A Security Analysis of Radio Remote Controllers for Industrial Applications For Blackhat Asia 2019

Jonathan Andersson, Marco Balduzzi, Stephen Hilt, Philippe Lin,
Federico Maggi, Akira Urano, Rainer Vosseler

Trend Micro Inc.

Contents

1. Remote Controllers for Industrial Applications

Radio frequency (RF) remote controllers for industrial applications appear as rugged, colorful remotes with multiple buttons or joysticks. They are used for control and automation purposes in various "heavy industry" sectors like mining, material handling, and construction.

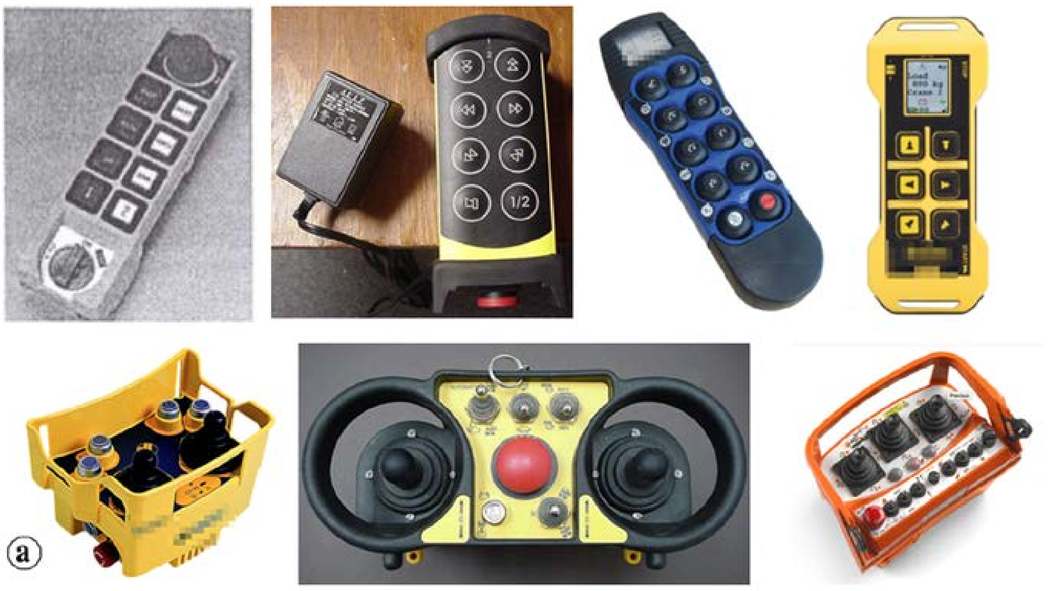


Figure 1. Evolution of hand-held and belt-pack remote controllers (main steps from 1997 to 2018), as seen in photos from user manuals and FCC documentation

Apart from the safety requirements, some of which are International Organization for Standardization (ISO) standards and requirements that can vary from country to country, we are not aware of any standardized architecture for industrial remote controllers. However, after carefully inspecting the technical documentation of the product lines of 17 vendors, we conclude that there are mainly three broad classes of remote controllers (pocket-sized, hand-held, belt-pack), which vary in size and number of buttons or joysticks. They can have capabilities ranging from carrying out simple open-close functions to implementing sophisticated variable-power actions, some of which require special training to be operated. Interestingly, one of the vendors that we have encountered even offers an authentication feature, which includes a role-based authorization system that dynamically enables or disables advanced functionalities depending on the operator's skills. Overall, however, remote controllers work similarly. The operator uses a transmitter (TX) to control pieces of machinery attached to the corresponding receiver (RX). Each command on the TX results in a distinct RF emission; the RX, configured on the same RF channel, demodulates it, and reacts to it, for example, by moving a load upward.

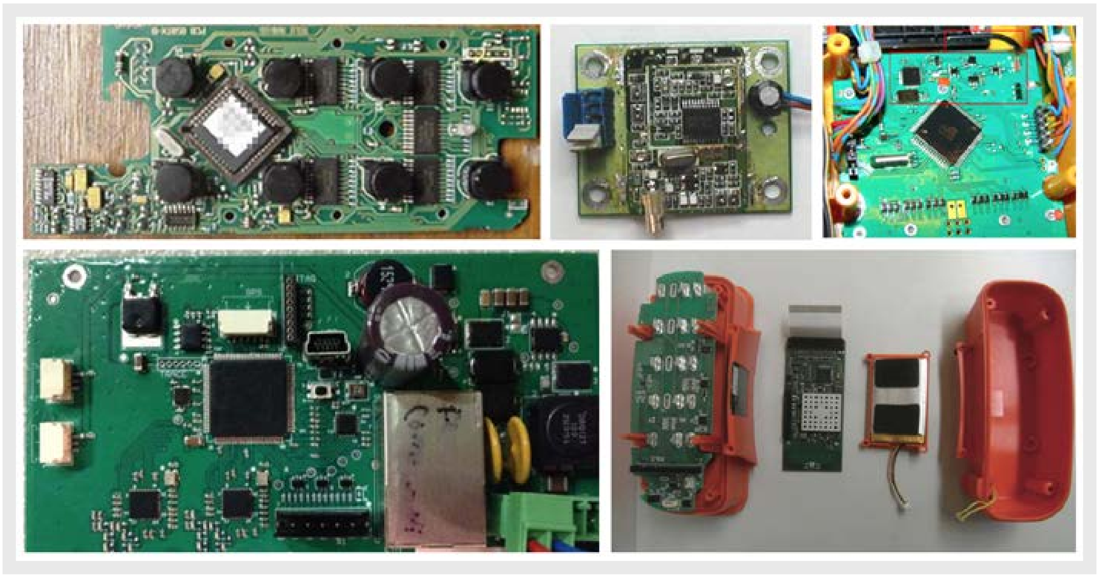


Figure 2. Evolution of the circuitry of industrial radio remote controllers (TX and RX) (main steps from 1997 to 2018), as seen in photos from user manuals and FCC documentation

1.1 Radio Protocols 101: Data Encoding and Modulation

Industrial radio remote controllers operate a set of remotely triggered relays, which activate various equipment. Most of the time, each relay is electrically connected to an actuating device such as a servo motor, siren, or other machinery that affects the physical world. Pushing a button on the remote triggers a certain relay.

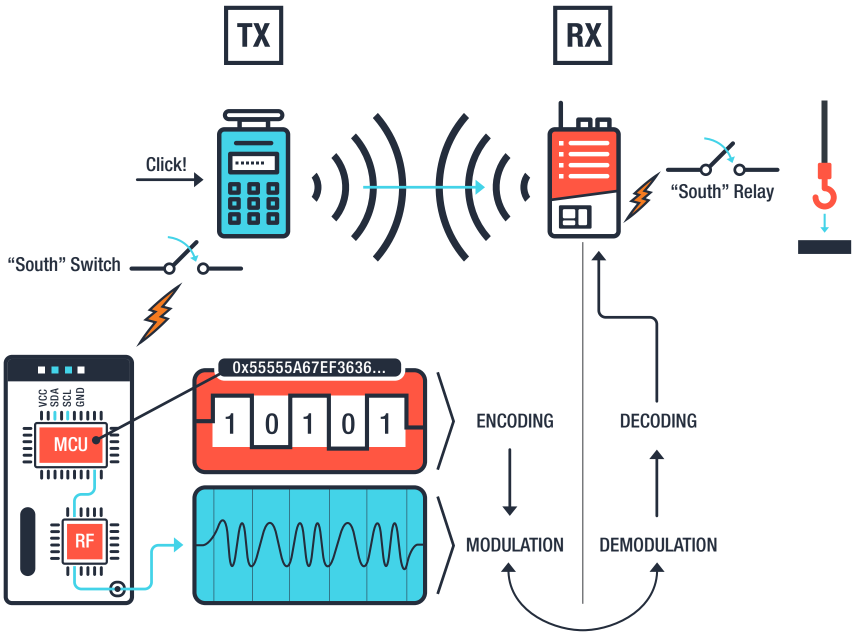


Figure 3. Schematized connection between a transmitter (TX, remote), receiver (RX), and controlled equipment

Industrial radio remote controllers usually operate in the ISM (industrial, scientific, and medical) bands, in which the most common carrier frequencies are 315, 433, 868, and 915 megahertz (MHz). Although the actual frequency band can vary from country to country, the ISM bands are internationally reserved for industrial, scientific, and medical requirements, and not for communications. More recently, industrial radio remote controllers in the 2.4 gigahertz (GHz) band are emerging, mainly as a solution to the overcrowded 315, 433, 868, and 915 MHz bands. In this research, we focus explicitly on ISM bands, given their popularity and worldwide pervasiveness.

On these sub-gigahertz carrier frequencies, a radio transceiver implements an RF modem, using a modulation scheme like frequency-shift keying (FSK), phase-shift keying (PSK), minimum-shift keying (MSK), or one of the many variants. Modulation tells how digital data (bits) is represented in the analog domain of radio waves (e.g., "lower frequency" means "zero," "higher frequency" means "one"). While this is interesting, this only describes the physical layer of the communication and does not say much about the application layer. Most of the time, the technical manual of an RF device indicates the modulation scheme because it's not considered a secret, unlike the application layer.

By way of description for the application layer, take for example when a button is pressed on the transmitter (e.g., "SOUTH"). This action triggers a set of instructions in a microcontroller, which in turn encodes and writes sequences of symbols on a bus representing the command. Note that to ensure reliable over-the-air communications (e.g., in case of transmission errors), these sequences of symbols are first encoded to increase redundancy by using dedicated codes (e.g., the Manchester code[1] is based on XORing, or

Exclusive ORing, the data with a clock, but there are several other line codes). On the RX side, the process is the same, just in reverse order: Given the parameters of the radio protocol (e.g., modulation scheme and bit length), the radio waves are demodulated to obtain the same sequences, which represent the actual encoded command. At this point, the microcontroller decodes and interprets them, finally writing the correct logic value on the correct gate, causing a relay to open or close. The way the commands are represented as sequences of symbols and the way symbols are encoded are part of the application layer.

On top of this basic architecture, vendors have evolved their product lines to incorporate more complex functionalities. However, this research focuses on the RF subsystem as described above. Over the years, vendors have shifted from embedding the entire RF circuitry onto the main printed circuit board (PCB) to a modular design, with a main board and a detachable RF daughterboard connected to the main board via a serial interface (e.g., Serial Peripheral Interface or SPI).

1.2 Typical Applications and Deployments

Radio remote controllers have a wide variety of applications, with an important aspect in common: They are all safety-critical applications. The following image provides a good bird's-eye view of the many industrial applications.

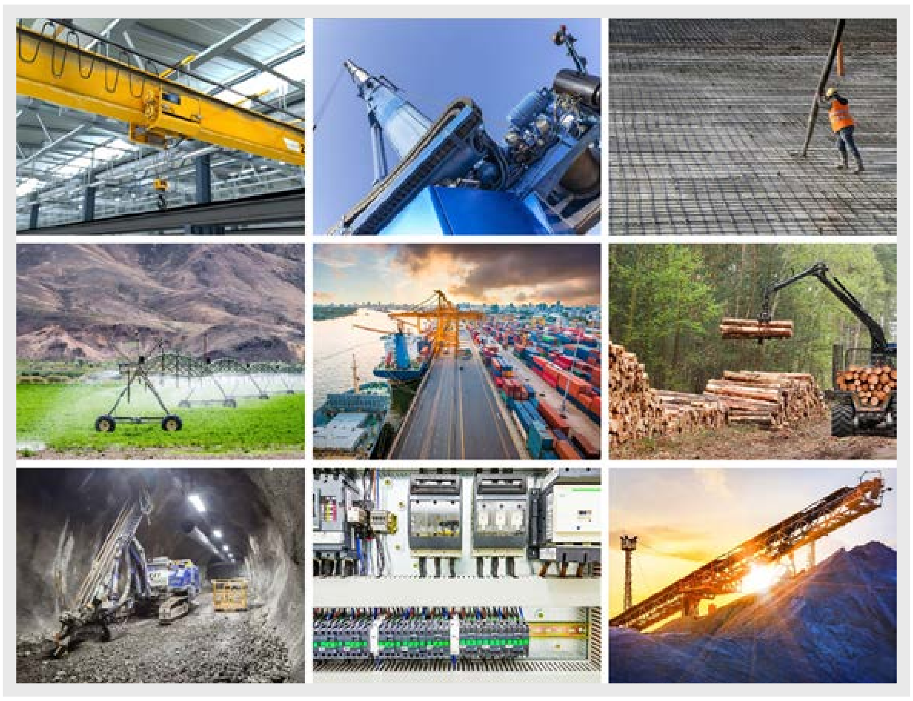


Figure 4. Overview of the application sectors, common across most vendors

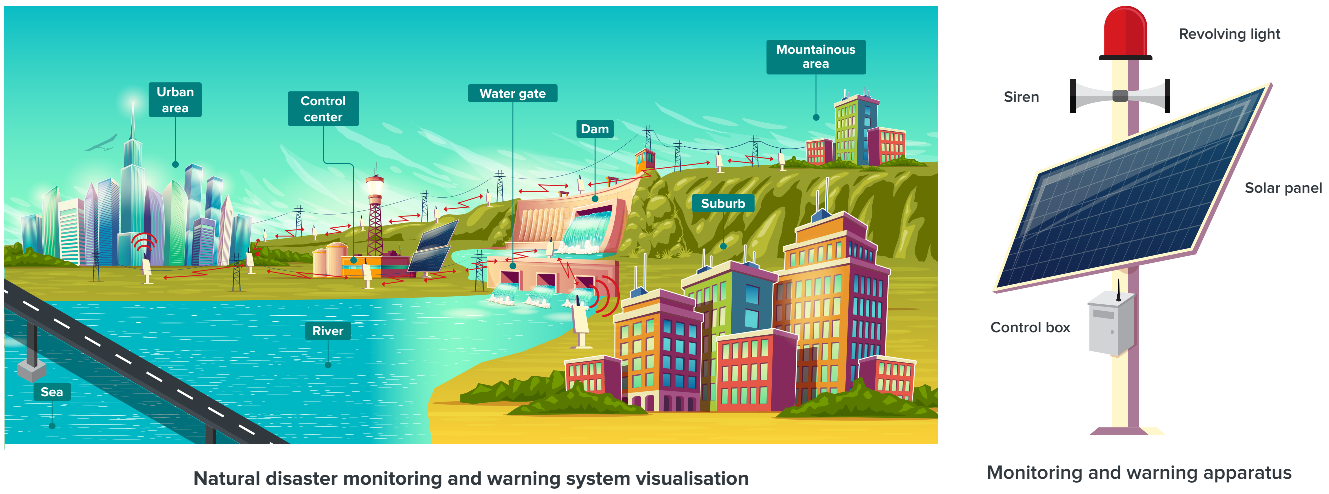


Figure 5. RF links used for natural disaster monitoring and warning systems, based on a figure from a technical information document by Circuit Design[2]

1.3 Size of the Problem: A Look at the Supply Chain

This research was motivated by three aspects: the use of proprietary RF protocols, the number of controllers currently deployed, and their long life cycle. Let's set aside the issue involving proprietary RF protocols momentarily. Assessing the number of controllers currently deployed is not trivial. As an example, driving randomly within an area of 10-km radius, we were able to find more than a dozen construction sites using these remotes, eight of which allowed us to run tests on their equipment. While it's not trivial to perform such a search at a large scale and at a fine-grained level, good indirect indicators are the number of vendors, their longevity and business size, and the number of facilities that likely employ these controllers (e.g., manufacturing sites, harbors, and construction sites), for which country-level statistics exist. Also, we need to know the life cycle of these remote controllers, to understand how easy (or hard) it is to upgrade them in order to fix the issues that we found.

1.3.1 Distribution of Industrial Radio Remote Controllers

Radio remote controllers for industrial applications are distributed across the world through vast sales networks, as revealed by the market research we conducted. This shows an ecosystem dominated by a dozen large vendors with long company histories. Most of them have deposited technical documentation to the Federal Communications Commission (FCC), which is made publicly available through its website and is searchable via third-party tools such as FCC ID.io.[3] Note that the FCC website is supposed to contain only nonconfidential material. However, some vendors have uploaded pictures of their devices' internals, which allow reconnaissance of the microcontroller and, more important, the radio transceiver.

The following table pertaining to the 17 vendors that we considered in our research provides a rough idea of the size of this market, providing indicators of the number of controllers installed or purchased worldwide.

On FCC ID.io since	Vendor (establishment)	Company fact	Price of controller (US$)	Headquarters	Distribution
2000	Autec (1986)[4]	€17.8 million annual revenue	300–1,300	Italy	Worldwide
1995	Hetronic (1982)[5]	410,000 installed systems	500–1,500	U.S.	Worldwide
1997	Saga (1997)[6]	Products sold in at least 35 countries	800–1,500	Taiwan	Worldwide
1974	Circuit Design (1974)[7]	US$18 million annual turnover	100–300	Japan	Worldwide
2014	Elca (1991)[8]	€2-5 million annual revenue	500–1,500	Italy	Worldwide
1992	Telecrane (1985)[9]	60,000 units sold per year	N/A	Taiwan	Worldwide
2013	Juuko (1994)[10]	Products sold in at least 40 countries	150–800	Taiwan	Worldwide
1998	HBC-radiomatic (1947)[11]	420 employees	200–1,000	Germany	Worldwide (60 percent export)
1982	Cattron (1946)[12]	125,000 installed systems	200 (used)	U.K.	Worldwide
1999	Tele Radio (1955)[13]	286,000 units sold in 2016	600–2,000	U.S.	Worldwide
1998	Scanreco (1980)[14]	30,000 units sold per year	N/A	Sweden	Worldwide
2002	Shanghai Techwell Auto-control Technology (2005)	10,000 units sold per year	N/A	China	China
1982	Remote Control Technology (1982)	Boeing and the U.S. military among its customers	N/A	U.S.	U.S.
1999	Akerstroms (1918)[15]	ABB Robotics and SAAB among its customers Uses RF transceiver from Circuit Design	400–1,000	Sweden	Worldwide
1999	Jay Electronique (1962)[16]	N/A	700–1,500	France	Worldwide
1986	Itowa (1986)[17,18]	N/A	N/A	Spain	Worldwide
2001	3-Elite (1995)[19]	Products sold in at least 25 countries	500–800	Taiwan	Worldwide

Table 1. List of vendors that we considered in our market research

After looking at what the market has to offer, we looked at the market demand for industrial remote controllers to complete the picture. We considered the question: *How, and where is it more likely to find these devices in the real world?*

Country-wise statistics about the number of manufacturing facilities, transportation hubs, and construction sites are very good indicators.

According to a recent research conducted by the World Economic Forum, the construction industry accounts for 6 percent of the global gross domestic product (GDP).[20] Clearly, there is no end-to-end connection between the vendors listed above and such numbers, but they give an aggregated indication of how widespread these devices are.

1.3.2 Supply Chain: The Long and Lonely Life of a Radio Module

Radio remote controllers for industrial applications follow a very straightforward life cycle. They're assembled from the vendor, configured — usually by system integrators — according to the clients' requests (e.g., which button does what action), installed at the facilities or equipment, and forgotten until they break down.

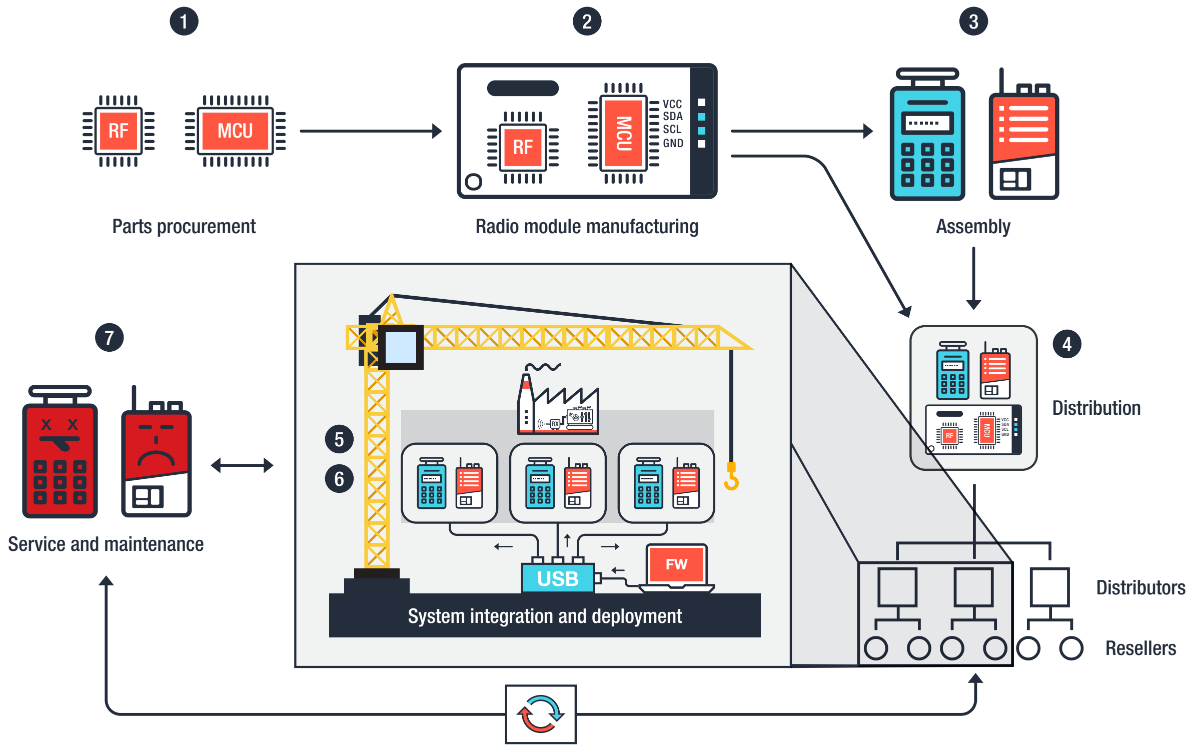


Figure 6. Supply chain flowchart of industrial radio remote controllers

Radio remote controllers go through the following phases in the supply chain:

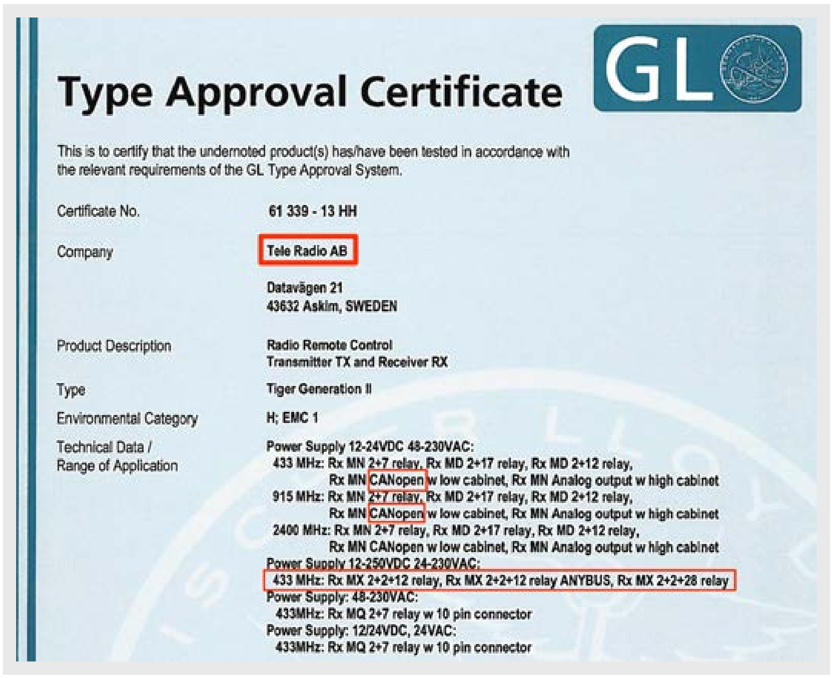
1. **Parts procurement.** Radio remote controllers can be quite complex, but in this research we are mainly interested in two core components, namely, the radio transceiver (RF) and the microcontroller (MCU). The vendors likely acquire the radio transceivers and the microcontrollers from semiconductor and integrated-circuit manufacturers.
2. **Radio module manufacturing.** From the information available on the public websites of the vendors, we understood that some vendors manufacture the radio modules in-house, while others outsource the production entirely. To some extent, the vendors have to develop or maintain a certain level of control over the firmware that is running on the microcontrollers, because they must be able to reprogram the final device according to the customers' needs. For example, we found at least one vendor (Akerstroms) that internally uses Circuit Design's RF transceivers (see the internal photos from the FCC website[21]).
3. **Assembly.** From the information available on the public websites of the vendors, we understood that most of the vendors assemble the radio controller units (TX and RX) into their product lines in-house.
4. **Distribution.** Most of the vendors sell worldwide, directly from their local business units, or through one to two levels of distribution and reselling. We have encountered small, national vendors that sell exclusively through local agents as well as large, international vendors that sell directly.
5. **System integration.** System integrators are often the suppliers of the controlled machines. For example, crane or hoist manufacturers provide the final products with the controllers installed and configured, ready to be used. Also, we have encountered system integrators that rent these parts, or the final products, given their high costs. For example, an indoor overhead crane capable of lifting 50 tons can cost several hundred thousands of U.S. dollars. Typically, system integrators are also able, directly or indirectly, to configure options on the firmware.
6. **Deployment.** Finally, the controlled equipment is deployed, and final users are trained to operate it. Some vendors (e.g., Juuko, Hetronic, Saga, Telecrane) allow the final users to configure options on the firmware.
7. **Service and maintenance.** Given the size and distribution of the sales networks of most of the vendors, the system integrators are usually able to provide service and maintenance, including — when feasible — firmware upgrades. However, upgrading the firmware on these devices requires reaching the RX unit (which is normally deployed on "the top" of the controlled machine), opening the enclosure, attaching a serial cable to a programmer, and updating the microcontroller with the new firmware. Most of these systems are designed to be heavy-duty, to require minimal maintenance, and to simply work. We confirmed this by talking with the operators of the industrial and construction facilities that we visited. A notable example was an overhead crane installed at a factory in 1997 with a wired pendant, which was replaced in 2012 with an RF one, and has never been touched since then.

Considering this supply chain, and based on our experience in the field, patching the devices is feasible. Some vendors have already implemented necessary countermeasures and made software updates available.

Now we can raise the question: *Are these vendors adopting proper security practices to keep their computer systems secure in order to minimize the possibility of attackers reaching the computers used to program the firmware of the remotes?*

1.4 Internet-Connected Radio Remote Controllers

That one of the largest crane vendors, Konecranes, is already using the term "smart cranes" suggests that future cranes will be more connected and equipped with technology from the information technology (IT) space. Although we can't be sure as to what devices are behind the 20 open Virtual Network Computing (VNC) servers that we found on Shodan by searching for the keyword "crane," there are several options to bridge industrial RF receivers with computer networks. For example, several vendors offer Anybus' CANopen, Profibus, or Modbus TCP interfaces on their receivers. These are particularly useful in mobile scenarios (e.g., mobile hoists, dump trucks) or in general when the controlled machines can be controlled via any of these standard buses. Remarkably, one of the leading vendors, Tele Radio, has an entire line of receivers that bridge RF and Anybus' CANopen.[22]



Type Approval Certificate

GL

This is to certify that the undernoted product(s) has/have been tested in accordance with the relevant requirements of the GL Type Approval System.

Certificate No.	61 339 - 13 HH
Company	Tele Radio AB
	Datavägen 21 43632 Askim, SWEDEN
Product Description	Radio Remote Control Transmitter TX and Receiver RX
Type	Tiger Generation II
Environmental Category	H; EMC 1
Technical Data / Range of Application	Power Supply 12-24VDC 48-230VAC: 433 MHz: Rx MN 2+7 relay, Rx MD 2+17 relay, Rx MD 2+12 relay, Rx MN CANopen w low cabinet, Rx MN Analog output w high cabinet 915 MHz: Rx MN 2+7 relay, Rx MD 2+17 relay, Rx MD 2+12 relay, Rx MN CANopen w low cabinet, Rx MN Analog output w high cabinet 2400 MHz: Rx MN 2+7 relay, Rx MD 2+17 relay, Rx MD 2+12 relay, Rx MN CANopen w low cabinet, Rx MN Analog output w high cabinet Power Supply 12-250VDC 24-230VAC: 433 MHz: Rx MX 2+2+12 relay, Rx MX 2+2+12 relay ANYBUS, Rx MX 2+2+28 relay Power Supply: 48-230VAC: 433MHz: Rx MQ 2+7 relay w 10 pin connector Power Supply: 12/24VDC, 24VAC: 433MHz: Rx MQ 2+7 relay w 10 pin connector

Figure 7. Receivers by Tele Radio have multiple connectivity options, including Anybus' CANopen

There are also customization features that allow end users to program the hand-held remotes through a simple USB cable. One of the vendors that we contacted specifically mentioned multiple inquiries from its clients, which wanted to **remove the need for physically pressing the buttons** on the hand-held remote, replacing this with a computer, connected to the very same remote that will issue commands as part of a more complex automation process, with no humans in the loop.

As a general trend, we observe that vendors want to offer their customers more control of their devices after the purchase, leaving them independent and free to customize the remotes. For example, Telecrane Italia is promoting and selling a tablet with the programming software pre-installed and pre-configured.[23]

In conclusion, considering all the factors, we believe that **computer-borne attacks are possible**. A truly remote attacker, who has taken control of the computer used to software-program or -control these remotes, can alter the functionality of the devices, thus implementing more persistent and sophisticated attacks. With the increasing interconnectivity promoted by the Industry 4.0 shift, we expect these attacks to become more prevalent and connected, and we thus believe that they are worth inspecting.

2. Safety and Security Features

By inspecting both the controllers and the available documentation (either from manuals or from the FCC ID.io website), we have compiled a list of safety features. We acknowledge that these features are actually safety features, in the sense that they're not meant for security or designed with a security mindset. We do recognize that they happen to offer some form of security — as a positive side effect — in the sense that they raise the difficulty of some attacks but do not completely prevent them.

Note: The criticality of the buttons can vary significantly. Simple "movement" buttons are inherently very different from "start" or "stop" buttons because the latter are meant to "enable" or "disable" the controlled machine. One would expect that special commands like the "start" and "emergency stop" buttons were encoded differently due to their criticality (for example, to make their transmission more resilient to errors). However, the reality is that we haven't found evidence in support of this, at least in the devices that we analyzed: Special commands are not treated differently from other commands.

Safety features are not designed with a cybersecurity mindset. Rather, they're meant to prevent injuries due to malfunction or unexpected external conditions. Upon analyzing the technical documentation of 17 vendors (and products from seven of them), we drew the following list of safety features.

Safety feature	Description	Issues prevented	Limitation
Pairing mechanism	Transmitter and receiver are paired with a (fixed) pairing code, which is used to recognize and accept commands only from known transmitters.	Interferences: Multiple transmitters (e.g. of the same model and brand) can work together in the same RF band.	Knowledge of the pairing code allows complete impersonation of a legitimate transmitter.
Passcode protection	The operator needs to enter a sequence (passcode) to operate the transmitter. This sequence enables the transmitter and starts the receiver.	Unwanted commands and unauthorized operators: Machinery can be controlled only upon entering the correct passcode.	Knowledge of the passcode allows anyone to use a transmitter.

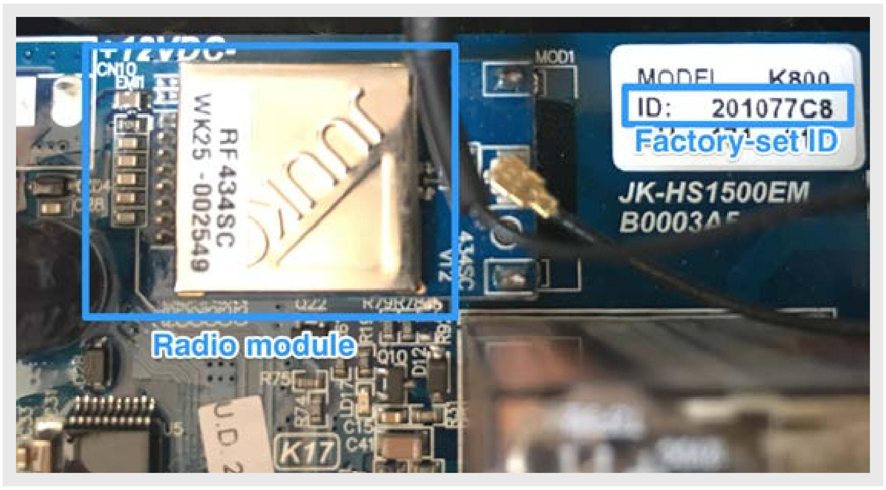
Safety feature	Description	Issues prevented	Limitation
Authorization	The transmitter implements an access control model that selectively enables or disables advanced features according to the level of the operator, who is identified using radio frequency identification (RFID) or an equivalent factor.	Inexperienced operators who might issue complex commands that could cause injuries.	RFID and equivalent factors can be stolen or cloned.
Virtual fencing	Transmitter and receiver communicate via an out-of-band channel (e.g., infrared) in addition to RF. When the transmitter is out of range, the receiver does not accept any commands.	Machines cannot be operated outside the "virtual fence" created by the out-of-band channel (e.g., the infrared range).	Knowledge of the out-of-band virtual fencing protocol allows mimicry of it.

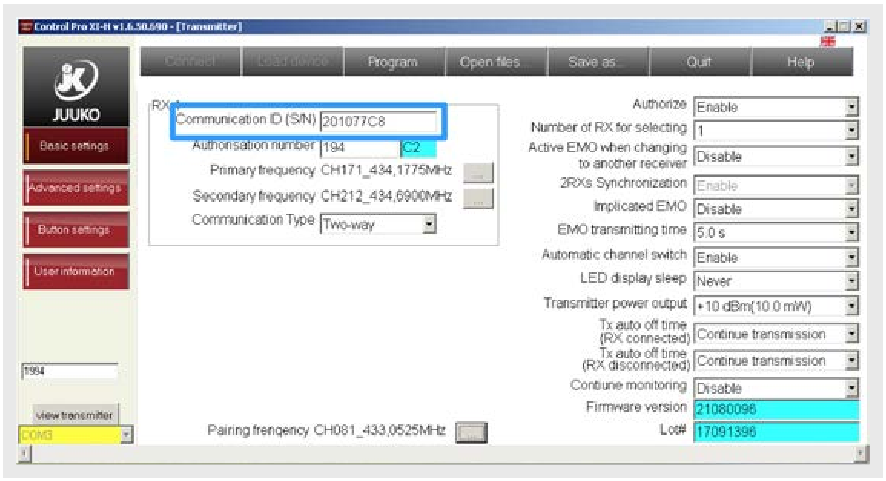
Table 2. Overview of the safety features implemented in radio remote controllers for industrial applications.

These safety features **do not prevent active attacks**, as they're not designed for that purpose.

2.1 Naïve (or No) Pairing Mechanism

All radio remote controllers are shipped with a pre-configured pairing code (the same on both transmitter and receiver). As seen later in the document, this code is properly encoded in the exchanged packets to avoid protocol-level interferences while working on the same frequencies. In some cases, this code is written on the device enclosure, which makes its identification trivial. Whenever such a code is used for mere TX identification on the RX side, this feature does not offer any security measure. All packets have this pairing information encoded in the same way, and the lack of rolling-code mechanisms makes these devices vulnerable to replay and other forms of attacks.





RX1
Communication ID (S/N) 201077C8
Authorisation number 194 C2
Primary frequency CH171_434,1775MHz
Secondary frequency CH212_434,6900MHz
Communication Type Two-way

Authorize Enable
Number of RX for selecting 1
Active EMO when changing to another receiver Disable
2RXs Synchronization Enable
Implicated EMO Disable
EMO transmitting time 5.0 s
Automatic channel switch Enable
LED display sleep Never
Transmitter power output +10 dBm(10.0 mW)
Tx auto off time (RX connected) Continue transmission
Tx auto off time (RX disconnected) Continue transmission
Contiune monitoring Disable
Firmware version 21080098
Lot# 17091396

1994

Pairing frenqency CH081_433,0525MHz

Figure 8. ID printed on the enclosure of a unit that we analyzed and on the programming software that allows the user to change it (consistently on TX and RX)

However, sometimes the user manuals could be misleading in the way they describe their "security features." For instance, one vendor, which we successfully verified to be vulnerable against replay and e-stop (emergency stop) abuse attacks, mentions that the pairing mechanism "prevents messages from other radio equipment from activating any system function," while our attacks have just proved otherwise.

2.2 Passcode-Protected Transmitter

A slightly more advanced security feature involves a passcode, or hardware key, on the TX unit. In recent devices, the vendor sets — or lets the client set — a special sequence of buttons to be pressed to enable the TX unit. Before that, any command is ignored. The TX must execute (i.e., transmit to the RX) a "start" command, effectively enabling — and sometimes, powering on — the controlled machine. In older devices, such a passcode is static, to the extent that an actual hardware key needs to be inserted and turned. With this feature, unauthorized personnel not in possession of the key will not be able to operate the device. Again, this is clearly meant as a safety feature, but if implemented correctly, could prevent an attacker from issuing movement commands before a proper "start." If this feature is absent, any pressure of a button on the TX will trigger a relay on the RX side.

From our assessment, we found out that the start sequence does not result in specially encoded transmissions. For example, it does not include authentication information in the transmitted packets, with which the RX could verify that the TX is not an impostor. So if the start sequence is just another simple command, then an impostor can power on the controlled machine without possessing the passcode sequence or hardware key. Note, however, that a proper way to secure radio remote controllers requires more than an authenticated and encrypted start sequence: All transmissions must be authenticated and encrypted.

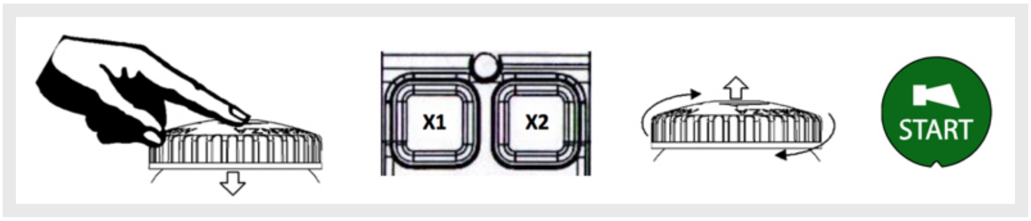
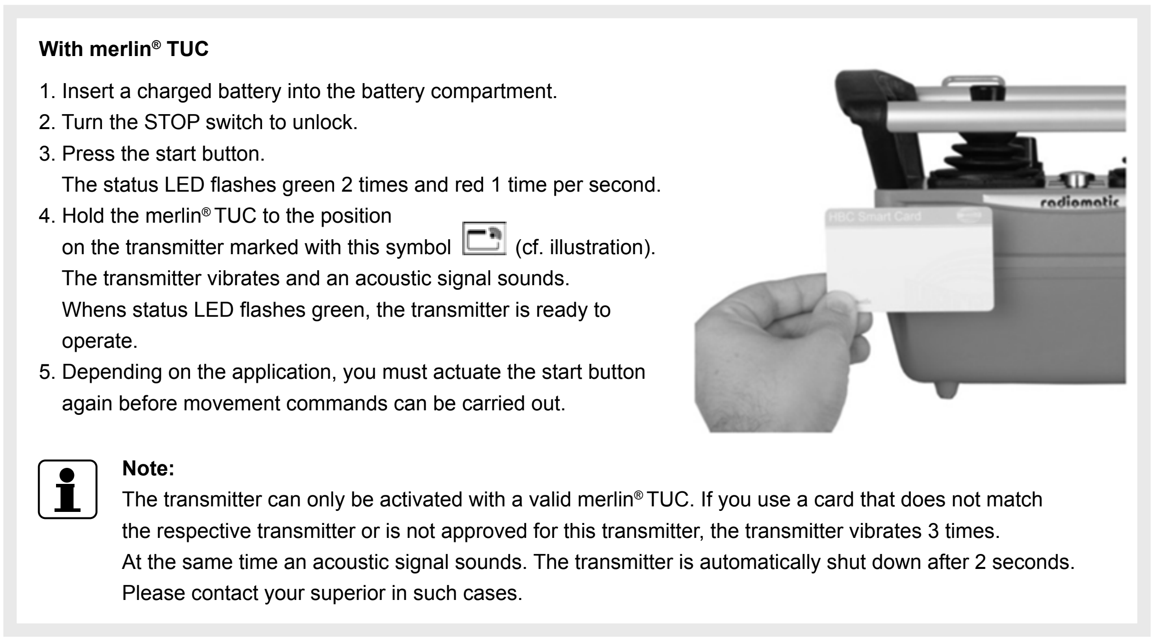


Figure 9. Default passcode sequence as described in the manual of a unit we analyzed, and as configurable via software[24]

2.3 RFID for User Identification

One of the vendors we considered, HBC-radiomatic, implements an interesting authentication and authorization functionality based on radio frequency identification (RFID). Each operator supposedly has a dedicated RFID card, which is read by the TX unit to enable the start function. In addition to this authentication system, the TX unit implements an authorization system that unlocks specific functionalities for each operator (e.g., according to their skill level).


With merlin® TUC

1. Insert a charged battery into the battery compartment.
2. Turn the STOP switch to unlock.
3. Press the start button.
   The status LED flashes green 2 times and red 1 time per second.
4. Hold the merlin® TUC to the position
   on the transmitter marked with this symbol (cf. illustration).
   The transmitter vibrates and an acoustic signal sounds.
   Whens status LED flashes green, the transmitter is ready to operate.
5. Depending on the application, you must actuate the start button again before movement commands can be carried out.

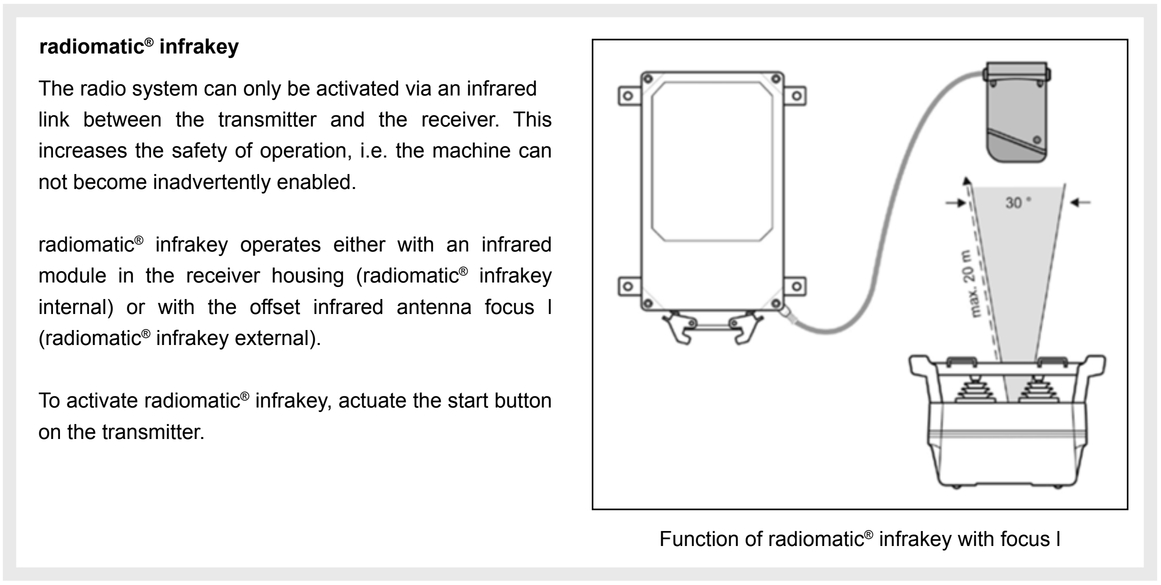
Note:
The transmitter can only be activated with a valid merlin® TUC. If you use a card that does not match the respective transmitter or is not approved for this transmitter, the transmitter vibrates 3 times.
At the same time an acoustic signal sounds. The transmitter is automatically shut down after 2 seconds.
Please contact your superior in such cases.

Figure 10. RFID-based user authentication functionality as described in HBC-radiomatic's user manual[25]

Although we haven't yet tested a controller with this specific feature, it's worth mentioning here, as it is actually an evolution over the previous two security measures. Passcodes have a limited key space compared to RFID smart cards and hardware keys, which can be easily lost or cloned. However, the research community is well aware of the vulnerabilities that affect RFID technology, which enable an attacker to bypass access control systems that rely on it.

2.4 Virtual Fencing

While testing our attacks in the field, we noticed that one crane (out of a dozen we considered in our on-site analysis) had an infrared (IR) receiver mounted at the middle, aimed at the ground on the construction site. After some observation, we found out that some vendors (e.g., Hetronic, HBC-radiomatic) implement a "virtual fencing" system using IR as an "out of band" technology. This mechanism is meant to enable the RX to accept commands only within the range of the IR beam. While meant primarily as a safety feature, this increases the barrier for an attacker who does not have access to the data sent via IR. However, the security of this mechanism is bound to the secrecy of such data.



radiomatic® infrakey

The radio system can only be activated via an infrared link between the transmitter and the receiver. This increases the safety of operation, i.e. the machine can not become inadvertently enabled.

radiomatic® infrakey operates either with an infrared module in the receiver housing (radiomatic® infrakey internal) or with the offset infrared antenna focus I (radiomatic® infrakey external).

To activate radiomatic® infrakey, actuate the start button on the transmitter.

Function of radiomatic® infrakey with focus I

Figure 11. IR-enabled virtual fencing system as described in HBC-radiomatic's manual[26]

Other vendors implement a form of virtual fencing by using heartbeat packets, which are exchanged periodically via RF by the TX, with the goal of informing the RX that the operator holding the remote controller is within range. This mechanism, however, loses the benefit of using the out-of-band channel (i.e., an independent, second communication channel, like the IR beams). An attacker who is within the RF range can analyze the structure of these packets and circumvent the protection mechanism. In particular, while testing remote controllers by Juuko, we were able to fool the RX that a TX was in range by simply forging valid heartbeat packets.

3. Security Analysis: Attacks, Risks, Threats

By exploiting various vulnerabilities that we discovered, we were able to move full-sized cranes deployed in production at construction sites, factories, and transportation businesses. In all of the cases, we were able to confirm and run the attacks very quickly. In each of the cases, we were able to switch on the controlled industrial machine even after the operator had issued an e-stop, which put the machine in a "stop" state.

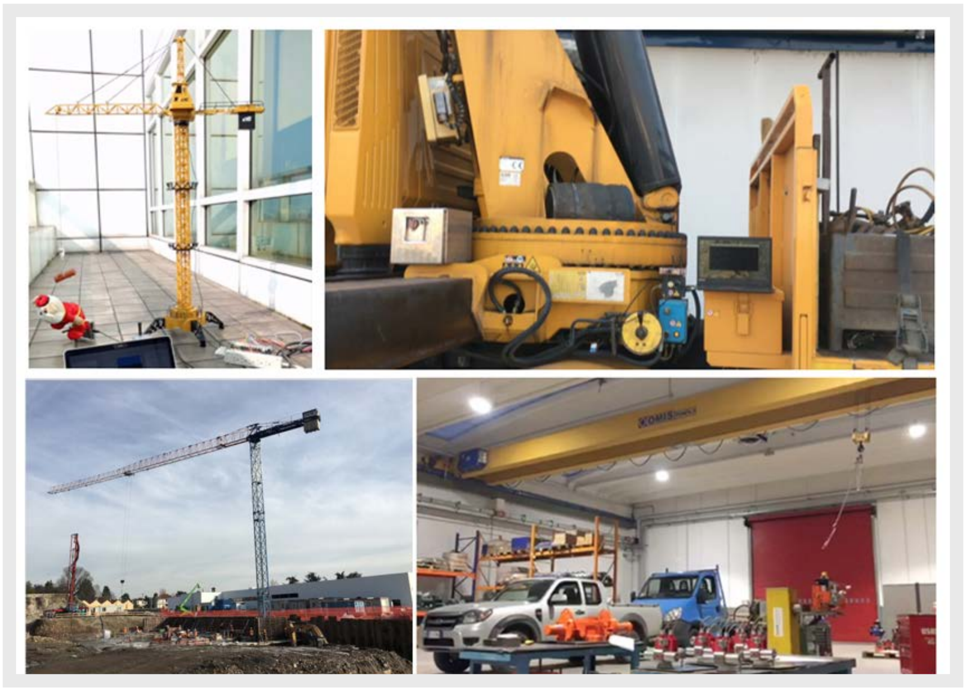


Figure 12. Overview of the conditions in which we tested our attacks: in-lab with a toy crane, at construction sites, in transportation businesses, and in factories

This section provides an overview of the attack classes that we identified, along with descriptions of the attacker models that we considered and three threat scenarios. (The next section focuses on the technical details of how we implemented and tested our attacks.)

Industrial radio remote controllers rely on proprietary RF protocols, designed decades ago and focused on safety as opposed to security — back then, the costs of RF attacks were quite high, whereas nowadays inexpensive RF-hacking toolsets are available. We found out that these systems are not robust against command spoofing, and thus an attacker (or attacking device) within range can capture a few seconds of radio traffic, selectively modify the packets, and craft arbitrary commands automatically.

Even if some of these remote controllers feature a pairing mechanism, this is not meant as a security measure against an active attacker, but only to prevent protocol-level interferences and allow multiple devices to operate simultaneously in a safe way. We show how an attacker who is within range — or is able to hide a coin-sized device with an inexpensive radio transceiver — can persistently and remotely control or simulate the malfunction of the attached industrial machines by crafting arbitrary packets. In comparison, consumer-level remote controllers for car or door locks tend to be more secure, as they implement some form of rolling-code mechanism.

3.1 Attack Classes

Attack class	Description	Difficulty	Access	Resources
1. Replay attack	The attacker records RF packets and replays them to obtain basic control of the machine.	Easy	Local or temporary local	Can purchase SDR equipment (US$100–2,000)
2. Command injection	Knowing the RF protocol, the attacker can arbitrarily and selectively modify RF packets to completely control the machine.	Intermediate	Temporary local	Can purchase target equipment for preparation (US$500–1,500)
3. E-stop abuse	The attacker can replay e-stop commands indefinitely to cause a persistent denial-of-service (DoS) condition.	Easy	Temporary local	Can purchase SDR equipment (US$100-2,000)
4. Malicious re-pairing	The attacker can clone a remote controller or its functionality to hijack a legitimate one.	Intermediate	Local or temporary local	Can purchase target equipment for preparation (US$500–1,500)
5. Malicious reprogramming	The attacker “trojanizes” the firmware running on the remote controllers to obtain persistent, full remote control.	Hard	Remote or temporary local	Advanced attacker (US$5,000–50,000)

Table 3. Overview of the five classes of attacks that we analyzed in this research. The term “temporary local” indicates that the attacker needs only to briefly drop by the target facility or use a drone; the rest of the attack will be carried out remotely.

From the list of security features that we have identified and from a preliminary analysis, this section outlines five classes of attacks, three of which (Attacks 3, 4, and 5) are specific to how radio remote controllers work and their deployment settings in industrial applications. In our case study, we describe how we implemented and tested the controllers against these attacks. Through these attacks, we were able to control construction cranes, industrial cranes, and mobile hoists on real production deployments. In addition, we demonstrate how an attacker can keep these devices in a permanent denial-of-service (DoS) state.

3.1.1 Attack 1: Replay Attack

In a replay attack, the adversary records a valid transmission and replays it (i.e., repeats it) with malicious purposes. The attacker captures the transmission with a software or hardware radio receiver and replays it "as is" using a radio transmitter. Since that transmission encodes a valid command that the TX unit is sending to the RX unit — which, if vulnerable, will accept it — a successful replay attack is indistinguishable from a valid transmission from a legitimate communication. Unlike in computer networks, the complete lack of attestation and forensics information also makes replay attacks against remote controllers ephemeral, and thus very hard to trace back to an actor or group.

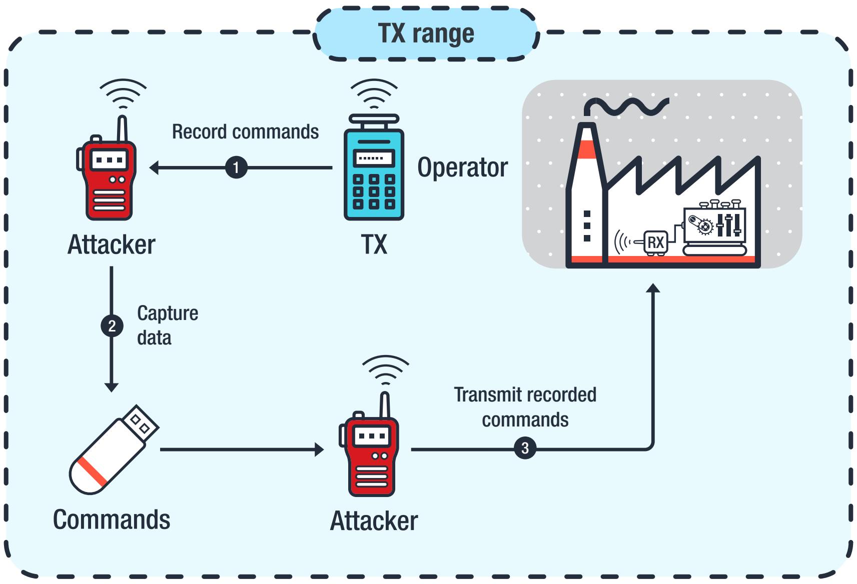


Figure 13. Conceptual representation of a replay attack on an industrial radio remote controller

Findings in brief. Ironically, while most of the consumer remotes (e.g., car locks, garage door openers) are secured against replay attacks — thanks to proper pairing and rolling-code mechanisms — *all seven industrial radio remote controllers that we tested were vulnerable to replay attacks*, both in-lab and on-site. If no rolling code is in place, this attack will always work.

The effect of replay attacks is proportional to the amount of knowledge that the attacker has about the target. In an industrial scenario, this translates to the amount of time that the attacker can spend sitting in close proximity to the target location, waiting for the commands to be issued, and recording them. In this way, the attacker can create a "library" of commands and use them at the right time. However, while in the past this would have been a daunting task requiring physical presence, the availability nowadays of inexpensive software-defined, miniaturized, and portable powerful radio transceivers makes this attack much more convenient. For instance, we built an RF research tool that is smaller than a credit card, is battery-powered and can interpret — in hardware — all of the modulations used by these controllers (e.g., FSK, PSK, MSK). This can be used to perform replay or other attacks. These devices can be hidden conveniently in an industrial facility and can be equipped with an RF transceiver and a cellular interface, allowing an attacker to maintain presence without physically being at the target location. This possibility of "becoming remote" poses other risks, even outside this specific domain.

3.1.2 Attack 2: Command Injection

In a command injection attack, the attacker must first gain a good knowledge of the radio protocol, to the extent that they can record a packet that represents "command X" and derive another packet that encodes "command Y." This can be performed through differential analysis of the RF packets captured under different conditions (e.g., different buttons, pressure duration, timing, and different devices). This attack is more powerful than a replay attack because it *allows forgery of arbitrary commands*, starting from a single packet (e.g., recorded on-site or after having purchased a copy of the target device).

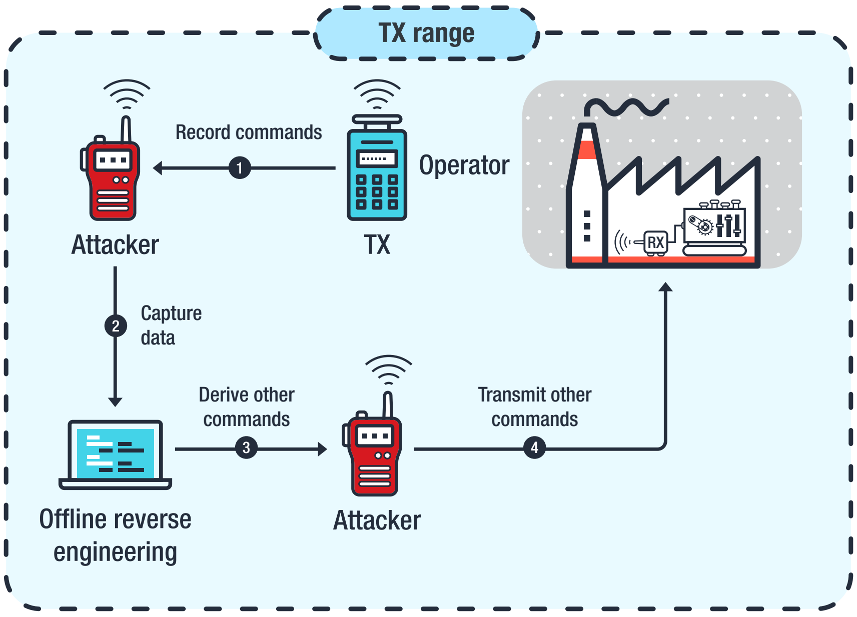


Figure 14. Conceptual representation of a command injection attack on an industrial radio remote controller

If the radio protocol does not use encryption or obfuscation, reverse-engineering the structure of the packets is a realistic option. For example, in our case study (described in Section 4), we encountered a vendor that uses a constant checksum function in all of the packets, making the process of fuzzing these packets even easier. Another vendor, which uses instead a weak form of encryption with a rolling code, doesn't enforce the check of the rolling code, leading to any forged packet to be accepted as valid.

Findings in brief. We tested in detail two products from two different vendors (Juuko and Saga) and confirmed the feasibility of this attack in a fully automated way, from both a local attacker and a remote one. In particular, we implemented our attack in a lightweight portable device (RFQuack, described in Section 5) that can be dropped on the target site or mounted on a drone. The device is programmed to look for known packets and modify them into any arbitrary command (e.g., the operator wants to move the crane down, but the attack device will automatically make it move up instead).

All the considerations made previously about the direct risks posed by replay attacks in the modern world also apply to command injection. An attacker equipped with a small device that can be deployed on-site for a certain amount of time will be able to collect enough data to successfully reverse-engineer the protocol and take over the site (or just purchase a copy of the identified target device and perform reverse engineering offline).

Replay and command injection attacks can be adopted as reusable attack primitives. As a result, an attacker builds domain-specific attacks that take advantage of functionalities that are specifically designed and very important for these devices. This is the case with the following three classes of attacks.

3.1.3 Attack 3: E-Stop Abuse

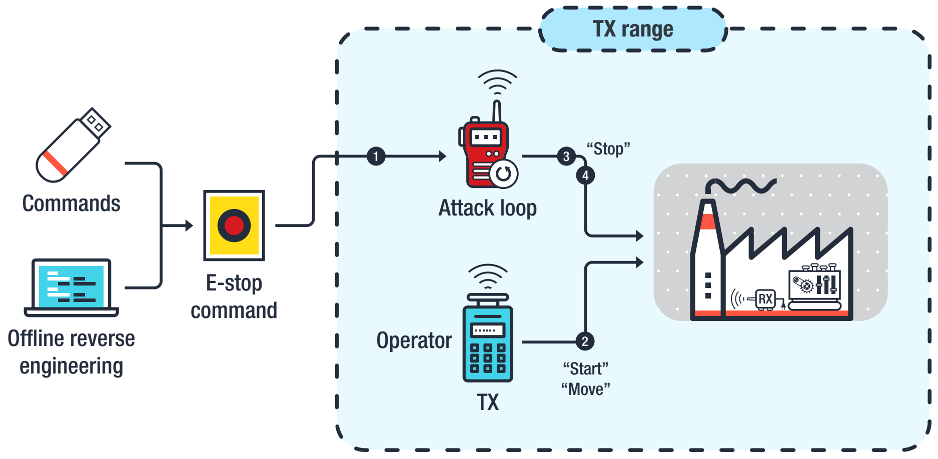


Figure 15. Conceptual representation of an e-stop abuse attack on an industrial radio remote controller

In an e-stop abuse attack, the attacker maliciously uses a special command to create a DoS condition. In fact, all remote controllers that we analyzed feature an emergency stop button, aka e-stop or EMO button, that sends a signal to the RX in case of an emergency. This cuts the power off from the attached load immediately. On the RX unit, this functionality is implemented by interrupting the power line with two so-called *safety* relays, positioned between the power source and the command relays. If the safety relays are disengaged, even if the command relays were engaged, the source line will be off. In the domain of radio remote controllers for industrial applications, from our experience, we conclude that “high priority” is more a matter of how the RX actuates the received e-stop command than how the TX sends it.

This is a safety requirement on all controllers. Depending on where a device is sold and deployed, there are strict regulations and certifications about the e-stop functionality. The most widely adopted standard in this scope is ISO 13850:2015.[27]

Findings in brief. None of the vendors that we analyzed use special encoding or higher transmission power when sending e-stop packets. As a result, an attacker who can simply record (using Attack 1) or forge (using Attack 2) an e-stop packet can transmit it indefinitely to keep the device consistently off, or to simulate a malfunctioning device. In turn, a miscreant can take control of an adversary facility and perform sabotage on it.

3.1.4 Attack 4: Malicious Re-Pairing

Most remotes offer a so-called “cloning” feature, which allows the creation of copies of a TX unit, or the association of an additional transmitter with an already deployed receiver. Note that the cloning functionality must not be confused with replaying a single command: All of the functionalities of an original TX unit are cloned onto a new one. As a result, this feature allows multiple operators to control a single receiver. Some products also have a “single transmitter to multiple receivers” scheme, which allows multiple receivers to be paired with a single transmitter.

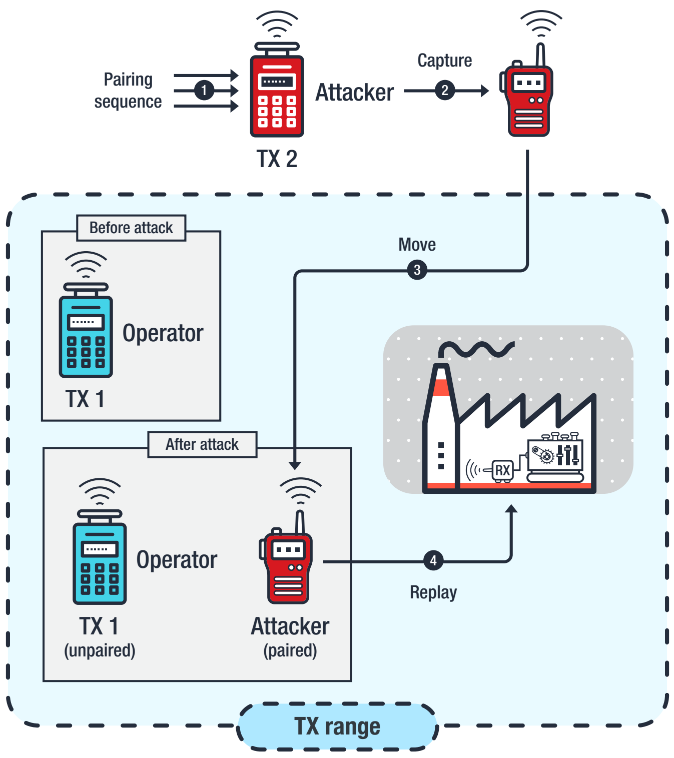


Figure 16. Conceptual representation of a malicious re-pairing attack on an industrial radio remote controller

Regardless of the "direction" of the cloning, to implement this feature, all paired transmitters and receivers must have a way to share the pairing code, so that all of the generated signals will be obtained from the same code. To avoid introducing a replay attack opportunity during the cloning session, the TX and RX must not directly exchange the pairing code itself because that could be intercepted and reused. Instead, they should exchange a unique code for each duplication session. Then, the actual pairing code will be computed from the unique code, together with a pre-shared secret (e.g., a key set by the factory on each device).

There are ways to securely implement a re-pairing functionality, an example of which requires only an accessible out-of-band channel (e.g., infrared) to manually put the RX in "re-pairing mode." Admittedly, the one vendor that we found vulnerable does not ship the devices that we procured with the "re-pairing" functionality enabled by default. Nonetheless, we believe that system integrators should be aware of these scenarios.

Findings in brief. We encountered one vendor that implemented this cloning functionality with the wrong attacker model in mind. The receiver *automatically* enters “pairing mode” whenever it’s powered on, and allows pairing for four minutes if the pairing is enabled in the configuration. During this time frame, an attacker is free to pair a malicious TX unit with any RX in radio range. This process does *not* require a physical action on the RX (e.g., keep a button pressed). (This is understandable, to some extent, because the RX units are typically deployed in hard-to-reach locations, e.g., top of the crane tower.) All it takes is to enter a certain sequence of buttons pressed on the new, malicious TX unit to convince the receiver that they have to pair with that transmitter. After this attack, the victim RX will discard all commands originating from the legitimate TX unit (i.e., accepts only the ones sent by the attacker) until an operator or a technician manually re-pairs the RX with the legitimate TX.

An attacker who can record or craft the packets emitted by the TX during the re-pairing phase and transmit them at the right time (e.g., using a miniaturized device, if not in range) will effectively pair with the receiver.

3.1.5 Attack 5: Malicious Reprogramming and Remote Attack Vectors

Lastly, we briefly look at remote attack vectors. We focus on an adversary model in which the attacker has already compromised a computer of the target organization, or worse, the computer used by the system integrator or final reseller to configure the devices prior to installation.

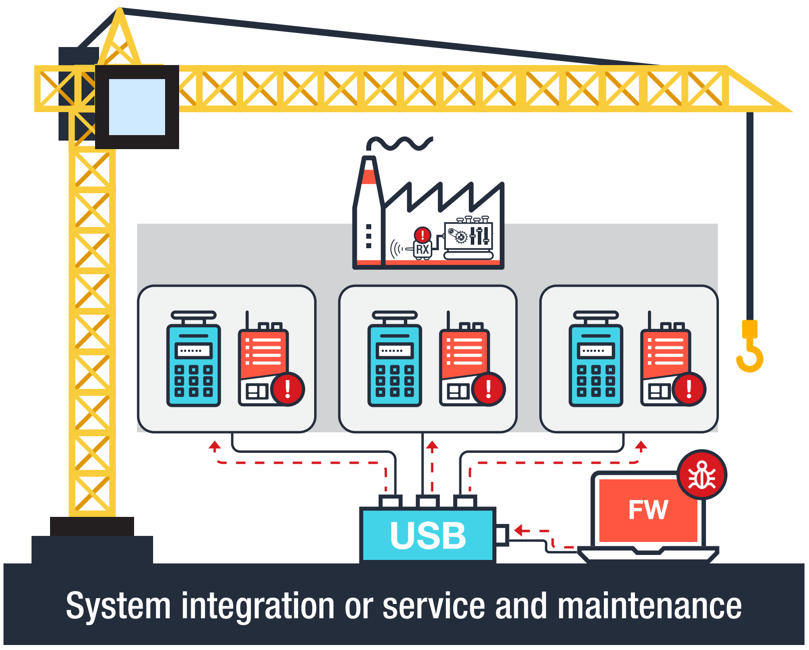


Figure 17. Conceptual representation of a malicious reprogramming attack on an industrial radio remote controller

Some of these devices expose programming pins and are typically sold with a programming "dongle" as an accessory. This way, the end users or the system integrators can customize the buttons, re-pair new units, change the fixed code, and so on.

However, if this process and the IT endpoints involved are not secured, any valid firmware can be flashed onto a victim unit when attached to the compromised computer. For example, we found out that in addition to allowing the users to flash the microcontroller's memory, some vendors even allow the extraction of the firmware (i.e., the microcontroller is not code-protected). Even in cases where fully re-flashing is prevented, an attacker can still reassign or disable buttons with the goal of causing damage or harming the operators. In another scenario, the attacker can install a backdoor that activates to a sequence communicated by the attacker and performs damaging operations like causing the crane to hit a person.

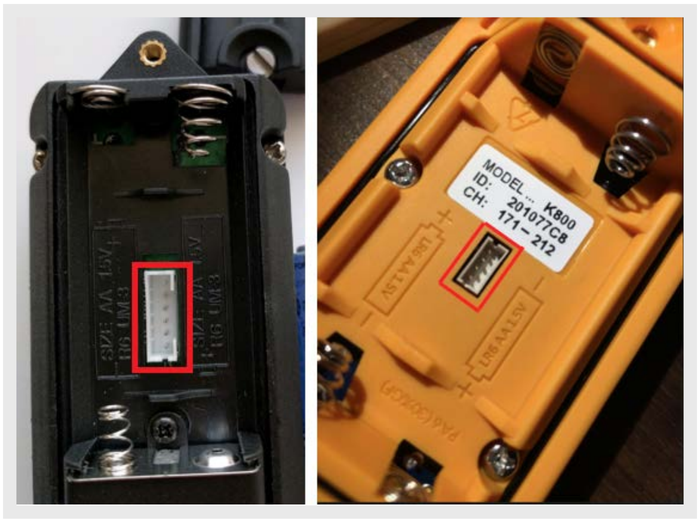


Figure 18. Programming interfaces on two of the units that we analyzed

Among the devices that we used, none of them had implemented any protection mechanism to prevent unattended reprogramming (e.g., operator authentication). Even inexperienced attackers (e.g., those not willing or able to write malware that communicates on the serial protocol), once the vendor-supplied software is installed on a compromised computer, can abuse the software itself through AutoIT-based[28] or similar ready-to-use malware to carry out this attack.

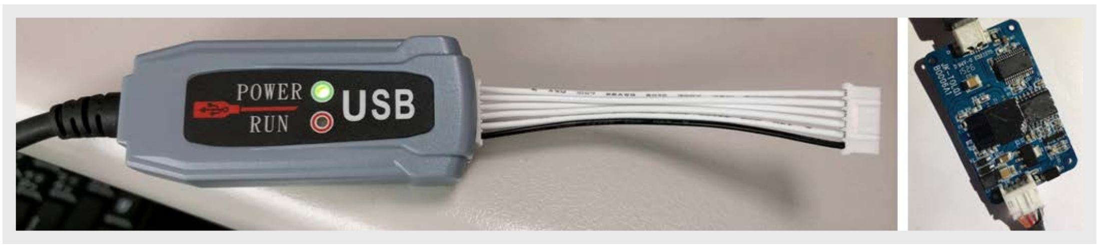


Figure 19. Saga and Juuko (opened) programming dongles. Telecrane uses the same programming dongle as Saga.

3.2 Risks to Industrial Facilities

Given the attacks outlined above, we now provide some potential threat scenarios to help evaluate the impact of such attacks in the real world. Considering that our findings are quite new, *these scenarios are by no means based on actual occurrences*. Rather, they are meant to provide food for thought to the reader, to better understand what variables are at play.

In these settings, the impact of basic vulnerabilities in the radio protocols can be significant because the controlled devices are safety-critical machinery, with serious consequences if compromised, including substantial financial loss, injuries, or loss of human lives. Given that these devices do not normally provide logs or other forensic artifacts, unless specifically configured to do so by attaching a proper data logger (some vendors offer such add-on devices, to keep track of the movements and usage of the controlled equipment), remote attacks on industrial applications are very difficult to be traced back to the attackers.

3.2.1 Scenario 1: Sabotage and Injury (Industrial Facilities, Construction)

Sabotaging an industrial facility is the first scenario that comes to mind. Depending on the role of the industrial facility (e.g., the goods that it produces or the services that it offers), the impact can be limited to the factory itself, or be broader, up to involving other businesses and even the society that depend on it. Any downtime in a factory, for example, will force long repair or replacement procedures with high costs and have subsequent impact on the final goods.

As another example, one or more incidents at a construction site can cost several months, if not years, of delays. If human lives are at stake, the construction work has to stop until the cases are probed and cleared. An attacker may also choose to target a company's reputation (e.g., cause a series of unlinked accidents over time at different sites). Apart from the incident itself, which may cause injuries and loss of human lives, in the long run, this could have an impact on the area that was supposed to take advantage of the future constructions.

A power plant may also be considered as an example. In 2013, at a nuclear power plant,[29] a heavy load fell during a refueling outage and caused the automatic shutdown of the reactor. The accident resulted in the death of one worker and injuries for several others. The damage to the plant by the dropped stator took weeks to repair. (In a generator, the stator converts the rotating magnetic field to electric current. In power stations, stators can have a diameter of several meters.) The cause was determined to be the temporary lifting machine that was used to lift the heavy load, which broke down during operation and dropped the heavy load. It is not far-fetched to envision a case where equipment such as the temporary lifting machine in this incident could be hijacked by an attacker.

3.2.2 Scenario 2: Theft (Harbors, Terminals, Logistics)

Advanced, fully automated harbors and terminals are remotely controlled (the Port of Rotterdam[30] being just one outstanding example), and a large number of industrial radio remote controllers are in place because these facilities are too large to make a wired-only deployment feasible. Even when not fully automated, traditional harbors and logistics facilities are filled with radio-controlled lifting and handling machines to move containers and loads.

An attacker who has compromised the radio communications can interfere with the lifting and handling operations to facilitate theft or hijack in-transit goods. This scenario envisions financially motivated attackers. Not only could the attacker steal goods, but they could also play a role in larger, supply-chain attack schemes.

3.2.3 Scenario 3: Extortion (All Sectors)

Over the years, cybercriminals have learned that the bigger the asset at risk is, the better they could exploit the extortion lever. And as noted in our recent report on the future of digital extortion,[31] the impact and methods of digital extortion can cross over to the physical realm. Given the high degree of interconnectivity of modern hardware, it is foreseeable that RF technology can be abused for extortion.

From this perspective, the previous two attack scenarios can be thought of in terms of extortion. An attacker can simulate a malfunction, causing repeated damage to industrial facilities or incidents, and then ask for a ransom to stop.

All of the attacks that we have described can be implemented through portable, small RF devices, conveniently hidden anywhere in the target facility. Such devices are not easy to be found, especially if one does not consider them a plausible threat.

3.3 Attcker Models

Rather than proposing self-contained attacker model definitions, we describe our attacker according to two main dimensions: *remoteness* and *skills*. In terms of remoteness, an attacker can be within or out of RF range. The skill level of the attacker influences the level of knowledge that the attacker can gain about the transmission protocol.

Although the attacker's budget makes a difference in what attacks (and attack scenarios) are feasible, we do not consider this variable, as it is difficult to envision an upper bound to an attacker's budget. We simply consider that, to carry out our attacks, a budget of a few hundred U.S. dollars is sufficient for the replay and e-stop abuse attacks, whereas the other attacks might require the attacker to obtain a copy of the target device, whose price ranges from a hundred to a few thousand U.S. dollars.

3.3.1 Local Attacks: Adversary Within Range

The transmission range of industrial radio remote controllers goes from a few meters up to roughly 300 meters, according to our measurements on-site. Using omnidirectional passive antennas designed for the specific wavelength, we were able to receive signals from a transmitter on the field up to 300 meters away, from within a car (with the antenna mounted outside the passenger window as shown in Figure 23). Clearly, using log-periodic directional antennas along with amplifiers, an adversary can transmit attack packets from kilometers away. For example, two battery-powered Texas Instruments CC1120 RF transceivers (like those used by Juuko, one of the vendors that we analyzed) 100 kilometers away from each other are able to communicate with zero packet loss.[32]

Local attacks may be carried out with these two attacker models:

- **Casual attacker (monkey style).** A side result of our work is that the risk in this domain is higher than thought, simply because the exploitation of some of the vulnerabilities does not require advanced skills. An attacker equipped with a software-defined radio (SDR) can record a command and replay it under hazardous conditions. This attacker model describes a casual attacker with no advanced skills whatsoever: They could be a simple script kiddie as well as a contractor, or even a disgruntled employee. Any of these attackers would be able to implement Attack 1 (replay) and Attack 3 (e-stop abuse) with a slightly higher skill level — just enough to write an infinite loop routine. In our view, sabotage and extortion are the most likely scenarios for this type of attacker.
- **Attacker knowing the RF protocol.** All of our envisioned scenarios apply to this type of attacker: An adversary who knows how to perform reverse engineering of a radio protocol goes beyond replay attacks. They could run Attack 2 (command injection), which requires neither any prior visit to the targeted site nor a recording of commands. Such an attacker could even obtain an exact copy of the target device, by learning the pairing code, and run the attack. We simulate this profile of an attacker by reverse-engineering the RF protocols of our tested devices.

3.3.2 Remote Attacks: Adversary Out of Range

An attacker who is out of the transmission range has two possibilities to launch our attacks: temporary physical access to the facility (e.g., a couple of minutes) or access to any computer used at any point in the supply chain for programming these devices. Remote attacks may be performed with these attacker models:

- **Remote and persistent attacker.** An attacker who can casually drop or plant a battery-powered, miniaturized embedded device will be able to maintain permanent remote access, until battery lasts. Nowadays, these devices are extremely easy to procure or build, and can include a cellular modem for convenient remote access. As a proof of concept, we built such a device to implement our attacks, as described in the following section.
- **Truly remote attacker.** Software attacks in the supply chain are among the most powerful ones. To gain such a level of access, however, an attacker would need to compromise a computer that is used, at some point, to program or configure the devices. We have already seen these forms of advanced attacks in the past (e.g., targeting critical systems in industrial applications by means of connected computers). To assess the risk level, we briefly looked at the security posture of the vendors that we considered. Using simple open-source intelligence (OSINT) sources, we found that *11 out of the 17 vendors run their public-facing websites on outdated and vulnerable server software*. One of them even has a public writable File Transfer Protocol (FTP) repository that should be *considered*

compromised, as it contained files uploaded by attackers to notify the system administrators about the breach. This activity is far from being comprehensive, but it offers a lower-bound indicator on their exposure.

3.3.3 Target Reconnaissance: A Real-World Case Study

One of the hardest things for an attacker in any RF attack is making sure they are looking at the right target. To understand how an attacker would act, we conducted an experiment considering an attacker who is looking for a target. As a result, we were able to profile a company in the town where one of the researchers lives and find its crane signals by pairing OSINT and RF reconnaissance. We were also able to figure out the specifications of the radios in use and how to attack them.

Attackers may have an idea of a few areas they want to make an impact on in a certain region of the world. They start to profile companies in the areas to find one that may be an ideal target for damage, injury, or even corporate espionage and brand reputation smearing.

With some searches on Google, one target became clear: a steel manufacturer that uses overhead cranes to move heavy loads onto trucks to be shipped around the country. We were able to find a YouTube video that showed a crane operator training video, presumably aimed at demonstrating how to safely operate the cranes to newly hired crane operators. At first, the video seems not particularly interesting to an attacker other than that it shows how the cranes are operated and what kind of loads they may be moving around. However, at a specific part of the video, the crane operator can be seen with his shoulder strap bearing the brand name of the vendor of the device he is using.

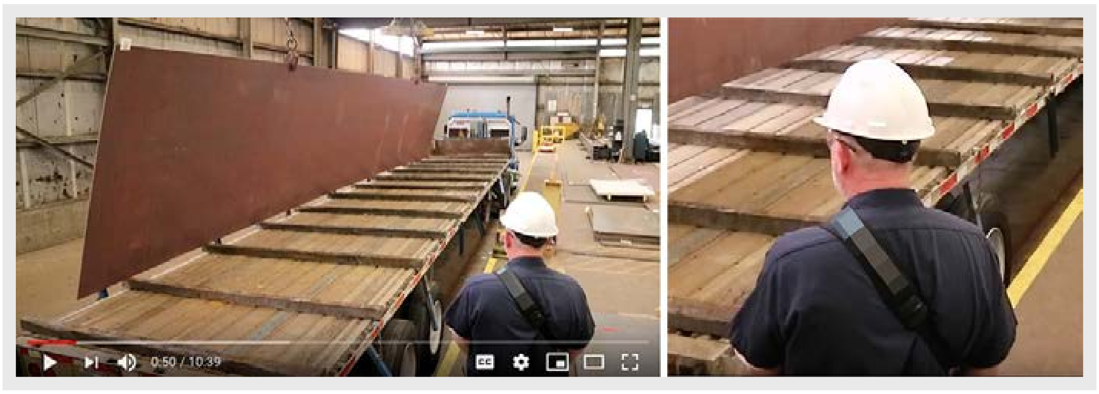
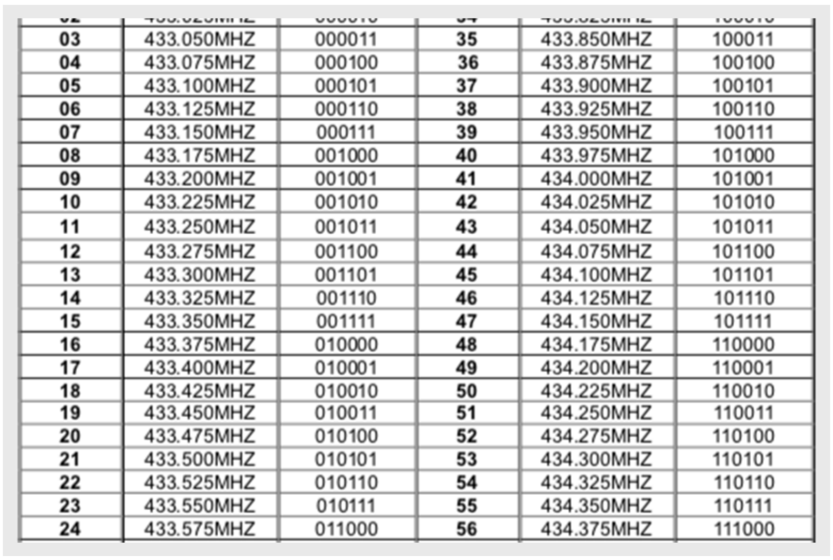


Figure 20. OSINT via YouTube with a close-up of the brand name (redacted) on the shoulder strap of the device used by the crane operator

Once we had an idea of what we were looking at, we performed some more OSINT to determine the specifications of the devices used in this case; most vendors in this space have all of their specifications and manuals online for public consumption. While reading some of the documentation for the vendor's products, there were multiple references about their operating within the 433–434 MHz range and using dual in-line package (DIP) switches to configure which frequency the TX and RX operate on.



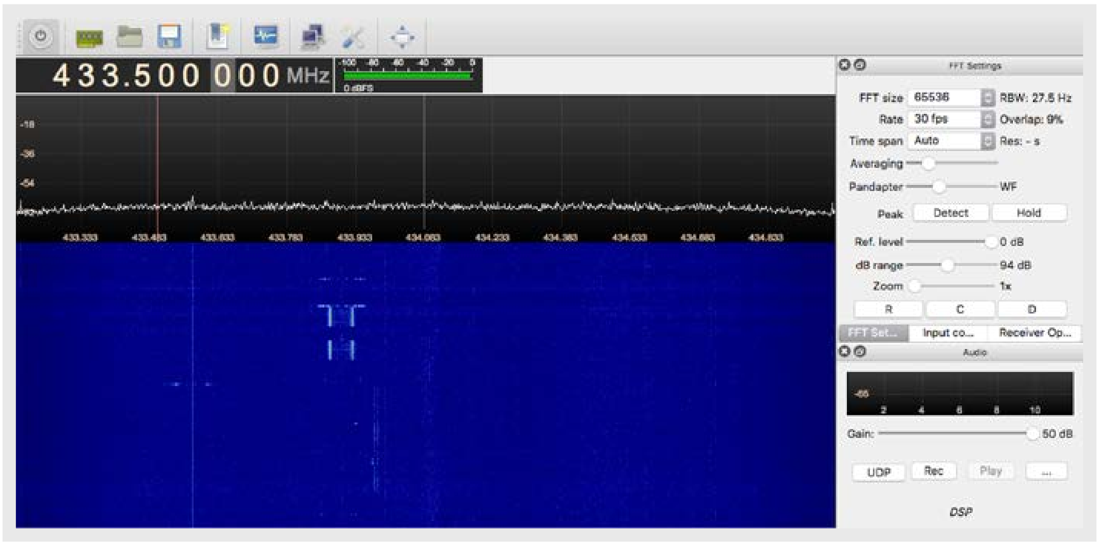
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
03	433.050MHZ	000011	**35**	433.850MHZ	100011
04	433.075MHZ	000100	**36**	433.875MHZ	100100
05	433.100MHZ	000101	**37**	433.900MHZ	100101
06	433.125MHZ	000110	**38**	433.925MHZ	100110
07	433.150MHZ	000111	**39**	433.950MHZ	100111
08	433.175MHZ	001000	**40**	433.975MHZ	101000
09	433.200MHZ	001001	**41**	434.000MHZ	101001
10	433.225MHZ	001010	**42**	434.025MHZ	101010
11	433.250MHZ	001011	**43**	434.050MHZ	101011
12	433.275MHZ	001100	**44**	434.075MHZ	101100
13	433.300MHZ	001101	**45**	434.100MHZ	101101
14	433.325MHZ	001110	**46**	434.125MHZ	101110
15	433.350MHZ	001111	**47**	434.150MHZ	101111
16	433.375MHZ	010000	**48**	434.175MHZ	110000
17	433.400MHZ	010001	**49**	434.200MHZ	110001
18	433.425MHZ	010010	**50**	434.225MHZ	110010
19	433.450MHZ	010011	**51**	434.250MHZ	110011
20	433.475MHZ	010100	**52**	434.275MHZ	110100
21	433.500MHZ	010101	**53**	434.300MHZ	110101
22	433.525MHZ	010110	**54**	434.325MHZ	110110
23	433.550MHZ	010111	**55**	434.350MHZ	110111
24	433.575MHZ	011000	**56**	434.375MHZ	111000

Figure 21. Excerpt from a table showing the DIP switch settings for each of the 62 frequencies (vendor redacted)

Once we had an idea of what we were looking for, we did some wardriving to see if we could identify the RF signatures of these systems. Driving around the area with our SDR equipment tuned to the range, we soon came to the realization that it wasn't going to be quite easy to find the cranes in use by driving around, and that there was no good parking location where we could sit for a while to observe the crane traffic. We thus scouted, via Google Maps, for the best places where we could set up our equipment and stay for a while without trespassing or being in anyone's way. Across a freeway from the target location is a public park, where we decided to sit for a few hours to see if we could find anything in the air.

Using a log-periodic antenna[33] pointing toward the target location, we were able to see the crane movements via RF signatures. This was confirmed by many hours spent watching the area, seeing RF signatures in what matches the specifications, and the communication looking similar to that of many other vendors. The facility in this case was using channel 37, as shown below in the waterfall plot taken at the time. As shown throughout this paper, from here we could start elaborating different attacks like command injection and malicious re-pairing if enough packets were captured and analyzed.

With this as an example, an attacker can in a matter of hours find a target in a region, deploy someone to sniff the area to positively identify the systems that are in use, build an attack, and carry it out whenever needed — without the target company knowing what's going on in its systems. In this case, it was a manufacturing company, but this same scenario can be performed in the power industry, in construction, in transportation, and anywhere else where these technologies are implemented.



FFT size 65536 RBW: 27.5 Hz
Rate 30 fps Overlap: 9%
Time span Auto Res: - s
Averaging
Pandapter WF
Ref. level 0 dB
dB range 94 dB
Zoom 1x
Gain: 50 dB

Figure 22. View of a command packet in a waterfall plot

4. Vulnerability Analysis and Attack Implementation

We tested our attacks on seven vendors under two conditions: in-lab and on-site. For in-lab testing, we purchased industrial radio remote controllers from two vendors, which distribute all around the world. We reverse-engineered their protocols and tested them against replay, command injection, e-stop abuse, malicious re-pairing, and malicious reprogramming attacks.

Region	Vendor	Tested devices	Attack 1: Replay	Attack 2: Command injection	Attack 3: E-stop abuse	Attack 4: Malicious re-pairing	Attack 5: Malicious reprogramming
APAC	Saga	2	Yes (in-lab)	Yes (in-lab)	Yes (in-lab)	Yes (in-lab)	Yes (in-lab)
APAC	Circuit Design	1	Yes (on-site)	Not tested	N/A (no e-stop feature)	N/A (only DIP switch pairing)	N/A (not programmable by end user)
EMEA + APAC	Juuko	1	Yes (in-lab)	Yes (in-lab)	Yes (in-lab)	Yes (in-lab)	Yes (in-lab)
EMEA	Autec	5	Yes (on-site)	Not tested	Yes (on-site)	Not tested	N/A (not programmable by end user)
EMEA + NABU	Hetronic	3	Yes (on-site)	Not tested	Yes (on-site)	Not tested	Not tested (software not available)
EMEA	Elca	1	Yes (on-site)	Not tested	Yes (on-site)	Not tested	Not tested (software not available)
EMEA + APAC	Telecrane	1	Yes (on-site)	Not tested	Yes (on-site)	Not tested	Yes (technical documentation*)

Table 4. Summary of our findings. “N/A” indicates that the device does not have the abused feature. “In-lab” indicates that the attack is confirmed in-lab, looking at the state of the relays, while “on-site” indicates that we tested the attack on a device deployed in production.
(*Verified through technical documentation.)

By analyzing and reverse-engineering the RF protocols of two popular vendors (Juuko and Saga), and by testing the above attacks against seven popular vendors, we highlight the following vulnerability patterns.

Vulnerability pattern	Description	Attack class	Patch development	Patch deployment
No rolling code	Each packet is self-contained and requires no dynamic secret to be interpreted. Any captured packet is always valid in the future.	1, 3	Very easy	Difficult in terms of accessibility: There are millions of units already deployed and the components are not easily accessible.
Weak or no cryptography	The data exchanged between transmitter and receiver is not encrypted or is weakly obfuscated and predictable.	1, 2, 3, 4	Easy	Difficult in terms of coverage: In addition to the patching problems explained above, firmware-only solutions may be insufficient because the hardware may not support cryptography.
Lack of software protection	The software used to upload firmware on the transmitter and receiver does not prevent unauthorized reprogramming.	5	Very easy	Easy: Vendors just need to implement proper access control in their software.

Table 5. Vulnerability patterns that we identified in our research

For capture and signal reconnaissance, we used a black-box approach that looked exclusively at the RF signal as transmitted over the air. In some cases, during our in-lab tests, we also adopted a gray-box approach by hooking onto the microcontroller's pins to confirm that the waveform captured via radio matched the signal sent by the microcontroller to the radio transceiver.

Black-box is inherently a difficult approach. But we decided to explore this way as well because it represents an attacker with low resources and limited knowledge of the target devices. We also show that even with a very low budget (a couple of hundred U.S. dollars), an attacker can control these devices.

According to the availability of the target devices, we tested our attacks in-lab or on-site (i.e., on a production system). When applicable, we made the in-lab tests slightly more realistic by attaching a mock industrial equipment (i.e., a toy crane hooked up to the real, industrial controller), for the sole purpose of demonstrating the effect of an attack on a smaller case.

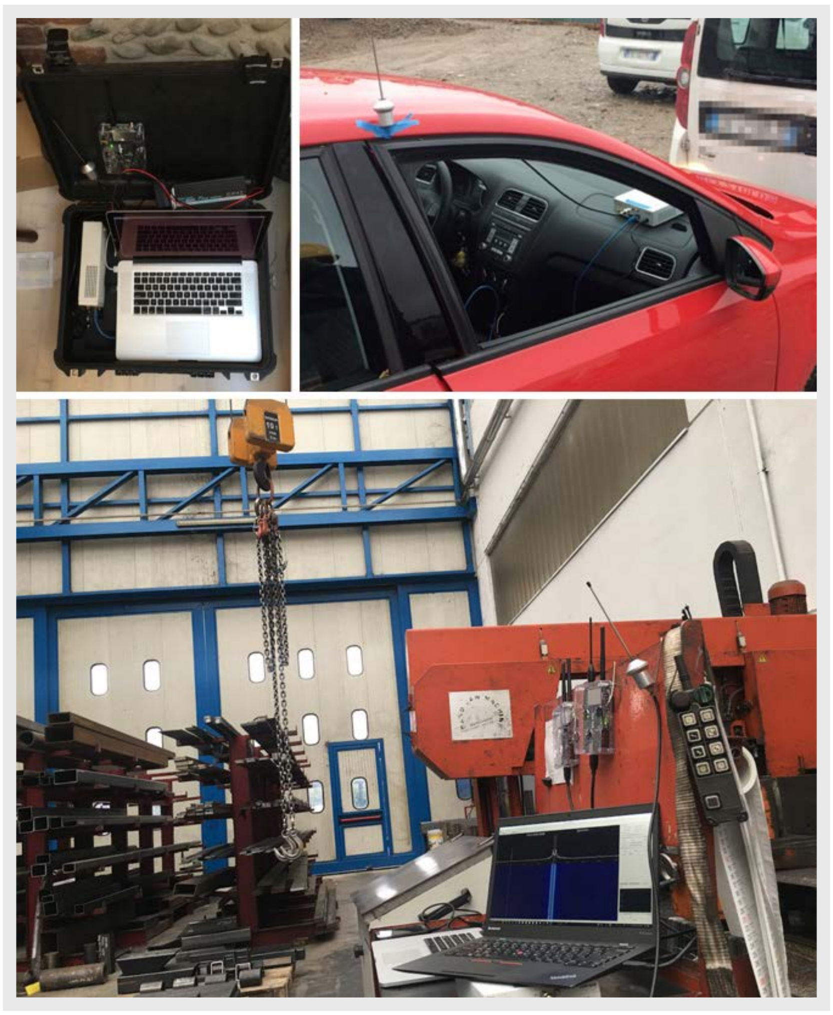


Figure 23. Various setups used during our experiments

4.1 Implementing Attack 1: Replay Attack

(ZDI-CAN-6183,* ZDI-CAN-6184,[34] ZDI-CAN-6185,** ZDI-CAN-6187,*** CVE-2018-17903,[35, 36] CVE-2018-17935,[37, 38] CVE-2018-19023[39, 40])

To test the remotes against the replay attack, we recorded individual traces of the following commands: start, button press (corresponding, for example, to a movement on the right), and e-stop.

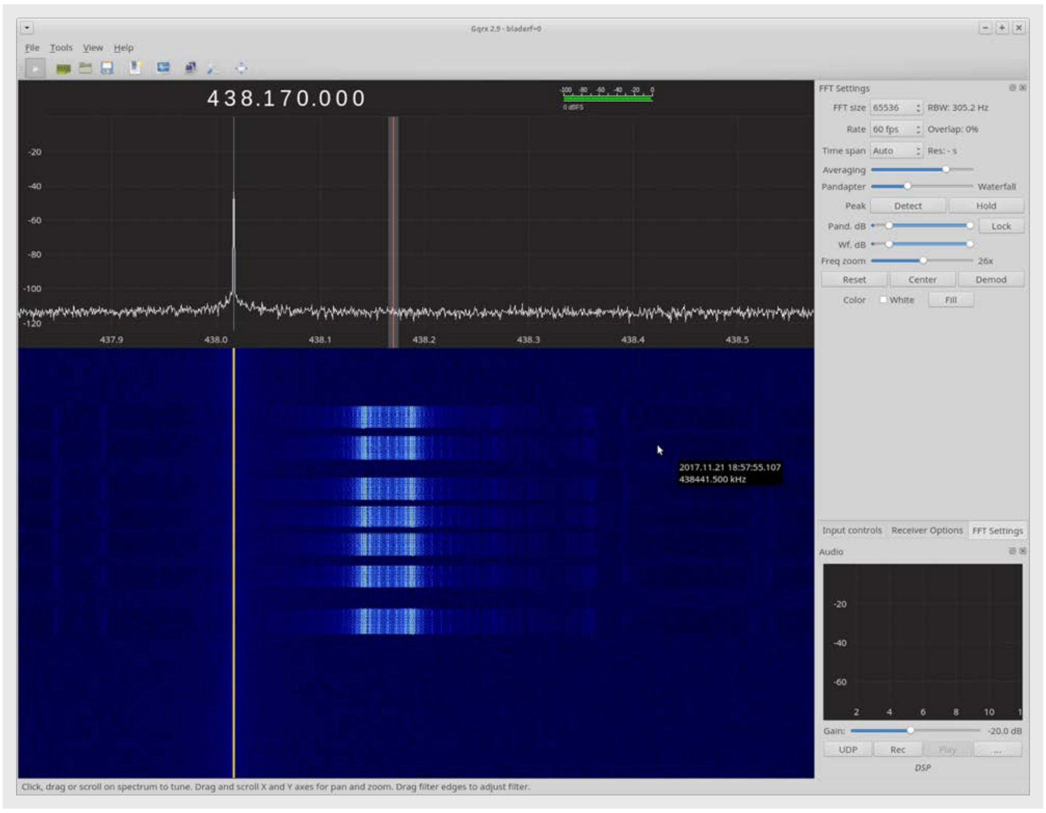
* The product under testing has reached end of life and is no longer supported by Autec.
** The product under testing was designed to specification and no fix is forthcoming from Circuit Design.
*** The product under testing has reached end of life and is no longer supported by Elca.

By replaying these commands individually, we confirmed three facts at once:

- The target device does not implement any security mechanisms like a rolling code because the RX will accept prerecorded data.
- An attacker can forcefully start a system if powered off.
- The e-stop command is encoded in a special way. This piece of information is useful for the implementation of Attack 3 (e-stop abuse).

In practice, we tested our attacks with various SDR devices, including an Ettus USRP N210, a BladeRF x115, and a HackRF. Considering that the output power is the main limiting factor for this specific attack, we believe that any SDR device with a transmitting antenna would be suitable to implement this attack.

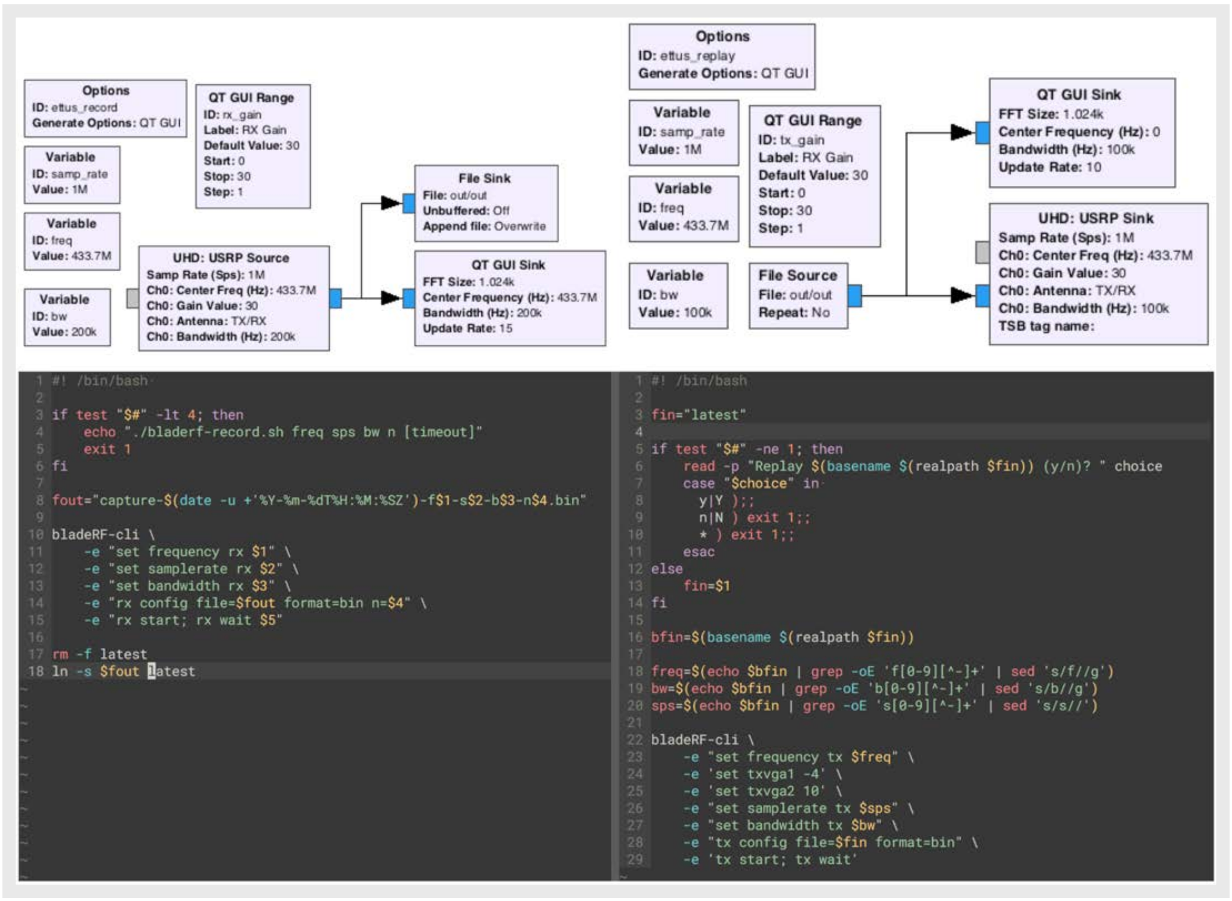
We implemented the attack both as a simple GNU Radio workflow (for the Ettus USRP) and as two shell scripts (for the BladeRF x115).



FFT size 65536 RBW: 305.2 Hz
Rate 60 fps Overlap: 0%
Time span Auto Res: - s

Click, drag or scroll on spectrum to tune. Drag and scroll X and Y axes for pan and zoom. Drag filter edges to adjust filter.

Figure 24. For ethical and legal reasons, before running the attacks we used a spectrum analyzer (GQRX) to ensure that no other signals were recorded (and thus replayed) within the bandwidth.



```
Options
ID: ettus_record
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: freq
Value: 433.7M

Variable
ID: bw
Value: 200k

QT GUI Range
ID: rx_gain
Label: RX Gain
Default Value: 30
Start: 0
Stop: 30
Step: 1

UHD: USRP Source
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 433.7M
Ch0: Gain Value: 30
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 200k

File Sink
File: out/out
Unbuffered: Off
Append file: Overwrite

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 433.7M
Bandwidth (Hz): 200k
Update Rate: 15
```

```
Options
ID: ettus_replay
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: freq
Value: 433.7M

Variable
ID: bw
Value: 100k

QT GUI Range
ID: tx_gain
Label: RX Gain
Default Value: 30
Start: 0
Stop: 30
Step: 1

File Source
File: out/out
Repeat: No

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k
Update Rate: 10

UHD: USRP Sink
Samp Rate (Sps): 1M
Ch0: Center Freq (Hz): 433.7M
Ch0: Gain Value: 30
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 100k
TSB tag name:
```

```bash
#! /bin/bash

if test "$#" -lt 4; then
    echo "./bladerf-record.sh freq sps bw n [timeout]"
    exit 1
fi

fout="capture-$(date -u +'%Y-%m-%dT%H:%M:%SZ')-f$1-s$2-b$3-n$4.bin"

bladeRF-cli \
    -e "set frequency rx $1" \
    -e "set samplerate rx $2" \
    -e "set bandwidth rx $3" \
    -e "rx config file=$fout format=bin n=$4" \
    -e "rx start; rx wait $5"

rm -f latest
ln -s $fout latest
```

```bash
#! /bin/bash

fin="latest"

if test "$#" -ne 1; then
    read -p "Replay $(basename $(realpath $fin)) (y/n)? " choice
    case "$choice" in
      y|Y );;
      n|N ) exit 1;;
      * ) exit 1;;
    esac
else
    fin=$1
fi

bfin=$(basename $(realpath $fin))

freq=$(echo $bfin | grep -oE 'f[0-9][^-]+' | sed 's/f//g')
bw=$(echo $bfin | grep -oE 'b[0-9][^-]+' | sed 's/b//g')
sps=$(echo $bfin | grep -oE 's[0-9][^-]+' | sed 's/s//')

bladeRF-cli \
    -e "set frequency tx $freq" \
    -e 'set txvga1 -4' \
    -e 'set txvga2 10' \
    -e "set samplerate tx $sps" \
    -e "set bandwidth tx $bw" \
    -e "tx config file=$fin format=bin" \
    -e 'tx start; tx wait'
```

Figure 25. Record-and-replay GNU Radio flowchart (above) and shell scripts that we used for testing against Attack 1 (below)

We tested our attacks both in-lab and on-site at various facilities like factories and construction sites. In all of the cases, *our attacks worked out of the box*. We have streamlined this testing procedure in order to collect evidence of our discovered vulnerabilities on additional vendors, with whom we are currently working for responsible disclosure.

4.2 Implementing Attack 2: Command Injection

The implementation of this attack required reverse engineering of the protocol, both at the physical layer (RF) and at the application layer.

As a general approach, we performed radio captures of different commands including start, e-stop, button press, and no-button press. We recorded all combinations of button pressures, for both long and short pressures. We then looked at the captured traces with the goal of understanding the demodulation schemes adopted by the different remote controllers. To this end, we made use of any documentation we managed to have access to (often resulting in little or misleading details) and the Universal Radio Hacker toolkit to inspect the “shape” of the radio signal. For one vendor (Juuko), which used a mixed 2/4-FSK modulation *within the same packet*, we had to implement a custom GNU Radio workflow because none of the available tools supported it. Finally, we developed dedicated attacking scripts to perform the attack. The case of the Juuko motivated us to orient our research efforts toward the development of a flexible RF hacking hardware device, nicknamed RFQuack and presented at the end of this section. We shared all of our findings and the research we produced with the affected parties.

The FCC documentation that we found turned out to be a double-edged sword. We discovered that the documentation was not up to date and at times quite misleading. This is understandable because the radio modules can go through several modifications over the years and these changes are not promptly reflected in the user manuals. We thus ended up trusting our results more than the manuals.

4.2.1 Analysis of the Saga Controller

(CVE-2018-17923,[41, 42] CVE-2018-17921,[43, 44] CVE-2018-17903[45, 46])

From the Saga technical documentation, we learned the exact frequency band (433.050–434.7875 MHz), modulation (FSK), length of the ID code (2^{20}), channel spacing (25 kHz), and use of Hamming code. During the analysis, however, we learned that the FCC documentation was outdated and did not match the radio signals. The actual ID code was longer (2^{24}) and Hamming code was not used.

Frequency and modulation. The actual frequency is printed on the box bought from Saga. The frequency can be changed by replacing the crystal oscillator, which is documented in the user manual. After capturing the signals on the printed frequency, we confirmed the modulation (2-FSK) by using baudline (see Figure 27).

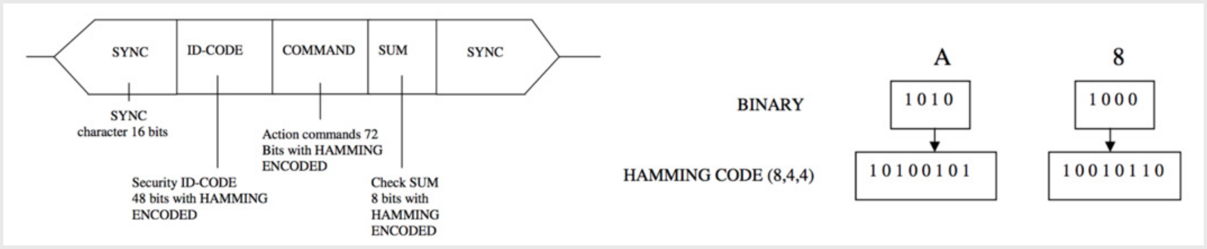

SYNC
ID-CODE
COMMAND
SUM
SYNC
SYNC character 16 bits
Security ID-CODE 48 bits with HAMMING ENCODED
Action commands 72 Bits with HAMMING ENCODED
Check SUM 8 bits with HAMMING ENCODED
A
8
BINARY
1010
1000
HAMMING CODE (8,4,4)
10100101
10010110

Figure 26. Packet structure and encoding according to the Saga technical documentation

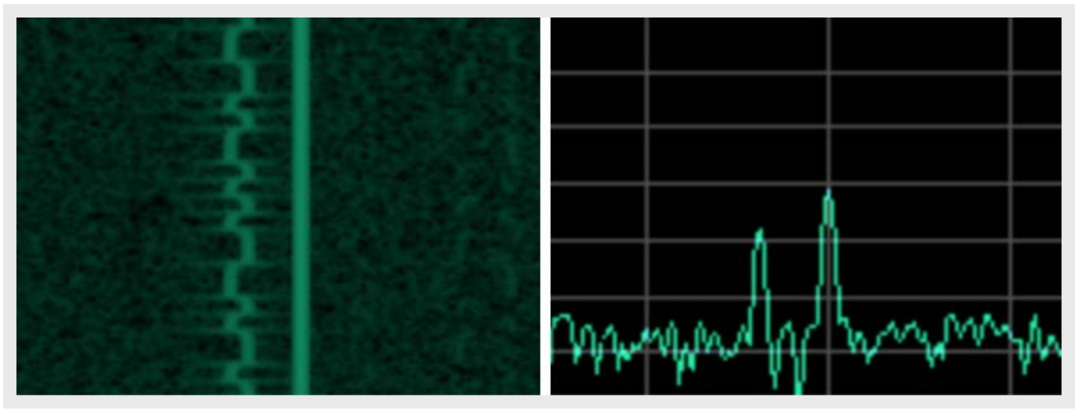


Figure 27. The transitions can be clearly seen between the two frequencies, a strong indicator that confirms the FSK modulation.

To confirm that the capture was correct, we looked at the data sent by the microcontroller to the radio transceiver, an Infineon TDA5101. By inspecting the PCB of the TX unit, we found out that the microcontroller was an MSP430F11, a widely used model from Texas Instruments. The TDA5101 datasheet didn't mention anything about the encoding scheme, so we assumed the encoding was done only by the microcontroller.

According to the FCC schematics (see Figure 28) and by following the traces on the circuit board, we learned that the transceiver was connected to the MSP430F11 via pin 8, and that the wire name was "FSKDATA". This was a strong clue that pin 8 might provide us with raw samples of the FSK-modulated data. As we noticed a (second) external clock wired to TDA5101, we could not reliably determine the clock speed. Here, we chose to use the highest sample rate on the logic analyzer to avoid undersampling.

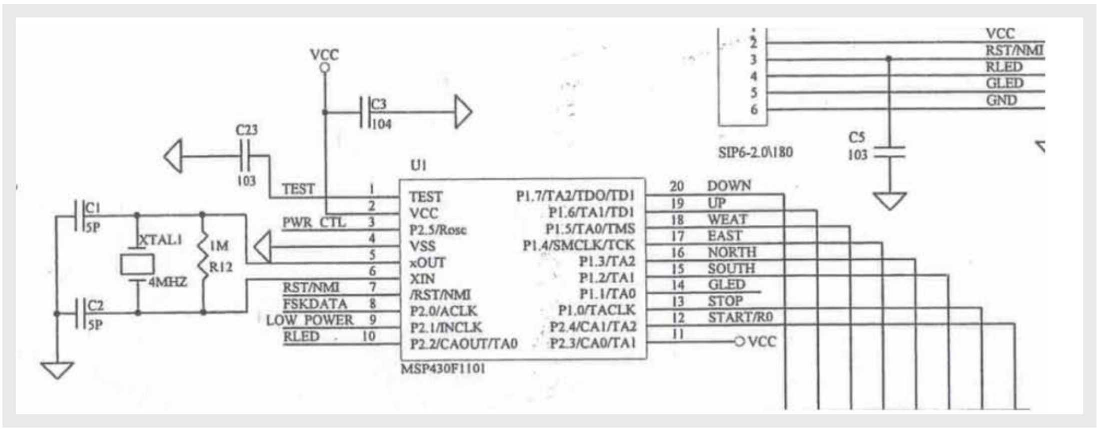


Figure 28. FCC schematics of the radio controller, indicating that the FSK-modulated data can be read from pin 8

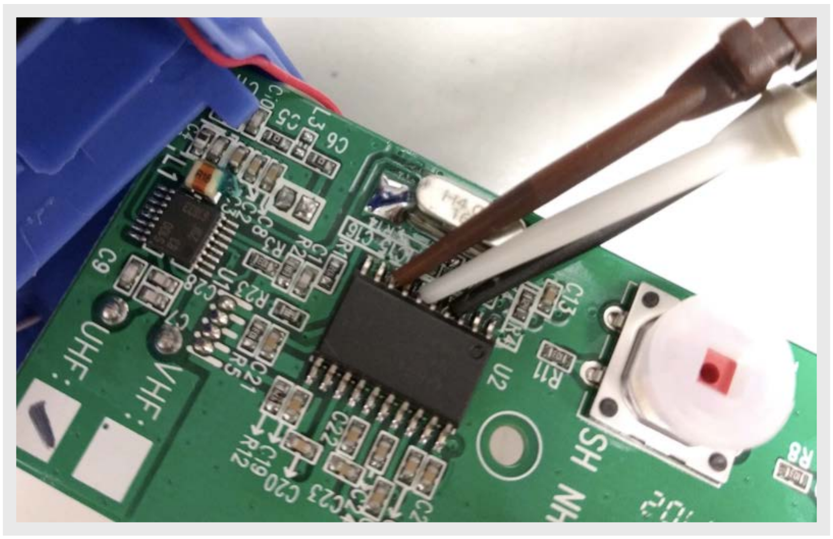


Figure 29. Clock and data pins hooked to the logic analyzer

Using the logic analyzer, we acquired the raw trace. At a sample rate of 2M and a bit length of 240 samples, we obtained an estimated transfer rate of 8,333 bits per second (bps), which is realistic in this type of equipment. At that point, we exported the FSKDATA data to a file that we then converted to a bitstream. We tried to align the resulting bitstream at various column widths, until we found a 224 bits long recurrent pattern (see Figure 30).

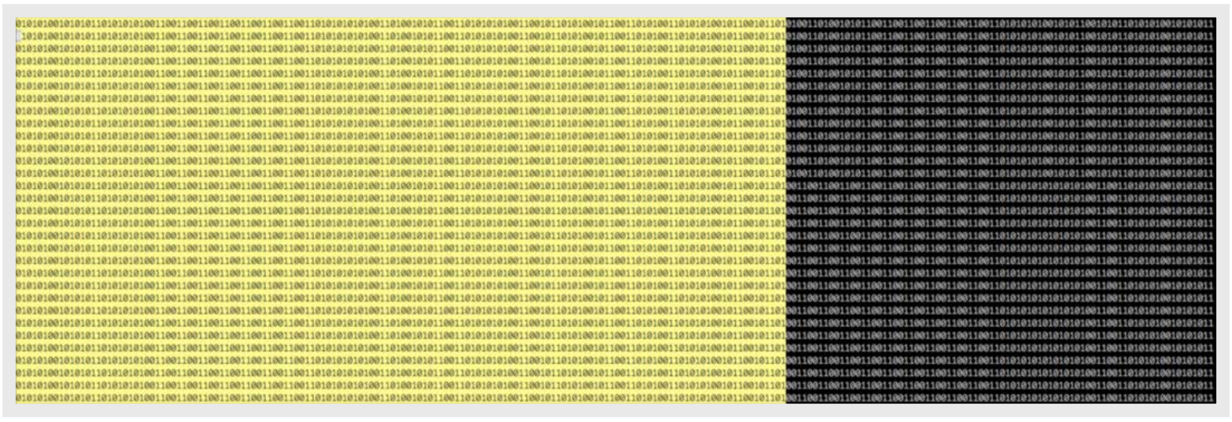


Figure 30. Repetitive patterns in the bitstream revealing the packets that encode (with Manchester coding) the commands

Packet reverse engineering. After decoding each packet with Manchester, we noticed the following recurring 72-bit sequence:

```
000 1111 0000 0101 0101 0101 0101 0000 0010 0111 0100 0001 0110 0011 0100 0100 0011 0110 0
```

We expected this to be the sync word plus the pairing ID: `0F 05 55 50 27 41 63 44 36`. As a further confirmation, we purchased a second device, with a different ID, and the preamble was the same. More precisely, the structure of the packet was:

```
0F 05 55 50 27 41 | 63 44 36 | Command (see table) | Checksum

0F 05 55 50 27 41 | 11 50 27 | Command (see table) | Checksum
```

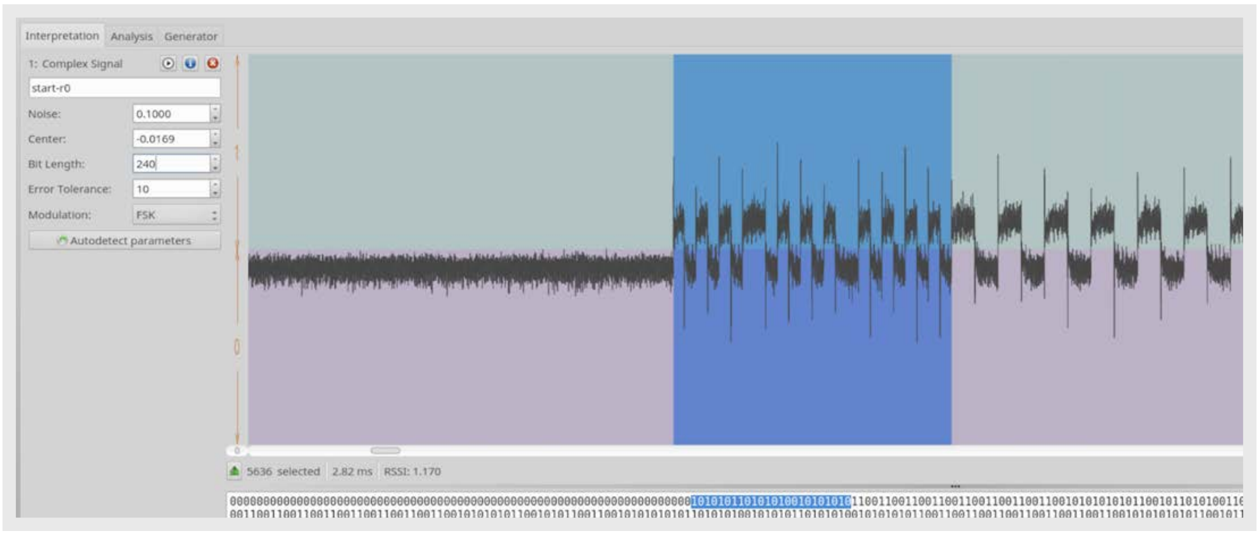
The second part of the packet was likely the pairing ID (fixed code). At this point, we were able to create a table of the packets associated with the various commands (see Table 6).

Button	Command (last byte is the checksum)
Reset (EMO)	55 50 50 11
Start	55 55 41 14
Up	66 55 50 36
Down	27 55 50 77
East	50 55 50 11
West	44 55 50 14
South	55 66 50 36
North	55 27 50 77
End of Packet (EOP)	55 55 50 05

Table 6. Decoded command and checksum fields resulting from the pressure of each button

We also found out that the implementation of the checksum for this particular vendor was rather weak. The command has three bytes, and the checksum is fixed (see the highlighted field in Table 6), that is, it does *not* change across multiple devices. As a result, we didn't need to reverse-engineer the checksum algorithm to be able to forge arbitrary packets. Knowing the preamble and the ID from a single packet, an attacker could forge a packet by encoding any command with any pairing ID.

Going back to the signal that we captured via radio, we confirmed our findings: The radio stream was modulated via 2-FSK and encoded with Manchester; no encryption or rolling code was used. The Universal Radio Hacker toolkit was very useful for our analysis.



start-r0
Noise: 0.1000
Center: -0.0169
Bit Length: 240
Error Tolerance: 10
Modulation: FSK

5636 selected 2.82 ms RSSI: 1.170

```
000000000000000000000000000000000000000000000000000000000000000000000000000001010101101010100101010101100110011001100110011001100110011001010101010110010110101001110
001100110011001100110011001100110010101010110010101100110010101010101101010100101010110101010010101010110011001100110011001100110011001010101010110010111
```

Figure 31. Pattern indicating FSK modulation and Manchester encoding

To prove our point, we used this toolkit to forge packets with given commands and verified that the crafted (malicious) message was correctly received and processed by the receiver. As a result, our testing crane moved according to a preconfigured script: It picked up the load, turned east, wired down, and unloaded the cargo.

4.2.2 Analysis of the Juuko Controller

(ZDI-CAN-6462[47])

While reverse-engineering the protocol for the Juuko controller, we found a vulnerability similar to the one we found in the Saga radio controller. The vulnerability lies in the fact that each packet is so weakly obfuscated (XOR-based) that the position of the fixed code and other fields is seen in each packet. The fixed code is set by the factory on both RX and TX but can be changed by the user using the software and programming dongle provided with the units. By exploiting this vulnerability, the attacker can forge new packets that encode arbitrary commands. As a side discovery, the weak obfuscation function adopted allows an attacker to obtain the secret fixed code by just capturing *one* radio packet. The rest of this section describes how we reverse-engineered the protocol and how the vulnerabilities can be exploited.

Integrated circuit and bus identification. We based our analysis on a Juuko JK800 transmitter with a Harmonized System (HS) code receiver. Both the units are based on Texas Instruments' CC1120, a powerful low-power, sub-1 GHz radio transceiver used in a variety of applications such as medical devices, smart meters, and building automation. On our devices, the CC1120 is driven by two microcontrollers: a PIC16L and a PIC18L. Although we haven't looked into their roles in depth, we gather that one of them is used to implement the communication (at the application layer) and the other is used to manage firmware loading and updating.

Unlike with the Saga, in which the radio transceiver takes an FSK bitstream and sends it over the air "as is," the transceiver used by Juuko needs to be initialized first. At that point, the transceiver is ready to receive packets in the form of byte-streams into a FIFO (first-in, first-out) buffer, which is then flushed upon each transmission. The CC1120 takes care of implementing the radio stack according to its configuration.

We started our analysis by removing the metal shield of the RF module. This allowed us to identify the radio transceiver (CC1120) by taking a close-up picture of the packet after removing the conformal coating. We used a Saleae logic analyzer to learn the pin allocation (as shown in Figure 32) and confirm that SPI is used as the communication interface.

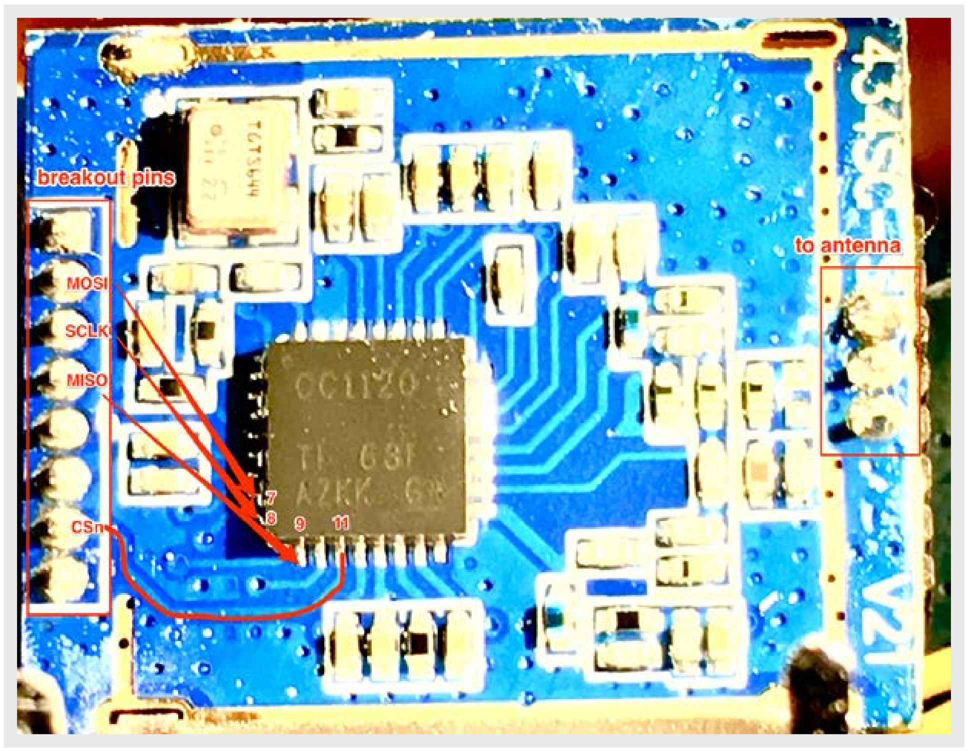


Figure 32. We confirmed the breakout pins by hooking a logic analyzer at each of the SPI pins on the package (MOSI, MISO, SCLK, CSn) as reported on the datasheet,[48] and compared the signal captured from each breakout pin until we found a match.

Reliably tapping into the SPI bus is needed to know how the microcontroller communicates with the radio transceiver. In this phase, we used the information called from the SPI logs to derive the radio frequency parameters used by the controller. These parameters are essential in the reverse engineering of the physical layer, i.e., in demodulating the signal. According to the programming user guide,[49] the transceiver is configured by setting appropriate register values. Parameters like the modulation scheme (GFSK, in our case), carrier frequency, sync word, and length of the data packets are all set by the microcontroller via SPI "commands."

Frequency and modulation. By looking at the radio communications we previously collected with our SDR equipment, we learned that the modulation scheme adopted by the Juuko is 4-GFSK (a variant of 4-FSK in which a Gaussian filter is applied to the data symbols to make the transmission smoother).

According to the programming user guide, the modulation is applied as shown in Figure 33.

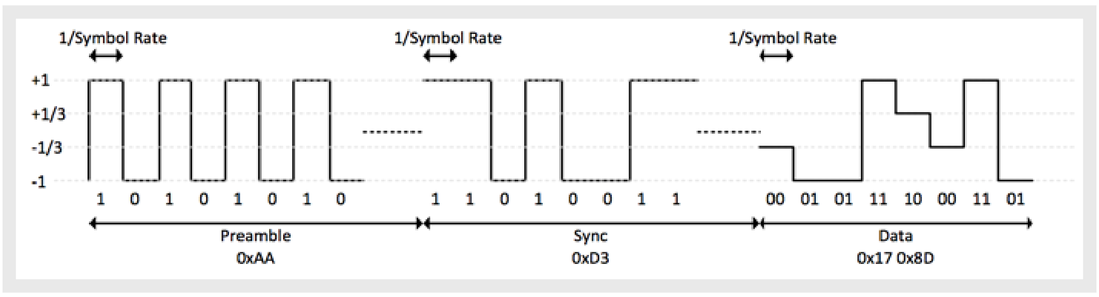
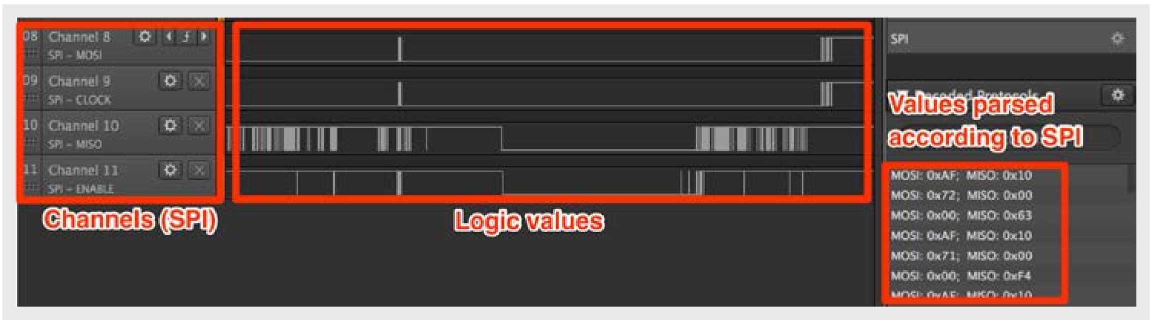


Figure 33. The modulation scheme adopted by the Juuko as documented in the manual. Note that the preamble (0xAAAAAA in this case) and sync word are modulated with 2-FSK, while the rest of the packet is modulated with 4-FSK. The symbol rate is consequently adjusted.[50]

(Courtesy of Texas Instruments Incorporated)

In particular, when configured to use GFSK, modulating and demodulating CC1120 packets is more complicated than just working with plain GFSK because the modulation switches dynamically from FSK to GFSK: It uses two symbols for the preamble and sync word and four symbols for the rest of the packet.

SPI tapping and protocol emulation. In addition to this challenge, we had to figure out the modulation parameters (such as the frequency deviations). However, the SPI logs as decoded by the logic analyzer are not directly informative.



```
MOSI: 0xAF;  MISO: 0x10
MOSI: 0x72;  MISO: 0x00
MOSI: 0x00;  MISO: 0x63
MOSI: 0xAF;  MISO: 0x10
MOSI: 0x71;  MISO: 0x00
MOSI: 0x00;  MISO: 0xF4
MOSI: 0xAF;  MISO: 0x10
```

Figure 34. Output of the logic analyzer (bottom right) while tapping on the SPI bus

Before we could extract the radio packets from the SPI logs, we needed to interpret them according to the configuration protocol described in the programming user guide.

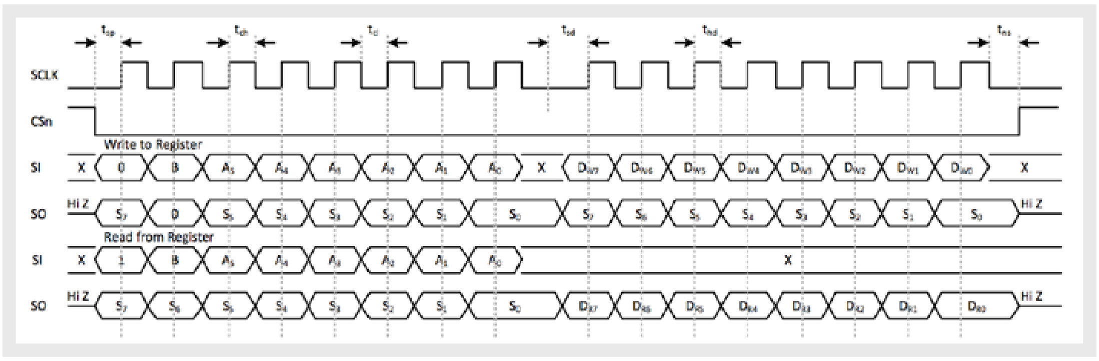
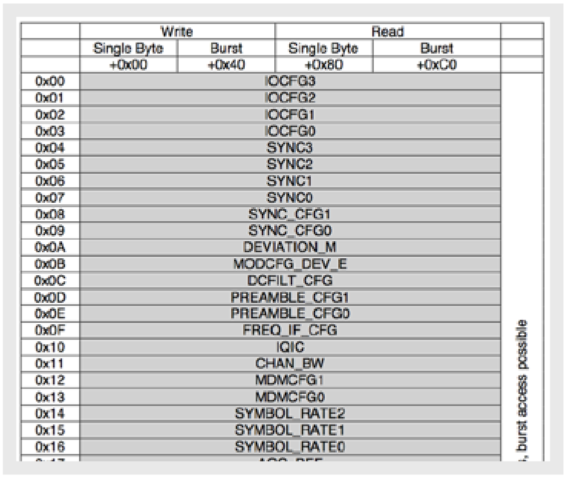


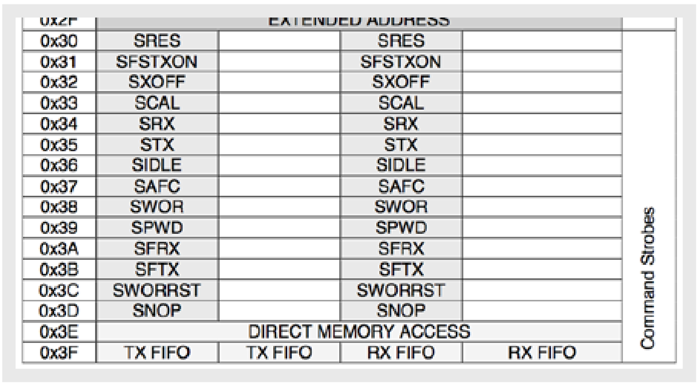
Figure 35. Configuration registers write and read operations via SPI on the CC1120[51]

(Courtesy of Texas Instruments Incorporated)

Using the programming user guide (Figure 35 and Figure 36) as a reference, which describes how the configuration registers should be set and how the memory map is organized, we implemented a custom emulator that takes as input the raw SPI logs and produces *decoded* SPI transactions. We used this emulator to reverse-engineer the SPI communication and learn the configuration registers set by the microcontroller to be used later in our analysis.



	Write		Read		
	Single Byte	Burst	Single Byte	Burst	
	+0x00	+0x40	+0x80	+0xC0	
0x00	IOCFG3				, burst access possible
0x01	IOCFG2				
0x02	IOCFG1				
0x03	IOCFG0				
0x04	SYNC3				
0x05	SYNC2				
0x06	SYNC1				
0x07	SYNC0				
0x08	SYNC_CFG1				
0x09	SYNC_CFG0				
0x0A	DEVIATION_M				
0x0B	MODCFG_DEV_E				
0x0C	DCFILT_CFG				
0x0D	PREAMBLE_CFG1				
0x0E	PREAMBLE_CFG0				
0x0F	FREQ_IF_CFG				
0x10	IQIC				
0x11	CHAN_BW				
0x12	MDMCFG1				
0x13	MDMCFG0				
0x14	SYMBOL_RATE2				
0x15	SYMBOL_RATE1				
0x16	SYMBOL_RATE0				



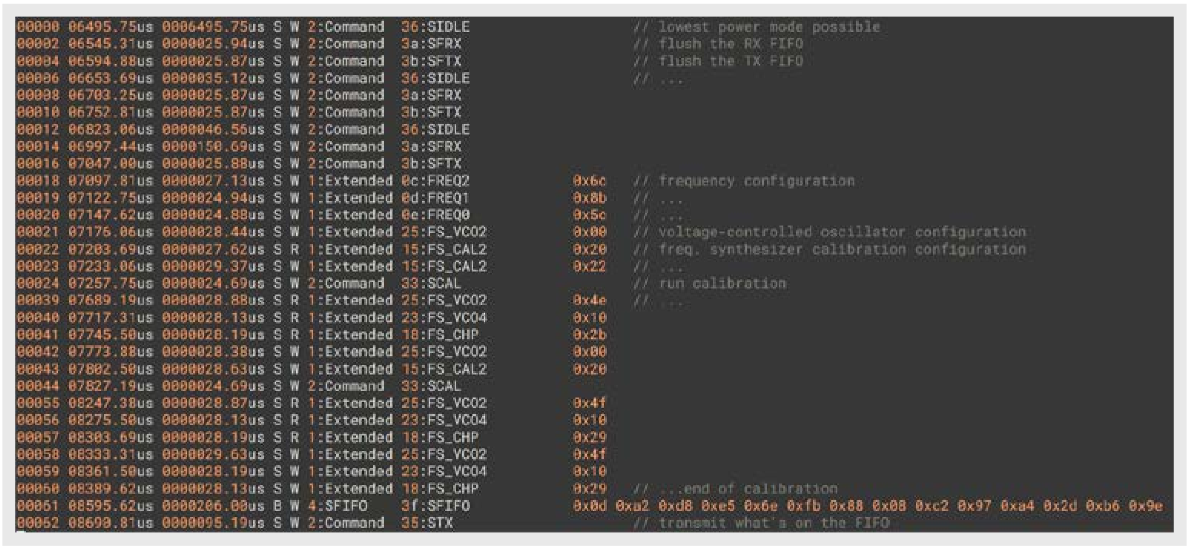
0x2F	EXTENDED ADDRESS				Command Strobes
0x30	SRES		SRES		
0x31	SFSTXON		SFSTXON		
0x32	SXOFF		SXOFF		
0x33	SCAL		SCAL		
0x34	SRX		SRX		
0x35	STX		STX		
0x36	SIDLE		SIDLE		
0x37	SAFC		SAFC		
0x38	SWOR		SWOR		
0x39	SPWD		SPWD		
0x3A	SFRX		SFRX		
0x3B	SFTX		SFTX		
0x3C	SWORRST		SWORRST		
0x3D	SNOP		SNOP		
0x3E	DIRECT MEMORY ACCESS				
0x3F	TX FIFO	TX FIFO	RX FIFO	RX FIFO	

Figure 36. SPI address space (with register names) of the CC1120[52]

(Courtesy of Texas Instruments Incorporated)

In particular, we were interested in decoding the register values that set the various radio parameters and the command transactions. For example, there is an SFIFO command used to write data onto a TX/RX FIFO buffer, an STX/SRX command to transmit/receive data, and so on. These commands are used by the microcontroller to "instruct" the CC1120 to "prepare the radio and send data."

Given an SPI log as input, our emulator generated this informative trace:



```
00000 06495.75us 0006495.75us S W 2:Command  36:SIDLE                   // lowest power mode possible
00002 06545.31us 0000025.94us S W 2:Command  3a:SFRX                    // flush the RX FIFO
00004 06594.88us 0000025.87us S W 2:Command  3b:SFTX                    // flush the TX FIFO
00006 06653.69us 0000035.12us S W 2:Command  36:SIDLE                   // ...
00008 06703.25us 0000025.87us S W 2:Command  3a:SFRX
00010 06752.81us 0000025.87us S W 2:Command  3b:SFTX
00012 06823.06us 0000046.56us S W 2:Command  36:SIDLE
00014 06997.44us 0000150.69us S W 2:Command  3a:SFRX
00016 07047.00us 0000025.88us S W 2:Command  3b:SFTX
00018 07097.81us 0000027.13us S W 1:Extended 0c:FREQ2          0x6c   // frequency configuration
00019 07122.75us 0000024.94us S W 1:Extended 0d:FREQ1          0x8b   // ...
00020 07147.62us 0000024.88us S W 1:Extended 0e:FREQ0          0x5c   // ...
00021 07176.06us 0000028.44us S W 1:Extended 25:FS_VCO2        0x00   // voltage-controlled oscillator configuration
00022 07203.69us 0000027.62us S R 1:Extended 15:FS_CAL2        0x20   // freq. synthesizer calibration configuration
00023 07233.06us 0000029.37us S W 1:Extended 15:FS_CAL2        0x22   // ...
00024 07257.75us 0000024.69us S W 2:Command  33:SCAL                   // run calibration
00039 07689.19us 0000028.88us S R 1:Extended 25:FS_VCO2        0x4e   // ...
00040 07717.31us 0000028.13us S R 1:Extended 23:FS_VCO4        0x10
00041 07745.50us 0000028.19us S R 1:Extended 18:FS_CHP         0x2b
00042 07773.88us 0000028.38us S W 1:Extended 25:FS_VCO2        0x00
00043 07802.50us 0000028.63us S W 1:Extended 15:FS_CAL2        0x20
00044 07827.19us 0000024.69us S W 2:Command  33:SCAL
00055 08247.38us 0000028.87us S R 1:Extended 25:FS_VCO2        0x4f
00056 08275.50us 0000028.13us S R 1:Extended 23:FS_VCO4        0x10
00057 08303.69us 0000028.19us S R 1:Extended 18:FS_CHP         0x29
00058 08333.31us 0000029.63us S W 1:Extended 25:FS_VCO2        0x4f
00059 08361.50us 0000028.19us S W 1:Extended 23:FS_VCO4        0x10
00060 08389.62us 0000028.13us S W 1:Extended 18:FS_CHP         0x29   // ...end of calibration
00061 08595.62us 0000206.00us B W 4:SFIFO    3f:SFIFO          0x0d 0xa2 0xd8 0xe5 0x6e 0xfb 0x88 0x08 0xc2 0x97 0xa4 0x2d 0xb6 0x9e
00062 08690.81us 0000095.19us S W 2:Command  35:STX                     // transmit what's on the FIFO
```

Figure 37. Decoded informative trace showing the initialization routine, the configuration of the registers, and the transmission of a packet (SFIFO command followed by STX)

The trace contained, in order, the initialization routine, the configuration of the registers, and the transmission of a packet (SFIFO command followed by STX).

Note: The content of a packet captured over the air or via SPI is the same except for the two additional bytes — handled by the RF transceiver and transmitted over the air — that indicate the quality of the communication (link quality indicator or LQI and received signal strength indicator or RSSI).

Custom SDR transponder. Using the information collected from these traces, we developed a custom SDR transponder in the form of a GNU Radio flowchart that implemented the physical layer of the communication protocol (i.e., the construction of the packet and its modulation/demodulation). Note that during the construction of the packet, we had to consider the computation of the checksum (CRC-16) which, on the Juuko remote, is computed by the RF transceiver.

We verified that our flowchart worked well in practice by comparing the data transmitted with the flowchart and the data received on the SPI, which were the same. In practice, we verified that the content of the transmission matched the content of the FIFO right before the STX command. Receiving a packet follows the same procedure (with SRX instead of STX). By reading the register configuration, with the aid of Texas Instruments' SmartRF Studio (a development software that eases the configuration of various Chipcon radio transceivers, including the CC1120), we were able to confirm that the microcontroller was not using a custom packet format. Thus, we could rely on the standard packet format as reported in the CC1120 programming user guide.

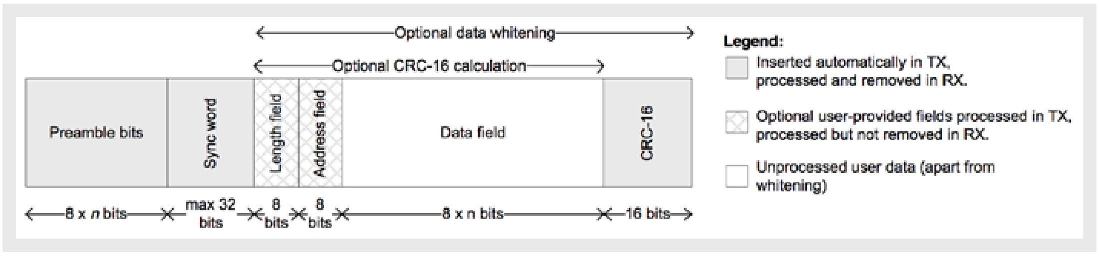

Optional data whitening
Optional CRC-16 calculation
Preamble bits
Sync word
Length field
Address field
Data field
CRC-16
8 x n bits
max 32 bits
8 bits
8 bits
8 x n bits
16 bits
Legend:
Inserted automatically in TX, processed and removed in RX.
Optional user-provided fields processed in TX, processed but not removed in RX.
Unprocessed user data (apart from whitening)

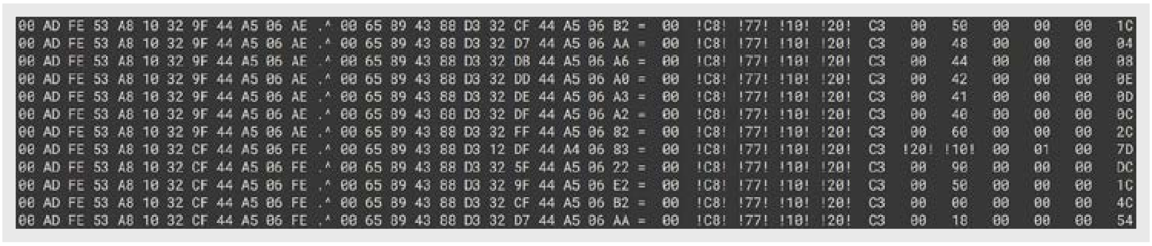
Figure 38. CC1120 standard packet structure, which is used by the Juuko industrial radio controllers *(Courtesy of Texas Instruments Incorporated)*

Packet reverse engineering. At this point, we had a reliable SDR transponder that we could use to transmit arbitrary data to the Juuko receiver, as well as for demodulating and receiving valid radio signals of the protocol. The next step in our analysis was understanding how the commands (like the pressure of a button on the remote) were encoded and communicated *at the application layer* to the receiver.

We started by capturing several traces corresponding to the pressure of one button. We then extracted the *data field* from these traces. An analysis of these values revealed that the very same button pressure was mapped onto a very diverse set of values. This indicated the presence of some form of obfuscation. We noticed, however, that the first byte of these fields was taking all of the 256 values of the "1 byte value range," i.e., 0x00–0xFF. Also, we could not find two identical packets having the same value on the first byte. This suggested that this first value could have been used as a sequential identifier (SID).

At that point, we enumerated all of these 256 “versions” by recording very long button pressures (five minutes) and removing duplicates. We repeated the same procedure for all commands (like “start button” or “button 1”).

By sorting these 256 “versions” and comparing them across distinct command values (CMD), we were able to derive the structure of the data field. To this end, we followed a differential approach. Using the serial programmer and software, we set the pairing ID to 0x00000000 on both TX and RX, in the hope that this would make binary math operations visible by just looking at several packets. At this point, we XORed each couple of packets with the same SID and CMD values and obtained the results shown below.



```
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 CF 44 A5 06 B2 =  00  !C8! !77! !10! !20!  C3   00   50   00   00   00   1C
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 D7 44 A5 06 AA =  00  !C8! !77! !10! !20!  C3   00   48   00   00   00   04
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 DB 44 A5 06 A6 =  00  !C8! !77! !10! !20!  C3   00   44   00   00   00   08
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 DD 44 A5 06 A0 =  00  !C8! !77! !10! !20!  C3   00   42   00   00   00   0E
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 DE 44 A5 06 A3 =  00  !C8! !77! !10! !20!  C3   00   41   00   00   00   0D
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 DF 44 A5 06 A2 =  00  !C8! !77! !10! !20!  C3   00   40   00   00   00   0C
00 AD FE 53 A8 10 32 9F 44 A5 06 AE .^ 00 65 89 43 88 D3 32 FF 44 A5 06 82 =  00  !C8! !77! !10! !20!  C3   00   60   00   00   00   2C
00 AD FE 53 A8 10 32 CF 44 A5 06 FE .^ 00 65 89 43 88 D3 12 DF 44 A4 06 83 =  00  !C8! !77! !10! !20!  C3  !20! !10!  00   01   00   7D
00 AD FE 53 A8 10 32 CF 44 A5 06 FE .^ 00 65 89 43 88 D3 32 5F 44 A5 06 22 =  00  !C8! !77! !10! !20!  C3   00   90   00   00   00   DC
00 AD FE 53 A8 10 32 CF 44 A5 06 FE .^ 00 65 89 43 88 D3 32 9F 44 A5 06 E2 =  00  !C8! !77! !10! !20!  C3   00   50   00   00   00   1C
00 AD FE 53 A8 10 32 CF 44 A5 06 FE .^ 00 65 89 43 88 D3 32 CF 44 A5 06 B2 =  00  !C8! !77! !10! !20!  C3   00   00   00   00   00   4C
00 AD FE 53 A8 10 32 CF 44 A5 06 FE .^ 00 65 89 43 88 D3 32 D7 44 A5 06 AA =  00  !C8! !77! !10! !20!  C3   00   18   00   00   00   54
```

Figure 39. The packets (data field only) on the left were recorded with the fixed code set to 0x00000000 and the packets on the right were recorded with the fixed code set to 0x201077C8, which was revealed (modulo swapped endianness) by the XOR operation.

At this point, we could make a better guess on the data field’s structure:

```
[SID][PACKET_CODE (4 bytes)][SUM1][0x00][CMD][0x000000][SUM2]
```

Our initial assumption was correct: The SID byte indicates the “version” of each obfuscation round. The PACKET_CODE is the result of a function applied to the fixed code and SID. We believe that the SID is used as a key in this function. SUM1 and SUM2 are two checksums. CMD encodes the command, with the exception of special commands (start and stop), which are encoded at byte 9 (instead of 10).

As a result, we found the first vulnerability, which allows an attacker to obtain the (hidden) pairing ID by capturing one single packet from the air and computing the differential analysis that we described. This vulnerability allowed us to isolate all 10 distinct commands:

- Heartbeat (sent periodically to notify the receiver that the transmitter is within range).
- Movement commands (to trigger the eight relays on the RX board).
- Start and e-stop commands.

To exploit the first vulnerability, an attacker needs to enumerate all of the 256 packets (for each CMD) upon setting the pairing ID to 0x00000000 and store them in a table indexed by SID and CMD. When capturing a target packet, the attacker performs a lookup in the table using the SID and CMD from the captured target packet. At this point, the target packet and the packet from the tables can be XORed to reveal the target packet's pairing ID. A compact table (only 256 entries) can be obtained by fixing CMD = 0x00 (heartbeat). This choice is practical and realistic: Because the TX and RX periodically (e.g., every one to 10 seconds) exchange heartbeat packets as a "keep alive" beacon, it's very likely that the attacker would be able to catch one.

The second vulnerability that we found allowed us to implement the command injection attack. By capturing an arbitrary packet, the attacker could just XOR the right value with CMD and SUM2 to obtain an arbitrary command. If the captured packet encodes a heartbeat command (which is very likely to be recorded and recognized since it's periodic), such a "right value" is just the command byte: 0x01 for button 1, 0x02 for button 2, and so on. Otherwise, if the captured packet encodes some other command, some simple XOR math is needed to obtain that right value. Here the attacker model is very simple: We only assume that the attacker is within the transmission range (i.e., hundreds or thousands of meters from the RX).

We believe that the SID is intended to be used to create a sort of rolling-code mechanism. However, the actual value of the SID is ignored from the RX. This is confirmed by the fact that we've previously demonstrated a successful replay attack, and when forging new packets, the RX accepts any packet that honors the intra-packet rules, which in turn implies that each packet contains enough information to calculate the per-packet code. To eliminate this vulnerability, the RX should enforce checks on both the SID and the fixed code. This can be obtained by using a clock to keep a counter up to date, from which the SID should be derived. In this regard, we're surprised that the heartbeat packets are not used to this end; they seem designed exactly for synchronization purposes. Given that the radio transceiver used by the target device (CC1120) supports up to 128-byte radio packets, we suggest fully taking advantage of this space.

4.3 Implementing Attack 3: E-Stop Abuse

(ZDI-CAN-6183,* ZDI-CAN-6184,[53] ZDI-CAN-6185,** ZDI-CAN-6187,*** CVE-2018-17903,[54, 55] CVE-2018-17935,[56, 57] CVE-2018-19023[58, 59])

* The product under testing has reached end of life and is no longer supported by Autec.
** The product under testing was designed to specification and no fix is forthcoming from Circuit Design.
*** The product under testing has reached end of life and is no longer supported by Elca.

The implementation of the e-stop abuse is a consequence of the work we described so far. It is essentially an infinite loop that transmits a reset (or e-stop) packet over time. We tested it against the controllers that we had at our disposal, and the effect was that, even if a legitimate TX unit was controlling the equipment attached to the RX unit, the e-stop packets transmitted by the malicious transmitter were received and interpreted anyway, causing the legitimate commands to be ignored. This happened because the RX kept the safety relays off all the time during the attack. This attack could be used by a miscreant for sabotage.

In a second experiment, we tried to forcefully restart the connected equipment using a legitimate TX unit to send one or more “start” commands rapidly. We simulated the scenario of an operator willing to restart a controller attacked by an e-stop abuse attack. However, as soon as the legitimate “start” command was received by the controller (causing the relays to turn on), the e-stop abuse attack would disable it immediately. Among the devices that we tested, some implemented a “busy channel” mechanism that keeps the selected channel occupied by continuously transmitting a heartbeat command. Regardless of this, a slightly more powerful, or closer, (malicious) transmitter is all it takes for the denial of service to succeed.

4.4 Implementing Attack 4: Malicious Re-Pairing

(CVE-2018-17921[60, 61])

To test this attack, we had to enable the pairing functionality of our tested remote controllers (Saga and Juuko) by connecting the TX and RX units to the serial programmers. Our controllers had this functionality disabled at the time of purchase. According to the reseller, the function can be enabled by technicians upon request.

We then initiated a pairing procedure by turning on the RX unit, and within a four-minute window, we pushed buttons in a given order on the new TX unit according to the feature documentation provided by the vendor. During this procedure, we captured the samples using the BladeRF and replayed them at the frequency of the victim RX. We verified that the victim RX stopped taking commands from the original TX and became responsive to our malicious TX. We then brought the RX back to the original state using the documented button sequence.

By looking at the protocol used to pair a Saga remote controller, we found it to be simple in structure and not encrypted. It is very similar to ordinary commands, except for the preamble (0F for an ordinary command):

Preamble	Sync	Device ID	Pairing	Sync
F0	05 55 50	27 41 63 44 36	55 50 50 11	0F

Table 7. Command used by Saga units to clone a remote

At this point, we emulated an attacker willing to maliciously pair their TX unit with the vulnerable RX unit, and we were able to forge a malicious re-pairing packet for any device ID. As a result, the original TX was no longer functional, as it had been taken over by the attacker. It is thus possible for the adversary to commit a massive DoS attack that prevents all industrial controllers in a factory, harbor, or construction site from working, or to combine the attack with Attack 2 (command injection) to cause severe damages.

This attack has been confirmed to work on the Saga remote controller. More notable, though, is that even if the Juuko controller requires the receiver unit to be physically opened to push a button in order to set it in "pairing mode," this attack can be implemented anyway *without access to the receiver*. Thanks to the code disclosure vulnerability that we have found, all the attacker needs to do is capture one single packet, from which they will be able to recover the fixed code. Recall that the fixed code is the only secret shared between the transmitter and the receiver. So, a receiver and a transmitter with the same code would effectively be paired. At this point, the attacker can just program the new transmitter using the stolen code.

4.5 Implementing Attack 5: Malicious Re-Programming and Remote Attack Vectors

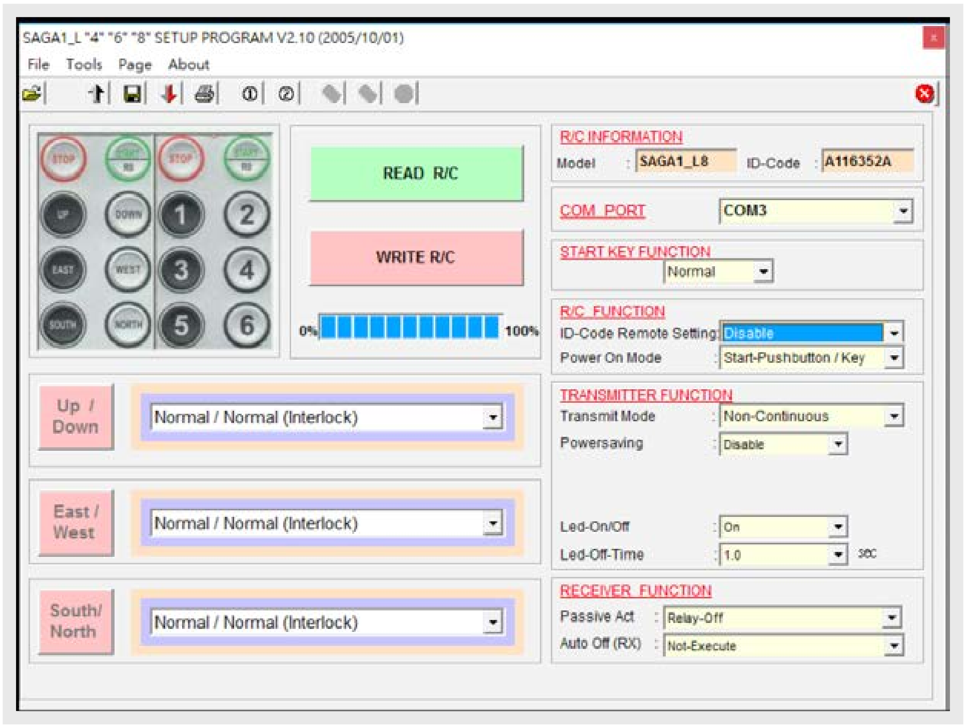
(CVE-2018-17921[62, 63])

To verify whether a remote controller could be reprogrammed without the user's consent (while attached to a compromised computer), we obtained a copy of the programming software and, when feasible, the dedicated serial dongle as well. During our research, this was possible for Juuko and Saga, while for Telecrane we sought confirmatory evidence from a technical manual.

Vendor	User-programmability	Protection
Saga	Yes	No
Juuko	Yes	No
Telecrane	Yes	No*
Hetronic	Yes	Not tested
Scanreco	Yes	Not tested

Table 8. Outcome of our test for Attack 5 (malicious reprogramming)
(*Verified through technical documentation)

For the tested Saga, Juuko, and Telecrane devices, we found no protection such as an authentication dialog against malicious reprogramming, as shown by the following series of figures. As a result, a malicious actor or a piece of malware could perform automated attacks and persistent compromises. Interestingly, the programming software is 100-percent based on Windows (and sometimes Windows-XP-only), which is the top operating system targeted by mass malware and advanced persistent threats (APTs).



SAGA1_L "4" "6" "8" SETUP PROGRAM V2.10 (2005/10/01)

R/C INFORMATION
Model : SAGA1_L8 ID-Code : A116352A

COM PORT COM3

START KEY FUNCTION Normal

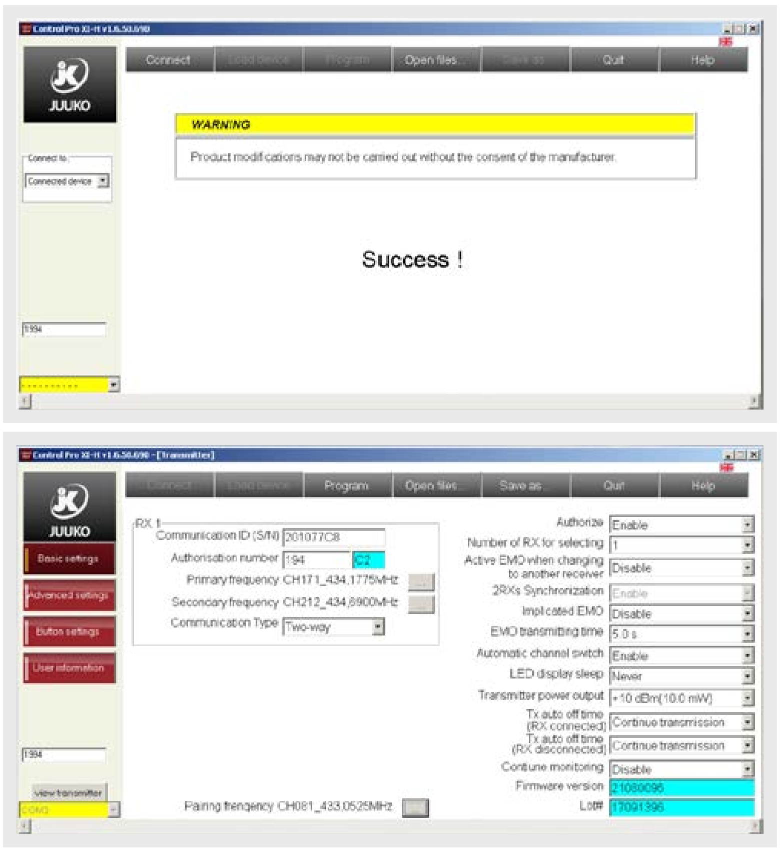
R/C FUNCTION
ID-Code Remote Setting: Disable
Power On Mode : Start-Pushbutton / Key

TRANSMITTER FUNCTION
Transmit Mode : Non-Continuous
Powersaving : Disable
Led-On/Off : On
Led-Off-Time : 1.0 sec

RECEIVER FUNCTION
Passive Act : Relay-Off
Auto Off (RX) : Not-Execute

Up / Down: Normal / Normal (Interlock)
East / West: Normal / Normal (Interlock)
South/ North: Normal / Normal (Interlock)

Figure 40. Screenshot of the Saga programming software, with no protection against unattended reprogramming



WARNING

Product modifications may not be carried out without the consent of the manufacturer.

Success !

Connect to: Connected device

1:594

Control Pro XI-II v1.6.50.6/90 - [Transmitter]

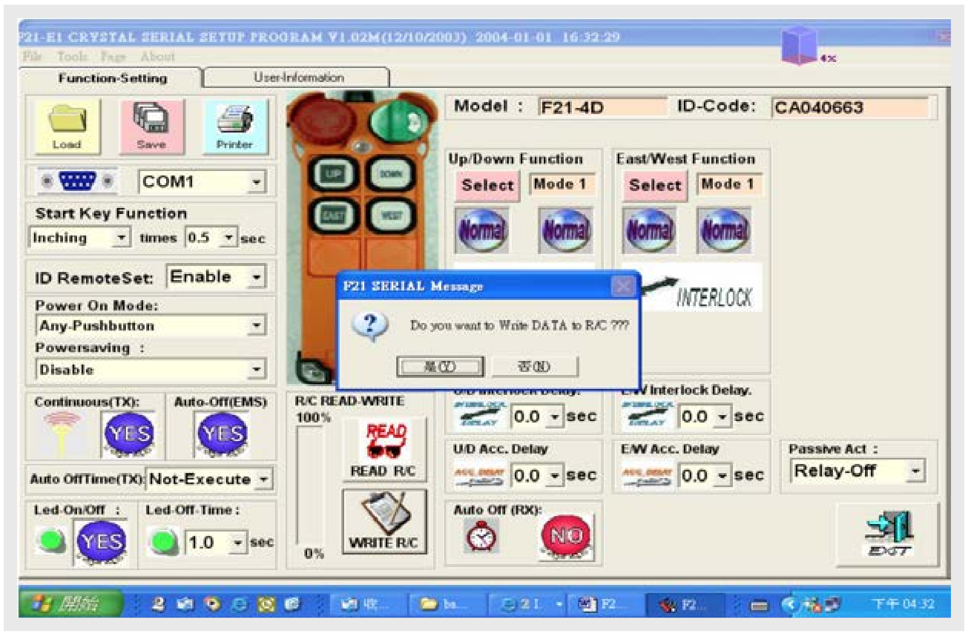
RX 1
Communication ID (S/N) 201077C8
Authorisation number 1:94 C2
Primary frequency CH171_434,1775MHz
Secondary frequency CH212_434,6900MHz
Communication Type Two-way

Authorize Enable
Number of RX for selecting 1
Active EMO when changing to another receiver Disable
2RXs Synchronization Enable
Implicated EMO Disable
EMO transmitting time 5.0 s
Automatic channel switch Enable
LED display sleep Never
Transmitter power output +10 dBm(10.0 mW)
Tx auto off time (RX connected) Continue transmission
Tx auto off time (RX disconnected) Continue transmission
Continue monitoring Disable
Firmware version 21080096
Lot# 17091396

Pairing frengency CH081_433,0525MHz

COM3

Figure 41. Screenshots of the Juuko programming software, with a warning about unauthorized reprogramming



Model : F21-4D
ID-Code: CA040663

COM1
Start Key Function: Inching times 0.5 sec
ID RemoteSet: Enable
Power On Mode: Any-Pushbutton
Powersaving : Disable
Continuous(TX): YES
Auto-Off(EMS): YES
Auto OffTime(TX): Not-Execute
Led-On/Off : YES
Led-Off-Time : 1.0 sec

R/C READ-WRITE 100% 0%

Up/Down Function Mode 1
East/West Function Mode 1

F21 SERIAL Message

Do you want to Write DATA to R/C ???

U/D Interlock Delay: 0.0 sec
E/W Interlock Delay. 0.0 sec
U/D Acc. Delay 0.0 sec
E/W Acc. Delay 0.0 sec
Passive Act : Relay-Off
Auto Off (RX): NO

Figure 42. Screenshot obtained from the Telecrane programming manual, not mentioning any authentication steps[64]

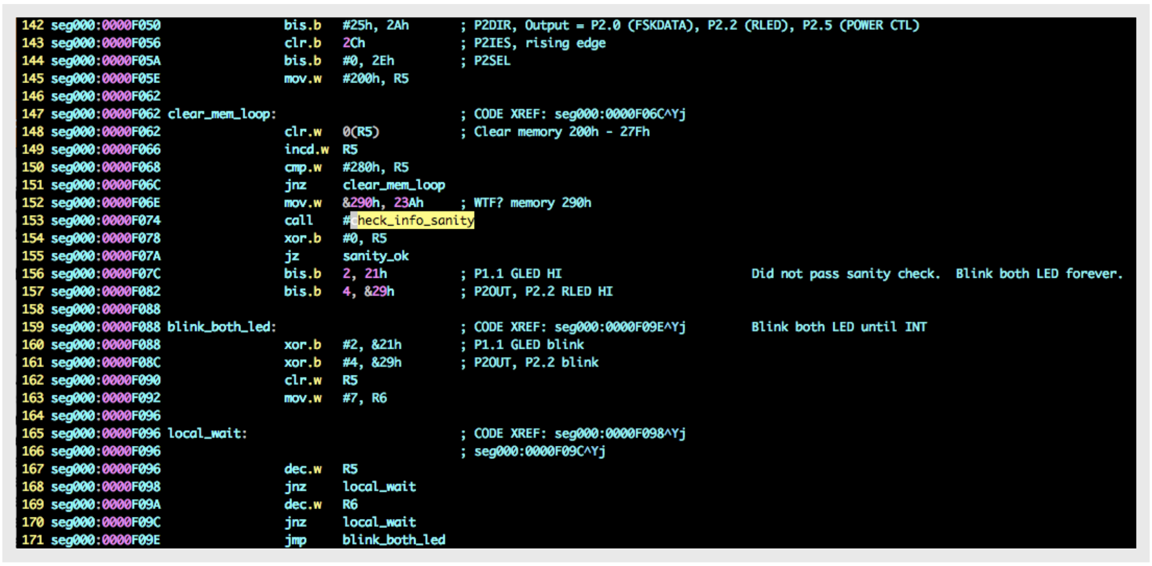
Firmware analysis. Moving one step ahead, we sniffed the Saga's traffic over the programmer cable and confirmed that an adversary would be able to inject malicious code into the flash memory when the controller is connected to a computer (e.g., for maintenance or configuration). The attacker model considers that the adversary has control of such a computer.

The controller is based on Texas Instruments' MSP430F1101A. While the JTAG (Joint Test Action Group) fuse is blown to protect unauthorized access, the mass-erase protection of the bootstrap loader (BSL) is not enabled for maximum security, because Saga has multiple versions of firmware in the market. As there is no hardware protection circuit, the BSL password can be sniffed with the purchased programmer cable (a standard UART TTL adapter with a six-pin connector) by using Bus Pirate or a logic analyzer.

As a result, we sniffed the traffic and recovered two BSL passwords, along with the parameters updated into the flash memory:

```
TX(UART) 80 (Sync)

RX(UART) 90

TX(UART)  80 10 24 24  E0 FF 20 00  00 F0 98 F4 98 F4 98 F4 98 F4 00 F0 72 F3 00 F0 00
F0 72 F3 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 9B 34 (80102424 = RX password, AX = FFE0h,
number of bytes = 0020h, checksum = 349Bh)

RX(UART) A0 (DATA_NAK) so we know it checks several possible BSL passwords. Mass erase
should have been disabled.

TX(UART) 80 (Sync)

RX(UART) 90 (DATA_ACK)

TX(UART) 80 10 24 24 E0 FF 20 00 00 F0 00 F0 00 FD 00 FD 00 FD 00 F0 00 FA 00 F0 00 F0
00 FA 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 9B 39 (RX password, AX = FFE0h, number of
bytes = 0020h, checksum = 399Bh)

RX(UART) 90 (DATA_ACK)
```

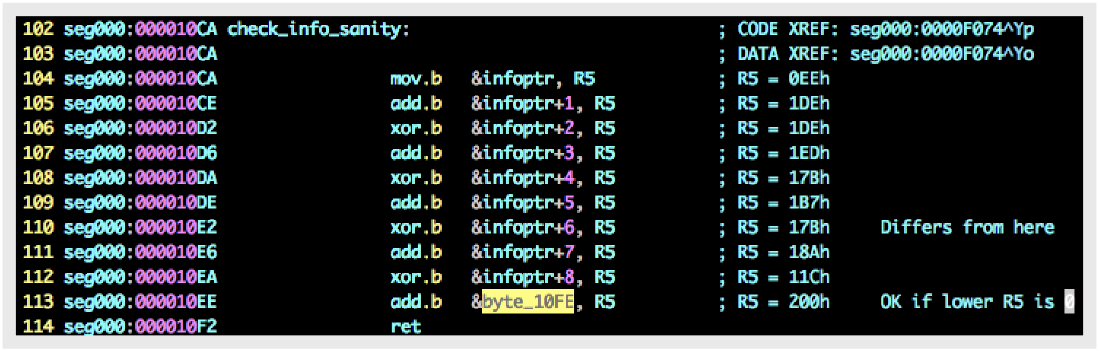
With the BSL password obtained over the programmer's cable, we were able to read the flash memory, change the device settings in the flash memory, update the firmware, and extract the firmware for research. We extracted the firmware and found a block of flash memory, which is also updated by the vendor's programming software, containing a subroutine that is executed in every boot.



```
seg000:0000F050                bis.b   #25h, 2Ah       ; P2DIR, Output = P2.0 (FSKDATA), P2.2 (RLED), P2.5 (POWER CTL)
seg000:0000F056                clr.b   2Ch             ; P2IES, rising edge
seg000:0000F05A                bis.b   #0, 2Eh         ; P2SEL
seg000:0000F05E                mov.w   #200h, R5
seg000:0000F062
seg000:0000F062 clear_mem_loop:                        ; CODE XREF: seg000:0000F06C^Yj
seg000:0000F062                clr.w   0(R5)           ; Clear memory 200h - 27Fh
seg000:0000F066                incd.w  R5
seg000:0000F068                cmp.w   #280h, R5
seg000:0000F06C                jnz     clear_mem_loop
seg000:0000F06E                mov.w   &290h, 23Ah     ; WTF? memory 290h
seg000:0000F074                call    #check_info_sanity
seg000:0000F078                xor.b   #0, R5
seg000:0000F07A                jz      sanity_ok
seg000:0000F07C                bis.b   2, 21h          ; P1.1 GLED HI                Did not pass sanity check.  Blink both LED forever.
seg000:0000F082                bis.b   4, &29h         ; P2OUT, P2.2 RLED HI
seg000:0000F088
seg000:0000F088 blink_both_led:                        ; CODE XREF: seg000:0000F09E^Yj        Blink both LED until INT
seg000:0000F088                xor.b   #2, &21h        ; P1.1 GLED blink
seg000:0000F08C                xor.b   #4, &29h        ; P2OUT, P2.2 blink
seg000:0000F090                clr.w   R5
seg000:0000F092                mov.w   #7, R6
seg000:0000F096
seg000:0000F096 local_wait:                            ; CODE XREF: seg000:0000F098^Yj
seg000:0000F096                                        ; seg000:0000F09C^Yj
seg000:0000F096                dec.w   R5
seg000:0000F098                jnz     local_wait
seg000:0000F09A                dec.w   R6
seg000:0000F09C                jnz     local_wait
seg000:0000F09E                jmp     blink_both_led
```

Figure 43. Some details of the disassembled Saga firmware (startup subroutine)

While this code was meant to check the integrity of the flash memory (see Figure 44), it could be replaced by malicious code. A determined adversary could also replace part of the firmware to create a “time bomb” that initiates unexpected behavior after a certain number of buttons or a predefined keystroke.



```
seg000:000010CA check_info_sanity:                     ; CODE XREF: seg000:0000F074^Yp
seg000:000010CA                                        ; DATA XREF: seg000:0000F074^Yo
seg000:000010CA                mov.b   &infoptr, R5    ; R5 = 0EEh
seg000:000010CE                add.b   &infoptr+1, R5  ; R5 = 1DEh
seg000:000010D2                xor.b   &infoptr+2, R5  ; R5 = 1DEh
seg000:000010D6                add.b   &infoptr+3, R5  ; R5 = 1EDh
seg000:000010DA                xor.b   &infoptr+4, R5  ; R5 = 17Bh
seg000:000010DE                add.b   &infoptr+5, R5  ; R5 = 1B7h
seg000:000010E2                xor.b   &infoptr+6, R5  ; R5 = 17Bh     Differs from here
seg000:000010E6                add.b   &infoptr+7, R5  ; R5 = 18Ah
seg000:000010EA                xor.b   &infoptr+8, R5  ; R5 = 11Ch
seg000:000010EE                add.b   &byte_10FE, R5  ; R5 = 200h     OK if lower R5 is
seg000:000010F2                ret
```

Figure 44. The subroutine that checks flash integrity

We also verified from the extracted firmware that our analysis of Saga’s button encoding was correct, and we were able to calculate the combination of buttons.

In conclusion, an adversary could implement a persistent attack on a remote controller by taking control of the programming software, for example via malware or a malicious insider. An attacker could also modify the BSL password and enable the “mass erase” feature to make forensics and incident response harder, if not impossible.

5. Research and Attack Tools: Introducing RFQuack

During our research, we experimented with various attack devices: several SDR devices, the famous Yard Stick One RF-hacking dongle, and the recent PandwaRF tool. Given the obstacles that we encountered, we conclude that a step forward is needed to offer more versatile research tools. This realization prompted us to develop RFQuack.

Tool	Examples	Versatility	Development effort	Efficiency	Cost range	Maturity
Software-defined radios	RTL-SDR, BladeRF, Ettus	Very high	High	Low	US$20–2,000	Very high
Hardware tools	Yard Stick One, PandwaRF	Intermediate	Low	High	US$90–150	Product
Hybrid radios	RFQuack	High	Intermediate	Intermediate	US$20–60	Concept

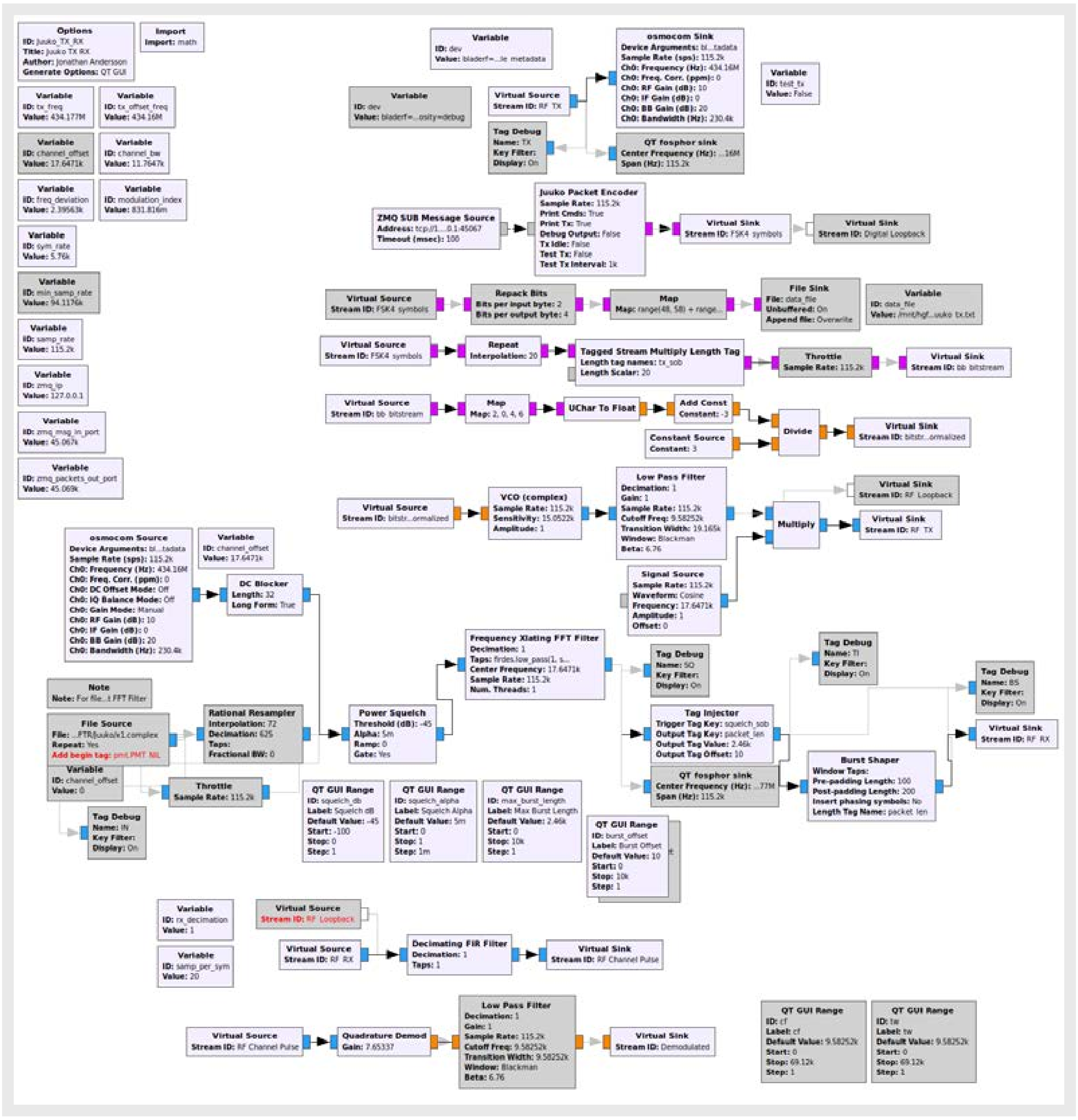
Table 9. Comparison of various RF research and attack tools

Similar in spirit to RFCat (the software used by the Yard Stick One and compatible dongles), RFQuack supports basic operations such as sniffing and transmitting RF packets. Additionally, we implement, on the firmware side (for more efficiency), on-the-fly packet modification.

RFQuack is designed to be modular: New RF transceivers with special features can be purchased and plugged to the standard serial interface. On the software side, almost all RF transceivers are already supported, and based on our experience, developing a new driver takes as little as a couple of days of development work.

5.1 Software-Defined Radios

Using SDRs is convenient only if the modulation scheme is already implemented in the software library, or can be easily and efficiently implemented. For vendors that use plain 2-FSK, such as Saga, we found no barriers and were able to implement an attack script to forge arbitrary packets. For vendors that use mixed 2-GFSK and 4-GFSK modulation, such as Juuko, we had to come up with our implementation, which, to be honest, was all but trivial, as Figure 45 shows. As a result, after we successfully performed our attacks via SDR, we also looked for a more efficient and practical research and exploitation method.


Options ID: juuko_TX_RX Title: Juuko TX RX Author: Jonathan Andersson Generate Options: QT GUI
Import Import: math
Variable ID: tx_freq Value: 434.177M
Variable ID: tx_offset_freq Value: 434.16M
Variable ID: channel_offset Value: 17.6471k
Variable ID: channel_bw Value: 11.7647k
Variable ID: freq_deviation Value: 2.39563k
Variable ID: modulation_index Value: 831.816m
Variable ID: sym_rate Value: 5.76k
Variable ID: min_samp_rate Value: 94.1176k
Variable ID: samp_rate Value: 115.2k
Variable ID: zmq_ip Value: 127.0.0.1
Variable ID: zmq_msg_in_port Value: 45.067k
Variable ID: zmq_packets_out_port Value: 45.069k
Variable ID: dev Value: bladerf=...le metadata
Variable ID: dev Value: bladerf=...osity=debug
Virtual Source Stream ID: RF TX
Tag Debug Name: TX Key Filter: Display: On
osmocom Sink Device Arguments: bl...tadata Sample Rate (sps): 115.2k Ch0: Frequency (Hz): 434.16M Ch0: Freq. Corr. (ppm): 0 Ch0: RF Gain (dB): 10 Ch0: IF Gain (dB): 0 Ch0: BB Gain (dB): 20 Ch0: Bandwidth (Hz): 230.4k
QT fosphor sink Center Frequency (Hz): ...16M Span (Hz): 115.2k
Variable ID: test_tx Value: False
ZMQ SUB Message Source Address: tcp://1....0.1:45067 Timeout (msec): 100
Juuko Packet Encoder Sample Rate: 115.2k Print Cmds: True Print Tx: True Debug Output: False Tx Idle: False Test Tx: False Test Tx Interval: 1k
Virtual Sink Stream ID: FSK4 symbols
Virtual Sink Stream ID: Digital Loopback
Virtual Source Stream ID: FSK4 symbols
Repack Bits Bits per input byte: 2 Bits per output byte: 4
Map Map: range(48, 58) + range...
File Sink File: data_file Unbuffered: On Append file: Overwrite
Variable ID: data_file Value: /mnt/hgf...uuko tx.txt
Virtual Source Stream ID: FSK4 symbols
Repeat Interpolation: 20
Tagged Stream Multiply Length Tag Length tag names: tx_sob Length Scalar: 20
Throttle Sample Rate: 115.2k
Virtual Sink Stream ID: bb bitstream
Virtual Source Stream ID: bb bitstream
Map Map: 2, 0, 4, 6
UChar To Float
Add Const Constant: -3
Constant Source Constant: 3
Divide
Virtual Sink Stream ID: bitstr...ormalized
Virtual Source Stream ID: bitstr...ormalized
VCO (complex) Sample Rate: 115.2k Sensitivity: 15.0522k Amplitude: 1
Low Pass Filter Decimation: 1 Gain: 1 Sample Rate: 115.2k Cutoff Freq: 9.58252k Transition Width: 19.165k Window: Blackman Beta: 6.76
Signal Source Sample Rate: 115.2k Waveform: Cosine Frequency: 17.6471k Amplitude: 1 Offset: 0
Multiply
Virtual Sink Stream ID: RF Loopback
Virtual Sink Stream ID: RF TX
osmocom Source Device Arguments: bl...tadata Sample Rate (sps): 115.2k Ch0: Frequency (Hz): 434.16M Ch0: Freq. Corr. (ppm): 0 Ch0: DC Offset Mode: Off Ch0: IQ Balance Mode: Off Ch0: Gain Mode: Manual Ch0: RF Gain (dB): 10 Ch0: IF Gain (dB): 0 Ch0: BB Gain (dB): 20 Ch0: Bandwidth (Hz): 230.4k
Variable ID: channel_offset Value: 17.6471k
DC Blocker Length: 32 Long Form: True
Note Note: For file...t FFT Filter
File Source File: ...FTR/juuko/x1.complex Repeat: Yes Add begin tag: pmt.PMT_NIL
Variable ID: channel_offset Value: 0
Rational Resampler Interpolation: 72 Decimation: 625 Taps: Fractional BW: 0
Throttle Sample Rate: 115.2k
Tag Debug Name: IN Key Filter: Display: On
Power Squelch Threshold (dB): -45 Alpha: 5m Ramp: 0 Gate: Yes
Frequency Xlating FFT Filter Decimation: 1 Taps: firdes.low_pass(1, s... Center Frequency: 17.6471k Sample Rate: 115.2k Num. Threads: 1
Tag Debug Name: SQ Key Filter: Display: On
Tag Injector Trigger Tag Key: squelch_sob Output Tag Key: packet_len Output Tag Value: 2.46k Output Tag Offset: 10
QT fosphor sink Center Frequency (Hz): ...77M Span (Hz): 115.2k
Tag Debug Name: TI Key Filter: Display: On
Burst Shaper Window Taps: Pre-padding Length: 100 Post-padding Length: 200 Insert phasing symbols: No Length Tag Name: packet_len
Tag Debug Name: BS Key Filter: Display: On
Virtual Sink Stream ID: RF RX
QT GUI Range ID: squelch_db Label: Squelch dB Default Value: -45 Start: -100 Stop: 0 Step: 1
QT GUI Range ID: squelch_alpha Label: Squelch Alpha Default Value: 5m Start: 0 Stop: 1 Step: 1m
QT GUI Range ID: max_burst_length Label: Max Burst Length Default Value: 2.46k Start: 0 Stop: 10k Step: 1
QT GUI Range ID: burst_offset Label: Burst Offset Default Value: 10 Start: 0 Stop: 10k Step: 1
Variable ID: rx_decimation Value: 1
Variable ID: samp_per_sym Value: 20
Virtual Source Stream ID: RF Loopback
Virtual Source Stream ID: RF RX
Decimating FIR Filter Decimation: 1 Taps: 1
Virtual Sink Stream ID: RF Channel Pulse
Virtual Source Stream ID: RF Channel Pulse
Quadrature Demod Gain: 7.65337
Low Pass Filter Decimation: 1 Gain: 1 Sample Rate: 115.2k Cutoff Freq: 9.58252k Transition Width: 9.58252k Window: Blackman Beta: 6.76
Virtual Sink Stream ID: Demodulated
QT GUI Range ID: cf Label: cf Default Value: 9.58252k Start: 0 Stop: 69.12k Step: 1
QT GUI Range ID: tw Label: tw Default Value: 9.58252k Start: 0 Stop: 69.12k Step: 1

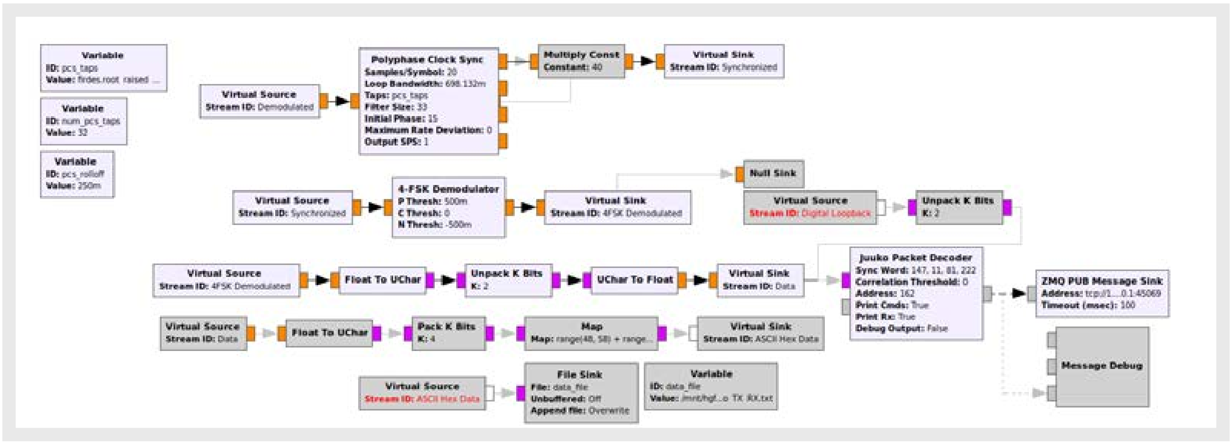
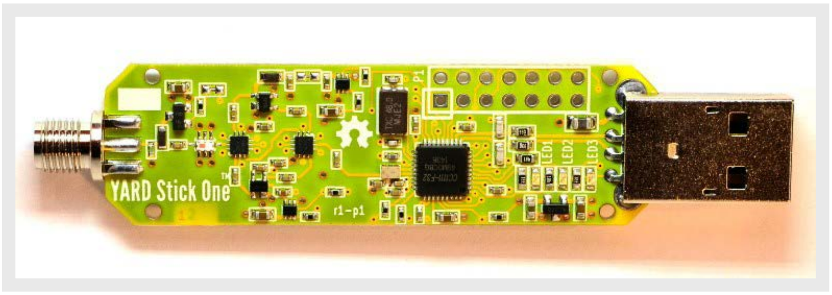

Variable ID: pcs_taps Value: firdes.root_raised ...
Variable ID: num_pcs_taps Value: 32
Variable ID: pcs_rolloff Value: 250m
Virtual Source Stream ID: Demodulated
Polyphase Clock Sync Samples/Symbol: 20 Loop Bandwidth: 698.132m Taps: pcs_taps Filter Size: 33 Initial Phase: 15 Maximum Rate Deviation: 0 Output SPS: 1
Multiply Const Constant: 40
Virtual Sink Stream ID: Synchronized
Null Sink
Virtual Source Stream ID: Synchronized
4-FSK Demodulator P Thresh: 500m C Thresh: 0 N Thresh: -500m
Virtual Sink Stream ID: 4FSK Demodulated
Virtual Source Stream ID: Digital Loopback
Unpack K Bits K: 2
Virtual Source Stream ID: 4FSK Demodulated
Float To UChar
Unpack K Bits K: 2
UChar To Float
Virtual Sink Stream ID: Data
Juuko Packet Decoder Sync Word: 147, 11, 81, 222 Correlation Threshold: 0 Address: 162 Print Cmds: True Print Rx: True Debug Output: False
ZMQ PUB Message Sink Address: tcp://1....0.1:45069 Timeout (msec): 100
Virtual Source Stream ID: Data
Float To UChar
Pack K Bits K: 4
Map Map: range(48, 58) + range...
Virtual Sink Stream ID: ASCII Hex Data
Message Debug
Virtual Source Stream ID: ASCII Hex Data
File Sink File: data_file Unbuffered: Off Append file: Overwrite
Variable ID: data_file Value: /mnt/hgf...o TX RX.txt

Figure 45. Custom GNU Radio receiver and transmitter that we built to support the reverse-engineering process of the Juuko protocol

5.2 RF-Hacking Hardware

The Yard Stick One is a programmable radio dongle based on the CC1111 transceiver, which supports many common modulation schemes in hardware. The client-side interface is a Python library (RFCat) that allows setting or getting register values on the CC1111 to configure the transceiver and send data.



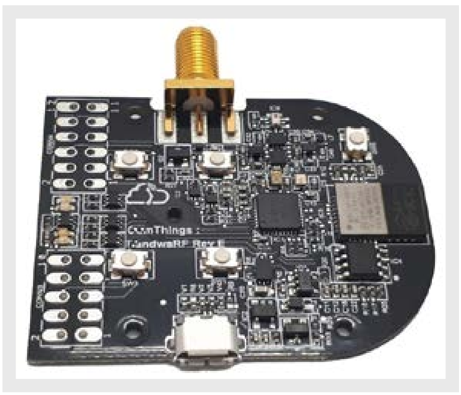
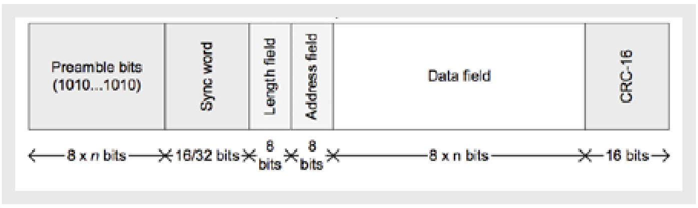


Figure 46. Yard Stick One dongle (left)[65] and PandwaRF device (right)[66]

Whenever the radio transceiver of the targeted device uses any modulation scheme and packet structure supported by the CC1111, using this device is very convenient: All the attacker needs to do is to use RFCat to send the desired data as a byte-stream, and the radio transceiver will take care of the rest. However, when this is not the case, using precooked hardware tools can become a nightmare.

In our case, the radio transceiver of the Juuko radio controller (CC1120) supports up to 32 bits of sync words, while the CC1111 supports up to 16 bits of sync words. The documentation of the CC1111[67] is a bit misleading, because it mentions 32 bits of sync words, which could lead the reader to believe that the CC1111 supports sync words of up to 32 bits. Instead, it actually supports up to 16 bits of sync words (optionally repeated twice, so 16 + 16 bits). Similarly, the support for 4-GFSK is undocumented and there is no easy workaround for using a mixed 2-GFSK and 4-GFSK modulation scheme. At this point, one may reconsider going the SDR way.



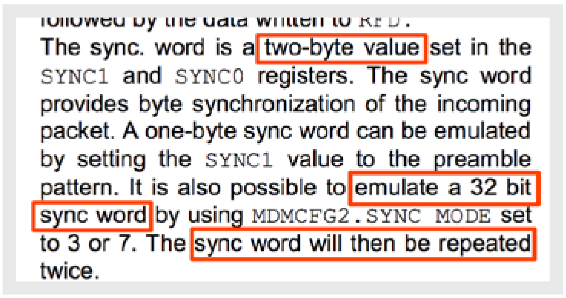
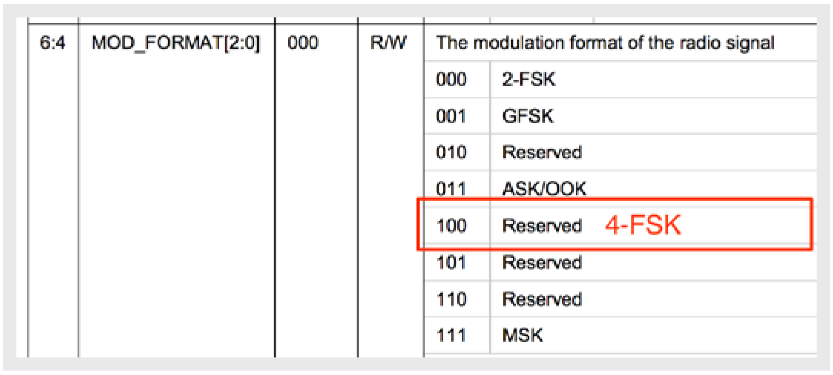

followed by the data written to RFD.
The sync. word is a two-byte value set in the SYNC1 and SYNC0 registers. The sync word provides byte synchronization of the incoming packet. A one-byte sync word can be emulated by setting the SYNC1 value to the preamble pattern. It is also possible to emulate a 32 bit sync word by using MDMCFG2.SYNC MODE set to 3 or 7. The sync word will then be repeated twice.

Figure 47. CC1111 packet structure as schematized on page 194 of the datasheet (left), with misleading detail about the 16/32 sync word support. On page 195 (right), further information clarifies that the 32 bits support is only emulated.

(Courtesy of Texas Instruments Incorporated)



6:4	MOD_FORMAT[2:0]	000	R/W	The modulation format of the radio signal	
				000	2-FSK
				001	GFSK
				010	Reserved
				011	ASK/OOK
				100	Reserved 4-FSK
				101	Reserved
				110	Reserved
				111	MSK

Figure 48. The 4-FSK support in the CC111 is undocumented and it is unclear whether the preamble and sync word are modulated with 2-FSK or not, as in the CC1120.

(Courtesy of Texas Instruments Incorporated)

In this situation, the only option is to configure RFCat to receive any packet that matches the preamble (0xAAAAAA) and filter matching packets on the client side. Sending packets is slightly more complicated — if at all feasible — because we would need to prepare a packet by determining which preamble and sync word *modulated with* 4-GFSK (without knowing how the logic values are mapped to frequency deviations) would create a radio packet that, when demodulated with 2-GFSK (the Juuko RX is set to 2-GFSK for preamble and sync word), would match the intended value expected by the RX. All of this while honoring the expected data rate and data length field.

Similar considerations apply to the recent PandwaRF, which is essentially a Yard Stick One with a Bluetooth Low Energy transceiver on a battery-powered PCB. Technically, it doesn't make any difference because it's still based on the CC1111. From an attacker viewpoint, however, the PandwaRF is more interesting because it's small and portable, it can be hidden anywhere, and thanks to the battery, it can run for days in idle mode. With such a device, with a price that is below US$150, the local attacker model becomes much more realistic.

5.3 RFQuack: Versatile, Next-Generation RF-Hacking Hardware

The versatility of SDR is unparalleled, but its efficiency (in both speed and development time) is very limited. We would like to have a hardware research tool that is as versatile as SDR, yet efficient. Thanks to the modularity of Arduino-compatible hardware, we believe that it is possible nowadays to create a modular hardware tool that can be adapted to support the "quirks" of the targeted RF transceiver. For example, if the target is a CC1120, we need not struggle to use an attack tool that uses the CC1111. And if the next target is a different radio, we could just remove the CC1120 and attach a different one, much like we do when we wire a new block into a GNU Radio flow graph. This is the intuition that motivated us to develop RFQuack.

Given the availability of inexpensive sub-1GHz radio modules with SPI, it makes sense to think that an attacker could just create a custom attack tool to overcome the limitations of SDRs and currently available RF-hacking hardware. The Juuko represented a good case, because the CC1111 was too limiting to target a CC1120 receiver. We purchased for US$12 a CC112x module from DigiRF[68] and an ESP8266-based development board (Adafruit Feather Huzzah) for less than US$20. We connected the CC112x as a slave SPI device for the ESP8266 and wrote firmware to intercept, modify, and send packets — all of this in an unattended way or in a remotely controllable manner. The remote interface was, in our case, a Wi-Fi transceiver that can be connected to a 4G hotspot. An alternative would be the Fona version of Adafruit's Feather series, which has an embedded cellular modem and would remove the need for a Wi-Fi hotspot. The total price was under US$40 (about US$60 for the cellular version).

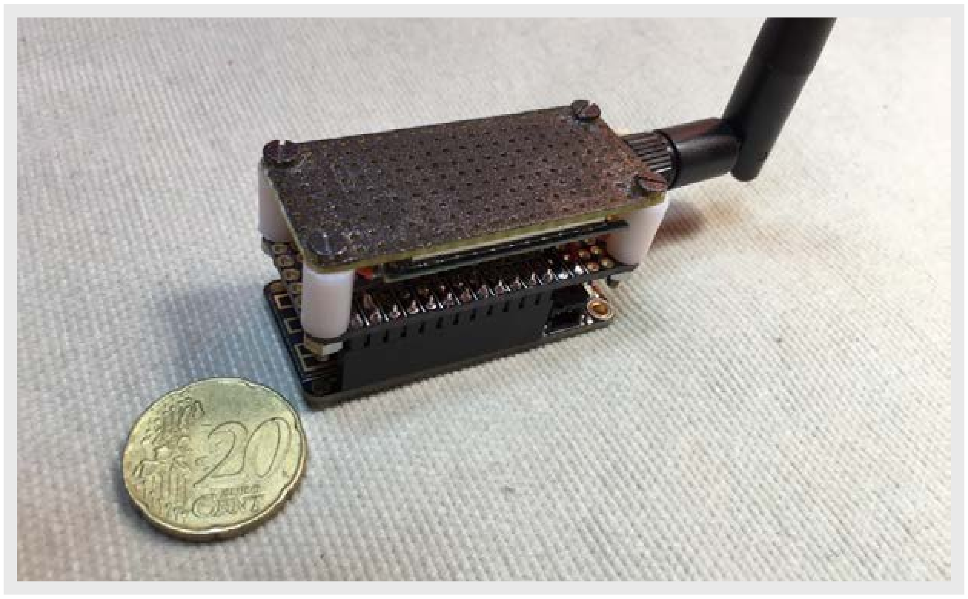
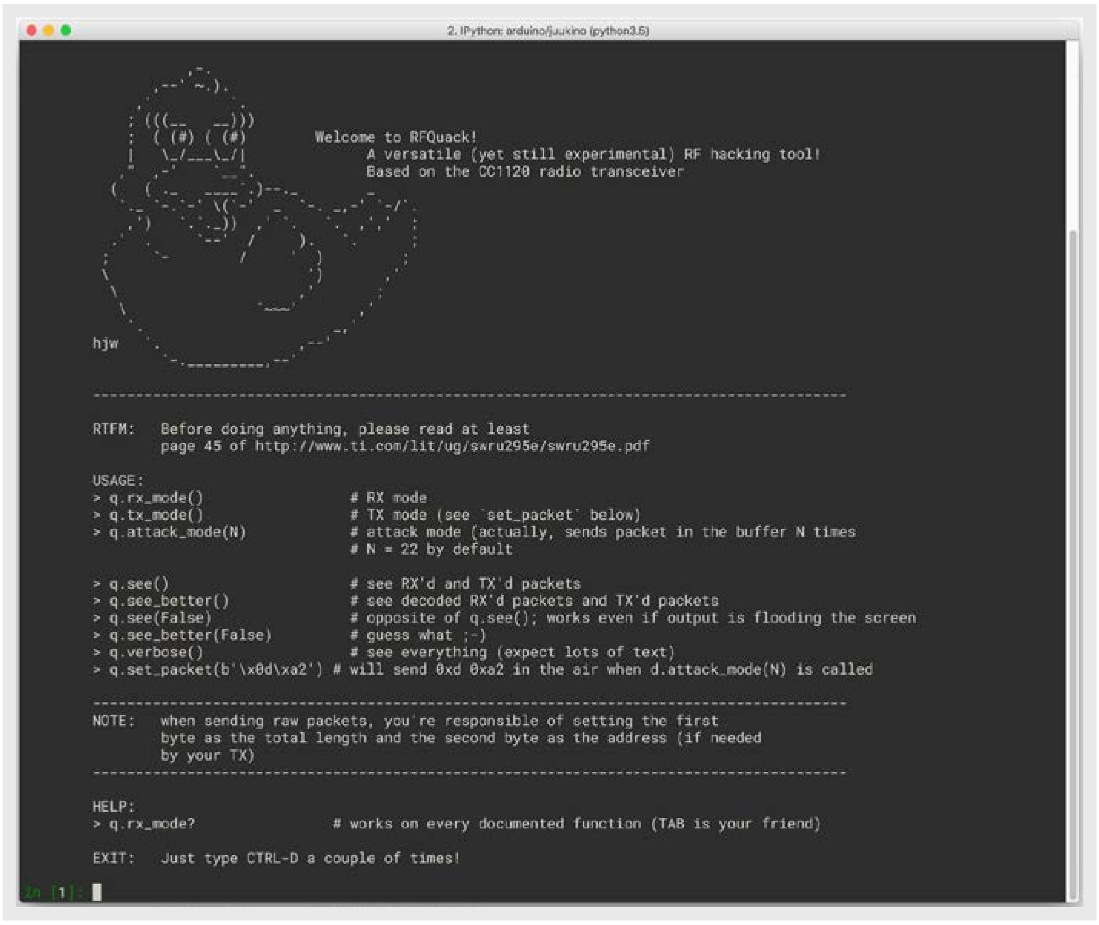


Figure 49. The first prototype of the RFQuack RF-hacking hardware device

RFQuack can be controlled remotely via Message Queuing Telemetry Transport (MQTT) messages, which can be sent from a client-side interactive console that we built around it. When powered up, the device does not do anything in order to save power. When put in receiving mode, it goes into deep-sleep mode and wakes up only when a valid radio packet is received. This enhanced wake-on-radio (WOR) feature is a characteristic of the CC1120 transceiver, which makes it a good candidate for these battery-powered applications. A valid radio packet can be configured, for example, to match a certain preamble and a given sync word, or to carry at least an amount of data (e.g., 14 bytes, configurable). While no valid packets are received, the radio will just stay in a deep-sleep state. When a valid packet is received, the default behavior is to resend it immediately N times (enough to make the target receiver “obey” the command). Optionally — and this is the most interesting feature — the user can set RFQuack to modify the packet on the fly right before retransmitting it, by specifying a set of byte-level modifications. For example, to implement Attacks 2 and 3 on the Juuko receiver, we knew that we had to modify only certain bytes of the packet, by XORing specific values at predefined positions to inject custom commands.



2. IPython: arduino/juukino (python3.5)

```
                                        Welcome to RFQuack!
                                              A versatile (yet still experimental) RF hacking tool!
                                              Based on the CC1120 radio transceiver

hjw

------------------------------------------------------------------------------------------

RTFM:   Before doing anything, please read at least
        page 45 of http://www.ti.com/lit/ug/swru295e/swru295e.pdf

USAGE:
> q.rx_mode()                  # RX mode
> q.tx_mode()                  # TX mode (see `set_packet` below)
> q.attack_mode(N)             # attack mode (actually, sends packet in the buffer N times
                               # N = 22 by default

> q.see()                      # see RX'd and TX'd packets
> q.see_better()               # see decoded RX'd packets and TX'd packets
> q.see(False)                 # opposite of q.see(); works even if output is flooding the screen
> q.see_better(False)          # guess what ;-)
> q.verbose()                  # see everything (expect lots of text)
> q.set_packet(b'\x0d\xa2') # will send 0xd 0xa2 in the air when d.attack_mode(N) is called

------------------------------------------------------------------------------------------
NOTE:   when sending raw packets, you're responsible of setting the first
        byte as the total length and the second byte as the address (if needed
        by your TX)
------------------------------------------------------------------------------------------

HELP:
> q.rx_mode?                 # works on every documented function (TAB is your friend)

EXIT:   Just type CTRL-D a couple of times!

In [1]: 
```

Figure 50. To ease the development of custom exploitation sequences,
we created a basic interactive terminal-based console to abstract the underlying
communication between the computer and the RF device.

6. Conclusion

The evolution of RF protocols, in most cases, started as a basic protocol supporting simple operations, where each command was sent over the air in clear text. One early problem with this was that if there were multiple systems in the area, one could control multiple pieces of equipment with one controller. To prevent this, DIP switches were put into different locations on the transmitter and receiver of the specific devices that had to be controlled. (We even evaluated a current industrial solution having this type of technology.) From there, manufacturers started implementing pairing solutions where the transmitter and receiver exchanged a code acting as a form of association: The receiver would only perform actions based on knowing that a command was generated by its "associated" transmitter.

Our research shows that there is a discrepancy between the consumer and industrial worlds. In the consumer world, the perceived risks have pushed the vendors to find reasonably secure, albeit imperfect, solutions such as rolling codes. In the industrial world, where the assets at risk are much more valuable than a fancy house or car, there seems to be less awareness.

Apart from leaked schematics, the only available "technical" documentation is limited to user manuals, and we are unaware of any public research about the digital security risks in this space. We hope that our findings will inspire the RF- and hardware-hacking communities to continue looking at these protocols, and to encourage vendors to focus on open, standard RF protocols.

It's also worth mentioning that a number of the vendors we considered have made significant strides toward taking accountability and acknowledging their shared responsibility in ensuring security. As a result, many of the vendors have been able to address the issues concerning them. In fact, in the case of at least one vendor, this is the first time that the vendor has ever released a patch for its products. This serves to highlight the importance of in-depth research and responsible disclosure, particularly where security is concerned.

6.1 Security Recommendations

Industrial radio devices have higher replacement costs and longer life spans than consumer ones, which means that vulnerabilities will persist for years, if not decades. However, patching the devices is possible. We have reached out and collaborated with the vendors in improving the state of security of their devices.

Some of them have already implemented the necessary countermeasures and even made software updates available. Some have also informed us that the vulnerabilities that we found will not affect next-generation devices, which will implement the necessary precautions.

Generally, there is a friction in patching because of the high downtime costs and business continuity constraints. Also, there's no such thing as "forensics" in this field. Incidents are scrutinized in the "physical world," and parts are just replaced to restore normal operations as quickly as possible. In other words, digital attacks are not considered a possibility in this field.

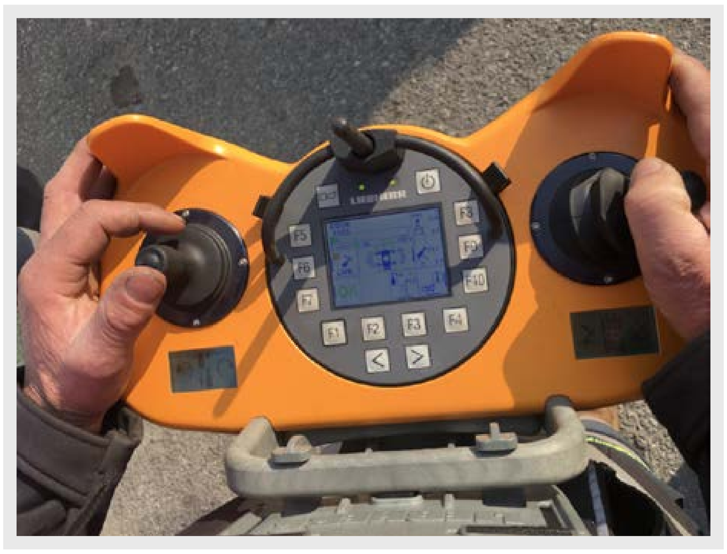


Figure 51. Liebherr industrial radio remote controller, which we found at a transportation business, using standard Bluetooth

The long-term solution to the problems that we have highlighted is that vendors should abandon proprietary RF protocols and focus on open, standard ones. Some vendors are already gradually switching to the 2.4 GHz bands — mainly to reduce interferences and limit the range — with Bluetooth Low Energy being one of the main, standard options. The benefit of the adoption of standard, open protocols such as Bluetooth Low Energy is clearly the increased security level and this would remove the burden on the vendors' part to design or integrate custom RF protocols.

The short-term recommendations go to the "users" — not the end users or the workers, but the system integrators. They should prefer devices that offer virtual fencing features, which disable the device when the remote is out of range. Despite not completely eliminating the vulnerabilities that we have highlighted, virtual fencing would make their exploitation window much narrower, because the attacker would need to either be on-site or know when the legitimate transmitter is enabled to perform the attacks. There is the possibility (to be verified) for an attacker to forge the out-of-band signaling that implements the virtual fencing.

6.2 Security Checklist

Without endorsing any vendors in particular, we hereby provide a practical checklist to the various subjects. Any actor throughout the supply chain should follow security best practices on their digital assets to reduce the exposure to cyberattacks that could compromise the integrity of the firmware images running on the radio transceiver and microcontrollers.

For users:

- Inspect the technical manuals before purchasing a device (most of the manuals are available online), and ensure that some form of configurable pairing is available.
- Periodically change the pairing (ID) code, if available.
- If the TX or RX units are programmable, keep the programming computer off the network, or harden its security as if it were a critical endpoint.
- Prefer remote control systems that offer dual-technology devices (e.g., with virtual fencing).
- Prefer devices that use open, well-known, standard protocols (e.g., Bluetooth Low Energy, 5G[69]).

For vendors:

- Implement a rolling-code mechanism and provide firmware upgrades to existing devices.
- Build on open, well-known, standard protocols (e.g., Bluetooth Low Energy).
- Consider radio transceivers that support encryption in hardware (e.g., CC1110Fx/CC1111Fx[70]).
- Consider future evolutions when designing next-generation systems. In particular, network-connected remote control systems, while in principle opening a wider attack surface, may offer an opportunity to implement over-the-air (OTA) firmware upgrade capabilities and distributed key exchange schemes.
- Use tamper-proof mechanisms to hinder reverse engineering. Although Saga and Juuko have tried to protect their firmware, we showed that Saga's firmware was still recoverable. Hardware mechanisms (e.g., mass-erase when chassis is opened, differentiate circuit to prevent programming cable from being sniffed, not to use RS-232) should be deployed to better protect the firmware.

In conclusion, given that the kind of machinery these remote controllers are managing can be dangerous if hijacked or disabled, manufacturers need to start thinking about moving to stronger open-source protocols rather than relying on security through obscurity. It could be challenging to balance the almost real-time requirements and secure RF transmission, but the hardware technology is there, ready to be used.

References

1. Eleif.net. *eleif.net*. "Online Manchester encoder/decoder." Last accessed on 16 August 2018 at http://eleif.net/manchester.html.

2. Tomihiko Uchikawa. (17 October 2002). *Circuit Design, Inc*. "Natural disaster monitoring and warning system using SRD." Last accessed on 9 October 2018 at http://www.cdt21.com/resources/pdf/TI_008.pdf.

3. FCCID.io. (2018). *FCCID.io*. "Searchable FCC ID Database." Last accessed on 16 August 2018 at https://fccid.io.

4. Autec s.r.l. (11 August 2004). *FCCID.io*. "Autec S.r.l. FCC Wireless Applications." Last accessed on 16 August 2018 at http://fccid.io/OQA.

5. Hetronic International Inc. (5 December 2014). *FCCID.io*. "Hetronic International Inc FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/LW9.

6. Gain Electronic Co., Ltd. (4 December 2012). *FCCID.io*. "Gain Electronic Co Ltd FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/NCT.

7. Circuit Design, Inc. (19 February 2009). *FCCID.io*. "Circuit Design, Inc. FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/V9X.

8. Elca S.r.l. (4 February 2014). *FCCID.io*. "Elca S.r.l. FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/2ABS7.

9. Lee's Hi-Tech Enterprise Co., Ltd. (12 March 2012). *FCCID.io*. "Lee's High-Tech Enterprise Co Ltd FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/LWN.

10. Elgato Systems LLC. (26 March 2013). *FCCID.io*. "SHUN HU TECHNOLOGY CO., LTD FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/RN4.

11. HBC-radiomatic GmbH. (17 December 2001). *FCCID.io*. "HBC-radiomatic GmbH FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/NO9.

12. Cattron-Theimeg Inc. (24 November 2010). *FCCID.io*. "Laird Controls North America Inc. FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/CN2.

13. Tele Radio. (22 January 2003). *FCCID.io*. "Tele Radio AB FCC Wireless Applications." Last accessed on 16 August 2018 at http://fccid.io/ONF.

14. Scanreco Industrielektronik AB. (24 April 2006). *FCCID.io*. "Scanreco AB FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/N5O.

15. Akerstroms Bjorbo AB. (22 March 2012). *FCCID.io*. "Akerstroms Bjorbo AB FCC Wireless Applications." Last accessed on 16 August 2018 at http://fccid.io/OG4.

16. Jay Electronique. (7 July 2009). *FCCID.io*. "Jay Electronique FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/OQM.

17. Investigation Total-Ware, SA. (16 February 2017). *FCCID.io*. "ANATEL 02267-09-05389." Last accessed on 16 August 2018 at https://fccid.io/ANATEL/02267-09-05389.

18. Investigation Total-Ware, SA. (16 February 2017). *FCCID.io*. "ANATEL 01199-09-05389." Last accessed on 16 August 2018 at https://fccid.io/ANATEL/01199-09-05389.

19. 3-Elite Join Industrial PTE Ltd. (31 December 2004). *FCCID.io*. "3-Elite Join Industrial Pte Ltd. FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/PCS.

20. World Economic Forum and The Boston Consulting Group. (2016). *World Economic Forum*. "Shaping the Future of Construction: A Breakthrough in Mindset and Technology." Last accessed on 16 August 2018 at http://www3.weforum.org/docs/WEF_Shaping_the_Future_of_Construction_full_report__.pdf.

21. Akerstroms Bjorbo AB. *FCCID.io*. "FCC ID OG4BC8518." Last accessed on 16 August 2018 at http://fccid.io/OG4BC8518.

22. Germanischer Lloyd. (7 November 2013). *Tele Radio*. "Type Approval Certificate." Last accessed on 16 August 2018 at https://www.tele-radio.com/downloads/documents/docdir/CER-TG2-AP055-A01-ALL.pdf.

23. Telecrane. (10 April 2018). *Telecrane*. "Telecrane ASUS tablet with original software." Last accessed on 16 August 2018 at https://www.telecrane.it/en/tablet-with-original-software/.

24. Elgato Systems LLC. (26 March 2013). *FCCID.io*. "SHUN HU TECHNOLOGY CO., LTD FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/RN4.

25. HBC-radiomatic GmbH. (17 December 2001). *FCCID.io*. "HBC-radiomatic GmbH FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/NO9.

26. Ibid.

27. International Organization for Standardization. (November 2015). *ISO.org*. "ISO 13850:2015." Last accessed on 16 August 2018 at https://www.iso.org/standard/59970.html.

28. Xavier Martens. (25 August 2017). *SANS-ISC*. "Malicious AutoIT script delivered in a self-extracting RAR file." Last accessed on 16 August 2018 at https://isc.sans.edu/forums/diary/Malicious+AutoIT+script+delivered+in+a+selfextracting+RAR+file/22756/.

29. Dave Lochbaum. (2015). *Union of Concerned Scientists, Inc*. "Arkansas Nuclear One: Pictures of an Accident." Last accessed on 16 August 2018 at https://cdn.allthingsnuclear.org/wp-content/uploads/2015/02/FS-181-PDF-File-with-links.pdf.

30. Port of Rotterdam. *Port of Rotterdam*. "Port of Rotterdam." Last accessed on 16 August 2018 at https://www.portofrotterdam.com/en.

31. David Sancho. (30 January 2018). *Trend Micro*. "Digital Extortion: A Forward-looking View." Last accessed on 16 August 2018 at https://documents.trendmicro.com/assets/wp-digital-extortion-a-forward-looking-view.pdf.

32. Terje Lassen. (17 March 2015). *YouTube*. "More than 100km range with CC1120." Last accessed on 16 August 2018 at https://www.youtube.com/watch?v=wgqtEu5PfAw.

33. Ettus Research. *Ettus Research*. "LP0410 Antenna." Last accessed on 16 August 2018 at https://www.ettus.com/product/details/LP0410.

34. Trend Micro's Zero Day Initiative. *ZDI*. "(0Day) Juuko JK-800 Replay Attack Vulnerability." Last accessed on 9 January 2019 at https://www.zerodayinitiative.com/advisories/ZDI-18-1336/.

35. The MITRE Corporation. *CVE*. "CVE-2018-17903." Last accessed on 29 October 2018 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17903.

36. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. "Advisory (ICSA-18-296-02)." Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

37. The MITRE Corporation. *CVE*. "CVE-2018-17935." Last accessed on 29 October 2018 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17935.

38. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. "Advisory (ICSA-18-296-03)." Last accessed on 26 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-03.

39. The MITRE Corporation. *CVE*. "CVE-2018-19023." Last accessed on 8 January 2019 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-19023.

40. Industrial Control Systems Cyber Emergency Response Team. (3 January 2019). *ICS-CERT*. "Advisory (ICSA-19-003-03)." Last accessed on 8 January 2019 at https://ics-cert.us-cert.gov/advisories/ICSA-19-003-03.

41. The MITRE Corporation. *CVE*. "CVE-2018-17923." Last accessed on 29 October 2018 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17923.

42. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. "Advisory (ICSA-18-296-02)." Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

43. The MITRE Corporation. *CVE*. "CVE-2018-17921." Last accessed on 29 October 2018 at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17921.

44. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. "Advisory (ICSA-18-296-02)." Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

45. The MITRE Corporation. *CVE*. "CVE-2018-17903." Last accessed on 29 October 2018 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17903.

46. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. "Advisory (ICSA-18-296-02)." Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

47. Trend Micro's Zero Day Initiative. *ZDI*. "(0Day) Juuko DATA Packet Command Injection Remote Code Execution Vulnerability." Last accessed on 6 December 2018 at https://www.zerodayinitiative.com/advisories/ZDI-18-1362/.

48. Texas Instruments. (July 2015). *Texas Instruments*. "CC1120 High-Performance RF Transceiver for Narrowband Systems." Last accessed on 16 August 2018 at http://www.ti.com/lit/ds/symlink/cc1120.pdf.

49. Texas Instruments. (2013). *Texas Instruments*. "CC112X/CC1175 Low-Power High Performance Sub-1 GHz RF Transceivers/Transmitter." Last accessed on 16 August 2018 at http://www.ti.com/lit/ug/swru295e/swru295e.pdf.

50. Ibid.

51. Ibid.

52. Ibid.

53. Trend Micro's Zero Day Initiative. *ZDI*. "(0Day) Juuko JK-800 Replay Attack Vulnerability." Last accessed on 9 January 2019 at https://www.zerodayinitiative.com/advisories/ZDI-18-1336/.

54. The MITRE Corporation. *CVE*. "CVE-2018-17903." Last accessed on 29 October 2018 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17903.

55. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. “Advisory (ICSA-18-296-02).” Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

56. The MITRE Corporation. *CVE*. “CVE-2018-17935.” Last accessed on 29 October 2018 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17935.

57. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. “Advisory (ICSA-18-296-03).” Last accessed on 26 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-03.

58. The MITRE Corporation. *CVE*. “CVE-2018-19023.” Last accessed on 8 January 2019 at https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-19023.

59. Industrial Control Systems Cyber Emergency Response Team. (3 January 2019). *ICS-CERT*. “Advisory (ICSA-19-003-03).” Last accessed on 8 January 2019 at https://ics-cert.us-cert.gov/advisories/ICSA-19-003-03.

60. The MITRE Corporation. *CVE*. “CVE-2018-17921.” Last accessed on 29 October 2018 at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17921.

61. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. “Advisory (ICSA-18-296-02).” Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

62. The MITRE Corporation. *CVE*. “CVE-2018-17921.” Last accessed on 29 October 2018 at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-17921.

63. Industrial Control Systems Cyber Emergency Response Team. (23 October 2018). *ICS-CERT*. “Advisory (ICSA-18-296-02).” Last accessed on 29 October 2018 at https://ics-cert.us-cert.gov/advisories/ICSA-18-296-02.

64. Telecrane. *Meerholz Canada LTD*. “F21 Series Radio Control.” Last accessed on 8 October 2018 at https://www.meerholz.ca/uploads/File/F21-manual.pdf.

65. Great Scott Gadgets. *Great Scott Gadgets*. “YARD Stick One.” Last accessed on 2 October 2018 at https://greatscottgadgets.com/yardstickone/.

66. ComThings. *ComThings*. “Home - ComThings.” Last accessed on 2 October 2018 at https://www.comthings.com.

67. Texas Instruments. (2017). *Texas Instruments*. “Low-Power SoC (System-on-Chip) with MCU, Memory, Sub-1 GHz RF Transceiver, and USB Controller.” Last accessed on 16 August 2018 at http://www.ti.com/lit/ds/symlink/cc1110-cc1111.pdf.

68. Vchip. (12 April 2018). *DigiRF*. “VT-CC1120PL-433M Wireless Module User Guide VT-CC1120PL-433M.” Last accessed on 16 August 2018 at http://www.digirf.com/EN/ProView/86.html.

69. Koji Fujii. (16 February 2018). *BusinessNetwork.jp*. “日本中の建機の集中操作も夢ではない––KDDI、大林組、NECが5G活用した遠隔操縦の公開実験.” Last accessed on 16 August 2018 at https://businessnetwork.jp/Detail/tabid/65/artid/5943/Default.aspx.

70. Texas Instruments. (2017). *Texas Instruments*. “Low-Power SoC (System-on-Chip) with MCU, Memory, Sub-1 GHz RF Transceiver, and USB Controller.” Last accessed on 16 August 2018 at http://www.ti.com/lit/ds/symlink/cc1110-cc1111.pdf.