Group: Brendan Amorin, Jack Bachelor, Bjorn Issacson, George Moraites, Varun Shah, Jacky Wang
Professor Bell
EE193 Advanced Embedded Systems
13 May 2022

Project Report

Table of Contents

1. Introduction:

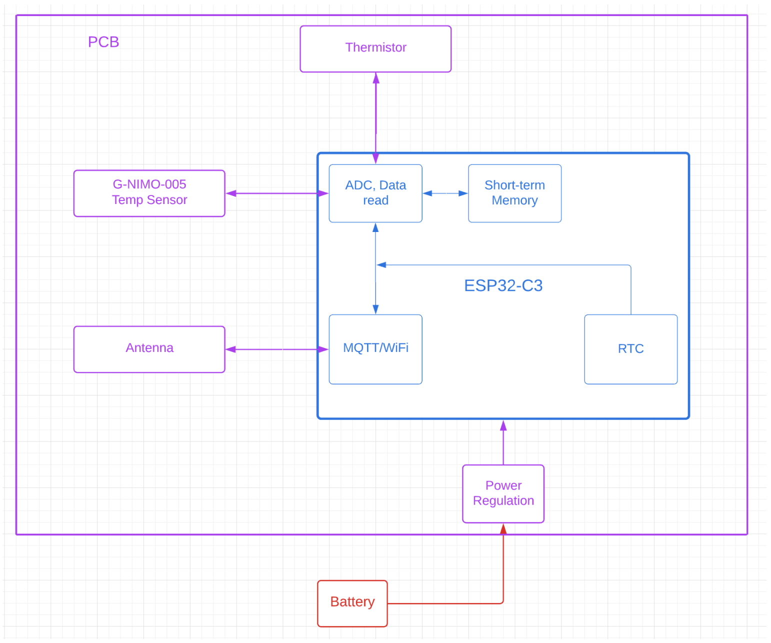
The internet of things (IoT) and embedded systems are becoming increasingly important to people's daily lives. Cutting edge technologies and systems such as smart homes or cities, aviation systems, or energy management devices wouldn't be possible without their creation. Embedded systems use microprocessors with software to perform a dedicated task either as a standalone item or a part of a larger system. IoT refers to the concept of connecting these embedded systems to the internet, for greater connectedness and collection of data.

In the context of the course, the class was tasked with creating temperature sensor nodes which were able to accurately take measurements every hour for a minimum projected time of six months around the Tufts campus. Part of the challenge and excitement of this course was to combine concepts like PCB design, wireless communication, and practicality to come up with a final product that achieved project and design goals.

2. Design Overview:

Using code and physical testing of our individual PCB designs as a basis for this project, the group made a block diagram of how we hoped our system would work. With respect to the PCB, the group decided to use a basic thermistor and a G-NIMO temperature sensor to sanity check both temperature measurements. Values from these components would be read by the analog to digital converter (ADC) and stored in the memory of the microcontroller. This information would be transmitted to a database on the Tufts campus and displayed someplace on the EE193 course website. The microcontroller chosen is the ESP32 C3, the latest and lowest power consumption variant of the ESP32 family. The PCB would be powered by batteries and the entire system would be kept in a weatherproofed housing.

Figure 1. Initial block diagram



The figure above shows a block diagram of the system designed during the project.

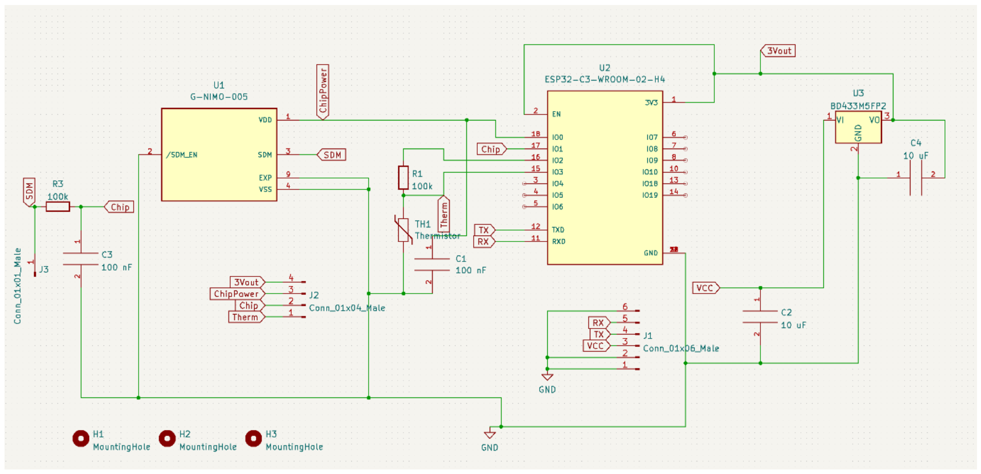
3. Detailed Design Overview:

Table 1. Part selection rationale

Part	Rationale
Lithium AA	These batteries were chosen because of their good performance in colder temperatures and high capacity.
4 AA battery pack	Provides secure mechanical and electrical mounting of the batteries.
Chip: ESP32-C3	After consulting the datasheets of this chip, along with other ESP32 chips (i.e. ESP32, ESP32-S2), it was reasoned that the ESP32-C3 was the most power-efficient chip at our disposal.
Voltage Regulator: (BD433M5FP2-CZE2)	This regulator was chosen to minimize the quiescent current while maintaining a reasonable output current (500 mA). Our original regulator had a quiescent current on the order of milliamps, so this new regulator with a quiescent current on the order of microamps was seen as a significant improvement.

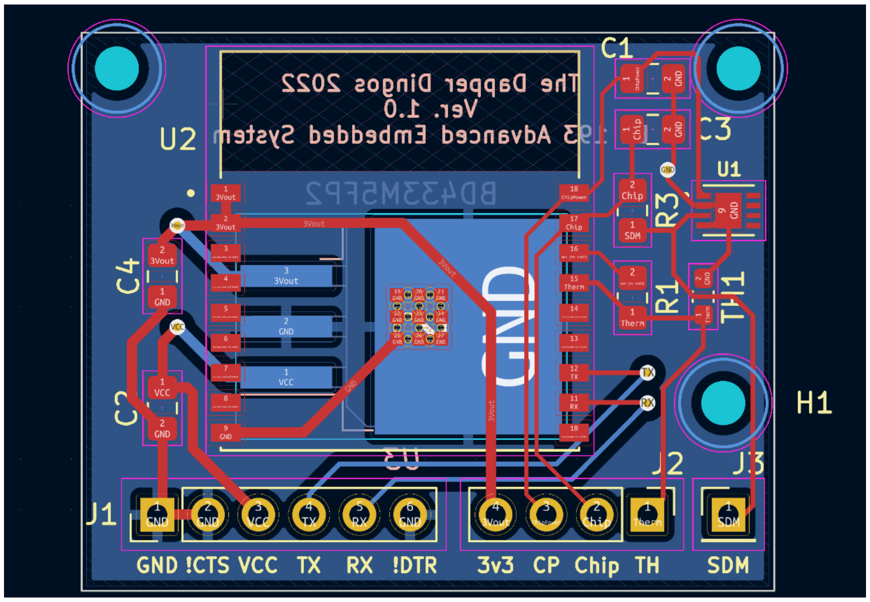
Temp Sensor: G-NIMO-005	If configured properly, this sensor could provide output voltage measurements without the need for the I2C protocol. Namely, integrating a lowpass filter at the output of the sensor converted the raw output reading to voltage such that the appropriate conversion to temperature could be performed.
Thermistor (B57350V2104F460)	The team had access to a complete PCB from a previous homework that included the chosen temperature sensor and this particular thermistor. This thermistor was found to produce accurate temperature measurements that corroborated the sensor's readings. Knowing the two work well together, this thermistor was chosen once the temp. sensor choice was finalized.
Resistor: 100k	The thermistor's room temperature resistance was 100k, so another 100k resistor was chosen to create an even voltage divider to measure temperature. Another 100k resistor was chosen based on a recommendation from the temp sensor's datasheet. This resistor is part of a low-pass filter at the output of the sensor.
Capacitor: 100nF	Again, recommended by the temp sensor's datasheet. This capacitor is used as a bypass cap between power and ground of the sensor, in order to reduce supply noise.
Capacitor: 220nF	This capacitor is the second component in the low-pass filter at the output of the sensor. Paired with the 100k resistor, the filter allows the voltage at the output to be read by the chip directly, without the need for complicated communication protocols. The schematic for this filter was given in the temp sensor's datasheet.

Figure 2. Schematic of prototype board



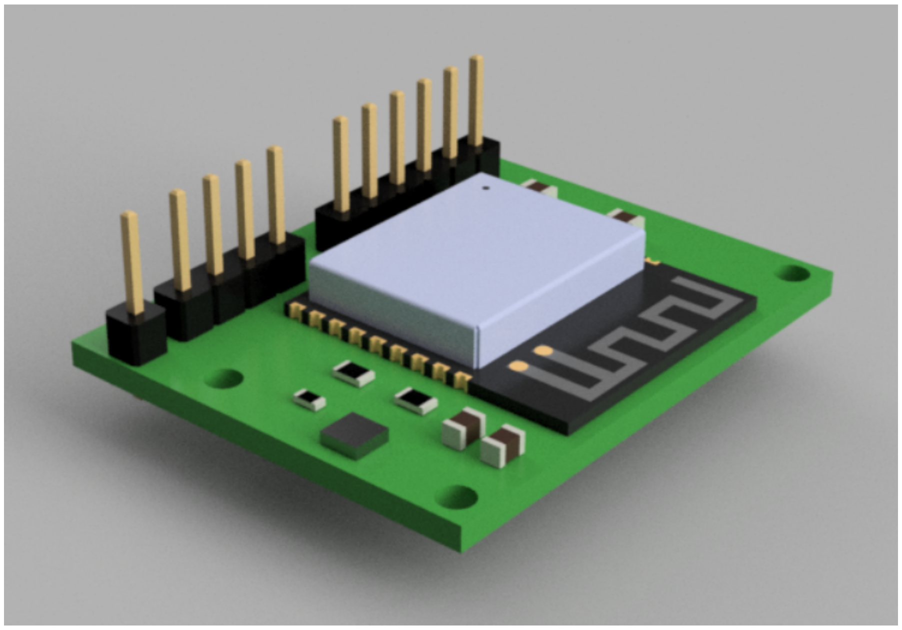
The figure above shows the prototype schematic used to design the PCB.

Figure 3. PCB layout design



The figure above shows the schematic of the PCB.

Figure 4. Board 3d rendering



The figure above shows a 3d rendering of the PCB and its associated components.

Figure 5. Enclosure rendering

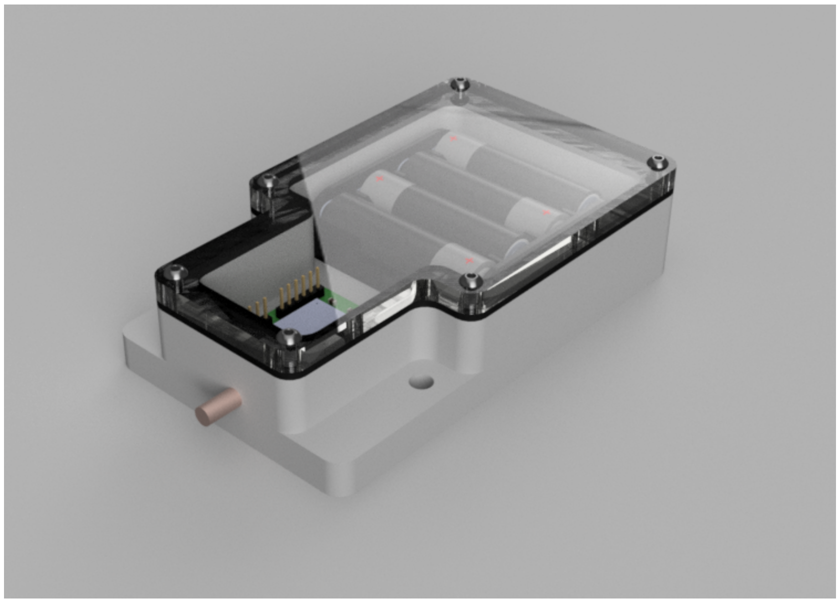


Figure 5. above shows a rendering of the enclosure that was designed to keep the PCB away from the elements, but able to take accurate temperature readings. The body is machined from HDPE, a plastic that is known for its environmental resistance. Machining from solid material is superior to 3D printing, as it is less porous and of higher mechanical strength. Acrylic is used as a cover, and it is also weather resistant. Laser cutting allows for accurate and simple fabrication, and clear acrylic allows for visual checks. Flanges in the HDPE body allow for mounting holes. 6 machine screws (ideally stainless steel) are used in conjunction with heat-set inserts for ease of tightening the Acrylic cover. Epoxy is used for mounting of the battery holder and board. A copper rod connecting the sensor to the environment allows for better thermal contact. Silicon caulking provides a seal. Ideally a rubber gasket is used, as the caulking doesn't allow the acrylic cover to be removed and reinstalled.

The system is entirely battery powered. Consideration was given to low consumption components, excess battery capacity, and longevity for battery chemistry. A voltage regulator drops the battery pack voltage to operating levels so that the components on the board receive a stable 3.3 voltage. Half the nodes are using lithium metal batteries, while the other half use alkaline for testing purposes. Lithium metal batteries are able to operate in -40 C conditions without degradation and are higher capacity, but are more expensive at $1.50 vs. $0.69 for traditional alkaline batteries. The longevity of IOT nodes may be limited by battery capacity, and so designing a low-power consumption system is extremely important. Because the ESP32 is often in deep sleep, the power draw of the voltage regulator is critical. The quiescent current mentioned by the datasheet (38 µA) and supply constraints were both drivers in selection and led us to pick the voltage regulator we did, despite its large physical size.

Temperature readings are taken through two possible ways. The primary method is the G-NIMO-005. This temperature sensor was chosen for its simplicity, and its use in prior homework assignments (in which it worked reliably). A thermistor is also tied to an analog input for backup readings. A downside with this particular component is that it's very small and needs to be put into the reflow oven as opposed to hand soldering.

The PCB was designed using KiCAD. This was a two layer board with most components on the top. The voltage regulator was placed on the bottom to keep its heat away from the sensors, and save space. A cut out was needed on the GND pour on the bottom layer to allow for good antenna performance. Header pins were used for power pins, as well as pins important for debugging or flashing the system. OSH park was selected as the producer of the PCB due to its ability to reliably ship PCBs at a low cost.

For more information on the components used in this project as well as any of the resources the group found helpful refer to the table below.

Table 2. Documentation Sources:

Resource:	Link:
Github Repo	https://github.com/tufts-embeddedsystems/Team_D_ESP32C3_G-NIMO-005
Plastic Choice for Enclosure	https://protoplastics.com/weatherproof-plastics-outdoor-structures/
What to do if chip won't connect	https://docs.espressif.com/projects/esptool/en/latest/esp32c3/advanced-topics/boot-mode-selection.html
Power Budget	https://docs.google.com/spreadsheets/d/18GI4imMPOJXvrP9uH3nrkdbgFkneUgSUCQqwxPxcfAU/edit?usp=sharing
Documentation Archive (The Team's Google Drive folder)	https://drive.google.com/drive/folders/16xUbRudf7gUwAs-hpwpd8nWw0RjgWU7h?usp=sharing https://github.com/tufts-embeddedsystems/Team_D_ESP32C3_G-NIMO-005
Mechanical CAD (Fusion 360)	https://github.com/tufts-embeddedsystems/Team_D_ESP32C3_G-NIMO-005
Electronic CAD (KiCad)	https://github.com/tufts-embeddedsystems/Team_D_ESP32C3_G-NIMO-005
G-NIMO-005 datasheet	https://octopart.com/datasheet/g-nimo-005-te+connectivity-59208841
ESP32-C3 WROOM 02 datasheet- Section 3 was particularly important	https://www.espressif.com/sites/default/files/documentation/esp32-c3-wroom-02_datasheet_en.pdf
Thermistor datasheet- R/T Number is 8502, corresponding to the middle column of the table	https://datasheet.octopart.com/B57350V2104F460-EPCOS-datasheet-78940024.pdf

Table 3. Team Contact Info:

Team Member and Project Focus Area:	Contact Info:
Bjorn Isaacson: Enclosure and Batteries / Power	bjornaisaacson@gmail.com
George Moraites: Soldering, assisting with debugging	gemoraites@gmail.com

Varun Shah: Hardware debugging	varunshah.2499@gmail.com
Jacky Wang: Software, hardware debugging, and PCB design:	jacky.w1204@gmail.com
Jack Batchelor: Bootloading, Hardware debugging	jacbatnc@gmail.com
Brendan Amorin: Software and hardware debugging	bmamorin00@gmail.com

4. Current State of the project:

As of 05/05/2022, our group can flash our code to a ESP-WROOM-32 DevKit to check and see if it's working, but we have been unable to get the code to go to our microcontroller on our test board. While testing our test node, we found that the pins that needed to be high or low were operating properly and that electrically speaking components were receiving the power they needed. As for the enclosure, we're confident that it'll work but we have not tested it against anything like water or heat to see if the electronics inside are safe and able to take accurate measurements.

As of 05/07/2022, we were finally able to flash code to a few of the C3 chips after numerous by-hand alterations, and breadboard-ed circuitry. The steps to flash the chip using the USB-serial breakout boards we purchased are listed below:

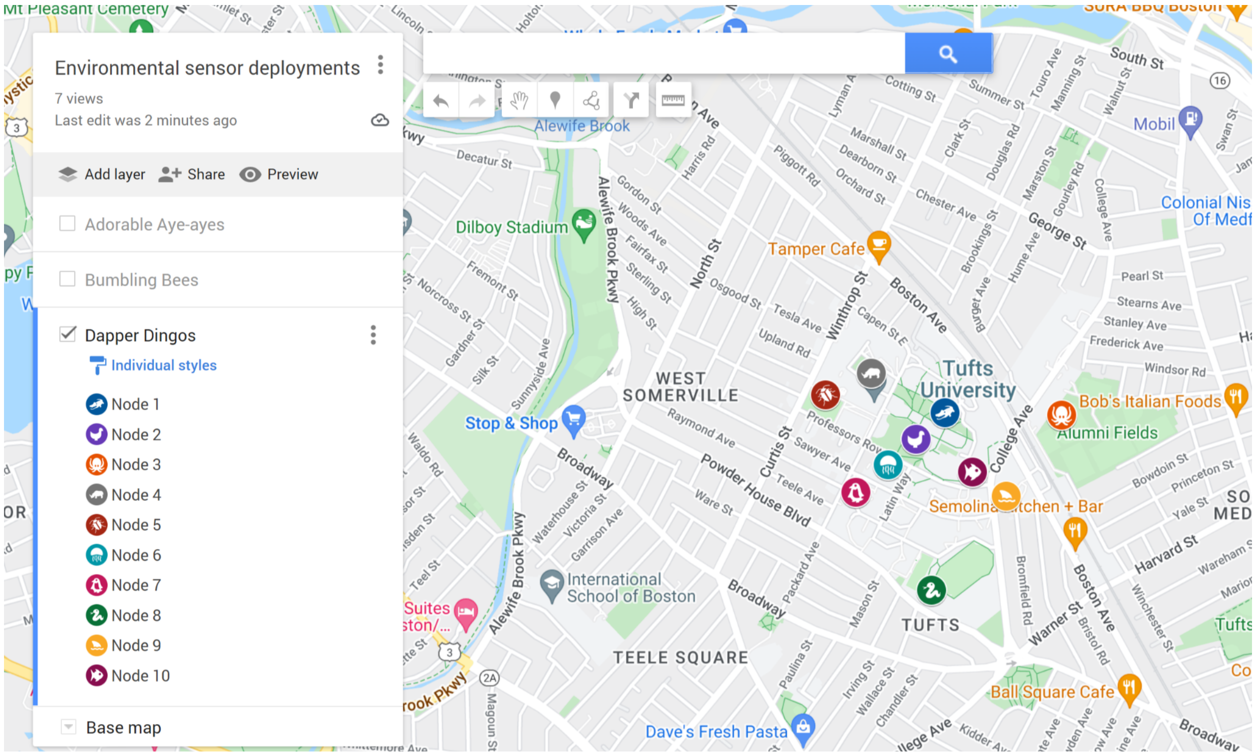
1. Cut the trace between Power and EN
2. Solder a chip to our PCB
3. Solder lengths of wire to pins of the chip:
    a. Pin 1: 3.3V Power
    b. Pin 7: GPIO8
    c. Pin 8: GPIO9
4. Connect soldered wires to breadboard

    a. Pin 1 to Power rail
    b. Pin 7 to 10k resistor to power rail (pull-up resistor)
    c. Pin 8 to button to GND. Include a 100uF debouncing capacitor across the pins of the button. This is the equivalent of the “boot” button on the devkit.
5. Our PCB design included breakout pin-headers on key traces for debugging. Connect these breakout pins from PCB to breadboard:
    a. GND to GND rail
    b. The pin on the PCB labeled Vcc is actually only connected to the Enable pin of the C3 chip, since we cut the trace to power in step 1. Connect Vcc to a button to GND. Include a 100uF debouncing capacitor across the pins of the button. This is the equivalent of the “reset” button on the devkit.
    c. TX to the USB-Serial module’s RX
    d. RX to the USB-Serial module’s TX
6. Finally, connect the power and ground rails of the breadboard to a DC power supply set to 3.3V, and connect the USB-serial module to a PC.
7. Hold the Pin 8 boot button down and press the Vcc reset button.
8. While still holding the pin 8 boot button, run the flash command from the PC.
9. When the terminal output says “Connecting….” let go of the pin 8 boot button

Currently, the team is calibrating the thermistor for more accurate temperature readings and attempting to successfully publish messages to the course dashboard from one of the ESP32-C3 chips. The code for this behavior successfully works on the aforementioned ESP-WROOM-32 DevKit, and the chip is able to successfully connect to WiFi, but fails to publish to the dashboard before a brown out occurs. We believe this could be caused by how we are powering the board and are planning to connect a larger capacitor to the power supply than the current 100 nF used to remove any noise or spikes from the source.

Five enclosures and battery setups have been created for if the PCBs can successfully publish to the server and be deployed in the environment. A list of proposed locations around the Tufts campus is shown below.

Figure 6. Proposed node locations



The figure above shows the proposed locations around Tufts where our group would deploy ten sensors.

As of 05/12/2022, the team was able to successfully complete one node to be deployed in the environment. To successfully power the board, the power pin and enable pin needed to be bridged together. In the process, this connection made contact with the ESP32-C3 chip, resulting in a short circuit. Fortunately, the voltage regulator limited the current to 500 mA, but once this contact was removed, the node was able to collect temperature readings and transmit them to the dashboard. Another node was encountering issues where no current was flowing into the circuit, but the team ran out of time to debug this. With more time, it was reasoned that the four remaining nodes could have been fully assembled.

5. Possible improvements, lessons learned:

Possible Improvements:

1. If there was more time, the biggest improvement would be to correct our PCB design by disconnecting the trace that's underneath the C3 chip between the VCC and enable pins. Since enable must sometimes be turned off, specifically when flashing the board, having these pins connected together caused major complications when trying to switch the chip into boot mode.
2. More rigorous testing and qualification of the system would be useful. This would involve the following steps, or expanding on them:
    a. Measuring the power consumption of the system over the cycles. Compare this to the predicted power budget to ensure the system can survive for the time required.
    b. Testing the system for reliability over cycles. Log the ability to connect, and operate.
    c. Compare sensor readings to a calibrated sensor.
    d. Environmental testing, such as operation in a freezer, or being submerged in water.

Lessons Learned:

1. One of the takeaways was to focus on having one unit working initially to produce a minimum viable product. This means we wouldn’t have soldered other boards prematurely.
2. Having more time to wait and debug one board would have been beneficial in allowing for group members to divide the work more efficiently.
3. Keep the same groups throughout the semester to keep continuity in teams and help students have an idea about what their final project should look like.
4. Be **really** careful when designing the PCB if the class is short on time, or allow for students to pick components, breadboard them for debugging, and then order their PCB so that they make fewer mistakes on the PCB and can iterate their design.
5. Be careful of the distinction between the chip’s datasheet and the Devkit’s datasheet, especially what components are internal to the chip vs external and on the devkit. We thought there was an internal pull-up resistor on GPIO8, meaning it would automatically be in the configuration for boot mode. Turns out, despite being shown on a page that included only the chip’s schematic, the pull-up resistor was external and included *only* on the DevKit.
6. Including traces to breakout pin headers proved to be the best choice we made in designing out PCB. Having those pins headers available to probe and manipulate key pins on our board was crucial.
7. If possible, it is better to solder jumpers or debugging wires to the PCB rather than the chip itself. By soldering to the chip itself, you run the risk of ripping the pads off and making the chip useless.

6. References:

https://www.ece.tufts.edu/ee/193AES/