Cathodic Protection Monitoring

JANET VOROBYEVA

UC San Diego, jvorobyeva@ucsd.edu

Corrosion is a significant and costly issue that affects all metal structures. Cathodic protection systems are a common tool used to prevent corrosion in structures exposed to moisture, such as bridges, pipelines, and water tanks. However, they require regular monitoring to remain effective, and many are monitored insufficiently or not at all. Fortunately, a cathodic protection system acts as a galvanic cell, producing a small amount of electrical power which could be harvested to power a remote sensing node. It should be possible to retrofit existing cathodic protection test points with sensors without needing to run external power supplies to them, enabling their monitoring to be automated. In this project I design and build an energy-harvesting sensor node which can charge itself from very low input power (~150uW), test it on a small benchtop corrosion cell, and have it regularly transmit data via LoRa radio to an internet gateway.

1 INTRODUCTION

1.1 Background

Corrosion is a significant and costly issue that affects all metal structures. A 2002 report estimated that the total direct cost of corrosion in the US was $276 billion dollars, a full 3.1% of the country's GDP, and that number doubles when you factor in indirect costs as well [5]. Corrosion can be especially aggressive to structures that come in contact with water, such as boats, bridges, home water heaters, and underground storage tanks / pipes, so extra effort is spent on prevention and mitigation in these situations.

Cathodic protection systems are a common tool used to mitigate corrosion in these aggressive environments. They work by attaching a "sacrificial anode" made of a strongly reactive metal, like zinc or magnesium, to the main structure. Metal corrosion is an electrochemical process in which one part of a metal structure acts as the anode (negative potential, is consumed) and another acts as the cathode (positive potential, is protected from corrosion). The magnesium or zinc acts as the anode, inducing a positive voltage in the main structure and causing it to be cathodically protected from corrosion. The sacrificial anode corrodes in place of the main structure and is consumed over time, needing to be replaced every 5-15 years.[1]

While cathodic protection systems can be incredibly effective in mitigating corrosion, most cathodic protection systems are not adequately monitored, and so may fail significantly earlier than designed. Monitoring their health means sending someone out once a year to each test point in the system, and verifying with a multimeter that they are still functioning properly; in practice this doesn't end up happening as often as it should.

A 2009 report from the Transportation Research Board had this to say on the subject of monitoring cathodic protection systems:

> "The respondents in the survey have made it clear that monitoring and maintenance of the cathodic protection system is too burdensome and that most agencies are finding it difficult to cope with this process. … Of the twenty-four [agencies] who have at least one cathodic protection system, only 10 monitor them as [discussed earlier] … Therefore, 14 agencies do not monitor their cathodic

[1] Note: this describes galvanic cathodic protection systems, which use sacrificial anodes. There are also systems which connect to a fixed power supply, called impressed-current cathodic protection systems, but these are typically used on larger scale installations like large oil pipelines.

> protection systems. … only nine agencies indicated that they have at least one person to monitor their cathodic protection systems and five believed that they had sufficient staff to monitor all of their systems" [4] (p36-37)

1.2 Proposed Solution

Developing systems to automatically monitor cathodic protection systems can remove the manual workload required and notify agencies when anodes need replacing, preventing premature failure of cathodic protections and extending the useful life of valuable infrastructure.

Further, cathodic protection systems may be uniquely well-suited to sensor monitoring: since they work by inducing a potential across a structure, some of this electrical energy can be harvested to power the sensor, removing the need for expensive wired connections, limited-lifespan batteries, or unreliable power sources like solar. While they produce a relatively small amount of power (milliwatts or hundreds of microwatts), an energy-harvesting system can slowly charge up over a long time, and send out a radio packet a few times a day, minimizing power requirements. Such an approach could allow sensors to be retrofit into existing installations with no accessible power sources, and these sensors would function reliably for as long as the cathodic protection system remains functional.

2 Related Works

Most of this work was based off of and expanded on a previous paper, by Dhananjay Jagtap, "Repurposing Cathodic Protection Systems as Reliable, in-situ, Ambient Batteries for Sensor Networks" [3]. He prototyped a version of this system on broadboards and characterized the electrical performance of a magnesium anode in a tank of water.

Some other projects have also proposed similar works. Kim et. al 2014 [6] proposes a similar idea. However, they have a much larger power budget (3mW, 350F supercapacitors), and their work focuses more on a custom energy-harvesting design. Qiao et al [7] also propose a similar system, but their work embeds a sensor into concrete as it's poured, rather than retrofitting existing test points.

3 Technical Material

3.1 Specifications

The plan for this project was to develop an energy-harvesting sensor. It had to power itself from an open circuit voltage as low as 0.6 V (cathodic protection systems start losing effectiveness below around 0.8V, so this would ensure our sensor wouldn't fail before the sacrificial anode did.). I didn't have clear numbers on minimum currents, but I initially assumed 1mA short-circuit current to be a reasonable minimum based on Dhananjay Jagtap's 2021 paper, giving us a power budget of 300uW (0.3V at 0.5mA). It had to be able to charge itself off of this power source and send out a radio packet with voltage measurements at least once a day. Over the course of the design, I ended up doing even better, and testing it running well off of only 150uW input power.

3.2 **Design**

To meet the above requirements, here were the major aspects of the hardware design:

Energy Harvester: I designed the system around the ADP5091 energy harvesting chip. It's a fully-featured energy harvesting module, and can be configured very flexibly (using about a dozen resistors of specific values). It can handle input voltages of 0.08V to 3.3V, cold-start itself from an input as low as 0.5V, and harvest energy at rates from 6uW to 600mW. It can also be configured to automatically output a "power-good" signal at configurable levels of charge, has a configurable power-point to harvest energy from, and has a builtin voltage regulator output. [1]

I configured it to run at an optimal power-point of 50% of open-circuit voltage, which should be optimal for resistive loads. I configured it to limit charge on the supercapacitors up to 3.3V (the highest my electronics could handle) and down to about 2.3V (slightly above the minimum working voltage of the chip).

The tricky part was making sure I had all of it configured correctly: I was able to test basic functionality with an APD5091 evaluation board I had access to, but many options were fixed in the hardware and couldn't be fully verified until I had assembled my own board. Therefore, I designed my mk1 board to also work as a testing platform, leaving many options accessible via pin-headers or through-hole resistors, and adding many test points to measure voltages at key parts of the energy-harvesting circuit.

Microcontroller: I designed the system around the Sparkfun Artemis microcontroller module, (built around an Ambiq Apollo3 microcontroller)[2]. The main reasons for this choice were its low power consumption (at 3.3V, it draws only 500uA while awake, and can go down to a deep sleep current of only 2.5uA), and the fact that my labmate Gabe had previously worked with Artemis boards, meaning I could use one of our Artemis dev boards for testing and reuse some parts of Gabe's firmware. For communicating with the board, I used a JTAG programmer/ debugger.

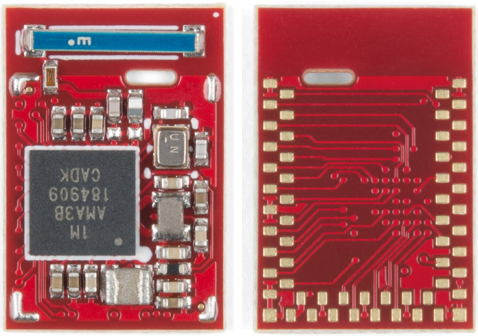


Figure: Sparkfun Artemis module, front and back.
(https://learn.sparkfun.com/tutorials/designing-with-the-sparkfun-artemis)

[2] Note: the reason for using the Artemis module over just the Apollo3 microcontroller is that the Apollo3 comes in a 9x9 ball-grid-array package, which requires complex routing, a 4-layer PCB with buried vias, a plethora of support passives, and would be incredibly difficult to properly reflow with the tools I had available. The Artemis module integrates most of the required passives, and breaks out the pins into a much more manageable layout for surface-mount assembly.

Radio Module: I designed the system to send data over LoRa, as it is both long-range and low-power. I used the RFM95W LoRa module, as it had previously been used for other projects in Pat's lab. At design time, I accounted for a maximum transmit current draw of 100mA for up to a second. In practice, I observed closer to 50mA current draw, and was able to use a less-robust transmission mode that took only about a tenth of a second to transmit, which meant I was able to transmit significantly more often than I had planned, and that I probably could have gotten away with a smaller energy store than I ended up using.

Power Storage: The highest power operation of the sensor is radio transmission, so the energy store had to be able to support that: To handle 100mA for up to a second, with an operating range of 2.3V-3.3V, this would require a capacitance of at least 100mF to sustain. However, at this current draw, the equivalent series resistance (ESR) through the supercap also becomes significant: even a 5Ω ESR would drop 0.5V, meaning ~20% of our energy would be wasted as heat, and the voltage drop would cut the operating range of our capacitor in half.

To account for the high current requirements, and to leave some extra room for multiple transmissions without recharging, 250mF would probably have been sufficient. I ended up using a pair of DGH105Q2R7 low-ESR supercaps which had 0.5F of capacitance (the smallest they went); and could handle up to 5V (though I only went up to 3.3).

Supercapacitor selection was additionally complicated by leakage current requirements: very low-leakage supercapacitors do exist, but often have significantly higher ESR (some I looked at were 20-100Ω), making them unsuitable for the high current needed for our radio. The DGH105Q2R7 struck a reasonable balance, with a low 0.8Ω ESR, but it did have a significant leakage current of up to 8uA.

Power Leakage: Handling leakage power was a major consideration in this design. My target power budget was for the circuit to be able to run off around ~150uW (50uA of current draw at 3V). The quiescent (deep sleep) current draw was therefore crucial to minimize, so as much of the input power as possible could go to charging the supercapacitors and transmitting data.

Main leakage sources: Our largest drain was through the supercapacitors: These could leak up to 8uA of current, which would be the largest contributor to our quiescent current. In practice, they should be leaking somewhat less, since we're only using them at around 65% of their rated voltage. The Artemis module could go down to 2.5uA in its deepest sleep mode. However even the second-deepest sleep mode consumed 70uA (far too much), so it was critical to accurately measure its current draw and ensure it reached the deep-sleep mode. The RFM95W radio module had terrible sleep performance (300uA in deepest sleep), so its power supply had to be gated behind a power-switch IC, dropping its leakage to negligible nanoamps.

Cold-start behavior was another important consideration: Once awake, the Artemis module could disable the radio and put itself into deep sleep mode. However, until the system charged up to 1.8V, the artemis would not be powered on and therefore couldn't control its own power consumption. To prevent this, a second power switch IC prevents power from leaving the energy harvester circuit until it charges to 2.3V, allowing it to safely charge until there's enough power to boot the system properly.

However, just gating the power supply to the microcontroller isn't enough. It needs to measure several voltages from the charge circuit, and power will leak through these analog input pins until it can be powered on. To limit leakage through these pins, I added some large (100kOhm, 500kOhm) impedances between the charge circuit and microcontroller, as well as 10nF capacitors to buffer the high-impedance signals and get accurate ADC measurements. A more robust solution could have been to use a low-power analog switch IC,

but simply adding impedance sufficiently limited current draw to ~10uA at maximum during cold-start (less in practice due to voltage drop in the microcontroller), and required fewer parts.

Note, it would have been possible to configure the main power switch to fully disable the microcontroller until transmit time, instead of relying on the 2.5uA deep sleep mode. However, the 2.5 uA draw is manageable within our leakage budget, and leaving the microcontroller powered allows the system to become responsive earlier.

Flexibility / Testability: The original plan was for the mk1 board to be a test platform to verify the design, and the mk2 boards to be more tightly integrated. While I ended up dropping the goal of ordering mk2 boards, the flexibility of the mk1 boards was invaluable.

A key feature was separating out ground planes for the Artemis microcontroller and the LoRa radio module, which allowed me to separately monitor power consumption of the microcontroller and radio, while isolating out other power consumers on the board such as LEDs or leakage through the charge circuit. This was invaluable in minimizing the Artemis' power consumption, and allowed me to catch several firmware bugs leading to increased power draw, e.g. the fact that attaching a JTAG debugger to the circuit prevents it from entering deep sleep until the circuit is power cycled.

Another useful feature was adding switchable jumpers to control the power source used, letting me run the board off USB power for testing, isolate the energy-harvesting circuit from the rest of the board to observe its performance directly, switch between direct and regulated power output of the energy harvester, and finally run the board directly off the energy harvester once the firmware was ready.

I also broke out many useful signals to pin headers for testing, including most key points of the energy harvesting circuit, a load button to drain the supercapacitors during testing, additional ADC pins, GPIOs, and an extra SPI bus, many of which proved very useful in debugging / fixing issues with the board.

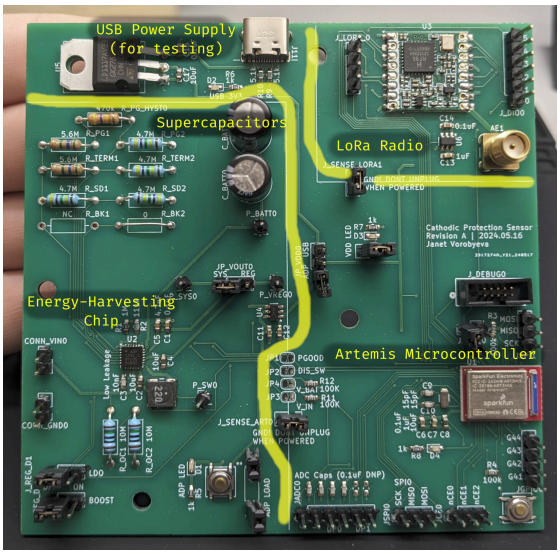


Figure: The assembled Cathodic Protection Sensor mk1

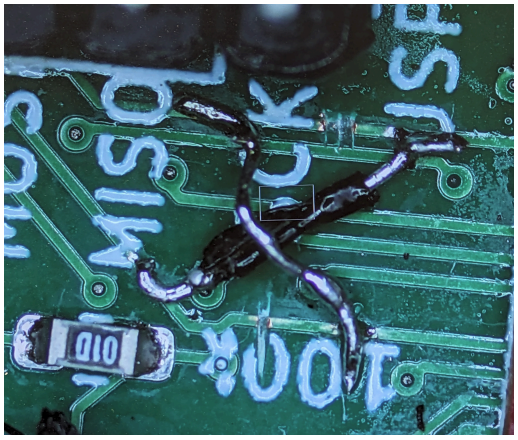


Figure: Trace Surgery (microscope view)

3.3 **Hardware Setbacks**

During testing of the board, several challenges presented themselves.

On assembly, I discovered I had incorrectly assigned two pins of the JTAG header, so I had to do some PCB surgery, cutting a couple traces and soldering jumper wires directly to the 0.2mm-wide traces.

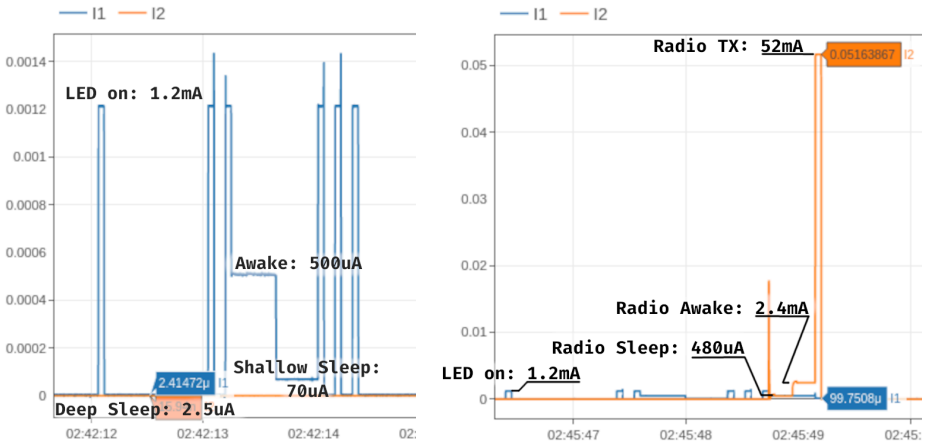


Figure: Current consumption of microcontroller(left) and radio (right)

3.4 **Power Testing**

Once the board was working, I was finally able to test the low-power firmware and measure power consumption of the various components. I was able to get the Artemis into its deepest sleep mode at 2.5uA, and wrote a test program to measure it and the radio's current consumption in various modes.

Figure: A test point in the wild (note the lack of labeling)

3.5 **Deployment Issues / Test Points**

I had originally planned to deploy this sensor at CP test points on campus, but ran into several issues. While I was able to locate several test stations, none of the connections were labeled, and none had the electrical setup I had expected to encounter.

Finding specific documentation on how CP test stations work was tricky. There are several designs of systems (galvanic vs impressed-current), and several kinds of test stations: Current-shunt test stations which require a connection to be broken to measure voltage, stations with no accessible shunt but that have a connection to a reference electrode buried in the structure, stations which require an external copper-sulfate reference electrode be used against the soil (and the soil to be moistened to ensure good connection) to get a good measurement.

The lack of documentation is also an issue, and I suspect it may be difficult to find who on campus knows how the cathodic protection systems are set up. In hindsight I should have spent more time on verifying the details of exactly what kinds of test points we have access to on campus, but by the time I got to this part there wasn't really enough time to dig into it properly.

Further, attaching my sensor to some of these might not work long-term (reference electrodes are meant for taking brief measurements, would they work for long-term current draw?), or might not be safe for the cathodic protection systems being monitored (if I break a current shunt to stick my sensor in between, am I disrupting the protective current flow? What happens if my device fails?). All of this would require additional design research / design work to handle properly.

Let's mark this section down as "Future Work"

3.6 **Gateway**

For the sensor to be useful, I needed a gateway to receive its LoRa packets and upload them to the internet. Connecting to both LoRa and WiFi are by now relatively standard operations, and so there are many sensible ways to solve this, such as dev-boards with both LoRa and Wifi support, or something even easier to program like a Raspberry pi with a LoRa shield.

Unfortunately, I neglected this part of the design until late in the project, and so did not pick any of these sensible options, instead cobbling together a gateway out of parts I had lying around. I used a Heltec Cubecell AB02A (LoRa board), and a NODEMCU ESP32-S dev board for wifi access, with the two of them

communicating over SPI. I was able to find Arduino libraries to program these in, which simplified the firmware to just get something working.

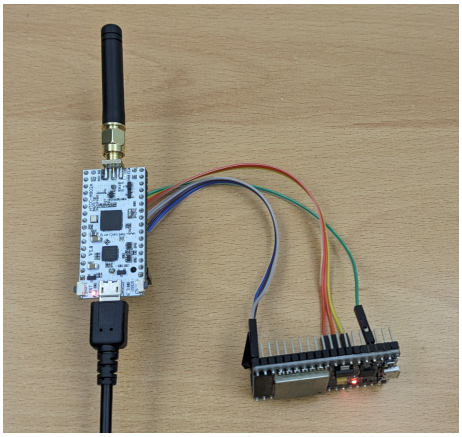
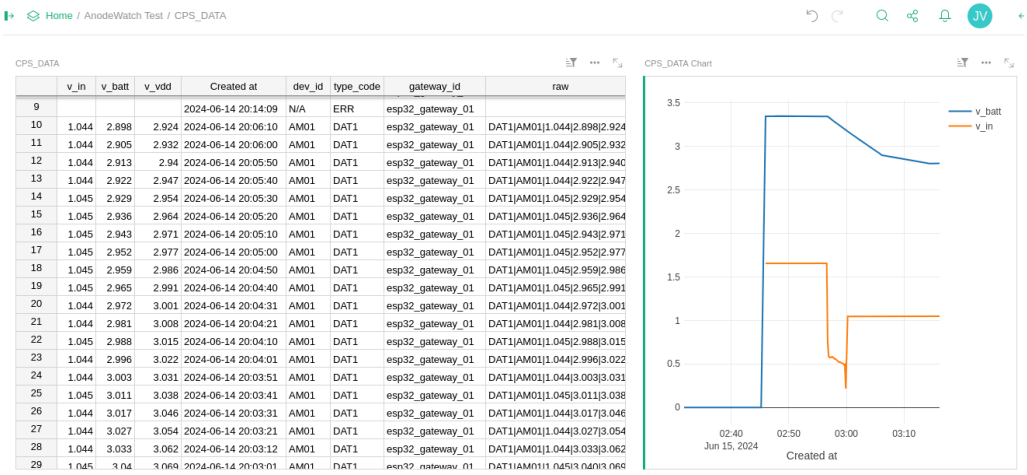


Figure: The Gateway

A further issue is that the campus WiFi is a nightmare to connect to; there exists a UCSD-DEVICE network which just takes a simple SSID and password, but it requires an approval process to get your device on the network, so for the demo in this paper I used my phone's wifi hotspot to test with.

I configured the gateway to push data into a Grist document. Grist is an open-source cloud spreadsheet application I'm fond of, and which might be well-suited to this application even in a more serious deployment, as it has good API support and granular permissions, making it easy to make parts of the data public while still ensuring it won't be tampered with.



	v_in	v_batt	v_vdd	Created at	dev_id	type_code	gateway_id	raw
9				2024-06-14 20:14:09	N/A	ERR	esp32_gateway_01	
10	1.044	2.898	2.924	2024-06-14 20:06:10	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.898|2.924
11	1.044	2.905	2.932	2024-06-14 20:06:00	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.905|2.932
12	1.044	2.913	2.94	2024-06-14 20:05:50	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.913|2.940
13	1.044	2.922	2.947	2024-06-14 20:05:40	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.922|2.947
14	1.045	2.929	2.954	2024-06-14 20:05:30	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.929|2.954
15	1.045	2.936	2.964	2024-06-14 20:05:20	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.936|2.964
16	1.045	2.943	2.971	2024-06-14 20:05:10	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.943|2.971
17	1.045	2.952	2.977	2024-06-14 20:05:00	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.952|2.977
18	1.045	2.959	2.986	2024-06-14 20:04:50	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.959|2.986
19	1.045	2.965	2.991	2024-06-14 20:04:40	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.965|2.991
20	1.044	2.972	3.001	2024-06-14 20:04:31	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.972|3.001
21	1.044	2.981	3.008	2024-06-14 20:04:21	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.981|3.008
22	1.045	2.988	3.015	2024-06-14 20:04:10	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|2.988|3.015
23	1.044	2.996	3.022	2024-06-14 20:04:01	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|2.996|3.022
24	1.044	3.003	3.031	2024-06-14 20:03:51	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|3.003|3.031
25	1.045	3.011	3.038	2024-06-14 20:03:41	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|3.011|3.038
26	1.044	3.017	3.046	2024-06-14 20:03:31	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|3.017|3.046
27	1.044	3.027	3.054	2024-06-14 20:03:21	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|3.027|3.054
28	1.044	3.033	3.062	2024-06-14 20:03:12	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.044|3.033|3.062
29	1.045	3.04	3.069	2024-06-14 20:03:01	AM01	DAT1	esp32_gateway_01	DAT1|AM01|1.045|3.040|3.069

Figure: Grist document storing logged data

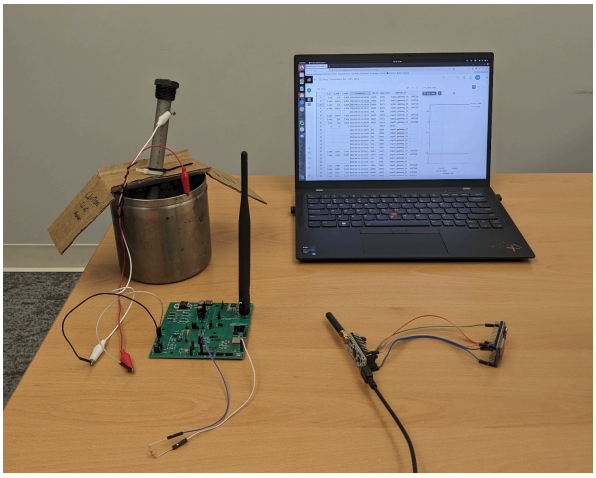


Figure: Working mk1 board, bucket anode, gateway, and spreadsheet

3.7 Final Deployment / MVP

Scaling back from my initial lofty goals of field deployment, I put together a test setup consisting of a magnesium anode rod in a steel bucket. I verified it was able to slowly charge up from an input power of 150uW (0.6V, 600Ω ESR).

Finally, I was able to put it all together and get data transmitted from the board, through the gateway, and up to the internet. Testing at a slightly higher power level of 250uW, it was able to charge much faster and transmit a radio packet roughly once every 5 minutes.

4 Milestones

4.1 Initial Milestones

These are the initial milestones I had set at the start of the quarter (and how they turned out)

- ✅ [DONE] Test Artemis dev-board
- ✅ [DONE] Schematic v1 and boards order (DONE)
- 🟨 [PARTIAL] Prototype dev-board with energy harvesting, sleep modes, radio
- 🟨 [PARTIAL] Find test points to deploy at
- 🕒 [DELAYED] Get radio working, set up gateway
- 🕒 [DELAYED] Assemble / Debug boards
- ❌ [NOPE] Schematic v2, cases
- ❌ [NOPE] Deploy on campus

By the mid-quarter update, I had gotten the build system, programmers, and SDK working with the Artemis board, and gotten basic communication established with the other modules on the board. I had also tested out the energy-harvesting ADP5091 chip and verified that it was able to charge off of low-power sources.

However, I was not yet able to fully power the system off of the ADP5091, as the Artemis dev-board had other components which would draw too much power, and it didn't have ports for me to accurately measure current

consumption. I therefore put off further development of the firmware, sleep modes, and radio gateway to prioritize designing and ordering the PCBs and parts.

My mid-quarter update milestones therefore looked like this: (and how they turned out)

- 🕒 [DELAYED] Assemble / Debug mk1 boards
- 🟨 [PARTIAL] Scout CP test points on campus
- 🟨 [PARTIAL] Plan enclosures / deployment plans for mk2 boards
- 🕒 [DELAYED] Write LoRa / Sleep mode Firmware
- 🕒 [DELAYED] Setup gateway/backend
- 🕒 [DELAYED] Documentation / Reports
- ❌ [NOPE] Order mk2 boards
- ❌ [NOPE] Deploy on campus

I was able to successfully assemble and test my mk1 board. Unfortunately, the order was delayed a week more than expected, and, once assembled, it took several more days of debugging and PCB-surgery to get it working. While the board did end up fully functional, the delay meant I had to drop the goal of building a mk2 version of the board, as there wouldn't be enough time to test the mk1, finalize the mk2 design, and get it shipped.

My goal of scouting deployment locations on campus was also a mixed success. I was able to locate 6 CP Test points and was able to get into 4 of them. However, none of the test points I looked at had any labeled cables or detectable voltage drop. In hindsight, I suspect that some of these may have been secondary test points that connected to sections of pipe but not anodes, while another one may have been a current-shunt that I would need to interrupt to be able to power my sensor off of it. Either way though, it was unclear if there was a straightforward way to deploy sensors at the actual test points safely; getting this done properly would require more research on cathodic protection systems, getting documentation from campus facilities about how the CP test points were set up, and some more serious design work to ensure that my sensor could reliably act as a current shunt without harming the performance of the cathodic protection systems I attached it to.

As a result of the setback with the mk2 boards and the lack of a clear deployment story, I scaled back the deployment goal to a desktop setup consisting of a magnesium anode rod in a steel bucket. I was able to complete the final milestones (firmware, and LoRa-Wifi gateway) this week and test them on my magnesium bucket, finally getting to a working MVP.

5 Conclusions

My goal was to develop a low power, energy-harvesting sensor system for monitoring cathodic protection systems, and to try to deploy it in the real world. I was successful in designing, building and testing an energy-harvesting sensor, and got a working end-to-end demo of sensor measurements being logged to the web, powered by the corrosion of a magnesium anode.

However, my goal of deploying it in the real world was less successful: I spent most of my focus on designing, implementing, and testing the electronics of the system, and neglected to verify the real-world requirements to deploy it until it was too late to change the design. While this is probably fine for a quarter-long class project to

develop skills with embedded systems, it is perhaps also a good cautionary tale on what not to do when working on projects intended to have real-world impact.

REFERENCES

[1] Analog Devices, “Ultralow Power Energy Harvester PMUs with MPPT and Charge Management,” ADP5091 datasheet, rev A, 2016. https://www.analog.com/en/products/adp5091.html

[2] Knowles, “DGH Low ESR Supercapacitor,” DGH105Q2R7 datasheet, March 2024 https://www.cde.com/resources/catalogs/DGH.pdf

[3] Dhananjay Jagtap and Pat Pannuto. 2021. Repurposing Cathodic Protection Systems as Reliable, in-situ, Ambient Batteries for Sensor Networks. In The 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021) (IPSN ’21), May 18–21, 2021, Nashville, TN, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3412382.345827

[4] National Academies of Sciences, Engineering, and Medicine. 2009. Cathodic Protection for Life Extension of Existing Reinforced Concrete Bridge Elements. Washington, DC: The National Academies Press. https://doi.org/10.17226/14292

[5] G. H. Koch, M. P. Brongers, N. G. Thompson, Y. P. Virmani, and J. H. Payer. Corrosion cost and preventive strategies in the United States. Technical report, United States. Federal Highway Administration, 2002.

[6] S. Kim, M. Lee and P. H. Chou, "Energy harvesting from anti-corrosion power sources," *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, La Jolla, CA, USA, 2014, pp. 363-368, doi: 10.1145/2627369.2627624.

[7] Guofu Qiao, Guodong Sun, Yi Hong, Yuelan Qiu, Jinping Ou, Remote corrosion monitoring of the RC structures using the electrochemical wireless energy-harvesting sensors and networks, NDT & E International, Volume 44, Issue 7, 2011, Pages 583-588, ISSN 0963-8695, https://doi.org/10.1016/j.ndteint.2011.06.007.