The Internet of Things: Vulnerabilities and Defense Mechanisms in a Hyperconnected World

Emanuel Mairoll, Julian Oberhofer

January 31, 2024

1 Introduction

The Internet of Things (often abbreviated IoT) is now a common part of our daily lives, with smart devices everywhere in our homes and workplaces. As IoT becomes more widespread, people often overlook the potential vulnerabilities in seemingly innocuous devices, such as smart lightbulbs. However, the security of these interconnected devices presents a significant challenge, as they become prime targets for cyber attacks. This paper will explore the current state of IoT security, focusing on the main threats, vulnerabilities, as well as the methods to protect against them.

1.1 Definition of IoT

Defining the Internet of Things (IoT) is a complex task due to the wide range of perspectives and interpretations that exist. After reviewing various sources and definitions, we have chosen to adopt a definition that captures the essence of IoT in a broad yet precise manner:

> *The term IoT describes devices and gadgets, that were traditionally not connected, have sensors and/or actuators, and can be controlled remotely.*

The definition we have adopted includes an extensive range of devices and systems, encompassing everything from smart light bulbs, roller shutters, and locks to pet feeders, cameras, and various kitchen appliances. It also covers larger items like white goods, access control systems, and even cars. Recently the Consumer Electronics Show (CES) in Las Vegas has showcased a multitude of new products in all these categories (even including an AI companion robot named "Ballie"[14]), highlighting the ever-expanding scope in the field of IoT.

2 Firmware

IoT firmware provides the APIs for controlling device functionalities, essentially defining each device's role, such as "I am a light bulb". With the evolution of IoT, various protocols and methodologies for firmware have developed, each bringing their own benefits and challenges:

Webserver: This implementation allows IoT devices to be accessed through a web interface via a web browser. It provides a straightforward but somewhat cumbersome method of interaction. Basic security (excluding implementation issues like memory safety) is maintained as long as access requires authentication by for example a password.
REST/RPC: Designed for integration into other systems, REST (Representational State Transfer) and RPC (Remote Procedure Call) offer simplicity and a minimal attack surface. However, security should be ensured through proper authentication.
MQTT: Unlike some other protocols, MQTT (Message Queuing Telemetry Transport) designates the IoT device as a client, connecting to an MQTT broker. It enables publishing and subscribing to specific "topics," making it suitable for scenarios like controlling a light by publishing messages like "ON" or "OFF" to the "living-room/light1" topic, which the subscribed device receives and reacts accordingly to.[13]
Vendor-specific Protocols: Major tech companies like Apple (HomeKit), Amazon (Amazon Smart Home API), and Google (Google Home) have introduced their own protocols for IoT. These all build upon similar principles,

coming with predefined services (e.g., light bulbs) and characteristics (e.g., brightness, color). They often employ cryptographic pairing for enhanced security and user-friendliness. However, as these protocols are not interoperable, they often lead to vendor lock-in.
Matter: Matter is an emerging open standard backed by major manufacturers, including Amazon, Apple, Google, Comcast, and the ZigBee Alliance. It offers a royalty-free, open-source connectivity standard and promises to bridge the gap between various manufacturers and their IoT products[1]. Currently, it is generating significant traction in the IoT community, evidenced by its prevalence in new products introduced at CES 2024.

2.1 Challenges in IoT Software Development

The development of software for IoT devices presents a unique set of challenges that require careful consideration:

Limited Computing Power and Weak Encryption: Many IoT devices operate with constrained resources, including limited processing capabilities and memory. This constraint can make implementing robust encryption and security measures challenging. Due to this limitation, some vendors also fall back on Security by Obscurity, where device functions and vulnerabilities are hidden rather than addressed. This often leads to catastrophic vulnerabilities only being found after rollout, as evidenced by NXPs Crypto-1 cipher employed in millions of MIFARE Classic NFC cards[19].

Embedded Development Complexity: Developing software for embedded systems, typical of IoT devices, is notoriously complex. It requires specialized knowledge and tools, making the development process more challenging. Moreover, IoT development often involves working with opaque documentation that is incomplete or not readily accessible, further complicating the development process. Additionally, the lack of standard implementations in IoT results in manufacturers creating their own, often half-stable code. This complex landscape is exacerbated by a significant portion of IoT software being written in C/C++, which lacks memory safety features. This can lead to vulnerabilities like buffer overflows and memory leaks, leading to exploit chains rooting devices remotely, bypassing all otherwise well thought out security measures.[11, 5, 3]

2.2 Vendor Clouds

Many manufacturers commonly use the approach of connecting IoT devices directly to their cloud services, bypassing local APIs. This practice grants them significant control and the ability to process a wealth of data from devices, such as usage patterns, sensor readings, and user interactions, which aids in improving their products and services. Yet, this centralization of control also introduces several substantial risks.
The primary risks stem from the potential misuse of sensitive user data. With access to detailed information, there's a possibility of selling or leaking this data, either intentionally or through security breaches. Additionally, the capability for remote firmware upgrades, while convenient, poses a risk of unauthorized or malicious updates that could compromise device functionality or user privacy.

Case Study: Tuya, a Chinese IoT platform-as-a-service company, enables manufacturers to quickly create IoT devices. Criticized by Michael Steigerwald in his "Smart Home — Smart Hack" presentation[16], Tuya's model was shown to have significant security flaws. The platform's devices often have unencrypted connections to the cloud and receive firmware updates remotely, posing the risk of unauthorized firmware modifications. Steigerwald also highlighted the extensive data collection by Tuya, including location and personal information, leading to major privacy concerns.

3 Project: Google Dorking for IoT Devices

As alluded to in the previous section, many IoT devices are connected to the public internet, often as exposed services, enabling their owners to control them remotely. It was also established that these connections should be secured with authentication and encryption, yet frequently, such measures are not implemented properly. Here we aim to explore the prevalence of such vulnerabilities and the potential risks they pose.

In this project, we will investigate a technique called "Google Dorking" to find exposed IoT devices. Search engines like Google use web crawlers to index all publicly available websites, which includes not only pages such as Wikipedia and PlusOnline, but also the web interfaces of numerous IoT devices connected to the internet.
By deliberately crafting search queries, we can uncover many of these exposed devices. Often, we can refine our search query to a point where we find devices that lack any password protection at all — a situation that poses a significant security risk, as it allows anyone to access, control, and monitor these devices.

3.1 The Search Query

We use the Exploit-DB from Offensive Security as a starting point, focusing specifically on the Google Hacking Database section[15]. After thorough investigation, we settle on examining CCTV management software, namely "WebcamXP" and "Webcam 7", with a particular focus on the latter[17]. Although this software features user management and password protection, our focus is on devices that have neglected these security measures.
The web interface of this software has a page that displays streams from all connected devices, accessible if the logged-in user has viewing rights or if no authentication is required. This page becomes our key target to find devices without password protection.
Subsequent analysis reveals that the title of all webpages associated with this software consistently includes "Webcam 7", and the specific page of interest contains "multi.html" in the URL. Armed with this information, we formulate the search query: "intitle:'webcam 7' inurl:'/multi.html' "

3.2 Results

Our search results uncovered unprotected devices worldwide, ranging from a Chicken Coop and an Animal Shelter to a Front Porch, various random Living Rooms, roundabouts, streets, and many others. Particularly concerning was a CCTV setup in a company, consisting of 12 cameras, monitoring the entire facility.
Furthermore, we found that every single connection used HTTP instead of HTTPS, meaning that all data is transmitted in "plain text". This makes credential interception trivial in cases where the device would require authentication. The issue is exacerbated by the fact that each web server is reachable via a different port, most commonly 8080, which is the default suggested by the software's documentation. On some occasions, we even found ports like 443 being misused, demonstrating a critical misunderstanding of HTTPS and security practices. It is noteworthy that using an obscure port, such as '32479' identified in one instance, is not a valid security measure and is akin to "Security by Obscurity".
To investigate further, we truncated the URLs to just the addresses and found login pages for a commercial network switch in one instance and a domestic router in another. One site even geoblocked us. We halted our investigation at this point to avoid legal issues, but given the lax "security" measures observed, we assume that bypassing these hurdles would likely be trivial.

3.3 Defense Mechanisms

In theory, the best defense against this kind of attack is to not connect the device to the internet at all. However, this is not always feasible, as some devices require an internet connection to function as intended. In such cases, ensuring proper access control and encryption is crucial. "Webcam 7" offers a user management system and supports HTTPS, but in the instances we uncovered, these security features were not utilized. Furthermore search engines like Google offer a way to remove a website from their index, or prevent crawling altogether, through the use of a "robots.txt" file in the root directory which outlines the pages that should and should not be indexed. Using a "noindex" meta tag in the header of every page that should not be indexed is also an option[8].

4 Communication

Wireless communication is favored in IoT devices due to its ease of installation and network integration, making it ideal for retrofitting where adding wires is impractical. This approach allows for scalable network expansion as more devices are added without the need for significant changes in physical infrastructure. Additionally, the mobility and flexibility provided by wireless communication are essential for devices like autonomous vacuum cleaners, where

movement is integral to functionality. Maintenance of these wireless IoT devices is generally straightforward, often limited to battery replacements, reinforcing their practicality in various applications.

4.1 Sub-GHz

Sub-GHz frequency bands, particularly around 433.92 MHz, are widely utilized in various control systems due to their advantageous characteristics in terms of range and obstacle penetration. These frequencies are known for their high efficiency in energy usage and the simplicity of the underlying technology. However, this spectrum is not without its limitations. One drawback is the relatively low data rate compared to higher frequency bands. However, this is oftentimes not a concern for IoT devices, as they typically do not require high data rates. Additionally, there are notable security concerns associated with Sub-GHz communications. A common issue is that transmissions in this band are often unencrypted, rendering them vulnerable to replay attacks. While the adoption of rolling codes, which change after each use, has been a strategy to enhance security, these too can be compromised through sophisticated rolljam attacks.

4.2 WiFi

WiFi is a established wireless communication standard. It has a high data rate and solid security features[21]. Yet the low computational power of many IoT devices makes it challenging to implement the necessary encryption and authentication protocols in a timely manner. Furthermore WiFi is also known for its high power consumption, making it unsuitable for battery-powered IoT devices. Additionally, WiFi is not as scalable as other wireless protocols, as it needs dedicated access points to extend the network. On the other hand, IoT devices can be easily integrated into existing WiFi networks without the need for additional infrastructure. It is worth noting that many IoT devices rely on older WiFi standards, which are known to have significant security vulnerabilities. For example, the KRACK attack, which exploits a flaw in the WPA2 protocol, can be used to intercept and decrypt WiFi traffic[18]. While this vulnerability has been patched, many IoT devices will likely never receive an update, leaving them vulnerable to attacks.

4.3 ZigBee

ZigBee is a wireless communication protocol designed for low-power, low-data-rate applications. It extends the IEEE 802.15.4 standard, operating most commonly in the 2.4 GHz band. The network topology is flexible but is most often based on a mesh network, allowing for increased reliability and range. To thwart replay attacks, ZigBee incorporates a 32-bit Transmission Counter (TC) in the header of each packet. Nodes store the last known TC of their neighbors, rejecting packets with a TC lower than expected. ZigBee also employs 128-bit symmetric encryption keys (AES) for securing communications. There a 2 types of keys, the Network Key, used for broadcast communication, and the Link Key, used for communication between 2 devices. Those keys are generated and distributed by the ZigBee Trust Center. The communication between the Trust Center and the devices is secured by a pre-shared key, known as the Trust Center Link Key. The issue, however, is that the Trust Center often utilizes a default value, hard-coded into the devices. The default value is specified in the ZigBee standard as '5A:69:67:42:65:65:41:6C:6C:69:61:6E:63:65:30:39', which translates to 'ZigBeeAlliance09' in ASCII[24, Page 409]. The problem is exacerbated by the fact that the flaw was known at the point of specification writing but was not addressed, and presumably, never will be. To quote the ZigBee specification from 2017: "During initial key transport the keying material used for protection may be a well-known key, thus resulting in a brief moment of vulnerability where the key could be obtained by any device."[24, Page 407].

4.4 Thread

Thread is the latest wireless protocol to gain traction in the IoT space. It serves as the physical layer for the aforementioned Matter protocol and is an open standard. It differs from ZigBee in that it is IP-based, and support multiple hubs, called Border Router, which can be used to connect the Thread network to the internet or other networks. Thread utilizes 6LoWPAN, a low-power wireless protocol, to transmit IPv6 packets over a PAN (Personal Area Network). 'No single point of failure' is a key design principle of Thread, which is achieved by utilizing multiple Border Routers and a interconnecting scalable mesh network. Thread also mandates the use of a secure connection between devices from the internet and the Thread network, ensuring that only authorized

devices can access the network[9]. Furthermore, a open implementation of Thread, called OpenThread, developed by Google, is available on GitHub, allowing for easy integration into new and existing devices[20].

5 Project: Sub-GHz Reconnaissance

As established above, Sub-GHz frequencies are a ideal band for wireless communication in the IoT space. The simplicity and effectiveness of Sub-GHz bands make them a popular choice for a range of devices, from home automation systems to car remote keys. Hacking or pentesting these devices has however become relatively straightforward with the use of specialized hardware. Common tools in the market include the RTL-SDR, Yard Stick One, and HackRF One, each with their unique capabilities in intercepting and analyzing Sub-GHz signals. Among these, the Flipper Zero stands out as a "Portable Multi-tool Device for Geeks" designed for pentesting and exploring various wireless systems.

5.1 Flipper Zero: A Hacker's Multitool

The Flipper Zero is a compact and versatile tool tailored for the pentesting community. It's not just limited to Sub-GHz communication; it supports a wide range of frequencies and protocols, including NFC, RFID, Wi-Fi, Bluetooth, and Infrared. For Sub-GHz operations, the firmware comes preloaded with a library of common protocols, allowing pentesters to readily interact with a myriad of devices right out of the box.

Sub-GHz Capabilities of Flipper Zero:
Frequency range: 300—928 MHz
Supported modulation types: AM270, AM650, FM238, FM476
In-built support for numerous protocols, like Nice Flo, CAME, DoorHan and Security+[23, 22]

5.2 Scanning Process

Using Flipper Zero for Sub-GHz reconnaissance typically is a two-step process: scanning for signals and then replaying them. The process starts by setting the Flipper Zero to the appropriate frequency band and modulation type. Once configured, it can scan and capture signals from various devices within its range. After capturing a signal, the Flipper Zero can store it for analysis or replay it to test the response of the targeted device.
Using Flipper Zero for Sub-GHz reconnaissance typically involves first capturing signals in 'Read' mode and then replaying them to test device responses. Initially, you set the Flipper Zero to the appropriate frequency and modulation, with can be found in the "Frequency Analyzer" mode. In the scanning phase, you have two key modes: "Read" and "Read Raw".
Read Mode: In this mode, Flipper Zero decodes and interprets the signal based on known protocols. It's useful for common devices where the signal structure is standardized and recognized by the Flipper Zero's firmware. This mode simplifies the process of understanding the signal's purpose and structure, making it easier to identify the type of device you're scanning.
Read Raw Mode: This mode is used when dealing with unknown or proprietary signals. Flipper Zero captures the raw signal data without attempting to decode or interpret it. This mode is useful for analyzing new or uncommon devices where the signal pattern isn't recognized by existing protocols in the Flipper Zero. It allows for a deeper level of analysis, as you can examine the raw signal patterns and timings, which can then be used to reverse engineer the protocol or create custom replay attacks.
After capturing the signal, whether in Read or Read Raw mode, Flipper Zero can store it for further analysis or immediate replay to observe the response of the target device.

5.3 Testing Devices with Flipper Zero

Continuing our investigation, we used Flipper Zero to test several devices, selected based on their common usage in IoT settings and differing communication protocols. This selection helped us assess the full scope of Flipper Zero's functions. For each device, we recorded key details like its communication protocol, operating frequency, and any implemented security measures such as rolling codes.

5.3.1 Remote Control Socket

Product: HM Müllner SDF3 Remote Control Socket
Frequency, Modulation: 433.88MHz AM
Protocol: Princeton 24bit
Uses Rolling Codes: No

We began our testing with the HM Müllner SDF3 Remote Control Socket, a simple device, expecting minimal security features. Our initial step was to determine the operating frequency. Using Flipper Zero's Frequency Analyzer mode, we identified the frequency as 433.88 MHz. We then set the Flipper to this frequency and switched to the standard Read mode, ready to capture the signal.
Upon activating the socket with its remote control, the Flipper Zero instantly captured the signal, displaying it in the list of received signals. We repeated the process to turn off the socket, which was also successfully detected. Next, we tested the replay functionality; the signal replayed by the Flipper immediately activated the socket. We were also able to switch the device on and off multiple times with the same respective sample, proving the lack of security features like rolling codes This device proved to be an excellent example for public demonstrations of Sub-GHz vulnerabilities, given the ease with which we could intercept and replicate its signals.

5.3.2 Effect Fog Machine

Product DJPOWER DSK-1500VS LED Vertical Fog Machine
Frequency, Modulation: 433.88MHz AM
Protocol: Princeton 24bit
Uses Rolling Codes No

As professional lighting technicians, we frequently work with fog machines, which can create stunning effects but are not without risks if misused. We were therefore quite surprised to find that the DJPOWER DSK-1500VS, a popular model among vertical LED foggers, was shipped with a standard 433 MHz remote control. This raised concerns about its security and potential misuse.
Following the approach we used with the socket, we located the operating frequency of the fog machine, which was 433.88 MHz, using the same Princeton 24 Bit protocol. Upon capturing and replaying the remote's signal with the Flipper Zero, the fogger responded by discharging fog at full intensity. This experiment demonstrated a significant security vulnerability, as the fogger could be triggered unexpectedly.
Fortunately, the remote control functionality of these foggers is automatically deactivated when they are integrated into a DMX control chain. However, when operated independently with a remote at events, this vulnerability could pose serious risks, potentially leading to hazardous situations if the device is triggered inadvertently.

5.3.3 Courtyard Gate

Product: Tousek Rotary Gate Control ST 51
Frequency, Modulation: 433.92MHz AM
Protocol: KeeLoq 64bit
Uses Rolling Codes: Yes

Seeking to challenge ourselves further, we decided to intercept a device with more sophisticated security features. We chose a courtyard gate, specifically the Tousek Rotary Gate Control ST 51, anticipating it to employ at least some form of rolling code for security.
The gate's remote had a single button to both open and close it. Intercepting the signal from the remote revealed a frequency of 433.92 MHz and the use of the KeeLoq 64bit protocol, which was unfamiliar to us at the time. Attempting a simple replay of the captured signal did not work. Upon researching KeeLoq, we discovered it is a widely used security method involving encrypted rolling codes, designed to prevent unauthorized signal replication. Considering the complexity of KeeLoq's encryption, we chose not to attempt reverse-engineering the hardware keys. Instead, we simulated a jamming attack. We positioned ourselves at a considerable distance from the gate and recorded three consecutive signal samples from the remote. Upon returning to the gate, we replayed the first captured sample, which successfully initiated an opening cycle. Subsequent attempts to reuse the same code

were ineffective. However, when we sequentially replayed the second and third samples, each worked perfectly, successfully mimicking the rolling code system and triggering the gate's operation.

5.3.4 Alarm System

Product: [Redacted for security reasons]
Frequency: 868.35MHz
Protocol: Unknown
Uses Rolling Codes: Seemingly

Eager to delve into more critical and impactful testing, we turned our attention to an alarm system installed at home. For security reasons, we will not disclose the product name, but it's a system from 2004, equipped with a remote that has two buttons for arming and disarming the alarm.
Initially, we focused on identifying the operating frequency. This time, it was a higher frequency at 868.35 MHz. The standard 'Read' mode of the Flipper Zero was unsuccessful, suggesting a proprietary protocol. We switched to 'Read Raw' mode to capture the signal. Surprisingly, we noticed that the remote didn't just send four short pulses when arming or disarming; the alarm system responded with four short, yet different pulses at a similar frequency.
After some experimentation, we successfully captured an 'arm' signal using the 'Read Raw' mode. Surprisingly, despite the system's seemingly robust security measures, replaying this signal caused the system to arm itself. However, when we attempted the same with the 'disarm' signal, it didn't work, suggesting enhanced security for this more critical function. Drawing on our experience with the courtyard gate, we captured the disarm signal from a distance and later replayed it near the system. This approach successfully disarmed the system, indicating that while the disarm function had enhanced security, it was still vulnerable to more sophisticated attack methods.

5.3.5 Car Remote Locking System

Product: VW Beetle Model 2016
Frequency: 434.42MHz
Protocol: Unknown
Uses Rolling Codes: Unknown

For our final test, we aimed high and decided to tackle the "champions league" of remote systems — unlocking a car. With the insights gained from previous experiments, we felt confident about our approach. We began by identifying the operating frequency of the VW Beetle's remote system, which was 434.42MHz. As expected, the standard 'Read' mode did not yield any known protocol, so we shifted to capturing a raw signal from a distance. When this signal was replayed in proximity to the car, to our amazement, it worked — the car unlocked.
To contrast this result, we attempted a similar experiment with a Mercedes A-Class from 2016, using the same approach. However, this time we were unsuccessful in unlocking the car. A closer inspection of the opening sequence using the frequency analyzer revealed communication occurring over multiple frequency bands. This observation suggested a more sophisticated handshake mechanism in place, one that couldn't be bypassed simply through a replay attack. This difference in security measures between the two vehicles underscored the varying levels of sophistication in car remote locking systems.

6 Hardware

In IoT security, understanding the hardware is as crucial as software. This section focuses on the core assumptions, interfaces, and vulnerabilities in IoT hardware.

6.1 Misconceptions and Realities

In IoT security, certain prevalent assumptions about hardware are often made by users or less informed developers. However, closer inspection frequently reveals these assumptions to be misconceptions.

Hardware is disassemblable: While IoT devices are often viewed as black boxes difficult to tamper with, many are surprisingly easy to access and disassemble, leaving them vulnerable to tampering.
ROM is writeable: Contrary to the expectation of tamper-proof, read-only firmware, numerous devices feature rewritable flash memory and often have unsigned updates, exposing them to potential manipulation.
Debug interfaces left open: Despite assumptions of secured interfaces, IoT devices frequently have unprotected access points like consoles, SSH, or debug protocols, creating opportunities for unauthorized control and access.
Next, we will examine two common debug protocols that can be used to read and rewrite the ROM.

6.2 UART Serial Interface

UART, a serial communication interface, typically involves four pins (power, ground, receive, and transmit) and can be often found as unpopulated header on breadboards. It's often unsecured, offering debug output or even console access to the device. This accessibility can lead to uses ranging from debugging to firmware reflashing[6]. The rooting of devices through UART has even become a popular challenge among tech enthusiasts, as evidenced by online demonstrations such as the hacking of a baby monitor on the "Low Level Learning" YouTube channel[10]. Notably, reflashing firmware via UART is a standard feature in many chipsets, including popular ones like ESP8266 and ESP32. They standardly feature a Firmware Download Mode, accessible via a button or pin, which can be used to write a new firmware using standard tooling. Apart from some soldering, this process is often quick and straightforward.

6.3 JTAG Debug Interface

The JTAG (Joint Test Action Group) debug interface is a standard tool for debugging firmware and printed circuit boards (PCBs). Often recognizable as a larger, unpopulated header on PCBs, it features the pins Test Data In (TDI), Test Data Out (TDO), Test Clock (TCK), Test Mode Select (TMS), and, optionally, Test Reset (TRST). There also exist variants such as 2-Wire JTAG and Serial Wire Debug (SWD) to fit different hardware requirements by optimizing the number of connections and the method of communication. The JTAG interface enables developers to debug by reading and writing to device registers, stepping through processes, and monitoring real-time execution of firmware, allowing thorough testing and troubleshooting of hard- and firmware. Additionally, JTAG allows for the reflashing of a device, enabling updates or complete firmware replacements as needed for maintenance or upgrades.[2]
Securing the JTAG interface is crucial, yet it's often overlooked. Ideally, hardware fuses could permanently disable debug access, or password protection could limit unauthorized use. Unfortunately, most devices lack these security measures, leaving them open to potential exploitation.

6.4 Numerous Attack Vectors

IoT devices with modified or malicious firmware pose significant risks to both users and networks. These devices can become conduits for various forms of cyberattacks, leading to breaches of privacy, network security, and more.
Backdoors into Private Networks: Compromised IoT devices can serve as gateways for unauthorized access to private networks, allowing attackers to infiltrate sensitive data and systems. Attackers can easily collect reverse shells from these devices, connecting to a centralized Command and Control (C&C) server to coordinate further attacks.
Illegal Traffic Generation: Maliciously modified devices can be hijacked to participate in Distributed Denial of Service (DDoS) attacks or other nefarious activities, such as the infamous Mirai botnet[4]. These compromised devices become part of a vast network of bots used to launch large-scale attacks on target websites or services.
Malware Distribution: IoT devices with modified firmware can be used to spread malicious software within a network. Attackers can preload these devices with malware, effectively turning them into carriers that propagate the payload to other vulnerable devices within the same network.
User Surveillance: Modified firmware can enable unauthorized surveillance of users, posing serious privacy concerns. A common target for this type of attack is security cameras. Attackers can gain access to these cameras, monitor live feeds, and potentially capture sensitive information about users and their surroundings.

Additionally, it is important to mention that bringing infected/modified devices to unsuspecting users is trivial and has occurred multiple times in the past. These malicious devices can easily enter circulation through various means:
Second-hand Market: IoT devices sold in the second-hand market may contain malicious modifications, endangering unknowing users[12]. For instance, cases have been reported where IoT devices purchased on Ebay were found to have preloaded spyware and other malicious firmware modifications.
Returns: Devices returned to manufacturers or retailers, if not adequately checked and reset, can potentially reintroduce security risks when resold to unsuspecting customers. Amazon, for instance, often handles returns and may resell opened products that show no visible difference, making it challenging to guarantee that the firmware hasn't been modified.
Gifts and Giveaways: Devices given away as gifts or promotional items can also be used to spread security threats. For example, a seemingly harmless smart light bulb gifted to a user may contain malware-loaded firmware, leading to the infection of other devices in the recipient's network.

7 Project: Introspecting an Eufy Robovac L70

To gain a deeper understanding of the risks associated with reflashing firmware in IoT devices, we introspected our own smart home environments. Our objective was to identify the device that would pose the most significant threat if compromised. For this investigation, we selected the vacuum robot "Eufy Robovac L70 Hybrid", primarily due to its mobile nature combined with a multitude of visual sensors — essentially functioning as a camera on wheels.

7.1 Pre-Disassembly Research

General Approach: Typically, pre-disassembly research involves gathering extensive information about the device.
Device Manual: Consulting the device manual for official guidelines on disassembly, particularly focusing on user-serviceable parts like the battery.
Teardowns: Investigating existing teardowns (e.g., iFixit) for a preliminary understanding of the device's internal structure and component layout.
FCC Database: Reviewing internal photos from the FCC's online repository for insights into the device's internal components and construction.
Community Projects: Exploring community-driven projects like Valetudo for insights on custom firmware compatibility and capabilities.
Before beginning the disassembly of the Eufy Robovac L70 Hybrid, we referred to online tutorials on battery replacement, showing how to remove the battery cover of the device. For an overview of the device's components and construction, we examined internal photos available in the FCC database. The document FCC ID 2AOKB-T2190 offered detailed visual references, allowing us to safely disassemble the vacuum cleaner with a minimized risk of damaging it.

7.2 Disassembling and Reverse Engineering the Hardware

The disassembly of the Eufy Robovac L70 Hybrid involved first removing three screws from the bottom and an additional screw for the side brush to loosen the battery cover. After safely disconnecting the battery, the removal of the battery cover exposed eight more screws. Before the vacuum could be divided into its bottom and top units, two connectors had to be detached, allowing the top part, which houses the main PCB, to be lifted away.

Upon inspecting the main PCB, densely populated with SMD parts and connectors, a prominently placed IC with many connections caught attention; shining a light on the chip revealed the label "GD32VF103R6T6". Attention was also given to any unused connectors and unpopulated headers. Notably, a 5-pin header labeled "Debug" was discovered, suggesting a potential for debugging or firmware analysis. This observation, alongside the noted unpopulated headers and unused connectors, marked significant points for further exploration.
Our research centered on the "GD32VF103R6T6" chip. Found on Gigadevice's website, this chip is part of the GD32VF103 series of MCUs, characterized by their RISC-V core architecture[7]. They are known for combining high performance with low power consumption. After locating the datasheet, we matched the pinout with the

configuration on the PCB. Key connections from notable headers and connectors to the chip's pins were traced using a voltmeter. In this case, the Debug Header was connected to VCC, GND, NRST, PA13, and PA14, aligning with a 2-Wire JTAG interface as per the datasheet.

7.3 Introspecting the Software

Considering the potential risk of bricking the device, we refrained from physically engaging with the JTAG interface, and our practical research was halted at this point. However, theoretically, the process would involve the following steps:
Connection: A JTAG adapter could theoretically be connected to the Robovac's debug header, linking it with a PC for communication.
Debugging Process: With the connection established, debugging tools such as GDB could be employed for comprehensive analysis and interaction with the device's internal processes.
Firmware Interaction: Theoretically, this setup would allow for the downloading of the device's current firmware directly from the SoC's flash memory. This step could also facilitate the extraction of sensitive information, such as login credentials stored within the firmware.
Establishing SSH Access: Another theoretical possibility could include the insertion of an SSH public key into the device's firmware. This modification could provide remote SSH access, potentially allowing for easier management or further modification.

This method shows how easily the vacuum's software could be accessed and changed, emphasizing the risk of malware and backdoors. It stresses the need to be cautious about IoT device security.

8 Conclusion

The landscape of IoT presents a paradoxical blend of revolutionary potential and significant security challenges. Despite often being dubbed as "digital dumpster fire" by security-aware professionals, IoT's potential for future advancements is undeniable. The key is to deploy IoT devices judiciously, being fully aware of their limitations and inherent risks. Essential considerations for a secure and sensible IoT implementation include:
Local-Only Operations: Restricting IoT devices to local network operations minimizes the risk of external attacks. This confines device activity to a more secure and controllable environment.
Risks of Vendor Clouds: Utilizing vendor-managed cloud services introduces significant risks, including potential data breaches and a heavy reliance on external infrastructures. These services, often targeted by cyberattacks, can compromise both user privacy and device security.
Avoiding Outdated Wireless Standards: Using the latest wireless communication standards is crucial. Outdated standards are often riddled with known vulnerabilities, while newer protocols offer stronger protections against unauthorized access.
Segmenting IoT into a Separate VLAN: Isolating IoT devices in a dedicated VLAN is an easy yet effective way to protect the main network. This separation acts as a barrier, preventing potential breaches in IoT devices from affecting other network areas.
Caution in the Second-hand Market: The second-hand IoT market demands careful scrutiny. Devices acquired from online marketplaces may carry hidden vulnerabilities or altered firmware. When acquisition of such devices is necessary, it is advised to perform a complete reset and update to the latest firmware to mitigate these risks and ensure the device's integrity.
Adopting Open-Source Alternatives: Open-source IoT solutions offer advantages in security, adaptability, and transparency. Notable projects include HomeAssistant for private, local control of IoT devices; ESPHome and Tasmota for open-source firmware on ESP8266/ESP32 devices; Tuya-Convert for replacing Tuya-based firmware without soldering; Adafruit and Olimex for DIY electronics and IoT projects; and Valetudo, specifically for vacuum robots, replacing the vendor cloud with a local service.

References

[1] Connectivity Standards Alliance. *Matter*. URL: https://csa-iot.org/all-solutions/matter/ (visited on 01/29/2024).

[2] Arduino. *JTAG*. URL: https://www.arduino.cc/reference/en/libraries/jtag/ (visited on 01/29/2024).

[3] Jiongyi Chen et al. "IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing." In: *NDSS*. 2018.

[4] Cloudflare. *Mira Botnet*. URL: https://www.cloudflare.com/de-de/learning/ddos/glossary/mirai-botnet/ (visited on 01/29/2024).

[5] K. Virgil English, Islam Obaidat, and Meera Sridhar. "Exploiting Memory Corruption Vulnerabilities in Connman for IoT Devices". In: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2019, pp. 247–255. DOI: 10.1109/DSN.2019.00036.

[6] Esphome. *UART*. URL: https://esphome.io/components/uart.html (visited on 01/29/2024).

[7] GigaDevice. *GD32VF103*. URL: https://www.gigadevice.com.cn/Public/Uploads/uploadfile/files/20230228/GD32VF103_Datasheet_Rev1.7.pdf (visited on 01/29/2024).

[8] Google. *blocl indexing*. Dec. 2023. URL: https://developers.google.com/search/docs/crawling-indexing/block-indexing (visited on 01/29/2024).

[9] Thread Group. *Thread*. URL: https://www.threadgroup.org/BUILT-FOR-IOT/Smart-Home (visited on 01/29/2024).

[10] Low Level Learning. *i hacked my son's baby monitor, for science*. URL: https://www.youtube.com/watch?v=qWzXYbCH3Ls (visited on 01/29/2024).

[11] Nataliia Neshenko et al. "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations". In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2702–2733. DOI: 10.1109/COMST.2019.2910750.

[12] Juniper Networks. *Secondhand IoT Devices, Firsthand Threats to Security*. Oct. 2018. URL: https://blogs.juniper.net/en-us/security/secondhand-iot-devices-firsthand-threats-to-security (visited on 01/29/2024).

[13] OASIS. *MQTT Version 5.0*. Mar. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf (visited on 01/29/2024).

[14] Samsung. *Ballie*. Jan. 2024. URL: https://news.samsung.com/global/video-ces-2024-a-day-in-the-life-with-ballie-an-ai-companion-robot-for-the-home (visited on 01/29/2024).

[15] Offensive Security. *Exploit Database*. Jan. 2024. URL: https://www.exploit-db.com/google-hacking-database?category=13 (visited on 01/28/2024).

[16] Michael Steigerwald. *Smart Home - Smart Hack*. Dec. 2018. URL: https://media.ccc.de/v/35c3-9723-smart_home_-_smart_hack (visited on 01/28/2024).

[17] Moonware Studios. *webcamXP - Webcam and Network Camera Surveillance Software*. Sept. 2016. URL: https://www.webcamxp.com/home.aspx (visited on 01/28/2024).

[18] Asma Terkawi and Nisreen Innab. "Major Impacts of Key Reinstallation Attack on Internet of Things System". In: *2018 21st Saudi Computer Society National Computer Conference (NCC)*. 2018, pp. 1–6. DOI: 10.1109/NCG.2018.8593071.

[19] Cihangir Tezcan. "Brute Force Cryptanalysis of MIFARE Classic Cards on GPU". In: *International Conference on Information Systems Security and Privacy*. 2017. URL: https://api.semanticscholar.org/CorpusID:21182728.

[20] Google Thread Group. *OpenThread*. URL: https://openthread.io/ (visited on 01/29/2024).

[21] WiFi-Alliance. *WiFi-Security*. URL: https://www.wi-fi.org/discover-wi-fi/security (visited on 01/29/2024).

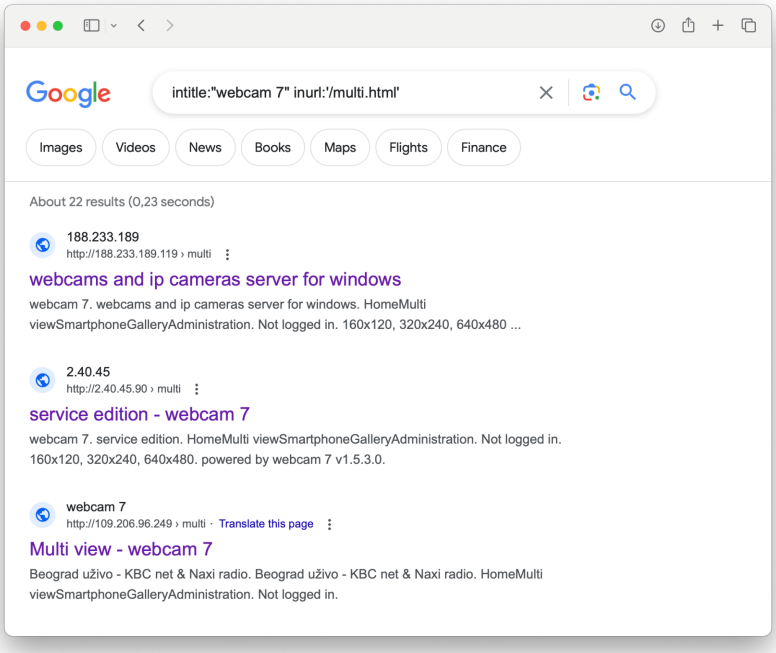
[22] Flipper Zero. *Addition new remotes*. URL: https://docs.flipper.net/sub-ghz/add-new-remote (visited on 01/29/2024).

[23] Flipper Zero. *Sub-GHz*. URL: https://docs.flipper.net/sub-ghz (visited on 01/29/2024).

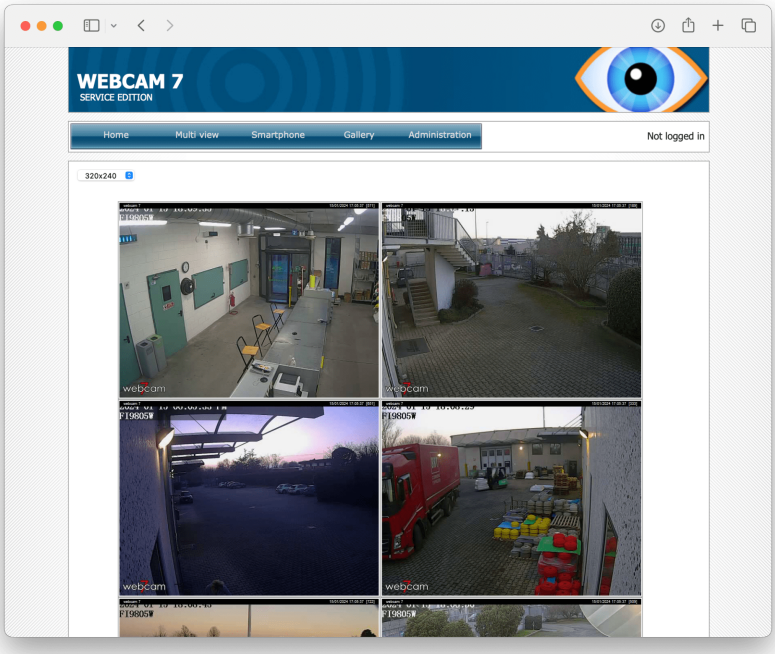
[24] Inc Zigbee Alliance. *zigbee Specification Revision 22 1.0*. Apr. 2017. URL: https://csa-iot.org/wp-content/uploads/2022/01/docs-05-3474-22-0csg-zigbee-specification-1.pdf (visited on 01/28/2024).

Appendices

Appendix I: Supplementary Illustrations

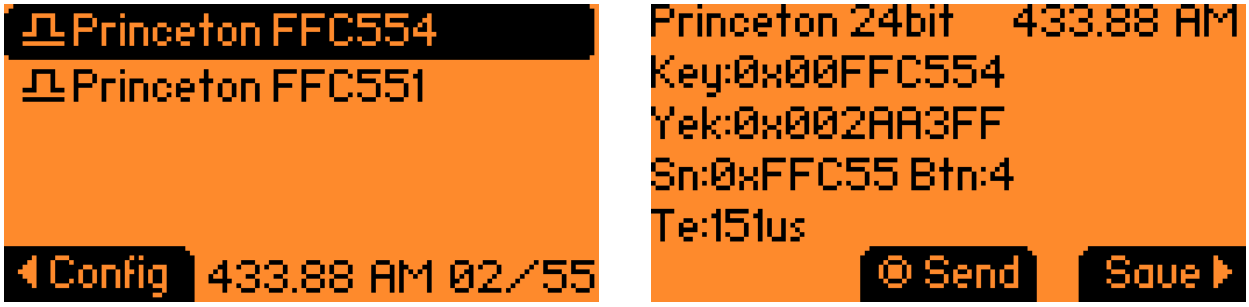


(a)



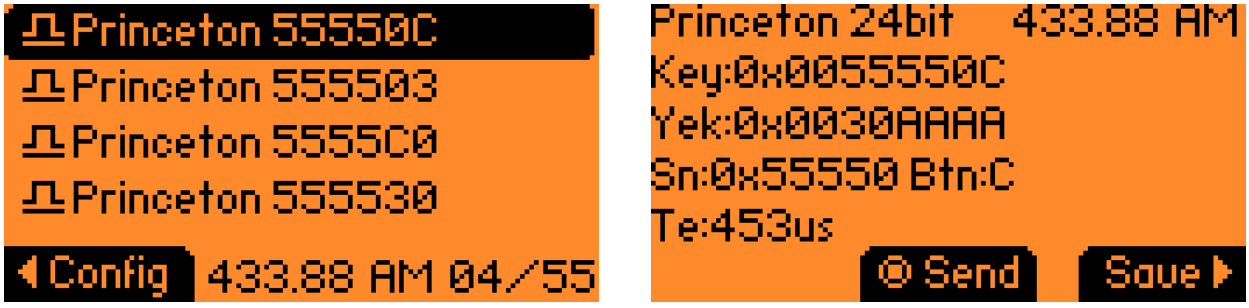
(b)

Figure 1: **Google Dorking for IoT Devices**: Screenshots outlining the process of identifying accessible IoT devices through Google Dorking. (a) A list of Google search results for the query "intitle:'webcam 7' inurl:'/multi.html' ", demonstrating the method's effectiveness in finding exposed devices. (b) The camera panel interface the aforementioned company, showing a meeting room, multiple backyards, and even trucks with clearly readably license plates.



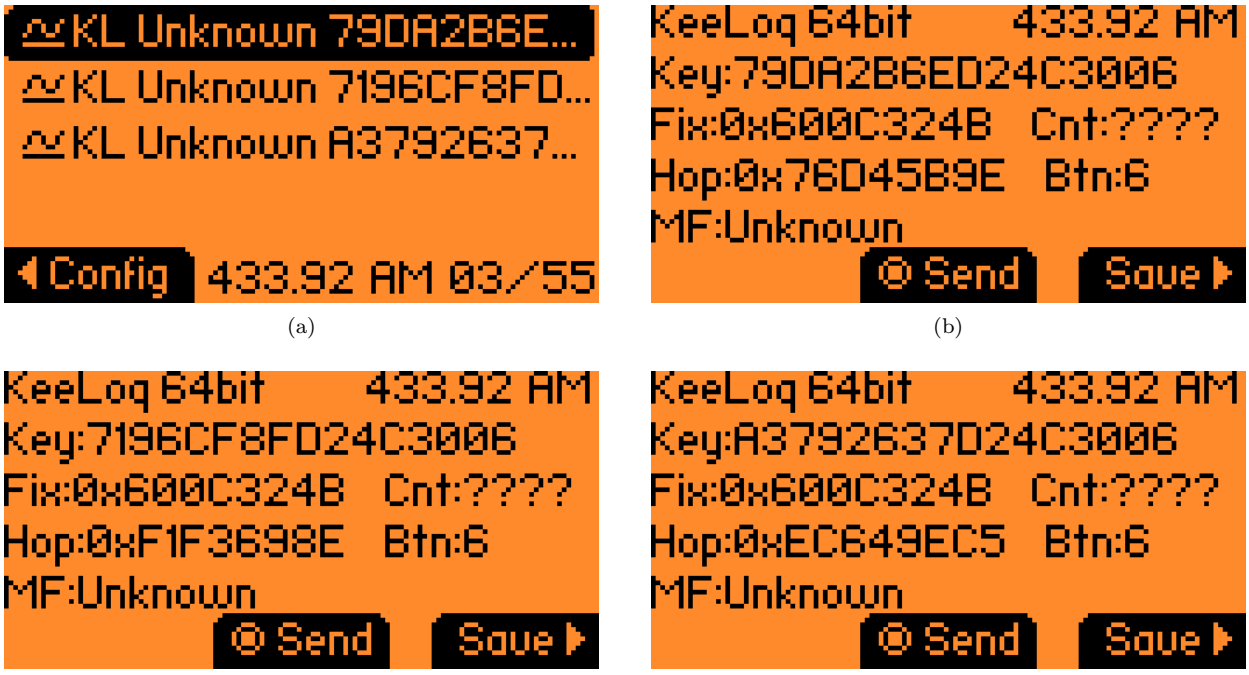
(a) (b)

Figure 2: **Signal Analysis for Remote Control Socket**: These screenshots from the Flipper Zero show Sub-GHz signal analysis of a remote control socket. (a) Features the "on" (Princeton FFC554) and "off" (Princeton FFC551) signals at 433.88 AM. (b) Provides details of the 'on' signal, showing protocol, frequency, key, and signal information.



(a) (b)

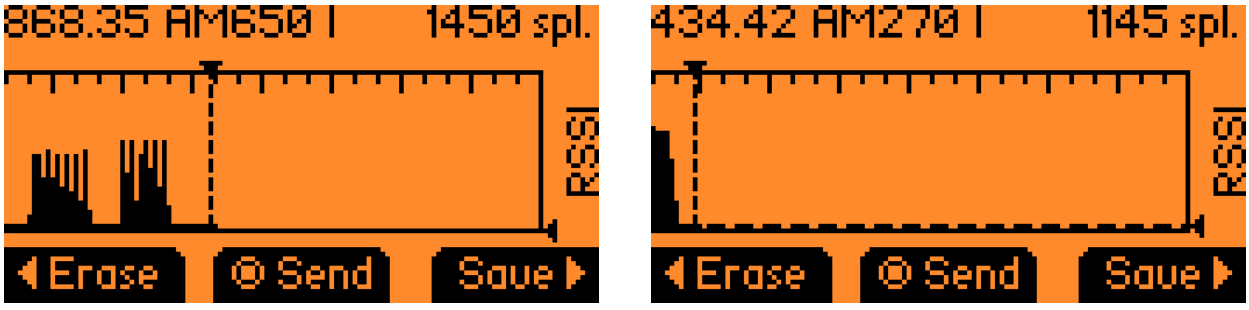
Figure 3: **Signal Analysis for DJPOWER Fog Machine**: These images present the signal analysis of a DJPOWER Effect Fog Machine's remote using Flipper Zero. (a) Shows the corresponding signals for all four buttons of the remote, labeled A (Princeton 555500), B (Princeton 555503), C (Princeton 555508), and D (Princeton 555530). (b) Details the signal for button A, showing the "Princeton 24bit" and detailing protocol, frequency, signal strength, button identification, and pulse timing.



(a) (b)

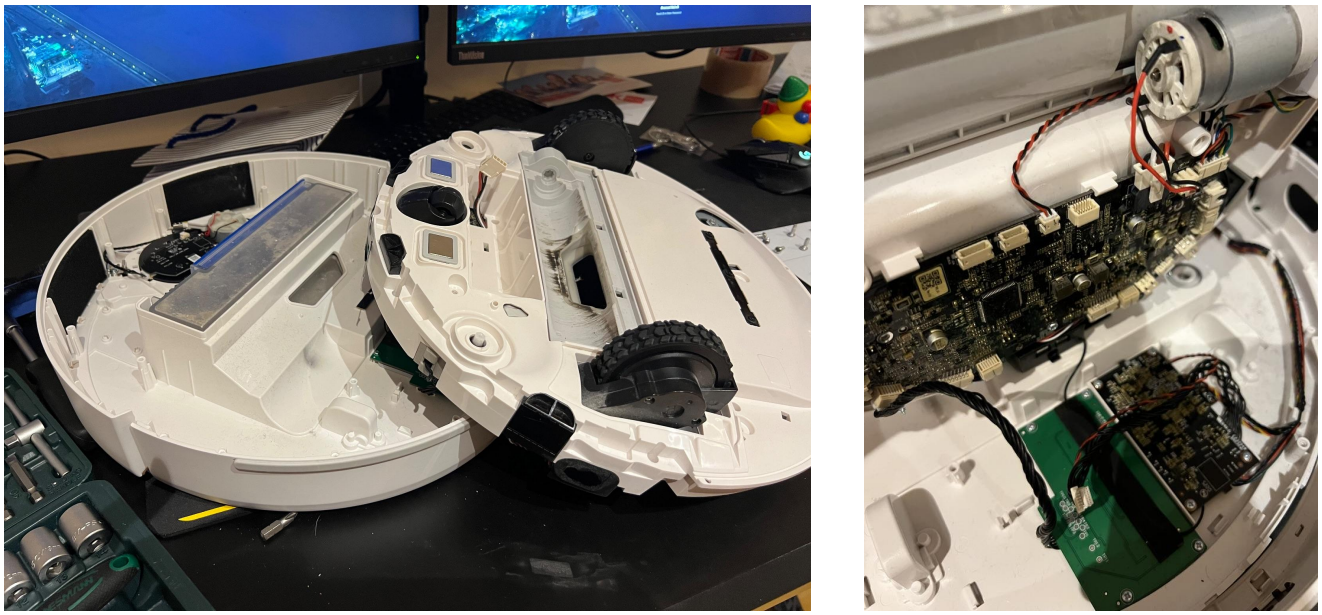
(c) (d)

Figure 4: **Signal Analysis for Courtyard Gate**: These images display the Flipper Zero's signal analysis of a courtyard gate's remote. (a) Shows three consecutively recorded codes, labeled as "KL Unknown" (KeeLoq with unknown manufacturer), with different key values. (b), (c), and (d) detail the characteristics of these three codes, like the use of the 'KeeLoq 64bit' protocol at 433.92 AM. Notably, the "Fix" component remains constant across the codes, while the "Key" and "Hop" values change with each signal.



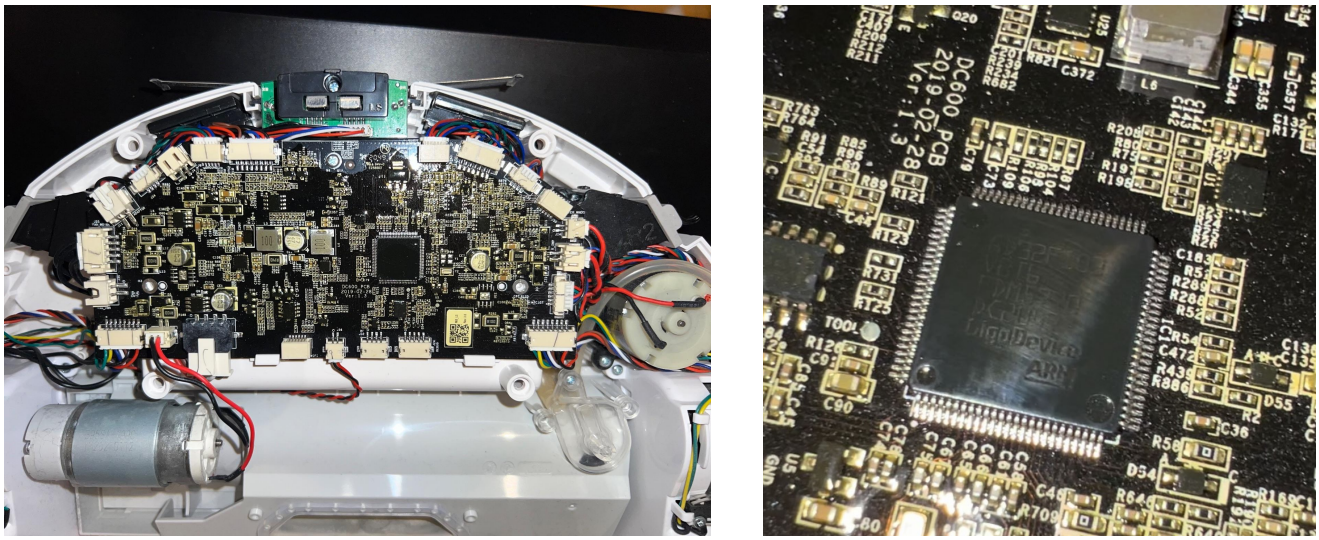
(a) (b)

Figure 5: **Sub-GHz Raw Signal Captures**: This figure illustrates the raw signal captures from the Flipper Zero for two different scenarios. (a) Displays the captured communication between a remote and an alarm system, where the initial signal peaks represent the transmission from the remote, followed by the response from the alarm system. (b) Shows the captured signal of a VW Beetle car remote, which was successfully used to unlock the vehicle, demonstrating the vulnerability in its remote locking system.



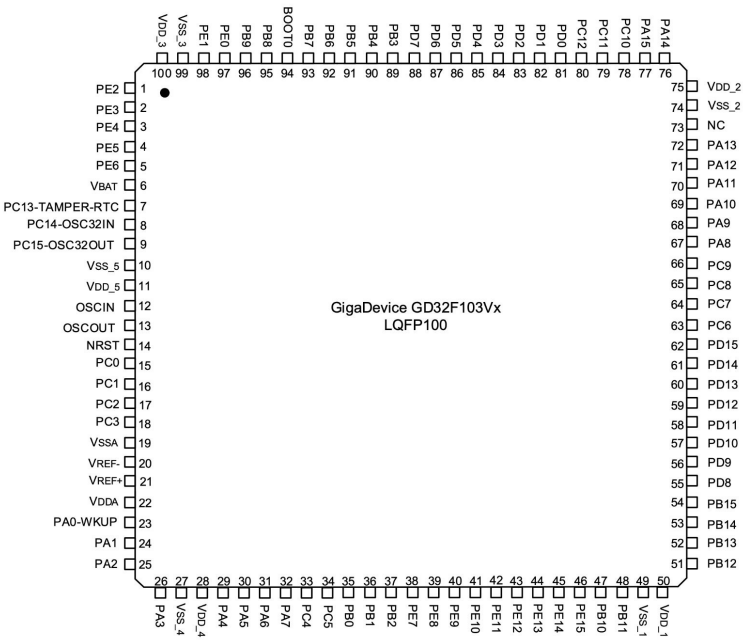
(a) (b)

Figure 6: **Robovac Disassembly Process**: (a) The Robovac split into top and bottom halves immediately after screw removal, still connected by internal cables. (b) Detailing the two cables that need to be disconnected from the main PCB to separate the halves.



(a) (b)

Figure 7: **Robovac PCB and Chip Analysis**: (a) Top-down view of the Robovac's main PCB, highlighting key components and layout. (b) Close-up of the illuminated chip with inscription, identifying it as a "GD32VF103R6T6".



(a)

PA8	67	I/O	5VT	Alternate: USART0_CK, TIMER0_CH0, CK_OUT0, USBFS_SOF
PA9	68	I/O	5VT	Default: PA9 Alternate: USART0_TX, TIMER0_CH1, USBFS_VBUS
PA10	69	I/O	5VT	Default: PA10 Alternate: USART0_RX, TIMER0_CH2, USBFS_ID
PA11	70	I/O	5VT	Default: PA11 Alternate: USART0_CTS, CAN0_RX, USBFS_DM, TIMER0_CH3
PA12	71	I/O	5VT	Default: PA12 Alternate: USART0_RTS, USBFS_DP, CAN0_TX, TIMER0_ETI
PA13	72	I/O	5VT	Default: JTMS Remap: PA13
NC	73			-
VSS_2	74	P		Default: VSS_2
VDD_2	75	P		Default: VDD_2
PA14	76	I/O	5VT	Default: JTCK Remap: PA14
PA15	77	I/O	5VT	Default: JTDI Alternate: SPI2_NSS, I2S2_WS Remap: TIMER1_CH0_ETI, PA15, SPI0_NSS
PC10	78	I/O	5VT	Default: PC10 Alternate: UART3_TX Remap: USART2_TX, SPI2_SCK, I2S2_CK
PC11	79	I/O	5VT	Default: PC11 Alternate: UART3_RX Remap: USART2_RX, SPI2_MISO
PC12	80	I/O	5VT	Default: PC12

(b)

Figure 8: **Robovac Chip Pinout and JTAG Interface Analysis**: (a) Diagram from the documentation, showing the full pinout of the chip. (b) Pinout table from the documentation with PA13 and PA14 highlighted, which are JTMS and JTCK connections.