johnson
    - Produce Gerber files
    - Order PCB
  - Wei Yan
    - Intercept 0 from graph in order to not plot 0 in graph.
    - Work with Eyal to set up energy trace with full system
  - Wenyu Bi
    - Continue to improve the test plan
    - Test and troubleshoot N2O sensor functionality(If we can get the sensor next week)
- **Questions / Problems**
  - Team Blocks
    -
  - Eyal Eynis
    - No questions.
  - Travis Johnson
    - None.
  - Wei Yan

        - No questions
    - Wenyu Bi
        - No questions

Week 14 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

04/15/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        -
    - Travis Johnson
        - Finalized the PCB.
        - Uploaded to OshPark and ready to place the order.
        - 1 week behind schedule.
    - Wei Yan & Eyal Eynis
        - We set up energy trace with the temperature and humidity sensor. Energy trace also gives us an approximate battery life when the system is run by a 2xAA battery. With just the temp/humid sensor the system can run for 1 month and 15 days. We encountered a new bug, which is the fact that the GUI can't open when there is more than 4 hours worth of data. We believe this is because the table can't store that much data. We will try to fix this by making the table show only a certain number of last sensor values.
    - Wenyu Bi
        - Completed the code to control the N2O sensor.
        - Continue to write part of the test plan.
- **Next week**
    - Team Plan
        -
    - Eyal Eynis
        - Fix the GUI problem mentioned above.
        - Fix the bug where when the GUI is in full screen then the table won't update when new data arrives.
    - Travis Johnson
        - Place the OshPark order. (Today)
        - Research and order components, cabling and interconnects.
        - Order enclosure.
    - Wei Yan

        - Fix GUI problem mentioned above.
        - Help make N2O sensor work.
    - Wenyu Bi
        - Test the N2O sensor and synchronize it with all previous sensors to test the function of the device.
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - No questions.
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - No questions.

Week 15 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

04/22/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        -
    - Eyal Eynis & Wei Yan
        - We figured out how to display the last 10 pieces of data from the log file onto the table, so now the GUI will always update to the latest 10 pieces of data. The issue we are experiencing is that when there is less than 10 pieces of data for one date and there are more than two dates in the log file then the dates do not span correctly in the table.
    - Travis Johnson
        - Ordered parts for PCB.
    - Wenyu Bi
        - Set up the N2O sensor with professor Burnett.
        - Improved sensor code to facilitate future testing
- **Next week**
    - Team Plan
        -

    - Eyal Eynis & Wei Yan
        - We will try to figure out how to eliminate the zeros from the plot. This would be one of the last changes needed in the GUI to get a solid GUI running as intended aside from small bugs. We will try to fix some minor bugs related to the dates mentioned above and GUI not updating when it is in full screen.
    - Travis Johnson
        - Build PCB.
    - Wenyu Bi
        - After the ESD kit arrives it is safe to carry out the overall test of the sensors part
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - No questions.
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - No questions

Week 16 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

04/29/2022

Team Leader: Travis Johnson

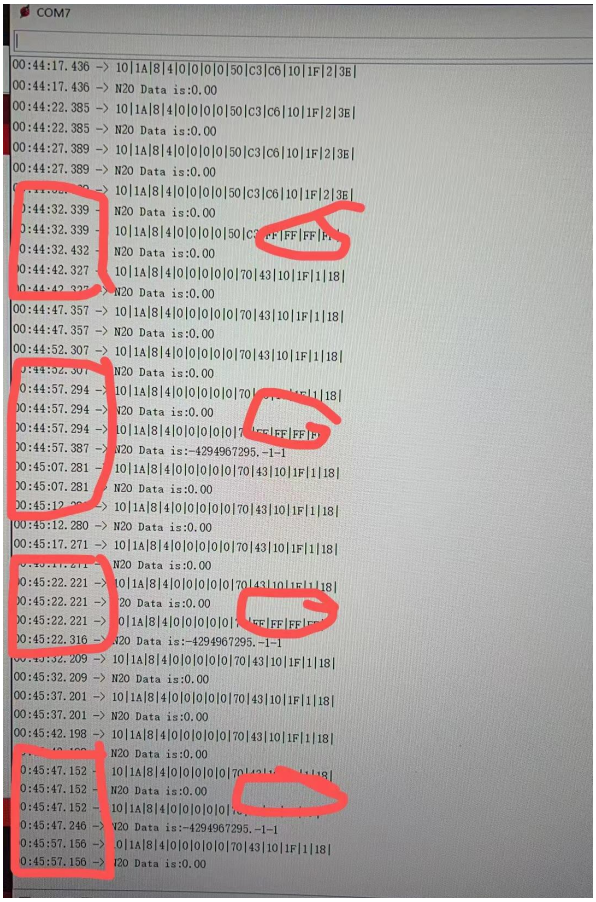
- **This Week**
    - Team Review
        -
    - Eyal Eynis & Wei Yan
        - We ran energy trace again and the current / power consumption for the system remained roughly the same at ~2mA and ~130mW. This may mean that the actual temp/humid sensor consumes very little power and the smartmesh mote consumes a lot of power.
        - Eliminated 0 from graph, but discovered a new bug. Whenever new data arrives data doesn't display properly when we switch to a different graph. Graphs were displayed properly when new data arrived but this was on a previous version of the code. Solving this

problem would require us to retrace our steps to find the previous working code which is a task for next week.
        - Time drift: We believe that energy trace shows time drift because the sample rate is sometimes 11 seconds, so this +1 extra second from our original sample rate of 10 seconds will add up eventually to make a big difference. This doesn't matter in our case because we create a new timestamp for every new sensor value that arrives.

```
├     MAC: 00-17-0d-00-00-32-dc-61
~  Status: OPERATIONAL
~   Coord: (None,None)
~ User ID: None
-- 04/27/2022
07:28:39, 2120, 4917, 1000
07:28:50, 2120, 0, 0
07:29:01, 2120, 4893, 0
07:29:11, 2121, 0, 1000
07:29:22, 2123, 4890, 0
07:29:33, 2123, 0, 0
07:29:43, 2124, 4880, 1000
07:29:54, 2125, 0, 0
07:30:05, 2123, 4883, 0
07:30:15, 2124, 0, 1000
07:30:26, 2123, 4929, 0
07:30:37, 2121, 0, 0
07:30:47, 2121, 4951, 1000
07:30:58, 2123, 0, 0
07:31:09, 2121, 4939, 0
07:31:19, 2121, 0, 1000
07:31:30, 2121, 4961, 0
07:31:41, 2120, 0, 0
```

    - Travis Johnson
        - Soldered junction board.
        - Built mote on board
        - Ordered enclosure and cable glands
    - Wenyu Bi
        - Preliminary testing of N2O sensor completed, N2O data can be obtained using MSP430

- **Next week**
    - Team Plan
        -
    - Eyal Eynis & Wei Yan
        - Fix the GUI bug mentioned above.
    - Travis Johnson
        - Order interconnects for N2O sensor
        - Build full mote
        - Fix SD Card function to take string which contains time stamp.
    - Wenyu Bi
        - Continue to fix the problem of N2O sensor in acquiring data



    - After several normal data acquisitions (five-second interval), there will be a situation where data is acquired twice at the same time in ten seconds, and the data acquired for the second time has a bunch of FFFF

- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - I need to look at N2O to get an idea of cabling to purchase.
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - When using the other soft serial ports, the device will not work properly. When using P6.1 and P6.0, it works fine, but as far as I know, these two serial ports are occupied by smartmesh.

Week 17 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

05/06/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    -
  - Eyal Eynis
    - HIH6120 sensor works with the current Energia code.
    - Recorded a video of the entire system as it currently stands: https://drive.google.com/file/d/12HwVu9Un4X2JcH3X0NANFac89ywSNqrf/view?usp=sharing
    - GUI bug from last week has been fixed.
  - Travis Johnson
    - Built Mote in waterproof enclosure. Connected HIH6120 sensor.
  - Wei Yan
    - Finally fixed major GUI bug from previous week. Status on GUI: All set except for small non crucial bugs that will be worked on in the side.
  - Wenyu Bi
    - Several tests were done to get the N2O sensor to work using an additional serial port
- **Next week**
  - Team Plan
    -
  - Eyal Eynis
    - Finalized sample rates will have to be implemented in the Energia code.
    - Have the SD card function log “millis()” so that sensor data can be associated with a timestamp.
  - Travis Johnson
    - Order parts for battery and holder
    - Order parts for creating cable to connect N2O sensor to mote.
    - Create a second mote with external ports and cabling for attaching N2O.
    - Begin documentation.
  - Wei Yan
    - We need to confirm final sample rate for every sensor because GUI will need to modified accordingly. GUI is currently able to support

following sample rates:
- Temp: 10s
- Humidity: 20s
- N20: 30s

■ GUI will somehow need to account for N2O sensor data while it is warming up.

○ Wenyu Bi
■ Do more research about how to make the transmission more stable
■ Complete the test for the N2O sensor and combine its code with the previous code
■ Help documentation work

- **Questions / Problems**
  - Team Blocks
    ■
  - Eyal Eynis
    ■ No questions.
  - Travis Johnson
    ■ None.
  - Wei Yan
    ■ What value does the N2O sensor output while it is warming up?
  - Wenyu Bi
    ■ The N2O sensor works perfectly when using the hardware serial port, I tried other ways to make it work as much as I can (you can see my test process in the N2O Test documentation. I think it would be better if we knew about the N2O sensor information earlier, we can choose a more suitable board), but the current results are not satisfactory.
    ■ https://drive.google.com/file/d/1gnmn8QrAAYt78sc80Pxa5al_8kKzQ5EJ/view?usp=sharing

Week 18 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

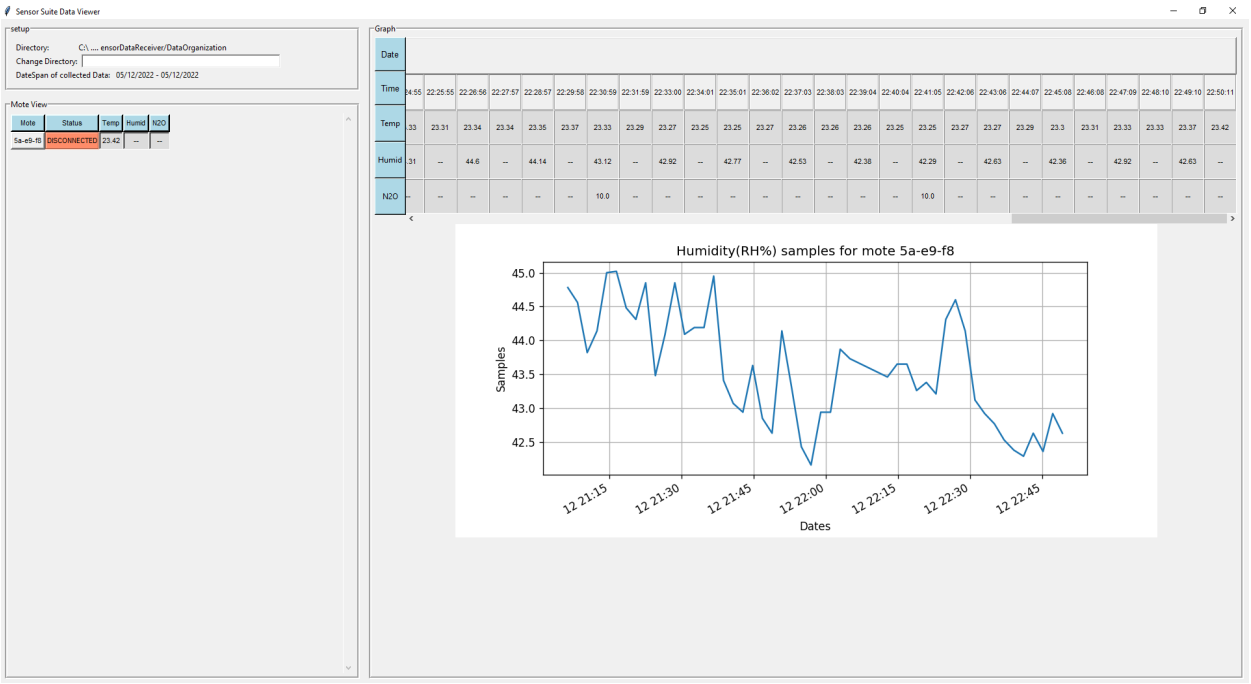
Team 13

05/11/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    ■
  - Travis Johnson
    ■ Began building second mote.

        - Behind schedule on creating N2O cable, and building battery holder.
    - Wei Yan & Eyal Eynis
        - Fixed GUI according to new sample rates as well as some other small bugs. GUI now displays last 100 pieces of data. Also, changed the null sensor values from 0 to 65535 in order to also account when N2O sensor value 0.
        - Worked with Wenyu to get initial power consumption of N2O with energy trace.



    - Wenyu Bi
        - Complete the configuration of the serial port, so that N2O can work normally
        - Do energytrace with Wei and Eyal

- **Next week**
    - Team Plan
        -
    - Eyal Eynis
        - Grab the N2O sensor from Wenyu and set it up with the current mote setup.
        - Do energy trace on the entire system.
    - Travis Johnson
        - Finish second mote with N2O sensor cable and battery mount
        - Begin documentation - bill of materials. Technical guide.
    - Wei Yan
        - Implement N2O code with current energia code and also work with Wenyu to implement milis function into SD card.
    - Wenyu Bi
        - Hand over the equipment to Eyal and help with the final functional

integration

- **Questions / Problems**
  - Team Blocks
    -
  - Eyal Eynis
    - No questions.
  - Travis Johnson
    - I have never assembled a dupont-style crimp connection before, and they are much more difficult than I anticipated. I think I will need to go with something more pre-assembled and solder the leads together.
  - Wei Yan
    - No questions
  - Wenyu Bi
    - No questions

Week 19 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

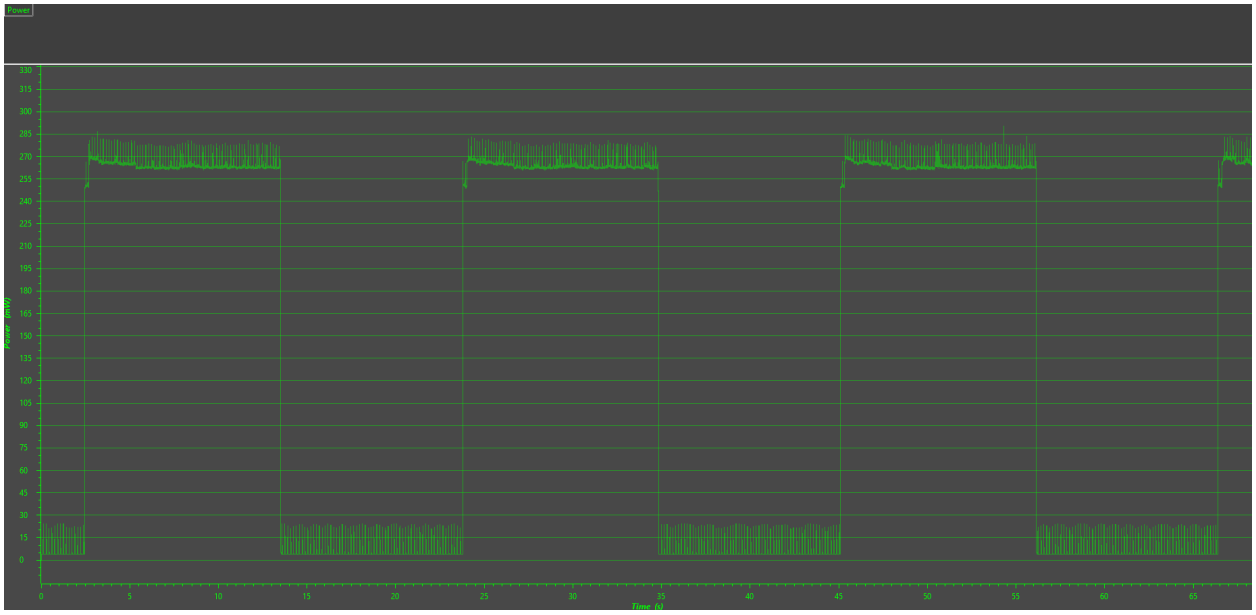
Team 13

05/20/2022

Team Leader: Travis Johnson

- **This Week**
  - Team Review
    -
  - Eyal Eynis
    - Tested out the transistor circuit that Travis built with the N2O sensor and the circuit functions correctly.
      - P4.3 on the MSP430 has been programmed to output a high (3.3V) for when we want the N2O sensor to be on and is switched to a low (0V) before the system is put into sleep mode. The signal from P4.3 is supplied to the gate of the transistor. Transistor circuit disconnects the GND of the N2O sensor.
      - Desired low power consumption is achieved whenever sleep mode is activated (~15mW). This result is only for the circuit containing the N2O sensor and the MSP430.
      - https://drive.google.com/file/d/1VTFv5io_XCNRBHuqiAgFBXmOSZNP06p8/view?usp=sharing
      - https://drive.google.com/file/d/1z7sFXAm_pShooT9faDpgKiQu62J5-8Cu/view?usp=sharing

Power consumption of N2O sensor in active (~270mW) vs sleep (~15mW) mode:



    - Travis Johnson
        - Built transistor circuit to be attached to the GND path of the N2O sensor, so that the sensor can be powered hard off.
        - [Completed] Built 2nd mote. This mote is the complete build, with battery holder and 4 18650 batteries (3400 mAh each) soldered in parallel and connected to the micro USB power cable that connects to the MSP430. The N2O cable is attached to the function board, and is ran through a water proof porthole outside the enclosure to be attached to the N2O sensor.
    - Wei Yan
        - Worked with team to come up with transistor solution.
        - Started working on Final Report
    - Wenyu Bi
        - Completed the improvement of the N2O sensor function (read data correctly, identify the warm-up state)
        - Helps measure the power consumption of N2O sensor
- **Next week**
    - Team Plan
        - Do Documentation!
    - Eyal Eynis
        - Pick up the second mote that Travis built.
        - Pick up the N2O sensor from Wenyu.
        - Integrate the N2O code into the main code and see if the system functions correctly.
        - Run energy trace on the entire system (N2O, temperature, humidity).
    - Travis Johnson
        - Begin documentation:
            - Bill of Materials
            - Technical build manual

            - Design files
        - Finalize SD Card functions.
    - Wei Yan
        - Work with team to get my laptop ready for live demo.
        - Continue writing Final report
    - Wenyu Bi
        - Help Eyal get N2O sensor working on mote
        - Start work on final report
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - The mote is complete. In the photo, the only thing missing is the MSP430, but Eyal will install this, and connect the cables. Please let me know in today's meeting if you see anything that does not meet requirements.
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - No questions.

Week 20 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

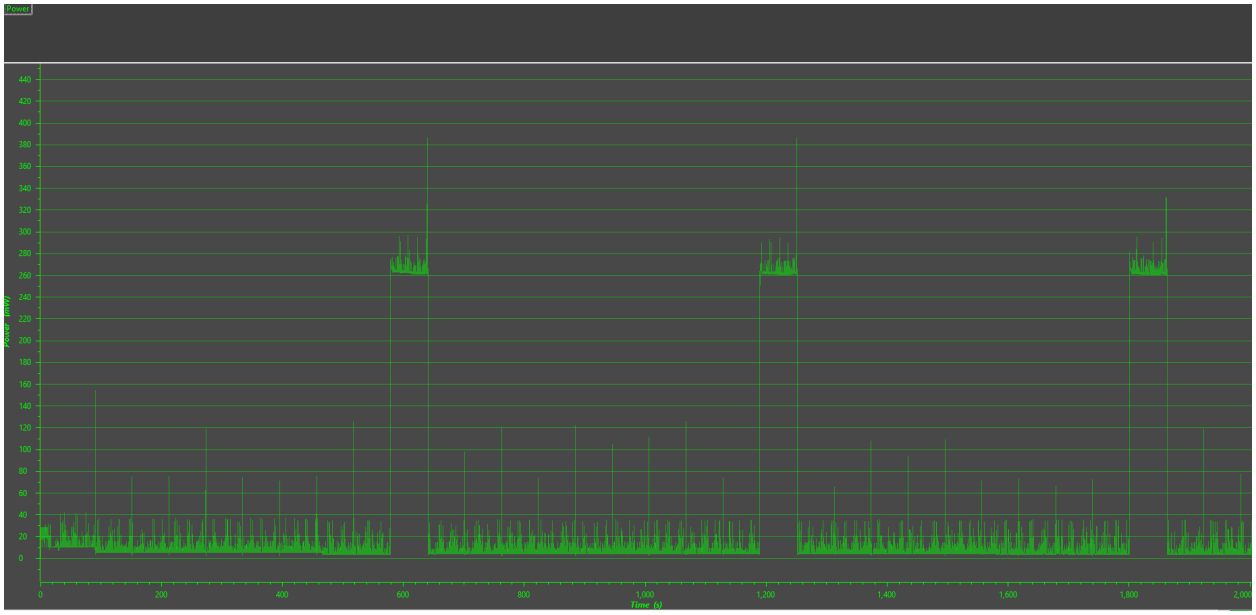
05/26/2022

Team Leader: Travis Johnsdon

- **This Week**
    - Team Review
        -
    - Eyal Eynis
        - Implemented the N2O sensor into the main system / code.
        - Implemented Wenyu's new SD card code (millis) into the main code.
        - Fit all of the components for the system into the enclosure (added switch to battery pack for easy operation).
        - Full working system (on batteries): https://drive.google.com/file/d/1egslNjGvPxgIx5SPKag1jc64QraBeB8P/view?usp=sharing
        - Energy Trace of full working system:

https://drive.google.com/file/d/18JJqDf2z5f-N3U_fnS4CupQUA_zm84uE/view?usp=sharing

Power consumption of entire system (N2O, temperature, humidity)



EnergyTrace™ Profile

Name	Live
System	
Time	2052 sec
Energy	58061.989 mJ
Power	
Mean	28.3248 mW
Min	0.0000 mW
Max	393.6869 mW
Voltage	
Mean	3.2915 V
Current	
Mean	8.6134 mA
Min	0.0000 mA
Max	119.8438 mA
Battery Life	Custom: 18 day 12 hour (est.)

*NOTE: Custom battery life is for one 3.7V 3400mAh battery

- Travis Johnson
  - Continued work on documentation.
- Wei Yan
  - Worked with Eyal to solve warm up timing problem. Full system works with 1 min N20 warm up time.
  - Continue filling in final report.
- Wenyu Bi
  - Help Eyal to modify the code of SD card
  - Send N2O sensor to Eyal for final equipment assembly and testing

- **Next week**
  - Team Plan
    -

    - Eyal Eynis
        - Write the user guide for the system / fill in other parts of the final report.
        - Help Wei get the system and code running on his laptop.
    - Travis Johnson
        - Finish poster and bill of materials.
        - Continue writing final report.
    - Wei Yan
        - Continue filling in final report.
        - Start installing all programs necessary in order to run full system on my laptop.
    - Wenyu Bi
        - Help with documentation
        - Help get our system working on Wei's computer
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        -
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - No questions.

Week 21 - Weekly Progress Report

Second Generation Low Power Environmental Sensor Suite

Team 13

06/03/2022

Team Leader: Travis Johnson

- **This Week**
    - Team Review
        -
    - Eyal Eynis
        - Worked on the final report.
        - Ran energy trace on the following setups:
            - MSP430 + Blink Internal LED
            - MSP430 + SmartMesh
            - MSP430 + Temp/Humid

                - MSP430 + N2O
                - Entire System (MSP430 + SmartMesh + Temp/Humid + N2O)
                - Energy trace data can be found here: https://drive.google.com/file/d/1RFVOwh5XeAJcJy8pjbNRIeZrp7tPojOW/view?usp=sharing
        - Travis Johnson
            - Wrote “how to build” in final report.
            - Finished Bill of Materials
            - Updated schematic
            - Began Poster
        - Wei Yan
            - Worked on final report
        - Wenyu Bi
            - Modified the sensor name error of SD card library
            - Add content to final report
- **Next week**
    - Team Plan
        -
    - Eyal Eynis
        - Continue working on the final report.
        - Help Wei get the mote system running on his laptop for the presentation.
    - Travis Johnson
        - Finish all documentation, including:
            - Poster
            - Final Report
            - Bill of Materials
        - Present at Poster event
        - Update Github
    - Wei Yan
        - Keep working on Final report and install all necessary libraries to my laptop for poster presentation.
    - Wenyu Bi
        - Continue to work on the final report
- **Questions / Problems**
    - Team Blocks
        -
    - Eyal Eynis
        - No questions.
    - Travis Johnson
        - No Questions
    - Wei Yan
        - No questions.
    - Wenyu Bi
        - No questions.