RetroIoT: Retrofitting Internet of Things Deployments by Hiding Data in Battery Readings

Victor Ariel Leal Sobral*
University of Virginia
sobral@virginia.edu

Nurani Saoda*
University of Virginia
saoda@virginia.edu

Ruchir Shah
University of Virginia
rvs4xt@virginia.edu

Wenpeng Wang
University of Virginia
wangwp@virginia.edu

Bradford Campbell
University of Virginia
bradjc@virginia.edu

ABSTRACT

Commercial Internet of Things (IoT) deployments are mostly closed-source systems that offer little to no flexibility to modify the hardware and software of the end devices. Once deployed, retrofitting such systems to an upgraded functionality requires replacing all the devices, which can be extremely time and cost prohibitive. End users cannot generally leverage deployed infrastructure to add their own sensors or custom data. However, we observe that IoT systems sometimes report battery voltage information to the cloud, and batteries are often user-serviceable. This indicates that perturbing the battery voltage to encode customized information could be a minimally invasive method to retrofit existing IoT devices.

In this paper, we propose a new approach, RetroIoT, to encode custom commands and data into the battery voltage channel of IoT systems and retrofit devices with enhanced capabilities. RetroIoT enables this functionality by replacing the device's original battery with a controlled power supply that manipulates the input voltages of the battery terminal. RetroIoT can encode both analog values and digital symbols which are later decoded once the battery voltage readings are stored in the cloud. This retrofit data channel enables transmitting additional data, sending new metadata, and even swapping batteries for energy-harvesting. This technique requires no modification to the IoT device beyond replacing the battery. We prototype this technique using two commercial LoRa devices and one BLE device. Results show a 95th percentile channel error of only 3.96 mV and 99% successful packet decoding with digital symbols.

CCS CONCEPTS

• **Computer systems organization** → **Sensor networks**; **Embedded systems**.

KEYWORDS

IoT, Deployment, Retrofitting, Energy-harvesting

*Both authors contributed equally to this research.

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9181-8/22/10.
https://doi.org/10.1145/3495243.3560536

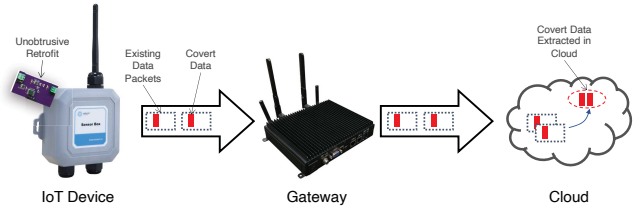


Figure 1: Many IoT devices sample and report their battery voltage, and by simply swapping the battery these devices can be repurposed to encode additional useful information. This retrofitting gives users new control to capture new data, upgrade to energy-harvesting, or strategically deactivate sensitive sensors.

ACM Reference Format:
Victor Ariel Leal Sobral, Nurani Saoda, Ruchir Shah, Wenpeng Wang, and Bradford Campbell. 2022. RetroIoT: Retrofitting Internet of Things Deployments by Hiding Data in Battery Readings. In *The 28th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '22), October 17–21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 13 pages.
https://doi.org/10.1145/3495243.3560536

1 INTRODUCTION

Commercial Internet of Things (IoT) systems today are commonly "walled gardens", as vendor lock-in advantages, first-to-market benefits, and interoperability overhead conspire to incentivize companies to develop their own end-to-end IoT solutions. This leads to closed-source implementations with few configurability or modification capabilities accessible to end users. As a counterpoint, open-source and maker-lead IoT platforms and systems offer significant flexibility to users and developers, with the potential for significant interoperability, but often at the cost of robustness, aesthetics, and ongoing support. Establishing design points between these extremes would enable IoT users and developers to leverage well-supported IoT infrastructure, while being able to customize their IoT systems for their own requirements and applications. Further, innovation often flourishes when open channels are introduced to previously closed systems and the broader community is able to experiment with and develop for the platform.

Enabling users to leverage the infrastructure of their existing IoT systems, including the sensors, wireless networks, gateways, cloud backends, and cloud APIs, without having to build their own devices or replicate the infrastructure could enable a series of upgrades to

existing IoT deployments. Users could add a new type of sensor that perhaps the manufacturer does not offer, and have its data flow to the cloud with the rest of the devices. Batteries could be replaced with energy-harvesting power supplies, and harvesting performance could be sent to the cloud. Additional metadata about a sensor's position or intended use could be added to the system, aiding deployment. Also, users could add an attestation device that verifies a sensor has not been moved or tampered with.

But without open standards, entirely new IoT nodes, or invasive hacking, how can users extend their existing IoT systems? Our observation is that many IoT devices report their battery voltage in addition to their sensor data. This enables notifying the user when the battery must be replaced, but most of the battery reports are effectively unused. We claim this channel can be used to encode completely new information beyond the device's initial intended application. We propose RetroIoT, an approach that replaces a standard battery with a "programmable" battery that can control its own voltage output and encode additional information into the battery voltage level. Later, the voltage readings can be retrieved from the cloud and decoded, and a new data channel is introduced without modifying the existing IoT devices beyond just replacing the batteries (Figure 1). As we show, the battery voltage channel of IoT devices can be repurposed using oblivious devices and prototype new capabilities to retrofit the initial use case. This technique can enhance existing devices to improve sustainability and privacy without waiting on manufacturers to produce battery-free or privacy-first IoT devices.

Replacing the battery is appealing for two reasons, first, as batteries need replacing they are typically easily accessible. Second, the battery voltage reports are largely unused in normal operation, except when the battery is actually at a low state of charge. We show how it is possible to convert the positive and negative terminals of the battery into a general end-to-end communication channel using two approaches. The first uses analog values, where analog measurements are converted into a typical operating range for a battery and transmitted directly. The second is a digital channel where certain voltage values represent communication symbols and entire packets can be transmitted to the cloud.

Creating this channel with oblivious devices requires addressing some key challenges. To synchronize the battery replacement hardware with the unknown operation of the device, we first monitor the power draw of the device and identify periodic peaks that likely indicate a wireless transmission. To calibrate with the battery measurement circuit, we sweep values to identify suitable voltages to act as communication symbols. And third, to support arbitrary length messages over this new low bitrate channel, we use reserved symbols to inform the receiver of the start of a new message and the end of a previous message.

Moreover, we attempt to understand the generality of the concept of hiding data in any underused IoT data channel beyond just the battery voltage channel. We identify several accessible channels in existing IoT channels which can be "hacked" to encode interesting information. For example, the Awair Glow air quality monitor [3] includes a switchable power socket with a physical button to control the plug socket. When the socket is switched on or off, the Awair reports that change back to the cloud and to the associated smartphone application. However, the Awair is a functional air quality monitor when the socket is empty, and that on/off channel goes unused. By using such channels, a user can send a new stream of data to the cloud to support new use cases independent from the original system.

After enabling the communication channel we show its utility by adding a new sensor and digital metadata to existing IoT networks, and converting a fully battery powered device to energy-harvesting. As batteries incur maintenance overheads [1, 8, 13] and there are billions of battery powered devices already deployed with more continuing to sell, the promise of battery-less devices is significant [9, 13]. Retrofitting existing devices with energy-harvesting will help accelerate the IoT away from battery waste. However, energy-harvesting devices often must adapt to their ambient harvesting conditions, and battery-powered devices assume a constant power supply. To alleviate this, we leverage the new channel to allow the energy-harvesting power supply to send the device's desired duty-cycle to the cloud, and then use existing update and control utilities to adjust the device's operation. This enables a feedback loop with otherwise oblivious devices.

We prototype RetroIoT to characterize the underlying channel properties in terms of encoding error, bit error rate and percentage of successful packet decoding. Our experiments show that the proposed encoding technique incurs only 3.96 mV 95th percentile error and we can decode 99% of the packets successfully at a channel resolution of 11.8 mV. We also perform a real world deployment study where we reprogram the battery voltage channel of three existing IoT devices with different wireless communication protocols: two LoRa devices and one Bluetooth Low Energy (BLE) device, and encode various types of additional information into the channel.

RetroIoT also highlights potential concerns and opportunities for IoT devices with user-serviceable batteries. Devices typically implicitly trust their batteries, but as we show a programmable battery can transmit data unbeknownst to the device. Further, a swapped battery could be very difficult to visually detect. Alternatively, however, batteries are of course necessary for a battery-powered device, and disconnecting the battery disables the entire device. Replacing a normal battery with a controllable one then provides the user with a method for entirely disabling a device without relying on the device's programming. Exploring the battery-device interface can yield new approaches for modifying future IoT devices.

The key contributions of the paper are summarized as follows:

- We propose and demonstrate a new technique to enhance the functionality of existing IoT deployments by encoding information into the battery voltage channel. IoT devices adopt RetroIoT by replacing their batteries with a controlled power supply, and do not require further modifications.
- We design two encoding-decoding techniques to send both analog and digital information over the channel, opening a wide range of retrofitting applications.
- We implement a prototype of the system in a custom hardware with a reasonable form factor that can be used to retrofit many existing IoT devices.
- We propose a design space for leveraging generic underused data channels, and discuss the potential security and privacy risks this technique uncovers.

2 EXAMPLE RETROFIT APPLICATIONS

Retrofitting existing IoT infrastructure is motivated by a wide range of application opportunities, for example, adding new sensing capabilities, inserting metadata tags, and replacing batteries with energy-harvesting power supplies. Although these applications can be enabled by designing a completely new IoT device, this leads to high time and cost overheads, even with existing gateway and cloud support. Instead, we advocate for solutions that re-use existing infrastructure to enable simpler retrofitting strategies. To highlight potential opportunities, we discuss some suitable retrofitting applications.

Analog Sensor Add-on. One interesting application is to encode the readings of an analog sensor in the battery output voltage. Users can then decode the reported battery voltage readings to retrieve the analog sensor readings in their cloud applications. This effectively adds a new sensor to an existing, legacy IoT device. Any analog voltage can be encoded using such a technique. For example, a sensor initially used to detect when a fire extinguisher is moved could be upgraded to report the fire extinguisher's storage temperature. We demonstrate this fire extinguisher application in Section 9.4 by retrofitting a door sensor device to also report temperature through battery voltage levels.

Digital Sensor or Metadata Tag Add-on. An IoT device can be augmented with a digital sensor reading or a new metadata tag. For this class of applications, digital symbols are encoded as voltage levels in the power supply. These digital values could represent new metadata tags, such as the deployment location or context of an IoT sensor that are usually only known at deployment time. Alternatively, digital sensor readings could be transmitted. In general, any digital value can be encoded using this same technique. As a first demonstration of this type of enhancement, we show how a door event sensor can be updated to transmit a new metadata tag with information such as in which room and on which object it is installed. In a second demonstration, we encode a timestamp for a temperature and humidity sensor to track not only ambient storage conditions but also the expiration date of perishable goods such as food or medications.

Energy-harvesting Add-on. Battery-powered devices can be made completely battery-less by accommodating an energy-harvesting power supply with RetroIoT. With energy-harvesting functionality, a device's lifetime is significantly improved, alleviating the need of battery replacement. We demonstrate this capability by replacing a soil moisture sensor's [5] battery with an energy-harvesting power supply. The energy-harvesting power supply monitors the energy harvesting rate of the sensor and optimizes the device's operation to achieve energy-neutrality. Achieving energy-neutrality is crucial for indoor light energy-harvesting scenarios where the amount of harvestable energy can be low [14, 23]. In our application, we demonstrate how a simple dynamic power management algorithm in the smart power supply can monitor stored energy and encode command messages using the battery voltage communication channel. These commands are then decoded by a cloud application and the communication frequency of the IoT device is adjusted.

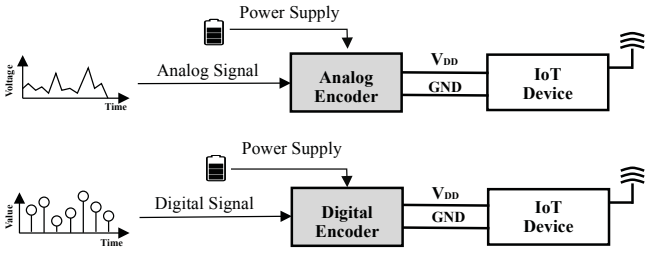


Figure 2: Overview of RetroIoT.

3 RETROIOT SYSTEM OVERVIEW

We introduce RetroIoT, an approach to modulate additional information using the battery terminals of an IoT device. RetroIoT replaces a conventional battery with a programmable voltage controller and augments the IoT device with desired functionality. It interoperates with many existing systems that already have functional hardware, software, and network infrastructure. Figure 2 shows the high-level overview of RetroIoT. Any analog and digital input data is mapped into the acceptable input voltage range of the attached IoT device. The main block of RetroIoT is the signal encoder that implements the modulation of symbols on the battery voltage channel from raw signal values. The encoder output includes the encoded voltage as well as provides power to the IoT device as the conventional battery would.

RetroIoT enables users, hobbyists, and IoT developers to take advantage of the existing infrastructure of closed source commercial devices, without requiring them to build the whole stack from scratch. This way, RetroIoT promotes re-usability and faster system development, and benefits deployments that require efficient and low-impact upgrades. It demonstrates a new design point for modifying existing IoT systems. RetroIoT is not a universal replacement for IoT redesign, but represents an option for applications where the value of a new data channel with existing devices is high and the limitations of increased power draw, power supply design effort, and limited data rate are acceptable. This is particularly true in cases where the alternative would require hardware and software updates in the IoT's gateway and server infrastructure. We elaborate on this further in Section 11.

4 DESIGN CHALLENGES

4.1 Minimal Modifications

To make retrofitting legacy IoT systems viable, integrating new capabilities into the existing IoT hardware and software infrastructure must require minimal changes. For instance, modifying the IoT device's software or wireless protocol is likely infeasible. We therefore assume a solution cannot require modifying the device's code, tweaking hardware settings, changing radio communication parameters, or introducing new networked devices. To meet this requirement, we only require replacing the battery with a new device, and as batteries are typically intended to be user-serviceable, this is a non-invasive option. However, we assume the IoT device sends the raw battery voltage values to the cloud for further processing, and the battery voltage information is retrievable by applications.

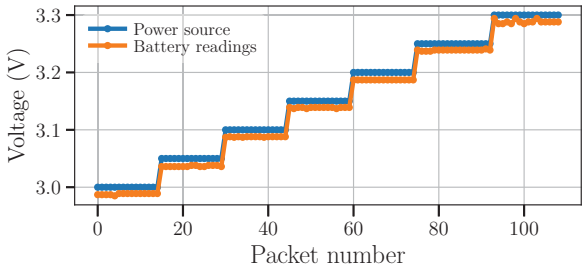


Figure 3: Battery voltage readings sent by a LoRa IoT device to the cloud via a LoRa gateway. This demonstrates the feasibility of encoding data in the battery voltage readings.

4.2 Power Supply Constraints

Replacing the battery with additional circuitry imposes several challenges. First, the retrofit must not interrupt the IoT device's normal operation. That is, the supply voltage and current from the encoder output must be within the expected range for the device's original primary battery. For example, an IoT device originally operating with 2 AAA batteries expects a voltage between 2.7 V and 3.3 V. This constraints the voltage range available to encode information. Second, the current draw of the legacy device is considered unknown, and a device with a high dynamic range of current draw can affect the output of the retrofit device. For example, if the expected encoded voltage is set to 3.27 V, this should be stable whether the device's current consumption is 1 mA or 20 mA. Further, as the retrofit device must replace the energy supply, it must be able to output the expected voltage regardless of the voltage of its own underlying energy supply. In addition to these short-term conditions, the energy consumption overhead introduced by the encoder must be minimized to not unduly shorten the IoT device's operating time.

4.3 Battery Reading Resolution

The retrofit device can optimize the resolution and accuracy of its programmable voltage supply connected to the IoT device. However, due to the minimal modification constraint, the retrofit is still constrained by the battery voltage monitoring circuitry and software used on the IoT device. For example, if the IoT device expects a maximum of a 3.3 V supply, and uses a 12 bit ADC to collect battery voltage readings, the voltage resolution of these readings is $3.3/(2^{12} - 1) = 0.806$ mV. That results in approximately 372 distinguishable voltage levels between 3.0 V and 3.3 V. This implies we could theoretically encode 8 bits of data by assigning voltage levels to the numbers 0-256. In practice, there is no standard for how should IoT devices report battery level. Manufacturers choose their own ADC resolution and the decimal precision of the battery voltage with which it can be retrieved from the cloud. Some also use battery level percentages instead of the actual battery voltage. According to our experience, it is common for IoT devices to report battery voltage readings with resolutions between 1 and 10 mV [5, 6, 10]. In addition to limited resolution, other battery voltage reading limitations are signal noise, nonlinearity, and offsets. We explore some of these challenges in more depth as well as an error mitigation strategy in Section 5.

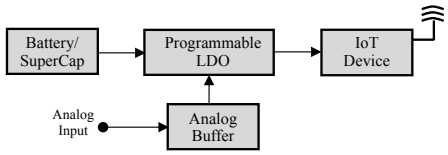
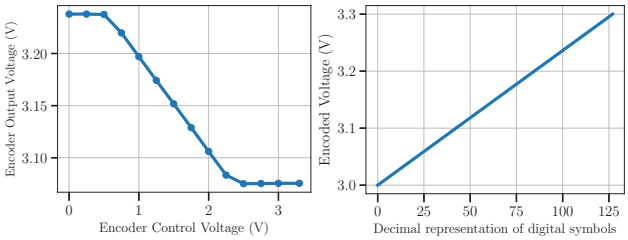


Figure 4: Analog encoder hardware design.



(a) Analog encoder transfer function.

(b) Digital encoder transfer function.

Figure 5: Transfer functions of the encoder.

4.4 Data Synchronization

From the analysis in Section 4.3, a single voltage reading can transmit a single byte of data. Sending more information will require using multiple battery voltage readings. However, this requires the retrofit device to loosely synchronize with the IoT device to set the voltage every time the device samples the battery voltage. Otherwise the same voltage value could be sent multiple times, or a value could be missed.

4.5 Recovering Transmitted Data

Once data has been encoded and the IoT device has (unknowingly) transmitted the data to its cloud backend, a processing algorithm must be able to recover the transmitted data successfully. This includes understanding how to divide the stream of transmitted voltage readings into the intended packets of data. We propose one possible and simple solution where two reserved symbols are used as a flag to signal the beginning and end of a multi-symbol message. These symbols will also be used in a decoding error mitigation step, discussed in more detail in Section 5.3.

5 RETROIOT ENCODER DESIGN

In this section, we design an approach to use the battery voltage channel to send analog readings and digital symbols over legacy IoT devices' network infrastructure.

5.1 Voltage Encoding Feasibility

As a proof of concept demonstration of the proposed approach, we attach a bench top voltage supply to the 3.3 V power rail of a LoRa IoT device and verify if we can receive the programmed voltage levels from the cloud. The device uses an ADC resolution of 12 bits. Figure 3 shows the battery readings collected in the cloud via the LoRa network against the ground truth input values. The close match suggests this approach is feasible.

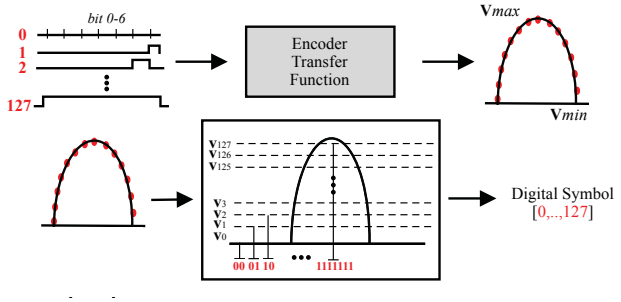


Figure 6: This diagram shows how the digital data is encoded (top) and decoded (bottom). The encoder converts the 7-bit digital symbol into a battery voltage value within v_{min} and v_{max}. The decoder function translates the encoded battery voltage back to a digital symbol.

5.2 Analog Encoder Design

The analog encoder directly translates an analog signal to a voltage suitable for the battery monitoring circuitry. The encoder accepts an input voltage ranging from 0 V to 3.3 V and adjusts the output of a low-dropout regulator (LDO) between 3.0 V and 3.3 V. The block diagram of the circuit is depicted in Figure 4. An analog buffer connects to an operational amplifier in a follower configuration to isolate the input signal source and the encoder control circuit. As the control voltage increases, the current flowing through the feedback resistor decreases, reducing the output voltage. Figure 5(a) represents the relationship between the control voltage and output voltage for the analog voltage encoder circuit. Although the usable control voltage range in this circuit is between 0.5 V and 2.3 V, this can be adjusted by adding appropriate gains and offsets with op-amp based analog circuits. The measured bias current for this circuit is 0.19 mA without any load connected to the encoder regulated output.

This simple analog voltage encoder supports directly connecting an analog input, for example an analog sensor, creating an easy-to-use option for retrofitting using the battery voltage monitoring channel. We demonstrate this with an end-to-end example using a temperature threshold detection alarm in Section 9.4.

5.3 Digital Encoder Design

Using a purely analog input voltage reduces complexity, but limits the amount and type of data that can be transmitted using this channel. To show how arbitrary data can be transferred, we describe a technique to encode digital data into a range of battery voltage values and how to decode the received battery voltage to retrieve the sent information.

5.3.1 Data Encoding-Decoding. First, the data to be transmitted must be converted to a series of symbols. We select 7-bit values to represent the symbols based on the capabilities of our DAC device and the voltage range available to encode information. With a 7-bit representation, there exist 128 unique digital symbols each translating into a distinguishable voltage level. We define the difference between two consecutive encoded voltage as the resolution of the encoding v_{rs}. For instance, the first encoded voltage can be calculated as $v_1 = v_0 + v_{rs}$. Voltages v_0 and v_{127} are respectively the minimum (v_{min}) and maximum (v_{max}) encoded voltages. Therefore, the n-th encoded voltage level can be denoted as $v_n = v_0 + nv_{rs}$, where v_n is n-th voltage level. Both the resolution of the encoding voltage and the IoT's reported battery reading resolution affects how well the symbols can be retrieved by a cloud application. In Figure 5(b) we plot the relation between encoded voltage between 3.0-3.3 V and decimal representation of the corresponding symbols.

To decode the information on the cloud application, the received battery voltage levels must be converted to symbols and then properly interpreted. A voltage level v is decoded as a unique symbol n if it satisfies $(v_n - \frac{v_{rs}}{2} < v < v_n + \frac{v_{rs}}{2})$. Figure 6 shows a block diagram of this process.

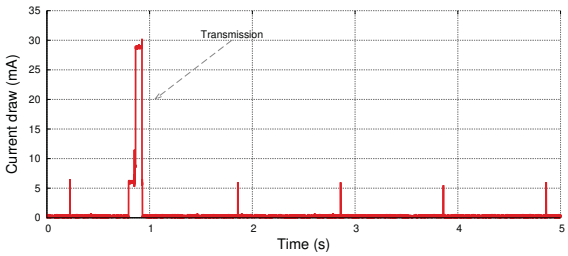


Figure 7: Current draw profile of a LoRa sensor [6] showing a distinct radio transmission spike.

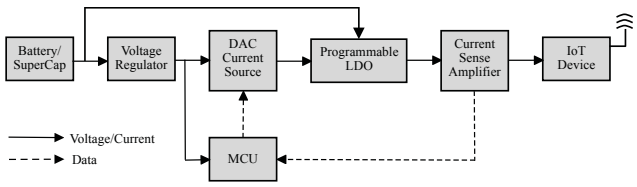


Figure 8: Block diagram of the digital encoder hardware design.

5.3.2 Data Synchronization. To ensure that the symbol to be transmitted is encoded approximately right before the device transmits a radio packet, the encoder needs to learn the device's transmission schedule. This is essential to support packets of data spread over multiple transmissions. We propose achieving this synchronization by measuring the current draw of the legacy IoT device, and observing spikes in the current trace. As battery powered devices must minimize their current draw, wireless transmissions will likely result in distinct spikes in the current trace as shown in Figure 7. The retrofit device can then update the voltage value every time it detects a transmission event. As a majority of the IoT sensing applications are fairly periodic, the battery voltage encoder observes the device's current draw and measures the time difference between two consecutive peaks resulting from a radio communication to determine the transmission interval. Then, it uses this interval to determine when to encode the next symbol.

5.3.3 Hardware Design. Figure 8 depicts the block diagram of the hardware design of the digital encoder. An I^2C-controlled digital-to-analog converter (DAC) current sink/source IC adjusts its output

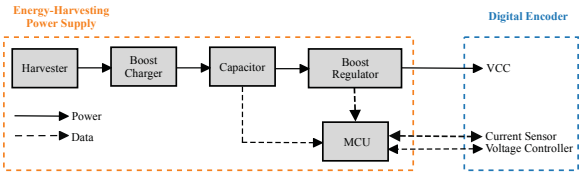


Figure 9: Energy-harvesting power supply module interfacing with the digital encoder.

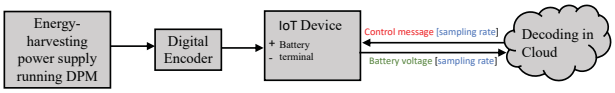


Figure 10: The retrofitting energy-harvesting power supply runs a dynamic power management algorithm locally and encodes the updated sensor sampling rate in the battery voltage. The sensor is then re-configured by the cloud control message to adjust device behavior.

current across 128 values to produce a variable output voltage signal. The variable output current of the DAC is injected into the feedback node of a voltage divider that feeds into an adjustable output voltage low dropout regulator. The DAC current source programs the LDO regulator to be configured at one of the 127 voltage output levels. We use an ultra-low power MCU to send symbols through the I^2C interface of the current DAC. The MCU uses a current-sense amplifier to monitor the IoT device's current draw and calculates the transmission interval of the device.

5.4 Decoding Error Mitigation

By running encoding and decoding experiments as depicted in Figure 3, we identify that the difference between the encoded power source voltage and the IoT's battery reading voltage can be modeled as the sum of a constant and a linear term, representing offset error sources from the voltage encoder and the IoT device's ADC. To mitigate these errors in the symbol decoding process, we estimate the encoded power source voltage before decoding the received symbol. To estimate the encoded power source voltage from the IoT device's battery readings, we perform a linear interpolation using the maximum and minimum IoT device's battery readings (v_{bmax} and v_{bmin} respectively), obtained from setting the encoded power source voltage to v_0 and v_{127}, respectively. Equation (1) shows how we estimate the encoded power source voltage v_{pwr} from the IoT device's battery voltage reading v_b.

$$v_{pwr} = v_0 + (v_b - v_{bmin}) * \frac{(v_{127} - v_0)}{(v_{bmax} - v_{bmin})} \quad (1)$$

To perform this interpolation we assume the maximum and minimum encoded voltages (v_0 and v_{127}) are special encoded voltage levels used only for calibration purposes (0 and 127 are then reserved symbols), while also periodically reporting them so they can be later used in the v_{pwr} estimation and decoding process. This approach results in reduced decoded bit error at the cost of decreased bandwidth due to the use of reserved symbols and special calibration messages as we will evaluate in Section 9.2.

6 ENERGY-HARVESTING RETROFITTING

One of the promising applications of upgrading a deployed IoT system is to replace batteries with energy-harvesting power supplies. However, successful energy-harvesting systems must adapt their execution based on available energy. A device designed with a reliable source of energy (e.g. a battery) will not have the programming or included logic to adjust its own operation based on the current harvesting conditions. In this section, we show how the retrofitting approach can address this challenge.

We start by replacing the battery with an energy-harvesting power supply that connects to the existing power and ground terminals, as shown in Figure 9. The new power supply replaces the energy store with a supercapacitor that is recharged with a harvester. A second stage voltage regulator regulates the capacitor voltage and supplies a constant voltage to the rest of the system.

In ideal harvesting conditions simply doing this swap would be sufficient. However, the available harvestable energy may not be sufficient to recharge the capacitor at the rate the legacy IoT device requires. To address this, we integrate a small microcontroller into the replacement power supply. The MCU observes the state of charge of the storage element and the incoming harvested energy. If it detects a shortfall, it configures the IoT device to reduce its operation to conserve energy. Since the new power supply is only connected via the old battery terminals, it cannot do this directly.

Instead, we leverage the underused battery voltage channel. To adjust the device's duty-cycle, the MCU creates a message by encrypting the recommended duty cycle in the battery voltage and transmits it to the cloud. An application hosted in the cloud receives the device's message and then tries to alter the device's operation to match the available energy constraints. This process is illustrated in Figure 10. IoT devices often times allow some degree of re-configuration through cloud APIs, particularly related to update rates. For example, control messages may be able to set the sampling period [5] or the keep-alive interval [6]. The cloud application uses one of the existing methods to send a control signal to the device to adjust the operation of the legacy IoT device.

Our proposed dynamic power management algorithm is shown in Algorithm 1. At each interval t_p, the power supply checks the gradient of storage voltage and if the gradient is either zero or has a positive value, the cloud is instructed to increase the sampling frequency of the sensor, and vice versa.

7 POTENTIAL RETROFIT CHANNELS

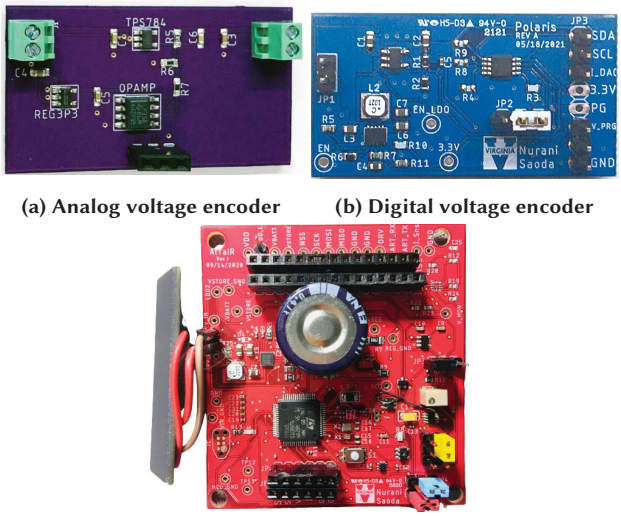
Though RetroIoT system builds upon the battery voltage channel, we also note a range of additional underused data channels that facilitate future retrofit opportunities.

Binary Inputs. Certain wall-plug sensors provide a switchable outlet in addition to their sensor (such as the Awair Glow). This binary channel can often permit bi-directional communication (i.e. physical control and signals from the app).

Add-on Sensors. Certain IoT devices, including smart sprinkler and outdoor lighting controllers, include connection points for external sensors (such as a rain sensor or light sensor) to allow users to customize the device for their use case. These sensor inputs

Algorithm 1 Dynamic Sampling Rate Control

```
while true do
  at each t_p:
  check for last δt:
  if ∇v_cap >= v_th or ∇v_cap = 0 then
    f_s+ = 1
    if f_s > f_max then
      f_s = f_max
    end if
  else
    f_s− = 1
    if f_s < f_min then
      f_s = f_min
    end if
  end if
end while
```



(a) Analog voltage encoder (b) Digital voltage encoder

(c) Energy-harvesting power supply

Figure 11: Prototype voltage encoder circuit boards and energy-harvesting power supply board.

are non-proprietary, and any analog signal can be used. This input could be used for a retrofit use case.

Unused Sensor Modalities. Various IoT sensors often integrate multiple sensing modalities, but not all may be needed for control decisions. Additionally, camera sensors are increasingly including motion detection beyond just video capture. These channels could serve as mechanisms to log data or detect events, even if the actual sensor or image data is not useful.

Interactive Devices. Many smart devices support user input, such as light switches or door locks. These devices can be used to capture input unrelated to the main use case. For example, some smart lights permit multiple taps to select a particular scene. However, even if no scene is sent the action is still transmitted to the cloud. For a door lock, entering an [intentionally] incorrect passcode might cause an event to be logged. These events can be identified and used in new applications.

8 IMPLEMENTATION

We implement the RetroIoT encoder and power supply designs using prototype PCBs.

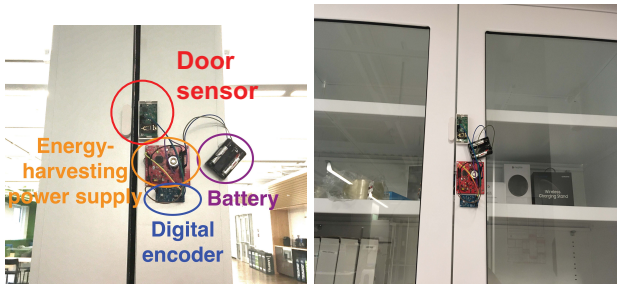
Analog Voltage Encoder. The analog voltage encoder is based on the low-dropout (LDO) regulator TPS784 [34] to adjust the output voltage. It uses the low power operational amplifier LP358 [33] to implement the buffer circuit for the analog input voltage. A LP2980 [32] LDO regulator with a fixed 3.3 V output voltage powers the operational amplifier. Figure 11(a) shows the prototype.

Digital Voltage Encoder. The digital voltage encoder board uses a Maxim Integrated DS4432 [16] DAC current source/sink amplifier. The current output of the IC can be controlled by I^2C commands to set a a variable output voltage of a LDO. We integrate a Texas Instrument TPS784 [34] as the LDO with an output voltage accuracy of ±.75%. The board also accommodates a Monolithic Power MPQ28164 [19] buck-boost switching voltage regulator with an efficiency above 85% at input voltage of 3.3 V that supplies voltage to the components. The assembled PCB is 4.3 cm by 2.3 cm. Figure 11(b) shows the hardware.

Power Supply. We adopt the Altair [24] hardware platform as the energy-harvesting power supply. Figure 11(c) shows the PCB of the energy-harvesting add-on module. The energy-harvesting power supply board accommodates an energy-harvesting front-end and an ultra-low power MCU to monitor the device current draw and send the appropriate commands to the digital encoder circuit. An ultra-low power battery charger boost converter SPV1050 [28] charges a supercapacitor from a solar or TEG harvester until it reaches 3.1 V. A nano-power boost regulator MAX17222 [17] with > 70% efficiency at 10 µA of input current regulates the supercapcitor voltage after its voltage reaches 2 V. We use monocrystalline IXYS solar cell as the harvester and a 470 mF supercapacitor with an ESR value of 25 Ω as electrical storage. We adopt an ultra-low power 32-bit ARM Cortex-M0+ STM32 [30] MCU to implement the dynamic sampling rate algorithm as explained in Section 6. The MCU monitors the load current draw by sampling a MAX9634 [18] current amplifier.

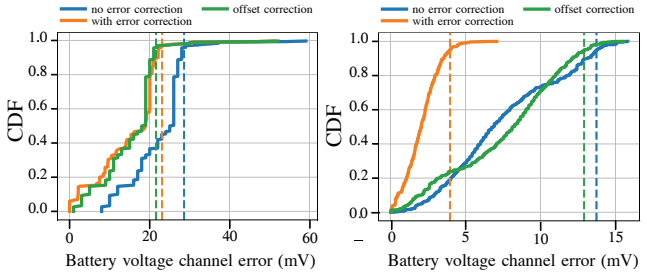
9 EVALUATION

To evaluate our system, we explore how accurately and reliably information can be retrieved from the voltage encoder through the battery voltage channel. We perform an extensive study to investigate the battery voltage channel characteristics in terms of voltage error, percentage of bit error per packet, and percentage of correctly decoded packets. We build two applications using commercial IoT devices to encode custom digital metadata, one application to retrofit with energy-harvesting, and one application to transmit readings from an analog sensor. We demonstrate how the proposed technique can help retrofit existing devices and how a functional end-to-end system can be built just by accessing the battery voltage terminal of the device.



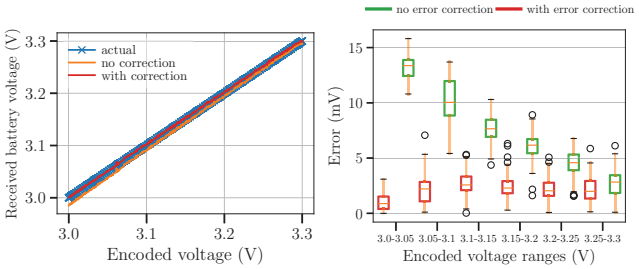
(a) On door (b) On cabinet

Figure 12: We deploy RetroIoT with a door sensor in different locations. Picture corresponds to two of the deployment scenario.



(a) Analog encoding (b) Digital encoding

Figure 13: CDF of the error in received battery voltage. The channel error is significantly reduced after calibration using the proposed error correction technique. The dash lines correspond to 95th percentile error values.



(a) (b)

Figure 14: The efficacy of the error correction technique on the battery voltage readings. After applying the error correction, the received voltage values match better with the actual encoded voltage.

9.1 Methodology

Experimental Setup. To investigate the battery voltage channel characteristics (Section 9.2), we connect the analog and digital encoder boards with a STMicroelectronics LoRaWan discovery kit [29]. The LoRaWan device measures readings from an attached MS5837-30BA [31] pressure sensor and transmits the encoded readings in the sampled battery voltage information twice every minute. We disconnect any power source from the discovery kit and replace it with the programmable voltage encoders by directly connecting it to the 3.3 V rail. We sample battery voltage with ADC resolutions of either 12, 10 or 8 bits and the reported voltage readings at the cloud have 1 mV resolution.

Retrofitted Devices and Applications. We retrofit two commercial LoRa sensing devices with upgraded functionality: 1) a door event sensor [6] and 2) a soil moisture sensor [5]. We upgrade the LoRaWan door sensor with an analog TMP37 [2] temperature sensor and a location metadata tag. We upgrade the soil moisture sensor with the solar energy-harvesting power supply. The goal of the sensor add-on experiment is to evaluate the fire extinguisher application scenario described in Section 2 by using temperature readings as an alarm to indicate unusual storage conditions. We artificially heated the sensor to simulate changes in ambient temperatures that would trigger the alarm. The door sensor sends a radio packet every minute with a door open/close event along with the battery voltage reading. The soil moisture sensor, by default, sends a reading every ten minutes. For these devices, the battery voltage is reported with 1 mV resolution. We also upgrade one off-the-shelf BLE temperature and humidity sensor [10] with long digital metadata representing a 32-bit timestamp value. This sensor reports battery voltage up to 10 mV resolution at approximately each hour.

Cloud Application. The LoRa IoT devices are connected to The Things Network gateways [35] and the messages are received and stored by a TTN application with storage and MQTT integration. For the door sensor applications, a Python script downloads the messages from the storage integration of the TTN application and then decodes the battery values. For the soil moisture sensor, a Python script running a MQTT client application connects to the TTN application's MQTT broker, then receives and decodes the sensor's messages to obtain the energy-harvesting retrofit commands. The Python script then sends the appropriate downlink command to update the wake up period of the LoRa IoT device. For the BLE sensor, the manufacturer provides a cloud API that allows sensor data and battery voltage information to be downloaded by our Python script.

Deployments and Experiments. We deploy the door event sensor with location metadata add-on functionality at four different locations: on a door, a cabinet, a fridge, and a drawer. Figure 12 shows the deployment.

9.2 Battery Voltage Channel Characteristics

In this section, we evaluate the error induced in the battery voltage channel and how the resolution of the channel affects successful decoding of information encoded in the battery voltage readings. Understanding these metrics is essential for further developments using such channels.

Received Voltage Error. To estimate the difference between the battery voltage sent from the voltage encoder and the battery voltage received at the cloud, we sweep the encoded voltage from the minimum (3.0 V) and maximum (3.3 V) values, report the value over a LoRa radio packet using the STMicroelectronics LoRaWan board. We collect two samples per minute for five minutes at each

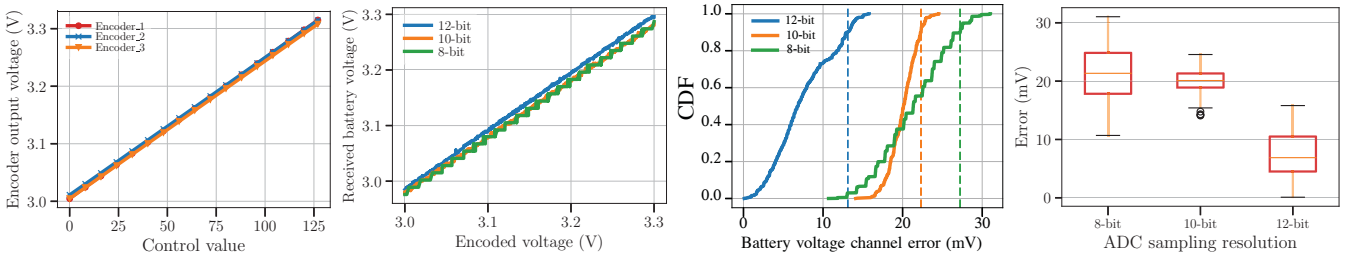


Figure 15: Understanding the effect of different sources of error due to hardware limitations of the design. a) captures the difference in encoder output voltage of three different boards. b) shows that with lower ADC resolution, the number of distinguished voltage levels is reduced, which compromises the bandwidth of the channel. c) and d) characterize the distribution of end-to-end channel error.

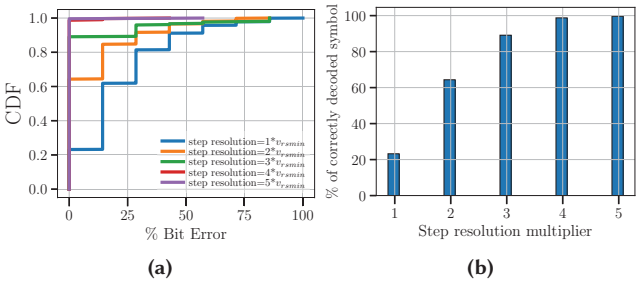


Figure 16: a) shows how the percentage bit error improves as we increase the step resolution of the voltage encoder. With step resolution, $5 * v_{rs}$ = 11.81 mV, we can correctly decode 99% of the sent symbols.

voltage level. After retrieving the battery voltages, we perform error correction on the data using Equation (1) as described in Section 5.3. We also perform a simple offset correction using just one of the two reserved symbols. We measure the CDF of error in the battery voltage, denoted by the difference between encoded voltage and received voltage values and analyze the results with and without error correction and the offset correction technique. Figure 13(a) and Figure 13(b) show the CDF of the errors while encoding analog and digital data, respectively. Battery voltages in IoT devices usually have a limited acceptable operating range below which the device is turned off. With more error induced in the battery voltage channel, the bandwidth of information that we can successfully decode decreases. The 95th percentile of the error is 28.42 mV for the analog data and 13.71 mV for the digital data without any error correction. With correction, the error can be bounded within 20.91 mV and 3.96 mV. Figure 14(a) shows the shift in voltage values after the error calibration on the digital data, which significantly reduces channel error. We further break down the error values across the whole spectrum of the voltage levels and show the variation in Figure 14(b).

Successful Decoding vs Encoder Resolution. The digital voltage encoder encodes a 7-bit data into the battery voltage. For a packet to be correctly decoded, the voltage error should be within the voltage difference corresponding to two symbols. The bandwidth of the channel is proportional to the number of achievable voltage levels. To evaluate how many bits per packet are incorrectly decoded, we analyze the CDF of percentage bit error with increasing step resolution (v_{rs}) starting from the minimum step resolution of the encoder at 2.36 mV. As shown in Figure 16(a), we observe that we can successfully decode 99% bits with a step resolution of $5 * v_{rs}$ = 11.81 mV. Figure 16(b) shows the percentage of symbols that are correctly decoded across different encoder resolution. For this experiment, we perform the error calibration before the analysis.

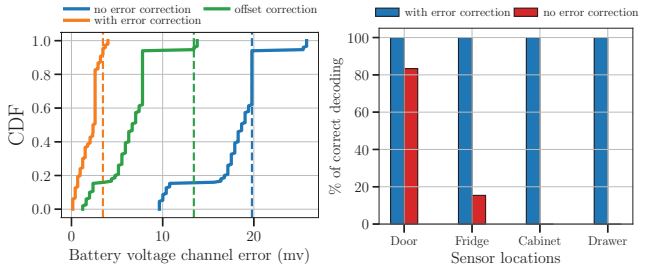
9.3 Hardware Variation Effect

We quantify the errors produced as an artifact of the hardware imperfections of the encoder itself and the IoT device. Specifically, we consider the variation in the encoder output voltage and variations in the reported battery voltage by the retrofitted IoT device due to different ADC sampling resolutions. Due to component variations, we expect the encoder output voltage to be slightly different across different boards. In Figure 15(a), we show the programmed output voltage of the encoder for three different boards as we perform a full voltage sweep. Though none of the encoder outputs violates the linearity of the transfer curve, encoders two and three have larger shift in between their transfer curve than encoder one and three.

Next, we quantify the variation in the battery voltage readings sampled by the LoRaWAN IoT device [29] with different ADC resolution. Different MCUs in the device can come with different resolutions among which 8, 10, and 12-bit are well-supported by most devices. In Figure 15(b), we see that higher ADC resolution allows us to achieve higher encoding resolution, while lower 8-bit resolution compromises the necessary voltage levels. The error distribution with different ADC resolution in Figure 15(c) and Figure 15(d) show that the maximum error can in fact be reduced by more than two times with 12-bit resolution.

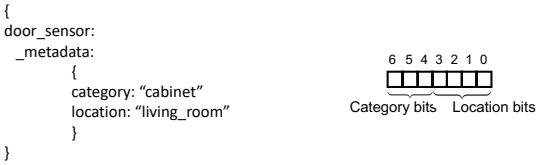
9.4 Real World Applications

We augment two COTS door event sensors with temperature sensing functionality and deployment specific metadata, one COTS BLE sensor with a timestamp metadata, and one soil monitoring sensor with a light energy-harvesting power supply. We investigate how



(a) (b)

Figure 17: Results from the door sensor metadata application. We can successfully decode all 12 unique metadata after error mitigation.



(a) (b)

Figure 18: Example of door sensor tag metadata in a). Figure b) shows how the 7-bit digital symbol can encode location and category information.

accurately the encoded sensor data can be retrieved and report our findings in this section.

Temperature Monitoring. For this application we attach a TMP37 temperature sensor to the analog encoder board and evaluate the sensor input voltage at normal operations. We measure the sensor output voltage as 0.55 V (equivalent of 27.5 °C) and measured the encoder regulator voltage output as 3.237 V, while the cloud application indicated a battery voltage of 3.246 V. Heating the temperature sensor raised its output voltage by about 1 V (equivalent of 50 °C), decreasing the encoder regulator voltage output to 3.1965 V, while the cloud application reported 3.198 V. This experiment demonstrated that the analog encoder regulator is capable of translating the temperature sensor readings into detectable alarm events at the cloud application with the threshold temperature being around 30 °C.

Digital Metadata Transmitting. We enhance the door sensor device with a simple digital metadata tag that informs what type of event it reports and where it is deployed. Figure 12 shows two installations. The digital metadata encoded in the battery voltage is unique for each sensor in a deployment area. Figure 18 shows an example of how the information can be encoded in a symbol. We categorize the sensor into four types based on the equipment it is monitoring: door, fridge, cabinet, and drawer and assign three location string for each of them based on which room they are located. These types of tags are useful for in smart home monitoring applications where the number of deployed sensors are only a handful. In total, we encode 12 unique symbols each representing different installed sensors.

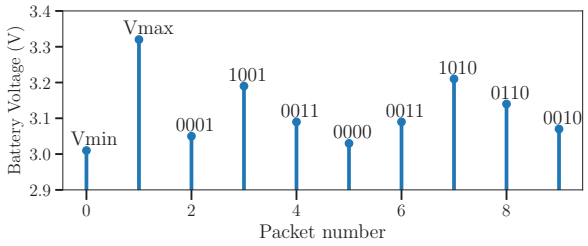


Figure 19: Detected voltage levels of the multi-symbol metadata transmission. Two reserved symbols equivalent of the maximum and minimum encoded voltages are used to mitigate decoding errors and as a flag for message start and end. The following encoded voltages represent the 32-bit unix timestamp for "2022-03-22 16:22:49".

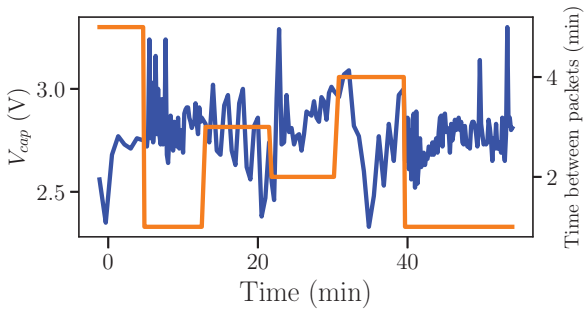


Figure 20: The dynamic sampling rate controlled by the energy-harvesting power supply with changing capacitor voltage over time.

We observe from Figure 17(b) that without proper error mitigation some of the metadata are not decoded correctly due to the high channel error shown in Figure 17(a). But after calibration, we could decode the metadata correctly for all of the samples. We could achieve this accuracy as the symbols are spread enough over the encoded voltage range. Though we can only encode 128 unique metadata tags, one can overcome the data bandwidth limitation by chaining multiple symbols together as we show in our BLE sensor experiment.

Multi-Symbol Metadata Transmitting. We upgrade a commercial BLE temperature and humidity sensor with an encoded 32-bit metadata message representing a unix timestamp. To transmit this message, we use ten symbols which are updated every hour. The first two symbols of the message are the reserved maximum and minimum voltages provided by the encoder regulator, used for decoding error mitigation as explained in Section 5.3. The following eight encoded voltages are 4-bit symbols representing subsections of the 32-bit unix timestamp. We show in Figure 19 the sequence of battery voltage readings for the timestamp corresponding to "2022-03-22 16:22:49".

Replacing Battery with Energy-harvesting. As a demonstration of how the battery voltage channel can be leveraged to convert a battery-powered sensor to an energy-harvesting one, we replace the battery of the soil monitoring sensor and plug in our

energy-harvesting power supply. We implement Algorithm 1 in the power supply board and report the optimized rate to the cloud in the battery voltage reading. The sensor period is updated every ten minutes. Figure 20 shows the instantaneous capacitor voltage and calculated sensor period.

9.5 Energy Overhead

As the encoder and power supply design result in an energy overhead, we evaluate the overall power draw of each module.

The Analog Voltage Encoder Module. We measured a total standby current of 190 µA for our analog module prototype without any sensor load. Adding the temperature sensor TMP37 resulted in a total current consumption of 212 µA.

The Digital Voltage Encoder Module. We measured a total standby current of 518 µA for our digital module prototype without any device connected to the I2C interface.

Energy Harvesting Retrofit. The retrofit module consisting of a power management IC and a low power microcontroller has a quiescent current of about 95 µA. The digital encoder module together with the energy harvesting retrofit consumes a total of about 613 µA.

Prototype Limitations. Our proof of concept design goal is to demonstrate how the control of the battery terminals voltage can be used as a new data channel, focusing on data encoding and recovery steps. Our prototype is not optimized to achieve minimal power consumption, and as such its standby power consumption can be too high for some battery-powered applications. Achieving lower standby current is possible by replacing the low-dropout regulator with a more efficient voltage converter and disabling unnecessary circuits while the sensor is in sleep mode. When retrofitting an IoT device to use energy-harvesting, the harvester should be selected with proper consideration of the energy consumption overhead.

10 RELATED WORK

Commercial IoT systems have been retrofitting legacy systems in condition monitoring, predictive maintenance, transparency in supply chain, etc. [4, 7, 12, 27] over the past decades. In this section, we briefly discuss work related to enhancing existing systems or interfaces with new techniques.

One of the possible ways of retrofitting the IoT network is adding sensing capabilities by attaching extra sensors or tweaking the radio. Penichet et al. presents passive sensor tags [21], where the IoT network can be augmented with a new sensor by placing a passive back-scatter sensor tag with the desired capability next to the already deployed devices using the IEEE 802.15.4 protocol. Since the proposed method lacks the media access control capabilities, it only demonstrates the prototype in low-density networks. LoRaBee [26] is presented as a LoRa to ZigBee cross-technology communication approach, which leverages the energy emission in the sub-1 GHz bands as the carrier to deliver information. LoRaBee tunes the LoRa's central carrier frequency and packet payload, such that a ZigBee device can decode the information carried by LoRa by sampling the RSS, and their result have shown a throughput of up to 281.61bps from LoRa to ZigBee. RetroFab was introduced to provide an end-to-end design and fabrication environment to retrofit the hardware interface of legacy devices [22].

The idea of augmenting versatile user interfaces of ubiquitous mobile devices have been explored in prior works. Kuo et al. designed HiJack [11] that exploits the exposed audio ports of mobile phone to encode additional data as well as harvest energy for operation. Nirjon et al. presented MusicalHeart [20] a wearable hardware platform to monitor the heart rate and activity level of the user which communicates the sensed data to the user mobile device using the audio jack of earphone. In our work, we focus on exploiting the battery voltage channel of smart IoT devices not only with the goal of adding sensor data, but also to eventually make the original device energy-harvesting and perpetual.

Another direction of retrofitting existing networks is to replace existing gateway with an generic gateway, whereas the devices itself remain unchanged, but the gateway would intercept their data stream at the next hop, and adding new sensors on generic gateways can add new sensing capabilities to the network. iGateLink introduces a pluggable design to allow data from different module sources that can be easily reused on edge without sending everything to the cloud [15]. This also speeds up the development of gateway applications. In real deployments, these approaches are not ideal for existing commercial IoT systems, due to a few reasons. The cost and complexity of recreating each layer is high. It is likely that existing devices cannot be changed since it requires specific software. Also, existing IoT platforms might be very rigid in the devices and the type of devices they support. Deploying an entirely new embedded-gateway-cloud system is another option. However, this approach is costly and does not leverage legacy systems, thus are not favorable for the end users.

11 DISCUSSION

Our prototype demonstrates the feasibility of augmenting existing IoT deployments with new data streams, and here we discuss some limitations, remaining challenges, and potential mitigations.

Increased Power Draw. Adding a controllable power supply and new sensors to an IoT device increases its overall power draw, and if the retrofitted device retains its original battery capacity, the IoT device will require more frequent maintenance interventions to replace batteries. To mitigate this maintenance overhead, the designer can adopt larger battery capacity in the retrofit power supply module or adopt an energy-harvesting solution compatible with the retrofitted IoT device energy requirements.

Data Channel Bandwidth. Our retrofit approach is constrained by the battery level reporting choices made by IoT device manufacturers, restricting the maximum achievable data bandwidth for a given application. For example, the Decentlab's soil humidity LoRaWAN sensor [5] reports its battery voltage with every uplink data message as a four-digit integer value with millivolt resolution (typically 2100 to 3300 mV), while the Seed Studio's LoRaWAN CO2 sensor [25] reports its percentage battery level after every 10 uplink data messages as a three-digit integer value (0 to 100%). While only very low throughput might be achievable under some IoT platforms, it can still be of great value to applications, for instance to enable alarm features or to support IoT deployment management by encoding a batch number or expiration date over multiple transmission as we demonstrated in Section 9.4.

Hardware Heterogeneity. Different hardware platforms may have different acceptable voltage ranges and resolutions for their battery voltage monitors. This essentially alters the data channel for the retrofit device. To accommodate this, a programmable range selector can be added to change the voltage output range. Also, using fewer voltage values could help with resilience at the expense of datarate.

Cloud API Access. We rely on the cloud API to retrieve the encoded battery voltage. For some signals, like the on-off of a button, this is likely readily available. But the battery voltage readings, may not be exposed through an API, either only used locally by the application provider or exposed only through a "battery low" alert. This limits the channels that can be used for this approach, or requires further consideration of the cloud-provided API when considering how the data to communicate is encoded. For example, a battery low alert could still be used as a low data rate channel.

Lossy Channels. The retrofit data channel may be constructed on top of a lossy underlying channel, and therefore data symbols can be lost. If the receiver is expecting to use multiple symbols to decode a packet, the protocol must handle the potential lossiness. Many standard data communication techniques could be used, including checksums and packet headers with length values.

Retrofit Synchronization. To synchronize the voltage encoder with the unmodified sensor we detect its sampling interval and only output new voltage readings before we expect the sensor to take its next reading. However, if the sensor is event-based, it may not follow a regular pattern when sending battery voltage state. This would hinder the ability to send packets of data without missing or duplicating symbols. One workaround is updating the voltage output only after a detected current spike, however, this would lead to an unpredictable datarate and perhaps stale data if events are infrequent. Some sensors both detect events and have a periodic transmission (such as a heartbeat packet), and a future version of this work could attempt to identify the regularly spaced packets and only transmit using those.

Another challenge related to our synchronization approach is that sensor devices also increase their power draw during receive mode, what could be falsely identified as a triggering event. However current peaks tend to be significantly lower for receiving modes, so the retrofit module controller can learn the IoT operation pattern and only use the highest current peaks as trigger events.

Another potential opportunity is the coupling between the energy harvesting rate of the devices in Section 6 and the datarate of the channel. More favorable harvesting conditions could lead to a better performing channel as the sensor is able to transmit more often. This increased performance may enable a secondary use of the channel and change how the energy-harvesting optimization algorithm works.

Temperature Variation Effects. Since outdoor sensor deployments can be exposed to a wide range of temperatures, more investigation is needed to understand what impact it can have on the encoder regulator retrofit. For instance, the manufacturer of the TPS784 voltage regulator indicates that the regulator output voltage accuracy varies by around 0.25 % in its recommended operation range from -55 °C to 125 °C for a 3.3 V output and 1 mA current. While the error mitigation approach presented in Section 5 is helpful to deal with voltage offset issues, fast temperature variations might result in reduced maximum achievable bandwidth.

Attack Potential. The ability to send data through the battery voltage channel, and that many devices are designed with user serviceable batteries, suggests that a possible attack vector is surreptitiously replacing the battery in the target IoT device with a "smart battery" that is controlling its own voltage output to exfiltrate data without any visual signs of tampering. The attacker would still need to be able to access the data once it is sent to the cloud, but the end-to-end attack may be feasible in conjunction with another vulnerability. Further analysis is required to understand the extent of this possible issue and future safeguards.

12 CONCLUSION

As IoT deployments grow larger in scale, designs and techniques that build on the existing device and network infrastructures can unlock many new applications and capabilities. Such design technique can not only enhance the functionality of existing systems, but also can significantly reduce the design time and developer overhead. We introduce one such technique that encodes information in the battery voltage enabling end-to-end communication, which otherwise just provides insight-less battery voltage information. We envision that this can lead to future explorations of other interesting underused channels in IoT deployments. Further, providing open and configurable channels can increase the solution flexibility and usefulness of new IoT devices and infrastructure. Open analog and digital ports and cloud API support to retrieve acquired data enable future users to customize IoT platforms for their own need at reduced cost and design effort.

13 ACKNOWLEDGEMENTS

We thank the anonymous reviewers and our shepherd for their valuable insights and feedback on improving this paper. This work was supported by the University of Virginia Strategic Investment Fund under grant SIF128, and the National Science Foundation under awards CBET-1735587 and CNS-2144940.

REFERENCES

[1] Mikhail Afanasov, Naveed Anwar Bhatti, Dennis Campagna, Giacomo Caslini, Fabio Massimo Centonze, Koustabh Dolui, Andrea Maioli, Erica Barone, Muhammad Hamad Alizai, Junaid Haroon Siddiqui, et al. 2020. Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 368–381.
[2] Analog Devices. 2015. TMP37. https://www.analog.com/en/products/tmp37.html.
[3] Awair. 2021. Awair Glow C . https://www.getawair.com/.
[4] Bosch-Connectivity. 2020. 3 Examples of How to Retrofit IoT Sensor Devices. https://www.bosch-connectivity.com/newsroom/blog/3-examples-of-how-to-retrofit-iot-sensor-devices/.
[5] Decentlab. 2018. Soil moisture and temperature profile sensor. https://cdn.decentlab.com/download/datasheets/Decentlab-DL-SMTP-datasheet.pdf.
[6] Dragino. 2021. LoraWan Door Sensor . https://www.dragino.com/downloads/downloads/LoRa_End_Node/LDS01/Datasheet_LDS01_Door_Sensor.pdf.
[7] Bruno V Guerreiro, Romulo G Lins, Jianing Sun, and Robert Schmitt. 2018. Definition of Smart Retrofitting: First steps for a company to deploy aspects of Industry 4.0. In *Advances in Manufacturing*. Springer, 161–170.
[8] Josiah Hester, Lanny Sitanayah, and Jacob Sorber. 2015. Tragedy of the coulombs: Federating energy storage for tiny, intermittently-powered sensors. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 5–16.
[9] Josiah Hester and Jacob Sorber. 2017. The Future of Sensing is Batteryless, Intermittent, and Awesome.

[10] HT1 Temperature and Humidity Sensor. [n.d.]. SensorPush. https://www.sensorpush.com/products/p/ht1.
[11] Ye-Sheng Kuo, Sonal Verma, Thomas Schmid, and Prabal Dutta. 2010. Hijacking power and bandwidth from the mobile phone's audio interface. In *Proceedings of the First ACM Symposium on Computing for Development*. 1–10.
[12] Theo Lins and Ricardo Augusto Rabelo Oliveira. 2020. Cyber-physical production systems retrofitting in context of industry 4.0. *Computers & industrial engineering* 139 (2020), 106193.
[13] Brandon Lucia, Vignesh Balaji, Alexei Colin, Kiwan Maeng, and Emily Ruppel. 2017. Intermittent computing: Challenges and opportunities. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[14] Xinyu Ma, Sebastian Bader, and Bengt Oelmann. 2020. Estimating Harvestable Energy in Time-Varying Indoor Light Conditions. In *Proceedings of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*. 71–76.
[15] Riccardo Mancini, Shreshth Tuli, Tommaso Cucinotta, and Rajkumar Buyya. 2021. iGateLink: A Gateway Library for Linking IoT, Edge, Fog, and Cloud Computing Environments. In *Intelligent and Cloud Computing*. Springer, 11–19.
[16] Maxim Integrated. 2008. DS4432. https://datasheets.maximintegrated.com/en/ds/DS4432.pdf.
[17] Maxim Integrated. 2020. MAX17222. https://datasheets.maximintegrated.com/en/ds/MAX17220-MAX17225.pdf.
[18] Maxim Integrated. 2020. MAX9634. https://datasheets.maximintegrated.com/en/ds/MAX9634.pdf.
[19] Monolithic Power. 2018. MPQ28164. https://www.monolithicpower.com/en/mpq28164.html.
[20] Shahriar Nirjon, Robert F Dickerson, Qiang Li, Philip Asare, John A Stankovic, Dezhi Hong, Ben Zhang, Xiaofan Jiang, Guobin Shen, and Feng Zhao. 2012. Musicalheart: A hearty way of listening to music. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. 43–56.
[21] Carlos Pérez-Penichet, Frederik Hermans, Ambuj Varshney, and Thiemo Voigt. 2016. Augmenting IoT networks with backscatter-enabled passive sensor tags. In *Proceedings of the 3rd Workshop on Hot Topics in Wireless*. 23–27.
[22] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. 2016. Retrofab: A design tool for retrofitting physical interfaces using actuators, sensors and 3d printing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 409–419.
[23] Nurani Saoda and Bradford Campbell. 2019. No batteries needed: Providing physical context with energy-harvesting beacons. In *Proceedings of the 7th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*. 15–21.
[24] Nurani Saoda, Wenpeng Wang, Md Fazlay Rabbi Masum Billah, and Bradford Campbell. 2022. An Energy Supervisor Architecture for Energy-Harvesting Applications. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 323–336.
[25] Seeed Studio. 2018. SenseCAP Wireless CO2 Sensor. https://www.seeedstudio.com/LoRaWAN-CO2-Sensor-US915-p-4308.html.
[26] Junyang Shi, Di Mu, and Mo Sha. 2019. Lorabee: Cross-technology communication from lora to zigbee via payload encoding. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.
[27] W. David Stephenson. 2019. Strategic Retrofitting: Older Machines Get an IIoT Update. https://www.industryweek.com/technology-and-iiot/article/21125857/strategic-retrofitting-older-machines-get-an-iiot-update.
[28] STMicroelectronics. 2018. SPV1050. https://www.st.com/en/power-management/spv1050.html.
[29] STMicroelectronics. 2019. LoraWan Discovery Kit . https://www.st.com/resource/en/data_brief/b-l072z-lrwan1.pdf.
[30] STMicroelectronics. 2019. STM32L010R8. https://www.st.com/resource/en/datasheet/stm32l010r8.pdf.
[31] TE Connectivity. 2019. MS5837-30BA . https://www.te.com/.
[32] Texas Instruments. 2020. LP2980. https://www.ti.com/product/LP2980-N.
[33] Texas Instruments. 2020. LP358. https://www.ti.com/product/LP358.
[34] Texas Instruments. 2020. TPS78401. https://www.ti.com/store/ti/en/p/product/?p=TPS78401DBVR.
[35] The Things Network. [n.d.]. https://www.thethingsnetwork.org/.