International Journal of Applied Technology & Leadership
ISSN 2720-5215
Volume 3, Issue 1, January 2024
ijatl@org

Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites

Tom Gallagher

Capitol Technology University (USA)

Abstract

Continuing the topics posed in "Understanding How System-on-a-Chip Data can Leak over Radio Transmissions", this article extends the exploration of data leakage of system-on-a-chip transceivers to real world small satellites. With the emergence of Low Earth Orbit communications networks, small satellites are poised to become an integral part of the world's communication fabric. The 2018 Screaming Channels research raised cybersecurity concerns about mixed-signal electronic devices including the transceivers used in small satellites. This article will propose a process for identifying data leaks in the broadcast of affected devices and then apply this framework to four target devices to demonstrate effectiveness. The results show how the proposed process identifies the known leak of a device from the 2018 Bluetooth research and potential concerns in additional devices associated with small satellite communications.

1. Introduction

As small satellites are becoming an important part of the world's communication networks, their security becomes increasingly important. In 2018, Giovanni Camurati published *Screaming Channels*, a new side channel attack vector for mixed-signal system-on-a-chip (SoC) circuits. Camurati's work showed how Bluetooth SoCs are vulnerable to this hardware weakness and the risk to encrypted communications. Part 1 of this article series, *Understanding How System-on-a-Chip Data can Leak over Radio Transmissions*, postulated that SoCs used in small satellite transceivers could be vulnerable to the same type of attack identified in *Screaming Channels*. This article will demonstrate the validity of the hypothesis presented in Part 1 and describe how *Screaming Channels* leaks can be identified in small satellite

transceivers. Specifically, a process for identifying leaks will be described followed by the results of applying that process to four different SoC transceivers. Part 3, the last article of this series, will present a framework for measuring the impact of identified leaks against cryptographic processes and propose countermeasures.

2. Background

This article assumes that the reader understands the background put forward in *Understanding How System-on-a-Chip Data can Leak over Radio Transmissions*. Concepts such as mixed-signal circuits, system-on-a-chip architecture, and the *Screaming Channels* phenomenon are described thoroughly in that article. Other than refreshing the core concept, this article will build on the established foundation.

2.1. Revisiting Screaming Channels

The *Screaming Channels* phenomenon is an unintentional modulation of a transmission from a mixed-signal circuit. Mixed signal circuits have digital components that make calculations and analog components that transmit/receive wireless signals. The unintended modulation is caused by digital components power consumption affecting the voltage of the shared source. Whenever digital components change state (e.g., unpowered to powered), the components cause a small drop or spike in voltage potentially impactful to other components sharing the power source. As analog components are highly susceptible to voltage changes, these variations cause measurable impacts on wireless broadcasts made by these components. In particular, the amplitude of the broadcast fluctuates with the voltage changes. Understanding this interaction and the cybersecurity implications is not the focus of this article but can be found in Part 1 of this series. Instead, this article will explain how a researcher can find *Screaming Channels* leaks in transceivers used by small satellites.

2.2. Legality

Before going further into the methodology used to identify *Screaming Channels* leaks, it is imperative to understand the legal and public safety ramifications of wireless signal research.

This paper does not constitute a legal opinion. No one with a legal background reviewed this content; it is a technical researcher's interpretation of United States regulations to be compliant and avoid disrupting authorized communications. Individuals performing this type of analysis should understand applicable regulations or seek legal guidance.

The United States Code of Federal Regulation Title 47, enforced by the Federal Communications Commission (FCC), prohibits unauthorized broadcasts in the US and territorial waters (FCC, 2019). This requires radio operators to be licensed and use certified hardware. However, Part 15 of Title 47 provides exemptions for the unlicensed operation of radio transmitters under certain circumstances. These exemptions permit low powered

transmitters with some limitations if they do not interfere with authorized broadcasts. Baby monitors and cordless phones fall into this category of low power device (FCC, 2017). Part 15 compliant devices must have an FCC ID etched into the transmitter indicating the hardware's certification. Exempted devices must avoid reserved frequencies, such as navigation signals and rescue communications. Additionally, broadcast power is limited based on frequency ranges (US Government, 2023, 47 CFR §15.209).

For US residents, abiding by the FCC regulations is an important consideration for this research. Not only does FCC levy extensive fines for violations, but careless broadcasting could interfere with sensitive communications and impact public safety. To ensure compliance, researchers should only test against devices with FCC certifications (i.e., FCC IDs) and limit power output to permissible levels. For example, FCC regulations impose a maximum power output of 200 μV/m at 3 meters for broadcasts in the 806-960 MHz range (FCC, 2023). While possible to measure the power output for compliance, doing so requires the device to be transmitting, potentially unlawfully. Instead, researchers can first calculate the expected theoretical output from a device based on the device specifications. For this purpose, Effective Isotropic Radiated Power (EIRP) is the total energy that can be radiated from an antenna under ideal circumstances (Bensky, 2019, 11-41). Compliance is assumed if EIRP is below the legal threshold. EIRP is calculated according to the formula (Mazar, 2016, 83-84):

```
EIRP [dBm] = Field Strength [μV/m] + 20log(Distance[meters]) - 134.77
```

Thus, for a field strength of 200 μV/m and a distance of 3 meters, we calculate the EIRP as follows:

```
EIRP [dBm] = Field Strength [μV/m] + 20log(Distance[meters]) - 134.77
EIRP [dBm] = 200 [μV/m] + 20log(3 [meters]) - 134.77
EIRP ~= 74.77 [dBm]
```

By configuring the device to transmit below 74.77 dBm, researchers can ensure that broadcasts in the range of 806-960 MHz will be compliant.

Broadcasting in an authorized manner is not the only legal concern with wireless analysis. A researcher must also be authorized to use the device itself. The Computer Fraud and Abuse Act is permissive of ethical hacking for devices and data that are owned and controlled by the researcher (US Department of Justice, 2022). Extending testing beyond devices owned by the researcher or accessing third party data could violate that law. Additionally, researchers could be subject to device End User License Agreements (EULAs). EULAs can explicitly prohibit certain activities, such as reverse engineering, which would otherwise be lawful in the researcher’s location.

3. Detecting Leaks on Suspicious Frequencies

The primary purpose of this article is to aid cybersecurity researchers in identifying potential *Screaming Channels* leaks on a target device. The background information and step-by-step testing procedures provided here make it easier for inexperienced cybersecurity enthusiasts to

perform this type of analysis. More attention to the hardware and physical layers should bring about stronger overall device security.

3.1. Environment

When analyzing a device for a *Screaming Channels* leak, it is valuable to adjust hardware characteristics such as broadcast frequency and output power. Programming is prone to human error, and a typo in the firmware for either of these characteristics could violate regulations or endanger public safety. Thus, as an additional precaution, wireless signal research should be performed inside a radio frequency (RF) shielded enclosure. An RF enclosure provides assurance that test signals stay within regulatory limits and limits external electromagnetic (EM) "noise" that could interfere with analysis. Limiting external noise with a shielded environment can help to identify subtle leaks in mixed-signal SoCs. If a real-world environment is required, then testing outside the enclosure can be performed once the device is confirmed to be broadcasting as intended.

The recommended setup, shown in Figure 1, requires several hardware and software components. First, researchers must have a device to test and the software to flash (i.e., load) firmware onto the device. Some SoC devices have built-in hardware interfaces that allow flashing; others require separate hardware. Next, researchers must have a software-defined radio (SDR) receiver. The analysis in this article was performed using the HackRF One by Great Scott Gadgets. GNU Radio Companion (GRC) was used to visualize the SDR input. Researchers should use an RF enclosure to minimize the risk of violating broadcast regulations and reduce external interference. A Ramsey STE2900 provided RF shielding for the analysis performed in this series. Depending on the software requirements of the chosen hardware, researchers may need a Windows or Linux system for developing/flashing the firmware or interacting with the SDR peripheral device. Because this article considers multiple devices, a dual-boot Windows and Linux Hewlett Packard laptop provided both environments. The laptop hardware included an AMD Ryzen 5 5500U, 32GB DDR4, and 1TB SSD.

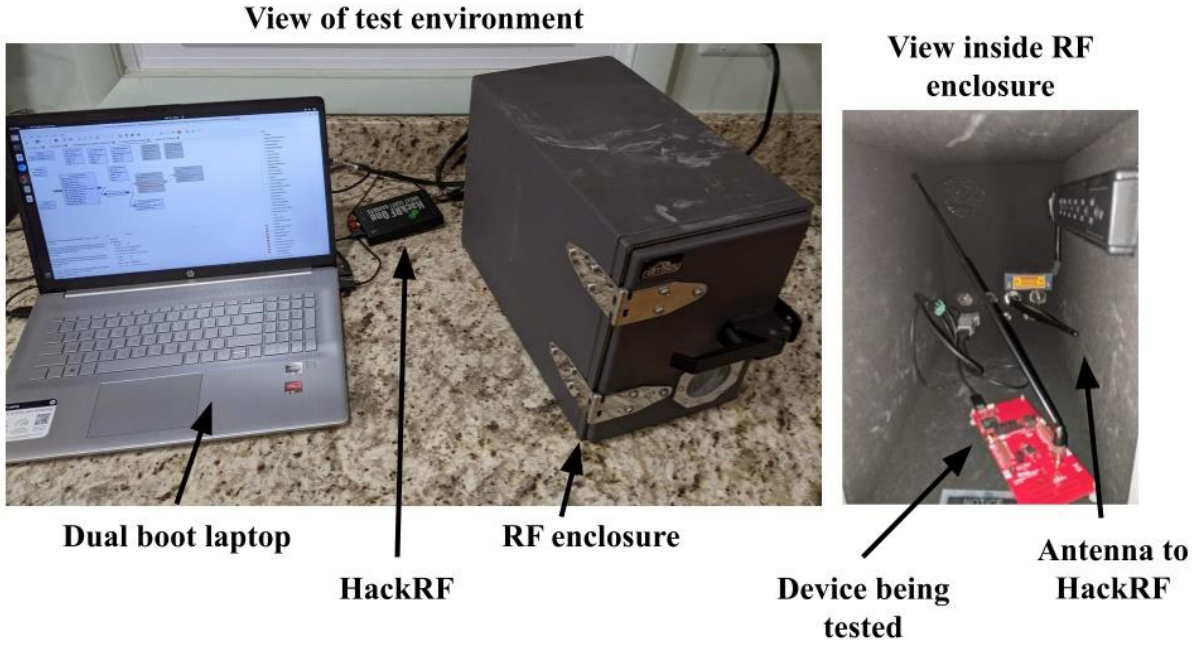

Figure 1: The environment used for this analysis.

3.2. Creating the Firmware

The methodology described here requires that firmware be created and executed on the transceiver. Programming firmware is challenging without experience and understanding of the device or a willingness to dig through reference documentation. An enthusiastic researcher without programming experience should not be deterred as the amount of firmware development is minor. Researchers pursuing this methodology should be familiar with basic firmware development and are legally permitted to perform the analysis.

The *Screaming Channels* phenomenon at its core is a hardware issue present regardless of the firmware. However, the leak may be more evident when the firmware exercises the hardware in certain ways. When testing for such weaknesses, it is useful to use firmware that makes the leak most evident and predictable. To show that the digital processor is causing unintended modulation, it is useful to know when the processor is active and inactive. Creating custom firmware provides researchers with this level of control and visibility.

While small satellite transceivers share many properties, these devices may have different capabilities. For instance, most transceivers provide a range of broadcast frequencies and transmit powers. Some provide encryption capabilities through hardware elements or software libraries. The sections below detail a minimal firmware implementation to identify potential *Screaming Channels* leaks. Pseudocode is used to illustrate program logic, but the real-world implementation will vary between device models. While software developer kits (SDK) simplify access to hardware, there is insufficient standardization for a single code solution to apply broadly. For this reason, developers will need to understand their device SDK to correctly implement firmware. A link to the source code used by each device analyzed for this article is provided in the *Contributions* section of this article.

3.3. Selecting Broadcast Frequency

While transceivers may offer a range of frequencies over which they can broadcast, researchers should not make this selection arbitrarily. Government organizations regulate transmissions depending on their frequency. For example, the frequencies 406 MHz and 121.5 MHz are reserved by the Federal Aviation Administration (FAA) for distress signals (FAA, 2022). Researchers should avoid transmitting at these and other reserved frequencies. Once the frequency is selected, the device should be configured to disable any features/protocols that automatically change the frequency, such as frequency hopping. Frequency hopping is a default Bluetooth feature that uses multiple frequencies for short intervals to enhance signal resilience. Researchers should understand the broadcast options provided by the SDK to ensure a static frequency is used.

3.4. Setting Broadcast Power

Like frequency selection, researchers should be cautious about regulations limiting broadcast power. Broadcast power at frequencies above 960 MHz is limited to 500 μV/m measured at 3 meters. Additionally, it is illegal to interfere with authorized signals, so researchers should avoid degrading their neighbor's Wi-Fi or other broadcasts (US Government, 2023, 47 CFR §333). *Screaming Channels* affect the amplitude of the susceptible transmissions, so fluctuations will be more evident when the amplitude is larger (i.e., when the broadcast power is greater). For this reason, researchers should use the maximum power output permissible.

3.5. Building the Application Logic

This article proposes a simple approach to detect a *Screaming Channels* leak with straightforward and intuitive application logic. The application needs to continuously transmit a "dummy" signal over the intended broadcast frequency and simultaneously exercise the processor to make the leak evident. As shown later, a simple way to accomplish this is alternating the processor between rest and execution at predefined intervals. For example, having the processor repeatedly sleep for two seconds and then perform calculations for two seconds would create a recognizable pattern in the broadcast if a leak exists. Depending on the available SDKs and capabilities of the device, it may be easier and just as effective to approximate these times using programming loops than integrating a timing mechanism into the application. If not using a real-world time source, researchers should trigger some external indicator, such as output across a serial connection or a device light emitting diode, to give evidence of approximate period timing.

Care must be chosen when selecting the processing operation as firmware compilation may optimize away simple operations, which could result in unexpected sleep periods. For example, if the firmware instructed the processor to execute the pseudocode `variable x = 1+1` one thousand times, there is a possibility that the compiler would recognize that x will always be 2 and remove the operation. While there are a number of ways to prevent optimization, the example below uses an encryption algorithm, which cannot be precalculated and removed by

the compiler. Furthermore, focusing the processor on encryption provides a means for measuring the leak's impact on the device's cryptography, which will be discussed in a subsequent article.

Below is pseudocode for the application logic discussed above:

```
main() {
      CypherText = "";
      initBoard(); //enable general device hardware
      initRFBoard(); //enables the Radio

      disableSpecialRFFeatures(); //such as Frequency hopping
      SetFrequency(900MHz);
      SetPower(11); //Power levels are vendor specific
      StartContinuousRadioTransmit(); //transmit a dummy signal
      while(1) { //loop this logic until device is powered off
            //sleep prevents disturbing the analog signal for 1 second
            sleep(2000); //2000ms = 2 second
            toggle_LED(); //toggle a device LED on or off
            start_time = time();
            while (start_time + 2000 < time()) {
                  rand = rand();
                  CypherText = Encrypt(rand, rand);
            }
            toggle_LED(); //toggle a device LED on or off
      }
}
```

3.6. Collecting the Data

Once the firmware is completed and loaded on the device, a researcher can use a number of methods to collect the signals. Use of a software-defined receiver allows the researcher to programmatically examine a range of frequencies. This is valuable since the *Screaming Channels* phenomenon could be visible at an unexpected frequency. HackRF is compatible with GNU Radio, an application that both configures the SDR receiver and processes the resultant signal.

Using caution to conform to regulations, the target device is placed in an RF enclosure. The HackRF is connected to the laptop and to a feedthrough connection on the enclosure that allows it to connect to an antenna inside. The laptop is running GNU Radio and GRC, which provides

a user interface to view received signals. GRC's time-frequency diagram, also known as a waterfall plot, provides a visual representation of the needed information. In spectrum analysis, the waterfall plot shows the intensity (i.e., amplitude) of each frequency in a range over time. As time passes, data moves upward in the diagram allowing newly captured data to appear at the bottom and the oldest information to trail off the top. The source code to generate a waterfall plot, as shown in Figure 2, is linked in the *Contributions* section of this article.

A key factor in configuring GNU Radio is selecting the frequencies to analyze. The device's configured broadcast frequency is a good place to start. However, as explained by Camurati in the original *Screaming Channels* research, SoC transceivers often transmit unintentionally over "nearby" frequencies (Camurati, 2018). For instance, as shown in Figure 2, the Texas Instruments CC1310 is configured to broadcast at 900 MHz, but the red vertical line indicates the presence of a strong signal at 909.6 MHz as well. SoCs are prone to broadcast unintentionally due to coupling between the analog carrier frequency and the digital clock. The coupling causes a third "beat signal" to be broadcast alongside the other signals. The beat signal frequency is given by the equation (Dvornikov et al., 2022, 30):

$$frequency_{beat} = |frequency_{broadcast} - frequency_{clock}|$$

Researchers should pay attention to beat frequencies while analyzing for leaking signals. While digital clock beat frequencies are one unintended signal, other device EM noise could interact unexpectedly and produce stray broadcasts. For this reason, researchers should iterate across wide ranges of spectrum analysis to identify all RF signals emanating from the device (Grimm & Rathmair, 2017, 1). Broadcast frequencies, including the unintentional ones, will display higher intensity colors (often orange/red) resulting in vertical lines in the waterfall plot.

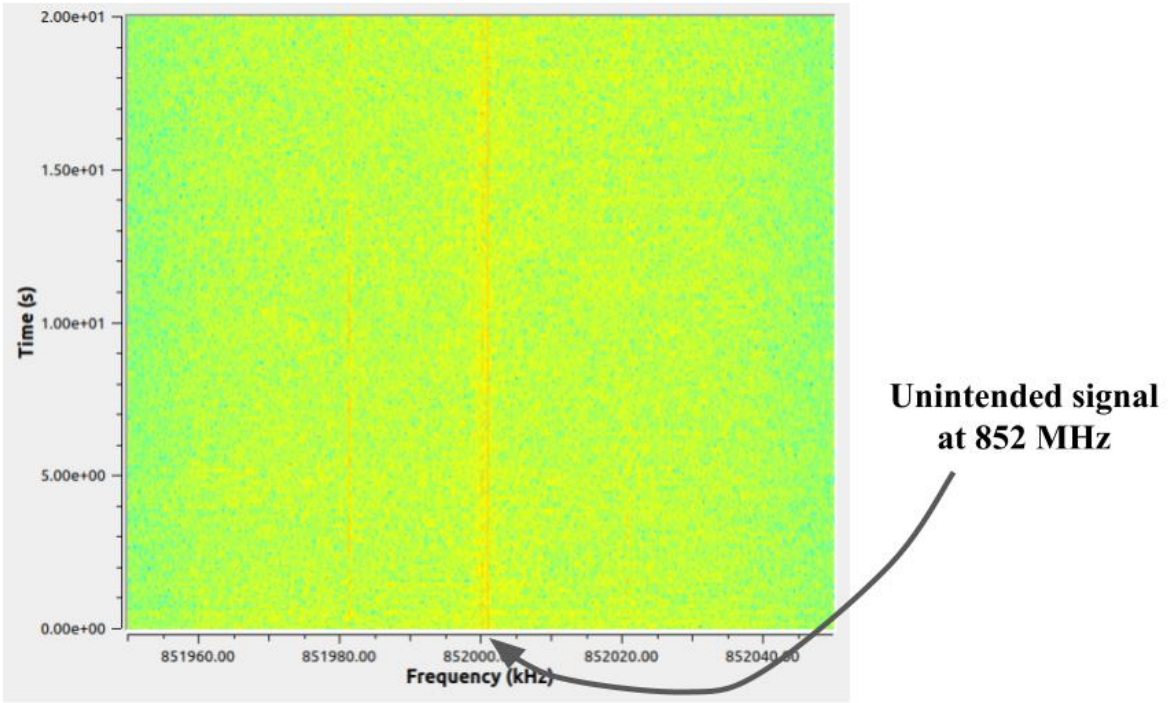

Figure 2: Waterfall plot of CC1310 device configured to broadcast at 900 MHz but shown here unintentionally broadcasting a beat signal at 852 MHz.

3.7. Interpreting the Data

The vertical lines on the waterfall plot identify broadcast signals, but the end goal is to determine which signals could be leaking data. Recall that a *Screaming Channels* leak affects the amplitude of the broadcast. Amplitude corresponds to color intensity of a waterfall plot; the higher the amplitude, the more intense the color. If the digital processor has no effect on a signal's amplitude, then the line's color will remain constant and continuous across the firmware's repeating sleep and encryption cycles. However, if a leak is present, then the sleep cycles will have a solid color and the active cycles will exhibit some color variation creating a dashed line pattern. This simple and intuitive analytic provides evidence that an analog signal is impacted by the digital processor, but it does not measure the extent of that leak or its impact on the device's cybersecurity. The next article in this series will expound in this area and present a framework for measuring a leak's impact with respect to cryptographic processing.

The original *Screaming Channels* research analyzed a Bluetooth SoC transceiver, the Nordic Semiconductor nRF52832. Figure 3 from that study shows how the waterfall plot visibly changes between the digital processor sleeping (represented by the "TX on" segment) and the digital processor actively encrypting (represented by the "TX on + AES" segment). The *Screaming Channels* figure does not use repeated cycles of sleep/encryption so dashed lines are not expected. However, the visual change is still dramatic.

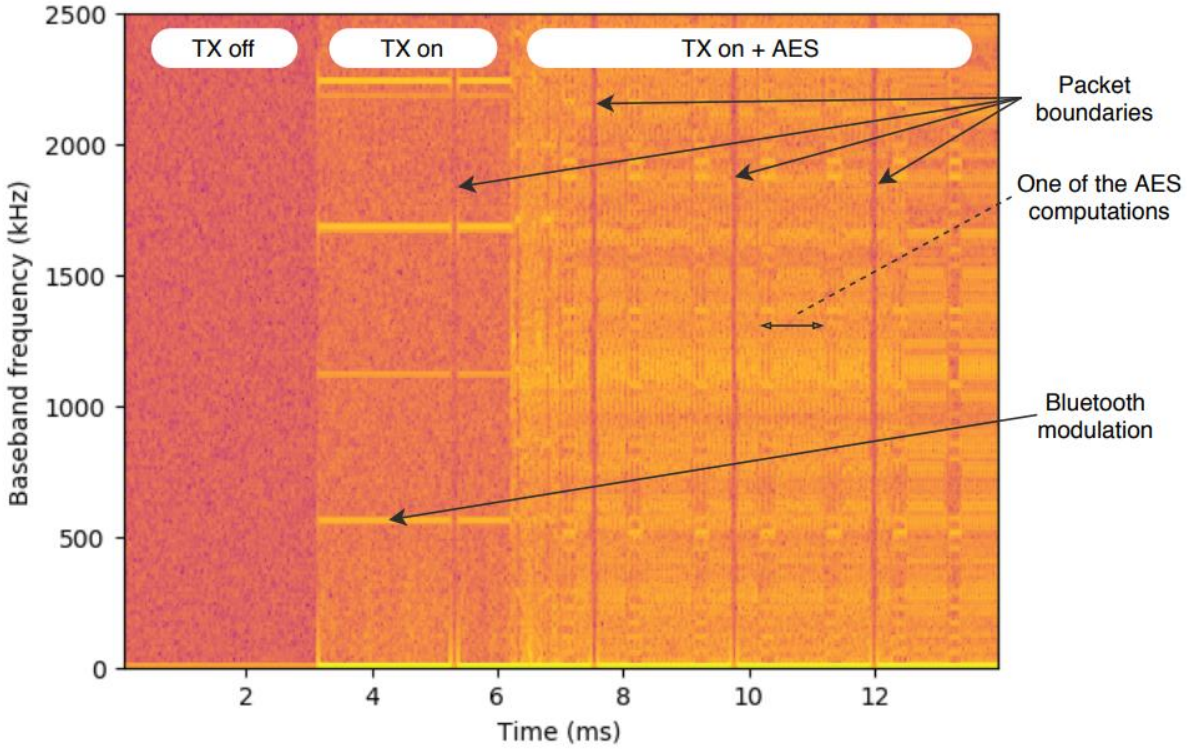

Figure 3: Waterfall plot of nRF52832 showing that digital processor is impacting the signal (Camurati, 2018).

4. Results

The analysis described above was performed on four separate SoC transceivers: Nordic Semiconductor nRF52832, Texas Instruments CC1111, Texas Instruments CC1310, and Seeed LoRa Wio-E5 Mini. The Nordic nRF52832 was analyzed during the 2018 Screaming Channels research and found to exhibit a leak, so it is included in the analysis here to illustrate the leak. The latter three devices are used by, or are similar to transceivers used by, operational small satellites as described in Understanding How System-on-a-Chip Data can Leak over Radio Transmissions. Three small satellites were selected for the study because their hardware specifications are readily available publicly. Hardware specifications for commercial small satellites, such as those used in communication constellations, were not found publicly. Future efforts could include industry partnerships to analyze commercial satellite transceivers. A link to the source code used in each device's analysis is provided in the Contributions section of this article.

4.1. Nordic nRF52832 (Bluetooth)

The Nordic nRF52832 was analyzed during the original *Screaming Channels* research and shown to exhibit a leak. The device is a short-range Bluetooth transceiver and is thus not feasible for small satellite communications. While Bluetooth transceivers are not the focus of this article, the device is included here to illustrate results for a known positive result. Understanding the visual for this known leak will help contextualize the results for the other three devices.

4.1.1. Device Configuration

The nRF52832 provides a max broadcast output of +4 dBm and a frequency range consistent with Bluetooth standards, 2.4 GHz to 2.4835 GHz. Because Bluetooth uses a frequency range shared by many protocols, there is a higher risk of interfering with authorized broadcasts, such as a neighbor's security cameras. For this reason, Bluetooth analysis should only be performed inside an RF enclosure. Confident that the broadcast would be attenuated by the enclosure, the nRF52832 was configured to broadcast at maximum power. Additionally, the device was configured to broadcast over a single frequency, 2.404 GHz, rather than using Bluetooth's frequency hopping mechanism. The firmware for the nRF52832 used in the 2018 research is available publicly (Camurati, 2018), so it was straightforward to repurpose for the analysis here. The resultant code and instructions to replicate the experiment is provided in the contributions section.

4.1.2. Observations

The 2018 *Screaming Channels* research illustrated the nRF52832 leak at 2.528GHz. Rather than focus exclusively on that frequency, the analytic process described in this article was applied from 2.3 GHz to 2.6 GHz to encompass the entire range of Bluetooth signals and beyond. As shown in Figure 4, the postulated dashed line pattern is clear on multiple frequencies. Each of the vertical (dashed) lines in the plot represents a frequency at which the device is broadcasting. The device is configured to broadcast continuously, so the vertical lines

should be a solid color. However, every two seconds, the digital components spend two seconds processing data, which draws sufficient voltage away from the analog components to result in a visible change in signal intensity. These results provide intuitive confirmation of the 2018 findings.

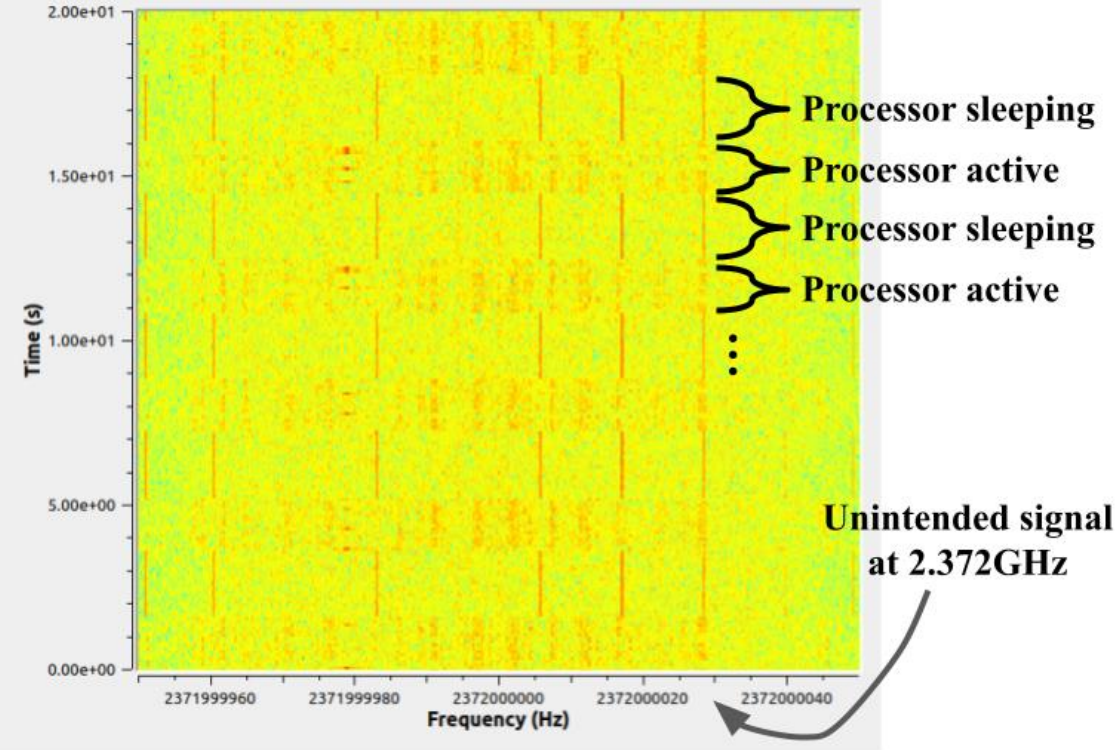

Figure 4: nRF52832 broadcasting at 2.404 GHz while fluctuating between periods of sleep and intense processing resulting in a dashed line pattern due to the leak.

4.2. Texas Instruments CC1111

The Tartan Artibeus is a “pocketqube” satellite the size of a fist. While the satellite is significantly larger than the two analyzed chip satellites, most of the additional bulk consists of solar panels and a large capacitor. The Tartan Artibeus, as described in its whitepaper and visible in Figure 5, uses a Texas Instruments CC1110 SoC transceiver for communications. While the CC1110 is a completely viable target for this analysis, information provided by the vendor shows that the CC1110 and CC1111 are nearly identical SoCs except that the CC1111 provides a USB interface for ease of use (Texas Instruments, 2019). The CC1110 and CC1111 both transmit in the ranges of 300-348 MHz, 391-464 MHz, and 782-928 MHz. Both SoCs also feature a configurable processor clock speed. For ease of testing over USB, the CC1111 was selected as a target for analysis over the CC1110. Findings are assumed to be similar for both devices.

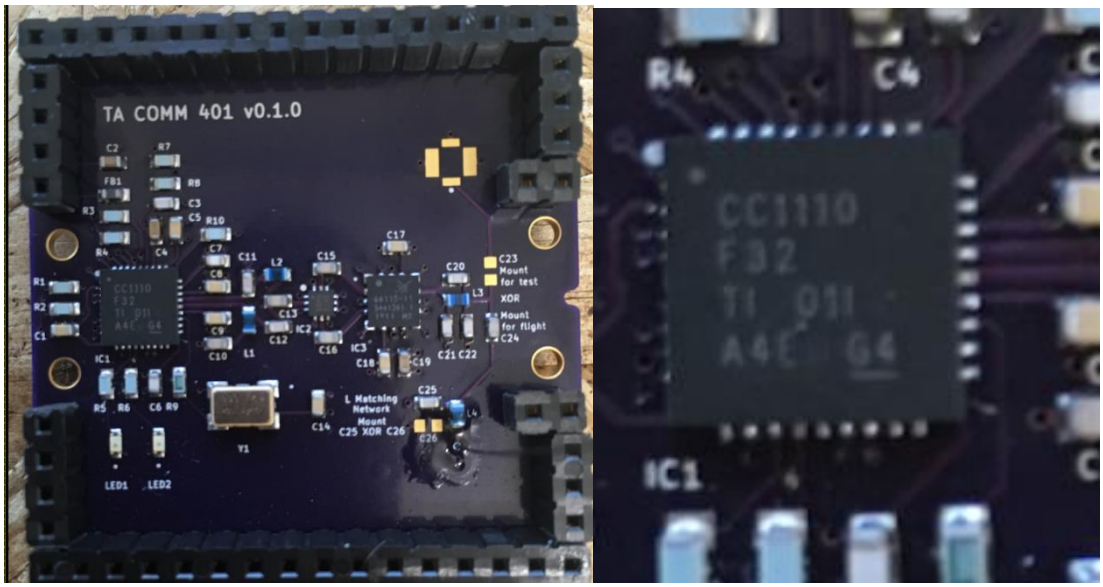

Figure 5: Image of Tartan Artibeus with enlarged image of transceiver chip.

4.2.1. Device Configuration

The CC1111 was configured to broadcast at a frequency of 868.3 MHz with the maximum power of +10 dBM, which is well under the allowed EIRP of 74.77 [dBm]. Unlike the other two development environments, the CC1111 recommended environment, IAR Embedded Workbench, imposed size restrictions without a purchased developer license. Because of those restrictions, the firmware was unable to leverage vendor SDKs making the transmitter configuration more complex. Flashing the firmware to the CC1111 requires a separate hardware device, the Texas Instruments CC Debugger.

4.2.2. Observations

The CC1111 defaults to a 24 MHz clock signal for its digital processor. Based on the previous discussion, the potential for clock and carrier frequency mixing emphasizes analysis of signals at 868 MHz ± (x * 24 MHz) where x is an integer. For completeness though, the range from 768 MHz to 968 MHz was analyzed. As shown in Figure 6, immediately surrounding the intended broadcast signal are dashed vertical lines. The solid parts of the dashed lines are each two seconds long; the blank part of the dashed lines represents each processing cycle of approximately 1.5 seconds. Because of their correlation with the processor state, the dashed lines show a clear *Screaming Channels* leak across the mixed-signal SoC. At this point though, the analysis only shows a leak is present; it is unclear the impact of the leak on the security of the device. The next article in this series will measure the impact of this leak on cryptography employed by the device.

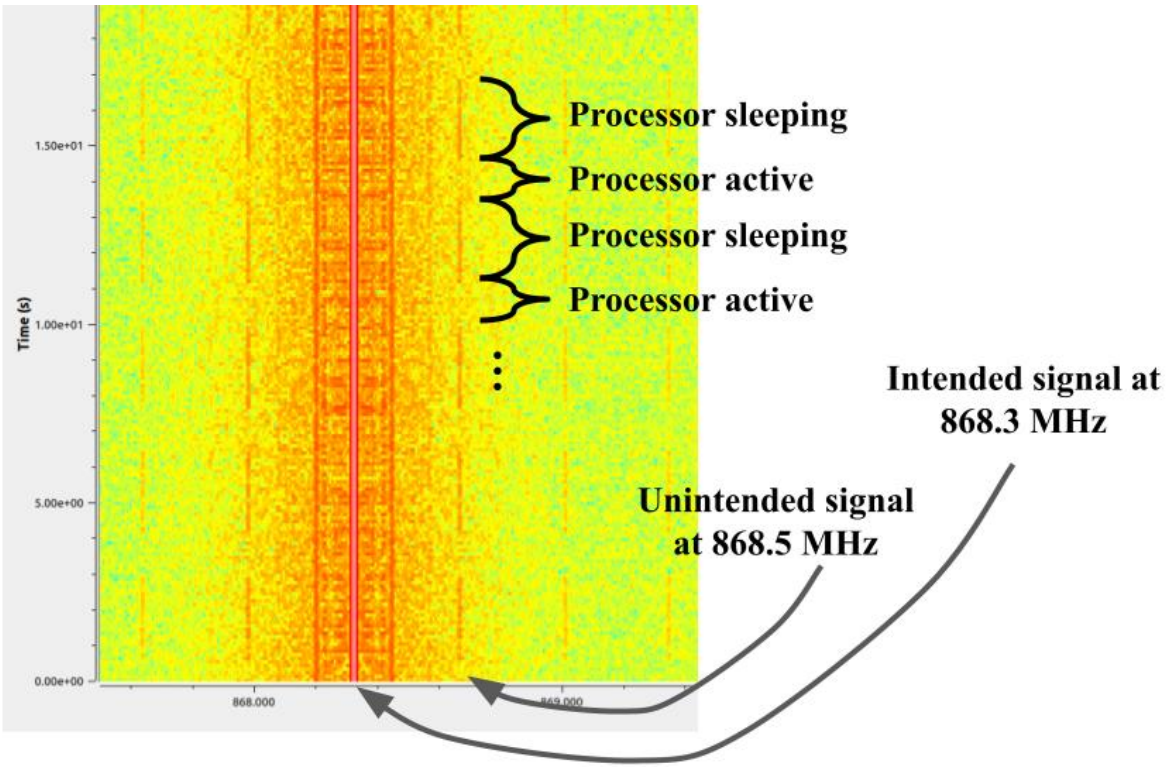

Figure 6: Dashed lines show leak of digital processor sleep/active periods on analog broadcast of CC1111 at 868 MHz.

4.3. Texas Instruments CC1310

A close inspection of the Monarch chip satellite image from Cornell's site, Figure 7, shows an integrated circuit labeled CC1310. CC1310 is a Texas Instruments SoC transceiver capable of broadcasting over the ranges of 287-351 MHz, 359-439 MHz, 431-527 MHz, 718-878 MHz, and 861-1054 MHz (Texas Instruments, 2018). The CC1310 is an attractive target for analysis because of the wide range of support and tooling provided by Texas Instruments. SmartRF studio provides functionality to run test signals on the device with no programming experience. Code Composer Studio provides a development environment with SDKs that abstract much of the complexity of firmware development. SmartRF Flash Programmer flashes the firmware to the device over a USB interface provided by the CC1310 development board.

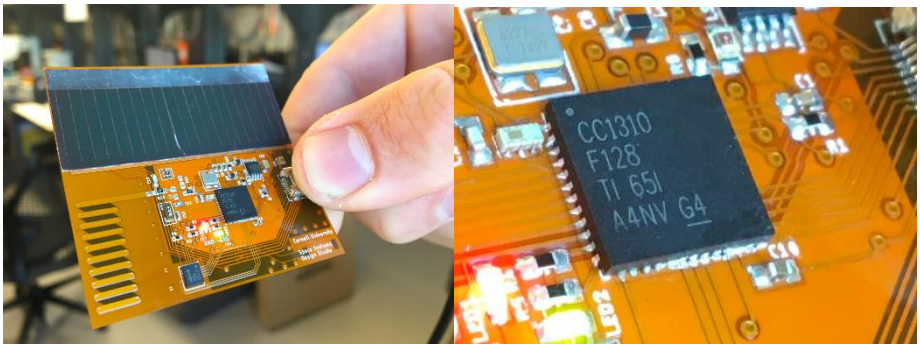

Figure 7: Image of Monarch satellite with enlarged image of transceiver chip

4.3.1. Device Configuration

The CC1310 was configured to broadcast at a frequency of 900 MHz, which is authorized for field strengths up to 200 µV/m and a distance of 3 meters or EIRP of 74.77 [dBm]. As there is no external amplification, this allows broadcasting at the device's maximum power of +15 dBm. If this analysis was not being performed inside an RF enclosure, then additional care would have been taken to ensure that the device was configured to broadcast at the correct frequency prior while using a lower power setting.

4.3.2. Observations

The CC1310 uses a 48 MHz clock for the digital processor (Texas Instruments, 2018). Based on the previous discussion, the potential for clock and carrier frequency mixing emphasizes analysis of signals at 900 MHz ± (x * 48 MHz) where x is an integer. For completeness though, the range from 800 MHz to 1000 MHz was analyzed. While no frequencies showed obvious dashed lines, there were a number of frequencies that showed some correlation with the encryption cycle. For example, at 848 MHz, there are clearly horizontal lines every four seconds. As shown in Figure 8, the horizontal lines represent an event that caused a noticeable amplitude change across a range of frequencies. In this case, the line corresponds to each time the processor went to sleep. While the finding is not as conclusive as the dashed lines for the CC1111, it shows a relationship between the digital processor and the analog transmission. The next article of this series will present a framework for measuring the impact of such potential leaks.

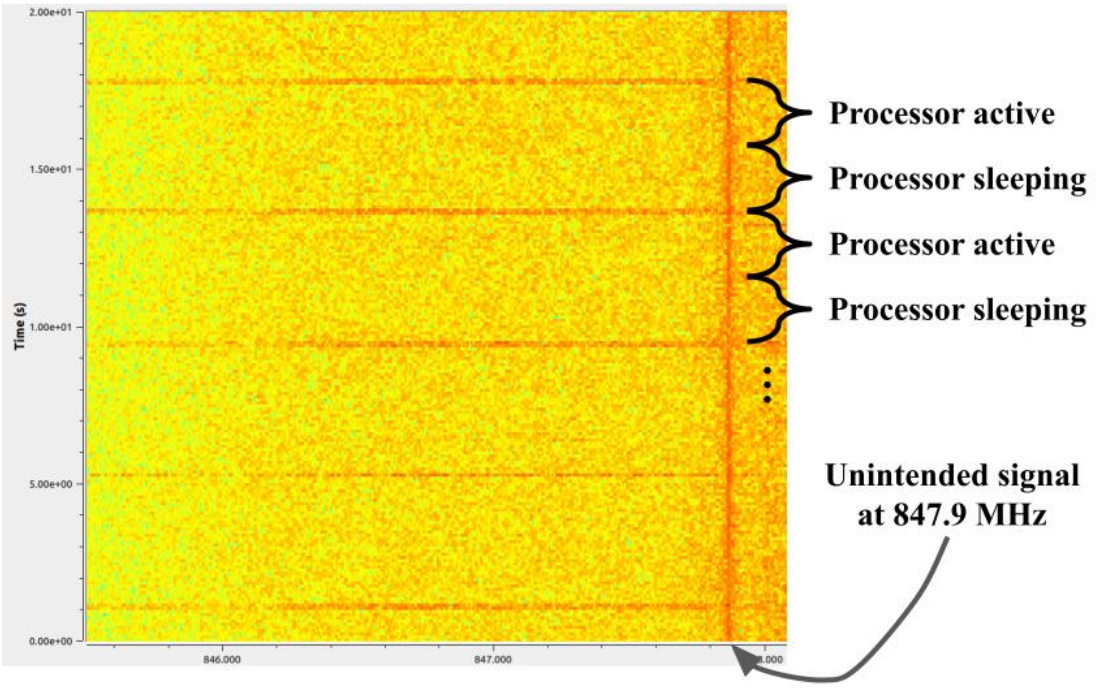

Figure 8: CC1310 broadcasting at 900 MHz while fluctuating between periods of sleep and intense processing resulting in an observable change (horizontal lines) when the processor sleeps.

4.4. Seeed LoRa Wio-E5

The AmbaSat project endeavors to provide hobbyists with the hardware and tooling needed to build a satellite. AmbaSat uses a Semtech LoRa RFM95 transceiver, Figure 9. The RFM95 operates exclusively at 868 MHz or 915 MHz with a maximum power output of +20 dBm. Unfortunately, the Semtech RFM95 is a transceiver module and not an SoC. In other words, the RFM95 does not have the capability to run custom firmware on a digital processor. Since the RFM95 is not compatible with the analysis techniques outlined in this project, it was necessary to find an SoC with similar specifications to the RFM95 that could be used in place of the RFM95. The Seeed Wio-E5 provides LoRa communications at frequencies 868 MHz and 915MHz with a maximum power output of +22dBm. The Wio-E5 uses an SoC architecture with a programmable digital processor. For the purposes of this research, the Wio-E5 provides a suitable target of analysis and could have been used by AmbaSat in lieu of the RFM95.

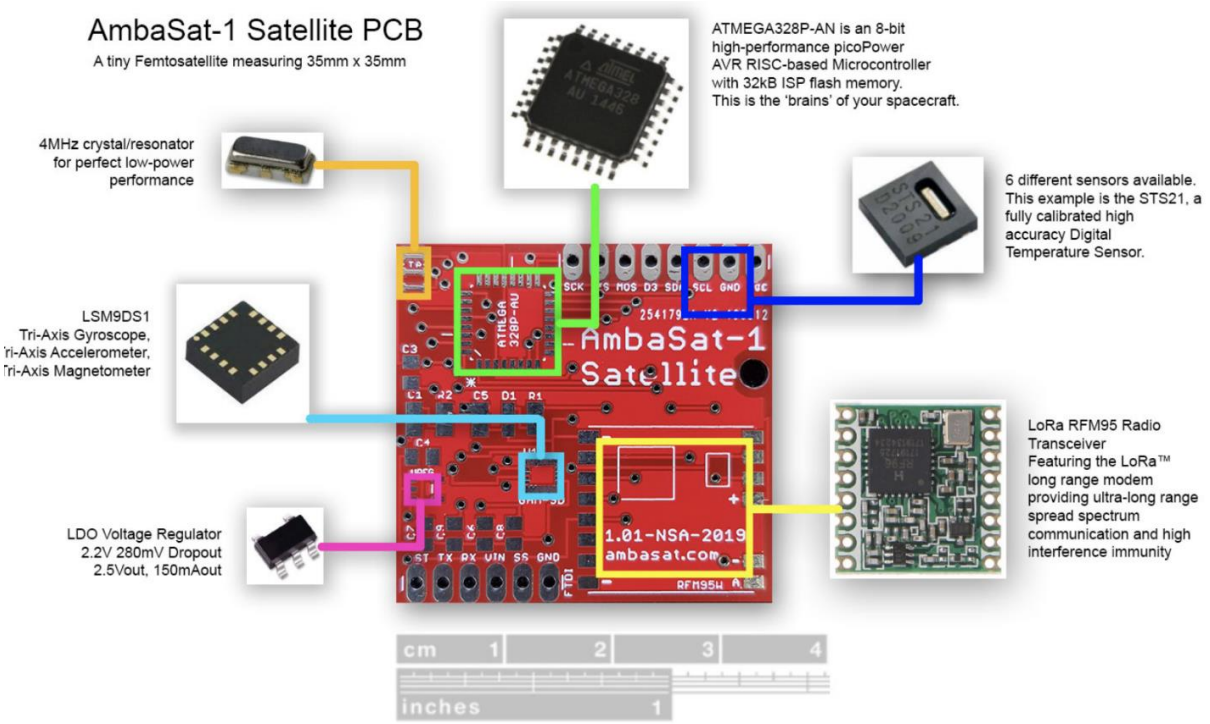

Figure 9: AmbaSat-1 chip satellite with component descriptions including the Semtech LoRA RFM95 transceiver (AmbaSat Ltd., 2022).

The vendor-provided development environment for the Wio-E5 seems geared towards using the LoRa communications stack, which is not ideal for this research due to the loss of frequency control. The STM32CubeIDE application provides an environment for developing firmware for a number of microcontrollers including the Wio-E5 and allows direct hardware interaction.

4.4.1. Device Configuration

The Seeed Wio-E5 was configured to broadcast at a frequency of 868 MHz with the maximum power of +22 dBm, which is well under the allowed EIRP of 74.77 [dBm]. Note that when configuring the Wio-E5 to broadcast a continuous test tone, it does not transmit on the

configured frequency. Instead, as discovered in the device reference manual (STMicroelectronics, 2023), the continuous tone is broadcast at:

$$\text{Tone Frequency} = 32e6 \times \langle\text{Configured Frequency}\rangle / 2^{25}$$

So, for a configured frequency of 868 MHz:

$$\text{Tone Frequency} = 32e6 \times 868\ \text{MHz} / 2^{25} = 827.789\ \text{MHz}$$

Because 827.789 MHz is authorized to broadcast with the same power as 868 MHz, it is still permissible to broadcast at maximum power. To use the radio components directly, the custom firmware bypasses the LoRa protocol allowing for a continuous broadcast at the fixed frequency. A hardware debugger, ST-LINK V2, was required to flash firmware onto the RFM95 device.

4.4.2. Observations

The Wio-E5 features a 48 MHz digital processor. Based on the previous discussion, the potential for clock and carrier frequency mixing emphasizes analysis of signals at 827.789 MHz ± (x * 48 MHz) where x is an integer. For completeness though, the range from 727 MHz to 928 MHz was analyzed. Despite a large number of unintended broadcast signals, no signal showed any direct relationship to the sleep/encrypt cycles of the digital processor. Signals from the Wio-E5 were generally solid vertical lines, such as the one at 870.7 MHz in Figure 10, as expected for a continuous, unmodulated signal. There were instances of faint, intermittent signals, such as the one at 872 MHz, but none showed patterns indicative of a leak. It is possible though that these signals are being modulated in an unpredicted way, so they will be further examined in a subsequent article.

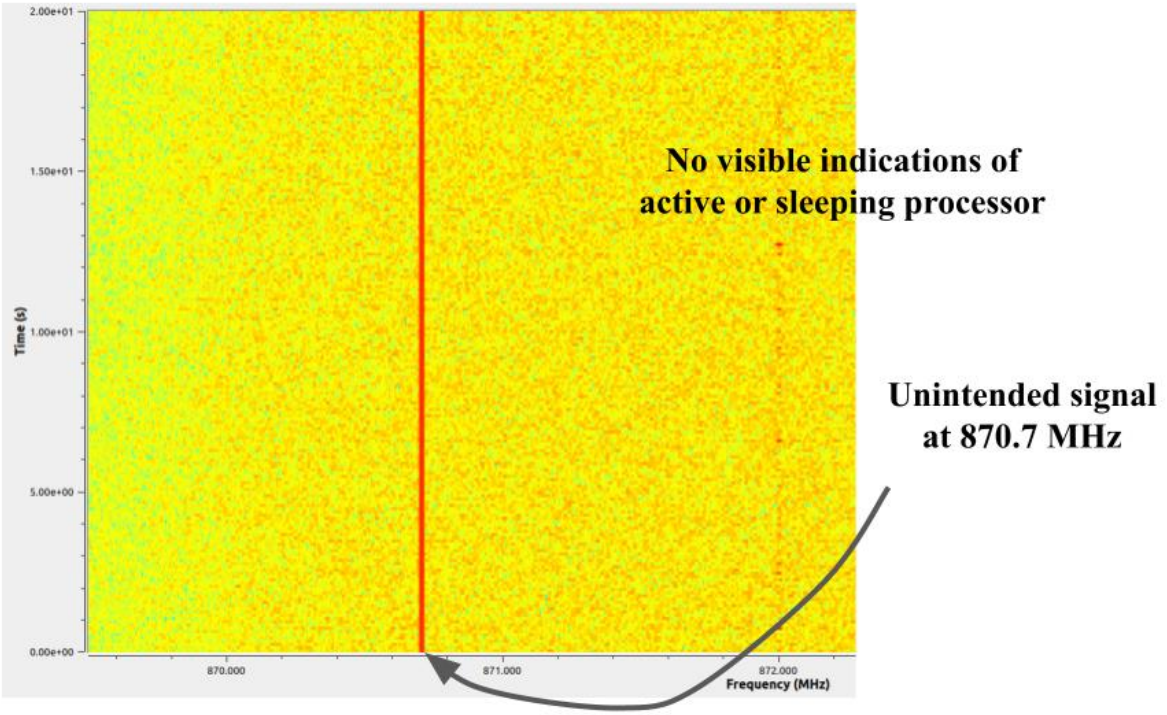

Figure 10: Wio-E5 broadcasting at 827.789 MHz while fluctuating between periods of sleep and intense processing resulting in an unexpected signal modulation that does not appear to be related to the sleep/active cycles of the processor.

5. Conclusion

This research postulated that exercising hardware in a specific manner would cause a *Screaming Channels* leak to be apparent on a time-frequency diagram. The results of the known-vulnerable nRF52832 lent credence to that hypothesis as they exhibited an obvious predicted pattern. Additionally, the CC1111 showed additional evidence that the proposed analytic can detect potential *Screaming Channels* leaks. While not conclusive, the CC1310 showed that the processor impact entering a sleeping state had a detectable impact on the analog signal. It is not yet clear if any of these detected anomalies are impactful to the security of the devices, but this impact will be further examined in a subsequent article. The Wio-E5 transceiver showed no perceptible relationship between the digital processor state and the broadcast.

Based on the observed results, one can conclude that the analysis process correctly identifies *Screaming Channels* leaks in some cases. Specifically, the process correctly identified the leak in the known vulnerable Bluetooth device and discovered a new potential leak in the CC1111. However, it is unclear if the other two analyzed devices exhibit leaks or not. For this reason, it is unclear if the process failed to identify leaks or correctly identified that no leak is present. While the CC1310 results did show the digital processor impacting the analog signal, the expected dashed line pattern was absent. In the CC1310 and Wio-E5 cases, it is possible that the leak is so subtle that the intensity change is too insignificant to be observed by the human eye. Despite the value of the analytic in intuitiveness and ease of demonstrating, there would be a benefit in evaluating the statistical variance of each signal to provide a metric beyond the visual change. The next article in this series will examine this idea in greater detail.

Contributions

To perform the analysis described in this article, a significant amount of software was created. Specifically, firmware was created to exercise each device in the described manner. GNU Radio Companion scripts were written to visualize signals at target ranges with configurable amplification. The source code for the project and step-by-step instructions for reproducing the analysis are available at https://github.com/GallagherTom/screaming_satellites.

References

1. AmbaSat Ltd. (2022, January 16). *AmbaSat-1*. GitHub. Retrieved October 3, 2023, from https://github.com/ambasat/AmbaSat-1/
2. Bensky, A. (2019). *Short-range Wireless Communication*. Elsevier Science. 10.1016/C2017-0-02356-X
3. Camurati, G. (2018, June 24). *Screaming Channels GitHub Repository*. GitHub. Retrieved October 3, 2023, from https://github.com/eurecom-s3/screaming_channels
4. Federal Communications Commission. (2017). *Low Power Radio - General Information*. Low Power Radio - General Information. Retrieved September 1, 2023, from https://www.fcc.gov/media/radio/low-power-radio-general-information
5. Federal Communications Commission. (2019). *Unauthorized Radio Operation*. https://www.fcc.gov/sites/default/files/unauthorized_radio_operation.pdf
6. Federal Communications Commission. (2023, 8 31). *47 CFR Part 15 -- Radio Frequency Devices*. Retrieved September 3, 2023, from https://www.ecfr.gov/current/title-47/chapter-I/subchapter-A/part-15
7. Grimm, C., & Rathmair, M. (2017). Dealing with Uncertainties in Analog/Mixed-Signal Systems: Invited. *Proceedings of the 54th Annual Design Automation Conference 2017, 2017*(Article 35), 1-6. 10.1145/3061639.3072949
8. Mazar, H. (2016). *Radio Spectrum Management: Policies, Regulations and Techniques*. Wiley.
9. STMicroelectronics. (2023, January 1). *STM32WL5x advanced ARM-based 32-bit MCUs with sub-GHz radio solution - Reference manual*. STMicroelectronics. Retrieved October 4, 2023, from https://www.st.com/resource/en/reference_manual/rm0453-stm32wl5x-advanced-armbased-32bit-mcus-with-subghz-radio-solution-stmicroelectronics.pdf
10. Texas Instruments. (2018, July 1). *CC1310 SimpleLink™ Ultra-Low-Power Sub-1 GHz Wireless MCU datasheet*. Texas Instruments. Retrieved September 21, 2023, from https://www.ti.com/lit/ds/symlink/cc1310.pdf
11. Texas Instruments. (2019, March 9). *CC1110Fx/CC1111Fx: Low-Power SoC with MCU, Memory, Sub-1 GHz RF Transceiver, and USB Controller*. True System-on-Chip with Low-Power RF Transceiver and 8051 MCU Datasheet. Retrieved October 4, 2023, from https://www.ti.com/lit/ds/symlink/cc1110-cc1111.pdf
12. United States Government. (2023). *Code of Federal Regulations*. Title 47 of the CFR. https://www.ecfr.gov/current/title-47

13. US Department of Justice. (2022, May). *Title 9-48.000 - COMPUTER FRAUD AND ABUSE ACT*. Retrieved September 1, 2023, from https://www.justice.gov/jm/jm-9-48000-computer-fraud
14. US Federal Aviation Administration. (2022, May 19). *Aeronautical Information Publication*. Retrieved September 19, 2023, from https://www.faa.gov/air_traffic/publications/media/aip_bsc_w_amd_1_dtd_11-3-22.pdf