WHITE PAPER

ANALYSIS OF CELLULAR BASED INTERNET OF THINGS (IOT) TECHNOLOGY

Deral Heiland - Principal Security Researcher (IoT), Rapid7

Carlota Bindner - Lead Product Security Researcher, Thermo Fisher Scientific

CONTENTS

INTRODUCTION

In recent years, the increased use of cellular communications in Internet of Things (IoT) technology has played a crucial role in enabling communication with IoT devices that lack wired internet access. In addition, cellular communication is now commonly relied upon as a backup solution for wired connections like ethernet or Wi-Fi should they fail. Common devices that have adopted cellular communication include cameras, GPS trackers, home and building security systems, and medical devices.

Given the significant potential for this trend to continue, security researchers must focus on comprehending this technology, its applications, and developing hardware testing methods to evaluate the end-to-end security of cellular-based devices. Consequently, this paper aims to address this topic and explore various testing methodologies.

CAT-M AND NB-IOT CELLULAR TECHNOLOGY OVERVIEW

Since the first cellular phone service appeared in 1983, cellular technologies have been rapidly adopted for portable computing, including notebooks, laptops, and smartphones. Despite cellular technology becoming ubiquitous, it often comes at a premium price, with the cost being proportional to the bandwidth used. Most IoT devices consume a fraction of the data used for email, browsing, and video streaming, making traditional cellular communications cost-prohibitive.

Additionally, reliance on low voltage batteries with limited power output and small form factors complicated cellular technology use by IoT devices. Yet, a growing number of IoT devices are leveraging cellular communications as either a primary or secondary communication platform. The introduction of CAT-M and NB-IoT has facilitated cellular communications for IoT devices.

CAT-M and NB-IoT are communication standards developed by the Third Generation Partnership Project (3GPP). These standards focus on providing low-power wide area network (LPWAN) radio communications for supporting IoT technology, which is focused on machine-to-machine (M2M) communication. 3GPP initially released these standards with Release 13 in 2015, followed by further improvements and updates in Release 14 in 2017.

CAT-M and NB-IoT are complementary standards that excel in different use cases. These standards now allow IoT devices to communicate directly to the cloud via cellular services and no longer require local aggregation via bridging technology.

CAT-M OR LTE CAT-M1/CAT-M2

Although the 3GPP standards include several variations of CAT-M, for consistency this paper will focus on the generally referenced standard LTE CAT-M1, which supports a bandwidth of 1.08 Mhz. The mode of operation supported by CAT-M1 is in-band LTE. In-band signaling is when the data is transferred within the standard LTE band or channel used for general communication.

CAT-M1 features include upload and download speeds of up to 1 mbps, which is higher than NB-IoT, and lower latency of 10 to 15 ms. In addition, the CAT-M1 standard supports half or full-duplex, and has enough bandwidth to support voice communication if needed. As a result, CAT-M1 is commonly used for IoT machine communication where more bandwidth is required. Examples of devices that use CAT-M1 include camera systems, alarm systems, and vehicle data collection devices.

NARROW BAND IOT (NB-IOT) OR CAT-NB1/CAT-NB2

Like CAT-M, the 3GPP standard for NB-IoT has variations. In this case, the overview will cover the overarching features of NB-IoT. NB-IoT has a maximum bandwidth of 180 kHz and only supports half-duplex communication. Similar to CAT-M1, NB-IoT uses in-band LTE, although it has the added feature of using either LTE guard bands or standard mode. Guard bands are narrow frequency ranges used for separating wider frequency ranges not typically used for communication, while standard mode leverages unused GSM bands.

Due to limited bandwidth, NB-IoT is best suited for IoT with limited complexity and low bandwidth requirements. Although it has lower bandwidth capabilities, NB-IoT excels in reduced power consumption compared to CAT-M1, making it well-suited for use in IoT technology dependent on battery power. This standard is often used by gas and water meters and low-power consumption sensors.

Both CAT-M1 and NB-IoT standards support Extended Discontinuous Reception (eDRX) and Power Saving Mode (PSM) functionalities. These features allow the device's cellular communication to go into a powered-down sleep mode when data transfers are not required, further reducing energy consumption. In the case of NB-IoT, this can extend battery life for years, potentially up to a decade. Furthermore, since the long-term cost associated with cellular is often based on bandwidth usage, these standards have significantly reduced operation costs, making it possible to deploy cellular technologies into increasingly more IoT products.

Since the initial release, 3GPP has published followup standards, including releases 15, 16, and 17. These three releases mainly focused on various improvements to current features, which involved:

Network operation and efficiency

- Access control
- Small-cell support
- Power management
- Improved support of 5G services and features

CELLULAR MODULE REVIEW

In 2022, global shipments of IoT cellular modules grew 14% year-over-year, according to Counterpoint's Global Cellular IoT Module and Chipset Tracker by Application report. In the first quarter of 2023, Quectel was the leading manufacturer in the IoT cellular market, accounting for 37.2% of global cellular IoT module shipments. Other manufacturers that accounted for just over half of IoT cellular module shipments included Telit Cinterion, Fibocom, China Mobile, and Sunsea.

During the initial selection of devices, we focused primarily on GPS trackers that used either NB-IoT or CAT-M for cellular communications and could be readily purchased on the open market. We later widened our device type search to other areas, including cellular modems and field cameras. While we performed reconnaissance on devices using FCC IDs when available, we could not identify the cellular modules used by all devices before purchase. We found 10 of the 14 devices we initially reviewed used Quectel or Telit Cinterion modules based on FCC documentation.

One of the Quectel modules encountered was the Quectel EG91-NAXD, part of the Quectel LTE EG91 series that supports LTE CAT-M1, found in two different field camera models from the same manufacturer. The Quectel LTE EG91 series is pin-to-pin compatible with Quectel's UG95, UG96, BG95, and BG96 modules. The pin-to-pin compatibility allows device manufacturers to swap modules, which allows changing modules for cost savings or if supply chain availability affects a particular cellular module version. As a result, two devices with the same make and model may have a different cellular module in use if they were manufactured at different times. Figure 1 shows the Quectel EG91-NAXD from one of the field cameras.

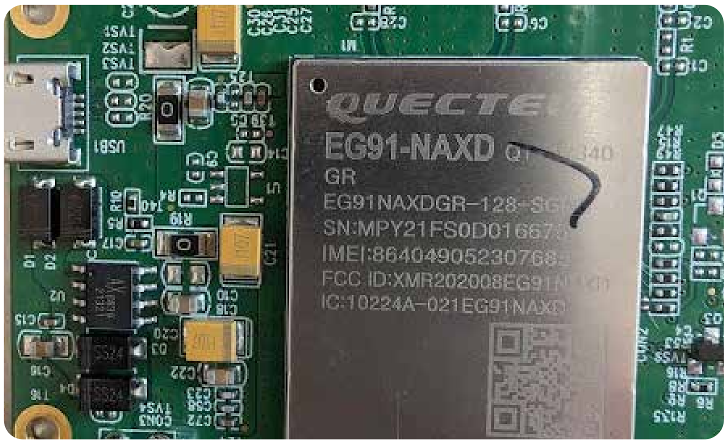


Figure 1. Quectel EG91-NAXD on Field Camera PCB

In addition to the field camera, a Quectel BC26 cellular module was found in a USB cellular modem, as shown in Figure 2. At the time of writing, a search on the Quectel website returned no information on the Quectel BC26 or product end-of-life notification that would indicate it had been discontinued. We found information for the Quectel BC26 from third-party seller listings and developer forum posts, which indicated it was still for sale and actively used for development. The Quectel BC26 provides CAT NB1 connectivity with support for various network protocols and multiple serial ports, including one for AT command and data transmission and two more for debug and serial connection.

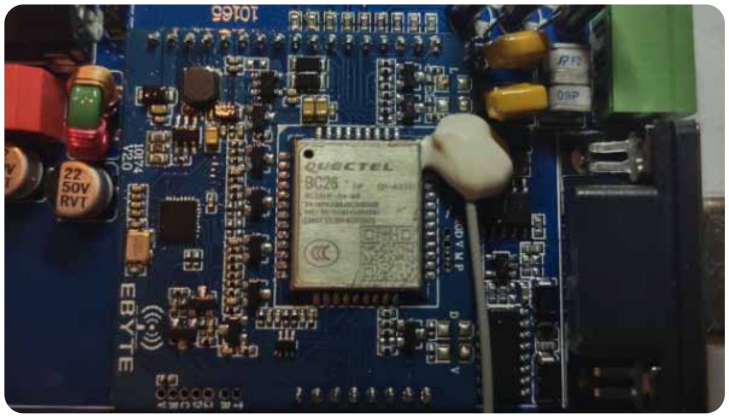


Figure 2. Quectel BC26 in Cellular Modem

In other devices, we found an array of Telit Cinterion modules, including some labeled as Telit and others as Cinterion. The label difference is due to Cinterion's incorporation of Thales's cellular IoT products in 2022. Figure 3 shows the Telit ME910CE-NA used in a compact cellular modem purchased for research. Like the Quectel BG91, the Telit ME910CE-NA supports CAT-M1 and NB1 communications and industry-standard interfaces, including USB, UART, SPI, and I2C.

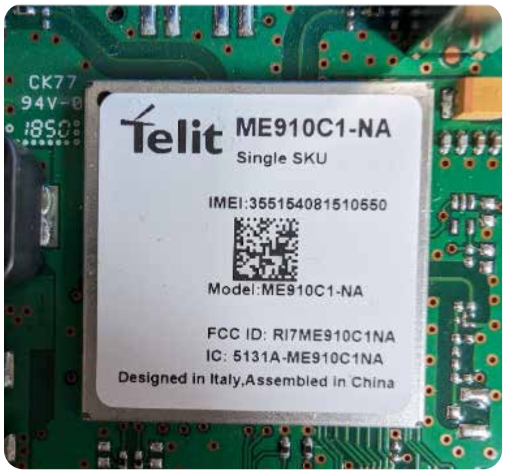


Figure 3. Telit ME910C1-NA in Cellular Modem

Another Telit Cinterion cellular module, the Cinterion S30960-S6200-A100-1, was found in a GPS tracker with a U-Blox SAM-M8Q smart antenna module, as shown in Figure 4.

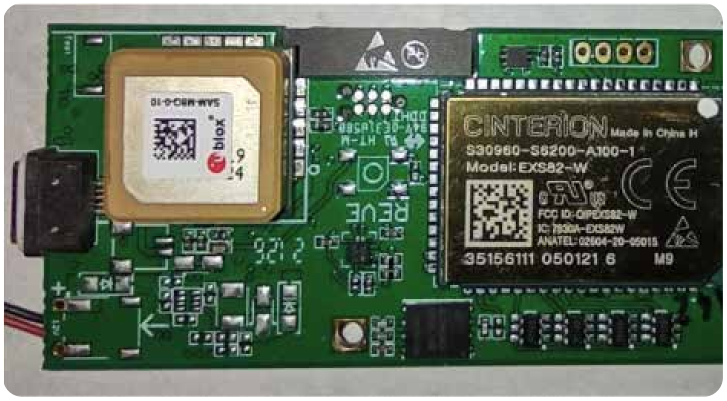


Figure 4. Cinterion Cellular Module in GPS Tracker

In addition to Quectel and Telit Cinterion modules, other cellular manufacturers were represented in GPS trackers purchased for research. After Quectel and Telit Cinterion, U-Blox cellular modules were third most common in the GPS tracking devices, including the SARA-R412M, SARA-U201, LISA-C200, and LEON-G100. Figure 5 shows the U-Blox SARA-R412M found in a pet tracker. The SARA-R412M provides multi-region support using both CAT M1 and NB1, features an integrated GNSS receiver for positioning, and supports multiple networking protocols focusing on MQTT for low-bandwidth communications.

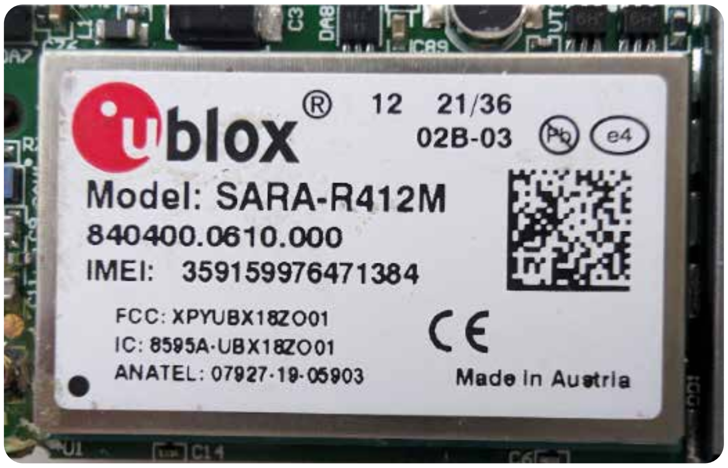


Figure 5. U-Blox SARA-R412M in GPS Tracker

While we intended to select devices using cellular modules currently on the market, two of the U-Blox cellular modules in vehicle trackers purchased for our research had been discontinued, the LEON-G100 and the LISA-C200. Figure 6 shows the LEON-G100 cellular module found in a vehicle tracker purchased in 2021, while U-Blox's End-of-Life announcement for the module was released on May 19th, 2017. Finding cellular modules no longer supported by the manufacturer in new devices demonstrates that End-of-Life (EoL) components continue to persist not only in currently deployed products but also in current inventory awaiting sale.

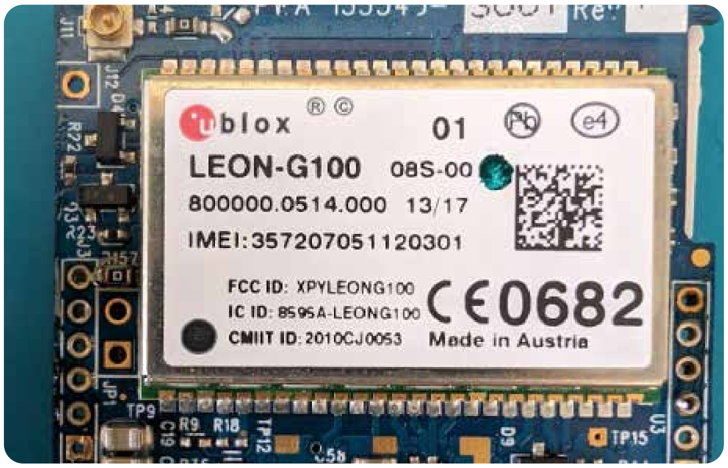


Figure 6. U-Blox LEON-G100 Cellular Module

In addition to U-Blox, we found one Nordic cellular module in a pet tracker. Figure 7 shows the Nordic nRF9160-SICA-R module, which had an ARM Cortex-M33 that supported LTE CAT M1 and NB1 communications and included SPI, I2C, and UART interfaces. The Nordic nRF9160-SICA-R was one of the smallest cellular modules with a package size of 10x16x1.04 mm LGA.

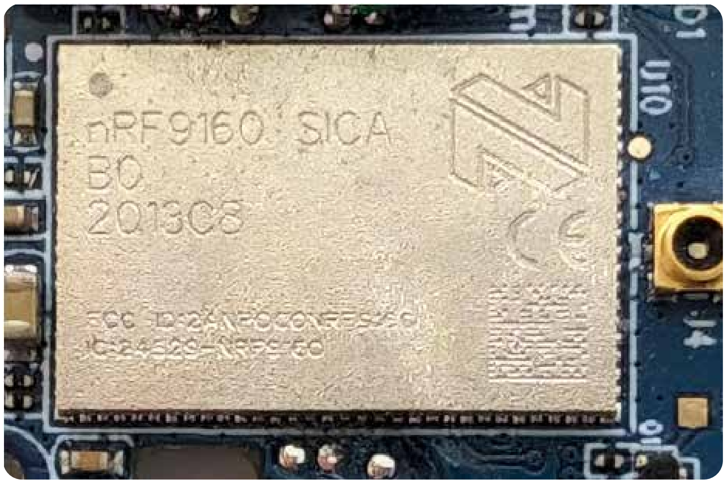


Figure 7. Nordic nRF9160-SICA-R Module

In addition to cellular-enabled devices, we also purchased development kits for cellular modules found in the consumer products purchased or in the same product line. This included a Telit Bravo Rev D board with a ME910C1-WW module and a U-Blox EVK-R4 evaluation kit with a SARA-R410Mc cellular module, as shown in Figure 8 and Figure 9, respectively.

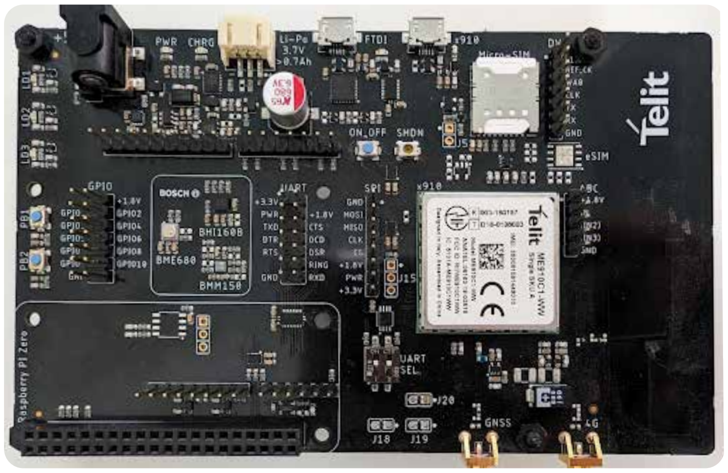


Figure 8. Telit Bravo Rev D Board

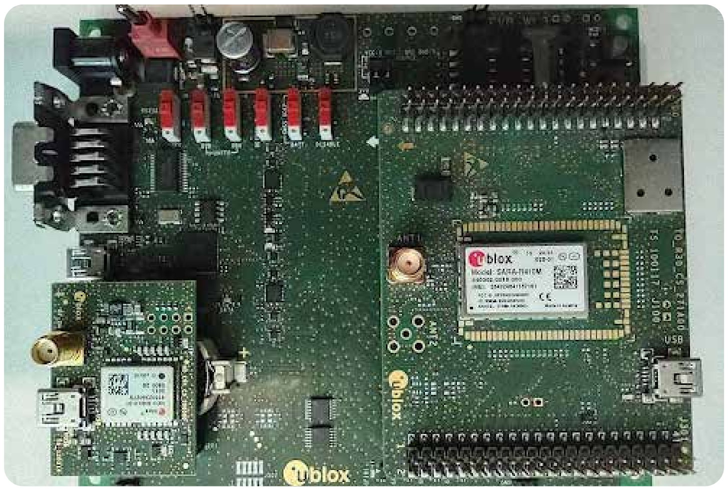


Figure 9. U-Blox EVK-R4 Evaluation Kit

Additionally, we found some development boards which supported different cellular modules using various connectors. Figure 10 shows the Cinterion LGA DevKit development board, which included a socket that could accommodate different Cinterion cellular modules.

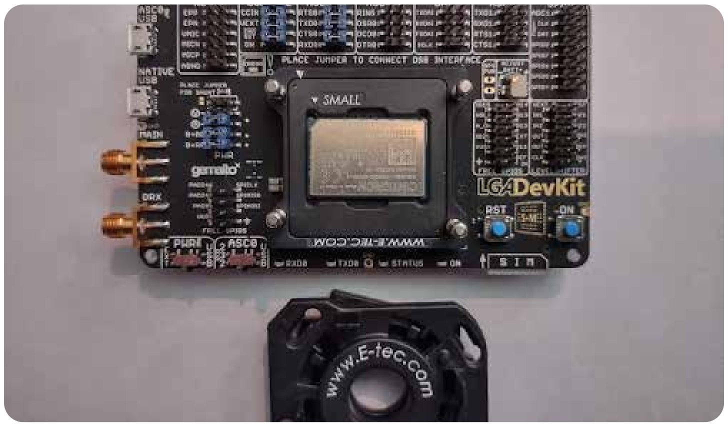


Figure 10. Cinterion LGA DevKit with Small Socket

The Quectel UMTS & LTE EVB kit supports four module types, including automotive, LPWA, LTE Standard, Wi-Fi/Bluetooth, and multiple cellular mobiles for both LPWA and LTE Standard.

Figure 11 shows the Quectel UMTS&LTE-EVB-KIT with an EG91-NA module.

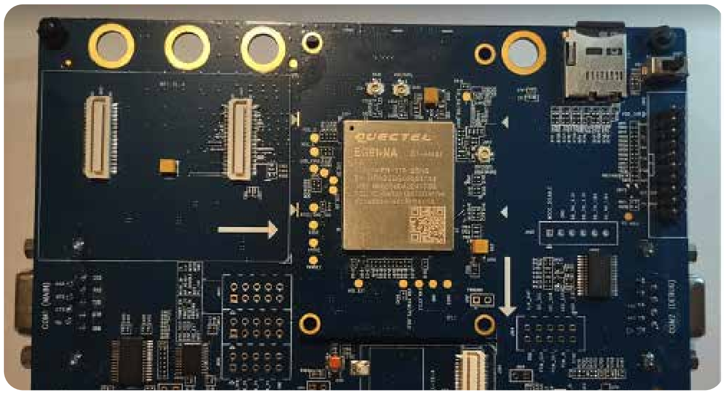


Figure 11. Quectel UMTS&LTE-EVB-KIT with an EG91-NA module

CIRCUIT BOARD LAYOUT ANALYSIS

When examining cellular modem modules on IoT devices, it is crucial to comprehend the circuit layout and orientations on the circuit board in relation to other components, such as the device's primary CPU. Additionally, you should consider the following factors:

- How are cellular modules oriented on the circuit board?
- How can this orientation information be used during testing to find potential communication test points?
- How do we access these communication test points even if they are not readily available on the circuit board's surface?

In this section, we will disassemble a couple of cellular-enabled devices to address the questions mentioned above and explore various examination and hardware interaction methodologies.

It is important to note the methodologies discussed here can be used on other circuit technologies besides cellular.

The first example we looked at was a Telit GE910 cellular module used in a home security system. Initially, we tried to understand the orientation of the cellular modem and its potential communication paths on the circuit board to identify connection points for testing, such as:

- Main UART
- Debug UART
- USB

Like most IoT devices, we examine the circuit board to identify all components, gather needed manufacturer's datasheets, and check to see if there are any good markings on the circuit board to identify points of interest. Once that analysis is complete, if you still need to better identify points of interest, the next method we use is a transparency overlay of the circuit board images.

By using the obverse (frontside) and reverse (backside) of the circuit shown below in Figure 12 and Figure 13, we produce an image as shown in Figure 14, which shows components on both sides of the board in reference to each other. This helps visualize key components on each side of the board in reference to each other and potential circuit paths.

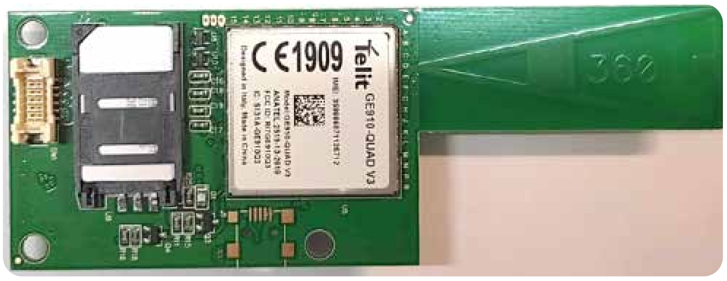


Figure 12: GSM Telit GE910 Cellular Board Obverse Image

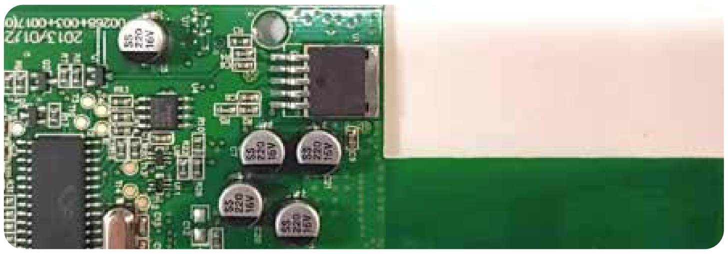


Figure 13: GSM Telit GE910 Cellular Board Reverse Image

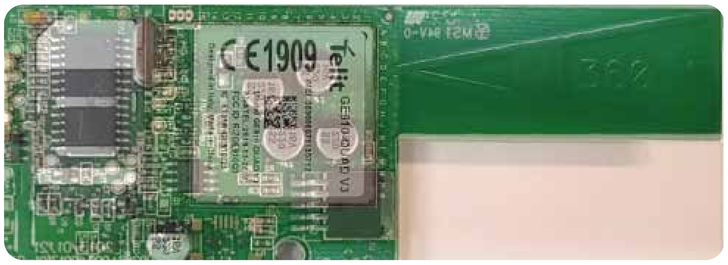


Figure 14: GSM Telit GE910 Cellular Board Transparency Overlay

In the final part, we will often overlay the cellular modem device's Land Grid Array (LGA) mapping from available datasheets. This allows us to quickly find where the cellular modem module's core communication connections are (UART, USB), and identify potential contact point locations on the circuit board so we can tap into them. An example of this concept is shown below in Figure 15:

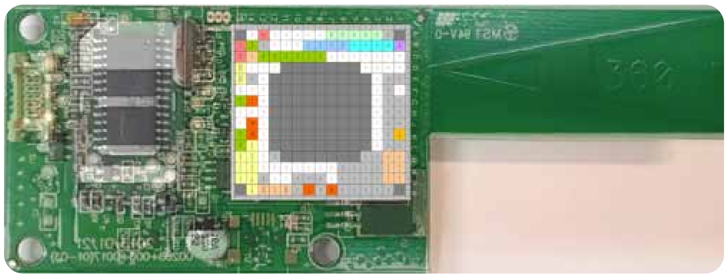


Figure 15: Telit GE910 LGA Pinout Layout Overlay on Transparency

The above method works well for establishing a general starting point for identifying the areas of the circuit board that are most likely to have a potential location for connecting to communication points on the circuit board for testing. Although, in some cases, we have found well-designed circuit boards that do not expose easily accessible communication points on the board's surface, but appear to be leveraging sublayers of the board. In this section, we also want to explore another method for connecting to communication points not directly exposed on the top layers of the circuit board.

We will refer to this method as submodule unit access (SUA). In this process, we take advantage of the fact that the most valid communication connection on cellular modules LGA pads typically runs around the outer perimeter of the cellular modules. For example, Figure 16 shows the measurement diagram from the BG95 hardware design manual and grid array of Quectel BG95 module, which shows that the points of interest are within .2 mm of the module's edge.

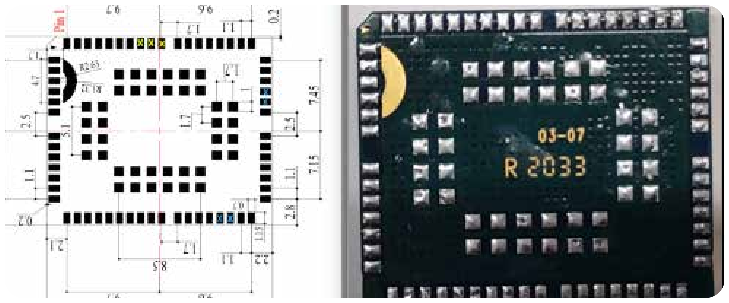


Figure 16: Quectel BG95 Pin-out Diagram and Module Side-by-Side Comparison

With that knowledge, it is possible to make entry from the side along the plane of the circuit board surface to establish a connection to needed communication points, such as USB, MAIN UART, and UART DBG. Figure 17 demonstrates a side view of the Quectel BG95 module attached to a circuit board showing a small gap and individual LGA pad connection points.

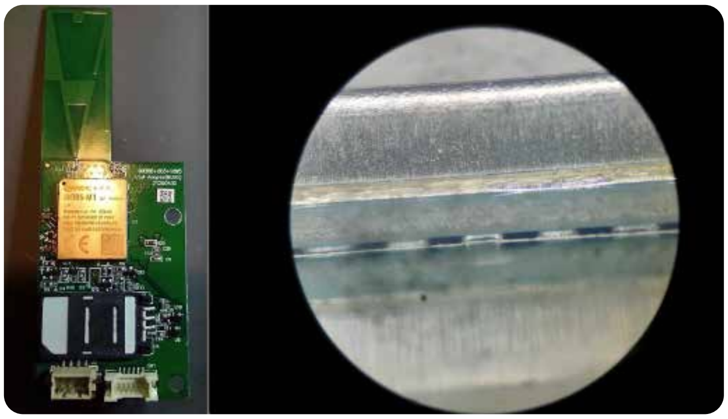


Figure 17: Wide-view and Side-view of Quectel BG95 Module

To attach to these contact points under the cellular modules, we used acupuncture needles (probes) to tap into the device's UART communication. Using different size acupuncture needles (.15 - .35 mm) we built several test probes. We purchased these probes — acupuncture-style needles — from online retailers selling them for cleaning out 3d printer nozzles.

Figure 18 shows an example of an assembled test probe consisting of an acupuncture needle and jumper wire connected using heat shrink butt connectors. During testing, we found that the larger needles (.35mm) were easier to work with when connecting to LGA modules where the pads were closer to the edge of the modules. The reason for this was the thicker probes were less flexible.

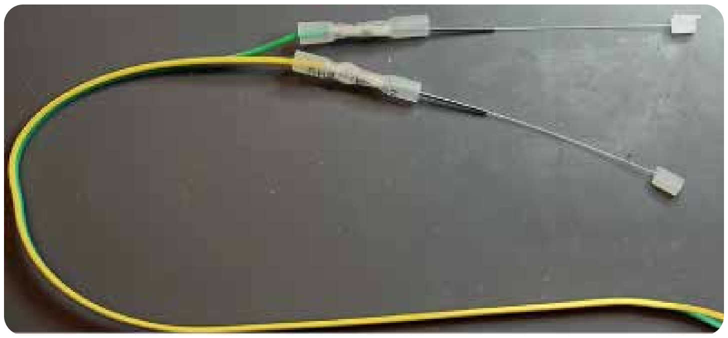


Figure 18: Test Probes Assembled from Acupuncture Needles

Before inserting probes under the cellular module edge, any exposed components on the circuit board should be covered with insulation tape to prevent shorting the probes. To do this, we used Kapton tape. It is also possible to wrap the probe needle, except the very end, in Kapton tape to insulate it from shorting. Also, during this process, we noted that entry under the modules must be made by remaining parallel with the PCB surface as much as possible to avoid damaging solder masking, which could lead to shorting out the probe. An example of this is shown in Figure 19.

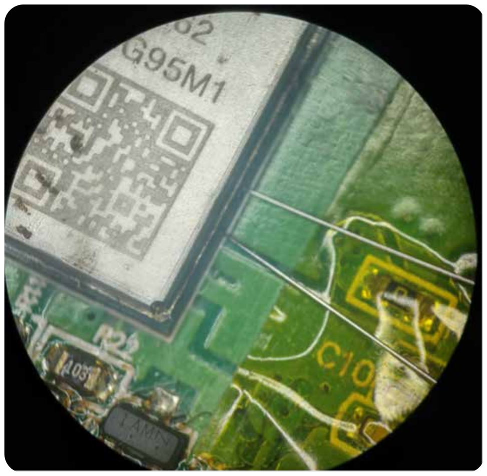


Figure 19: Probes Inserted Parallel to PCB with Kapton Tape

Once probes were inserted, getting them to remain in place was problematic. To do this, we used a flexible adhesive called E6000, which can be easily removed after work is completed. This adhesive had issues adhering to the stainless-steel probes and required us to add more E6000 covering a larger section of the probe body to prevent probes from being dislodged. Further adhesive testing was conducted using different adhesives, including a couple of different brand super glues and various forms of UV set solder masks and resins. Ultimately, we found the E6000 to be the most effective method for holding the acupuncture probes in place.

Once the probes were secured with adhesive, we tested the reliability of the overall process by reinstalling the cellular control board, attaching the probes to a Saleae logic analyzer, and powering up the devices, as shown in Figure 20.

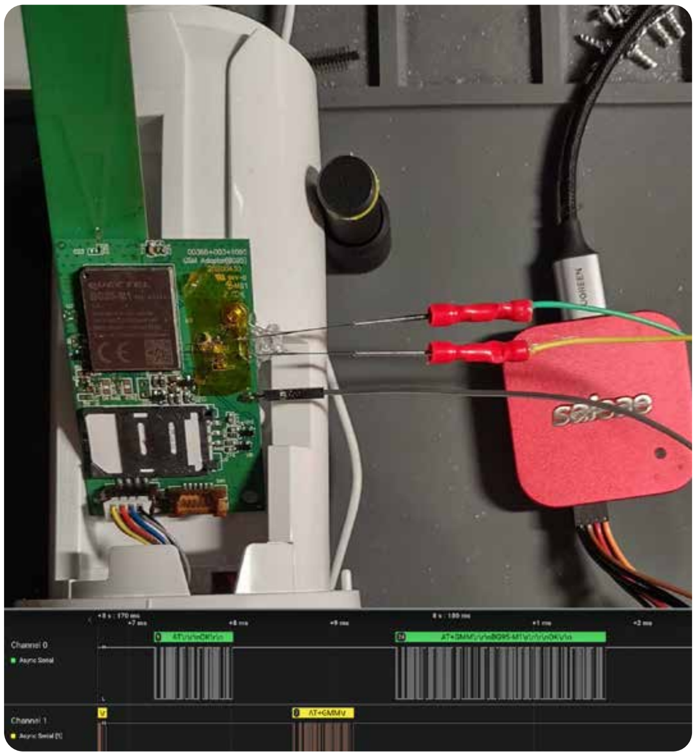


Figure 20: Saleae Connected to Probes Inserted Under the Cellular Module

We also examined the use of this method on larger cellular modules where the LGA contact points were further from the edge of the module. In the case of the Quectel EG91, the LGA connection points were 2.75mm from the edge of the module body with a gap along the edge that was not wide enough to properly insert any size acupuncture probes to make contact with LGA connection points. So this method was found to be most effective against smaller-sized cellular modules where the LGA lands are nearest to the module edges.

The final mechanical methods we want to examine in this paper are potentially more destructive and require actually removing and reinstalling the cellular module. If care and patience are taken, these methods can be very effective, allowing access to more easily map out critical circuit paths or modifying existing circuit board traces to allow for access to critical communication points. In this section, we will examine a Quectel EG91-NAXD module used in a trail camera, as shown in Figure 21.

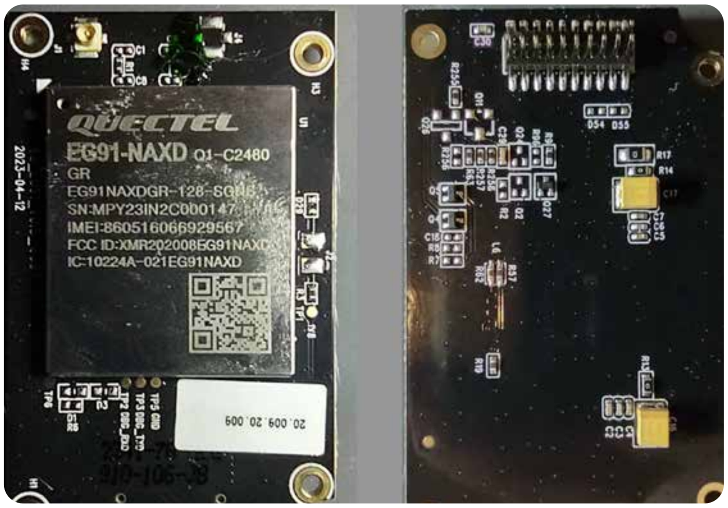


Figure 21: Quectel EG91-NAXD on Trail Camera's PCB

To remove a cellular module, the method we prefer involves using an Infra-Red (IR) reflow oven. However, this process can be accomplished with directed IR or hot air systems combined with a preheater hot plate. Setting up a temperature curve to heat the board and chip at a rate that does not shock or damage the chip is critical, and the hardest part of this process.

To accomplish this in an IR reflow oven, we must also consider the complete mass of the circuit board, which must be brought to the melting temperature of the solder (183 Celsius) before components, such as the cellular module, will be released. To do this, we set our IR reflow oven temperature curve to rise to match manufacturer recommendations, often defined within the datasheets, along with an extending time and higher peak temperature set to approximately 240-250 Celsius on average and vary the length of time the reflow oven stays at that temperature based on the size and thickness of the overall circuit board to which the cellular module is mounted.

With that said, it is important to remember that individual components on the circuit, such as memory and CPU chips, can easily be damaged if they reach a temperature of 215 Celsius or higher. Also, when the module is ready to be removed, caution should be taken, as all components on the circuit board are loose, and it is very easy to accidentally dislodge other components, especially when working with very small circuit boards.

Furthermore, when removing larger cellular modules with a suction tool, it is not uncommon to just pull off the shielding and not remove the module. In those cases, we have been able to either slide the module off the board without dislodging other components or use a cellular phone repair tool with a wide blade, as shown in Figure 22. The wider blade can be inserted under the module body to help lift the module from the surface.

Figure 22: Thin Blades for Removing Components

Once the module has been removed from the circuit board, we can examine the LGA layout, and use a multimeter set on continuity check to help trace out the communication lines we would be interested in accessing, as shown in Figure 23.

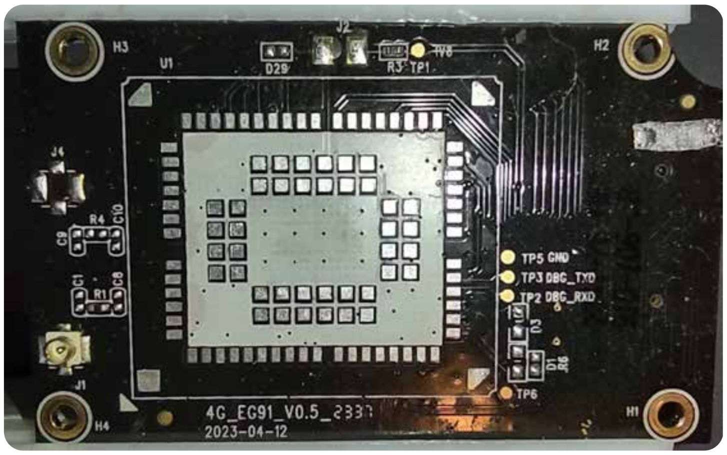


Figure 23: Footprint of Quectel EG91-NAXD on Trail Camera's PCB

For example, in Figure 24, we can see that the Main UART communication lines, labeled in yellow and blue, were routed in a direction under the module body, which would have impacted our transparency overlay method used for finding possible communication on the circuit board, as described at the beginning of this section. Although chip-off examination adds a level of complexity to the testing process, this example helps emphasize the value in chip-off examination methods.

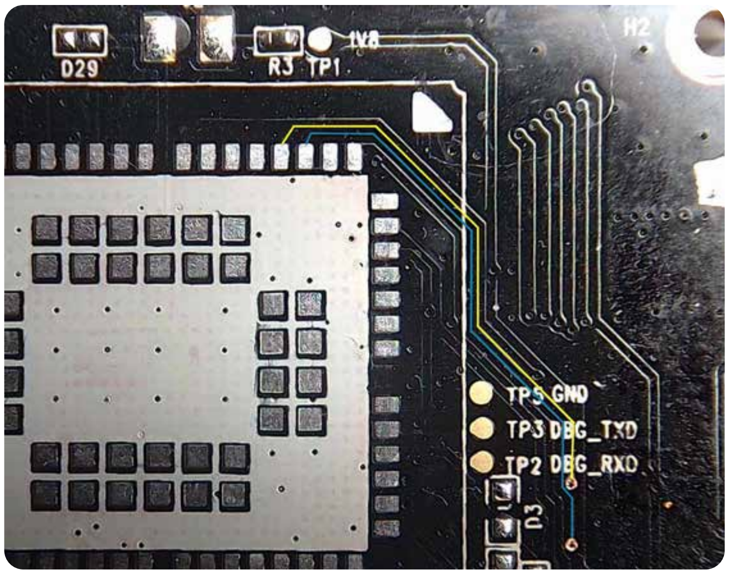


Figure 24: UART Traces Highlighted on PCB

In addition to examining the PCB traces for UART, we were able to map out USB communication lines. Cellular devices typically do not use both USB and the Main UART for transmitting data or controlling the cellular module. Most often, the one (USB or Main UART) not in use will have its LGA lands not connected to any circuit paths on the circuit board, which can indicate the one currently in use.

Additionally, while examining the board with the cellular modules removed and comparing other connection points to the pinout diagram within the EG91 Series design manual, we identified the USB_Boot circuit layout. USB_Boot is used to place the device into a mode for conducting firmware upgrade processes. Examining our device, we also see that the circuit board contains most of the circuit designed to support that functionality, as shown in Figure 25.

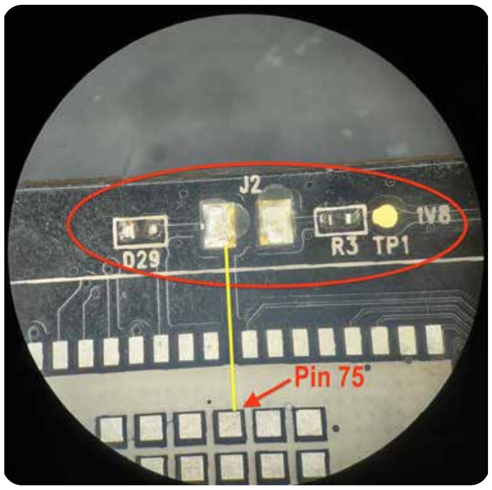


Figure 25: USB_Boot Pin on PCB

It can often be difficult to locate USB communication tap points on a circuit board. This is because, in many cases, those communication lines are routed via the sublayers on the circuit board and not easily accessible. However, many times we have found that the USB circuit path between the cellular module and CPU involved the installation of resistors inline. When that is the case, the USB circuit paths are routed to the circuit board's surface for that purpose. In some cases, to effectively map out those USB paths, it may be necessary to also remove the primary CPU if it is on the same circuit board. For the device we are examining, that was not the case, and we could easily identify various tap points on the circuit board, as shown in Figure 26. Also shown in Figure 26, the USB communication line does have resistors installed inline, as mentioned above.

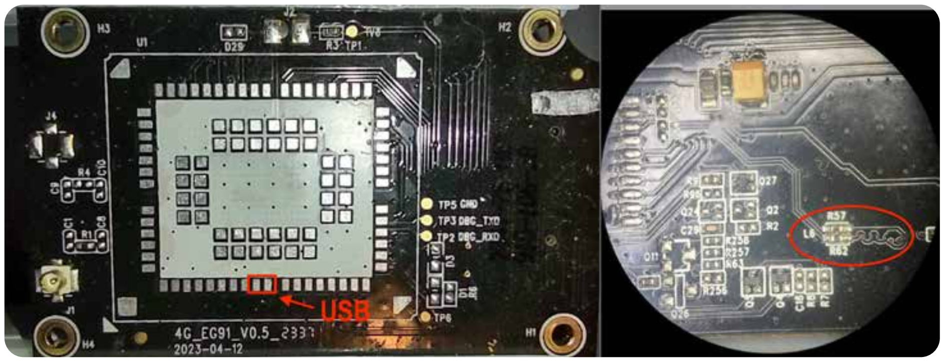


Figure 26: USB Inline Resistors on PCB

To prepare the USB communication for analysis, we also needed to install 20-40 ohm resistors between the USB D+ and D- lines and the testing gear. Adding these resistors helps to prevent impedance mismatching and signal echos that would degrade the USB communication signals. In our case, we installed surface mount device (SMD) size 0601 33 ohm resistors directly on the circuit board next to the inline resistors already installed, which would allow us to attach test gear later during signal capture and analysis. A sample of this is shown in Figure 27.

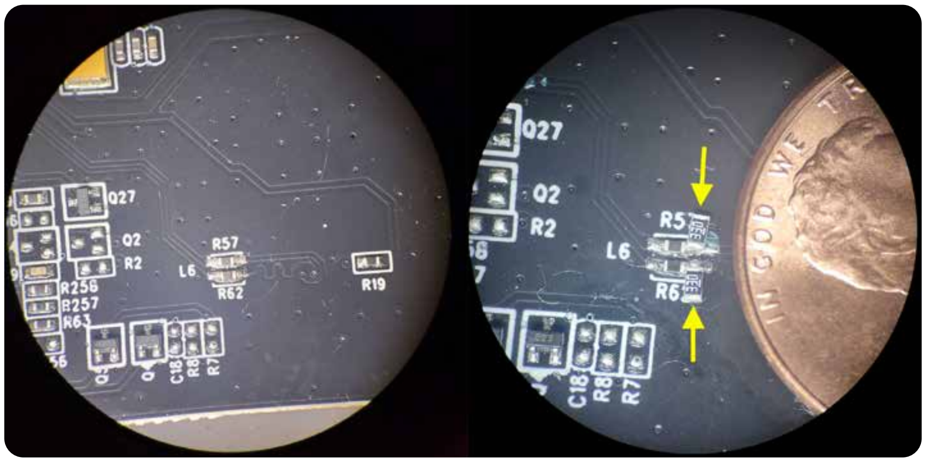


Figure 27: Installation of Additional Resistors

To better streamline this for future research, we decided to build a USB test jig, which could then be attached to the target device and or piggybacked onto the circuit board using E6000 adhesive. Building out the prefabricated test jigs, we purchased USB micro breakout boards and cut the circuit traces for D- and D+, then soldered SMD size 0402 33 ohm resistors in place, as shown in Figure 28.

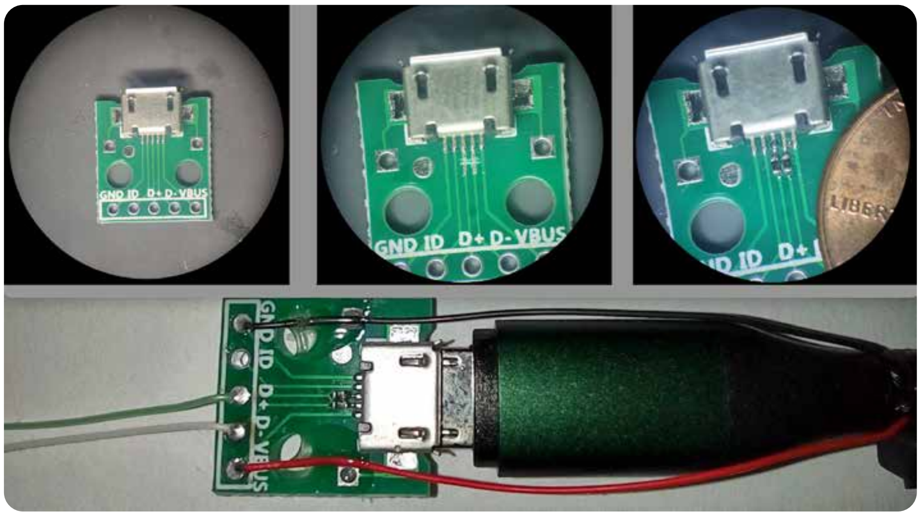


Figure 28: Installation of Resistors on USB Test Jig

In this final section of the Circuit Board Layout Analysis, we want to cover those rare cases where we may find a need to alter the circuit board design to better support the testing work and to gain access to communication features for the cellular device. Although the need for this is rare, it is important to understand the processes and also have the skills to do the work. Building and understanding the skills needed ahead of time will help facilitate a more streamlined and timely effort versus wasting time with trial and error efforts when required.

Although not needed in our test example, we decided to walk through the process of board alteration to add lines from the EG91 LGA pads for Main UART out to an easily accessed point on the circuit board. Figure 29 shows the LGA pads of interest.

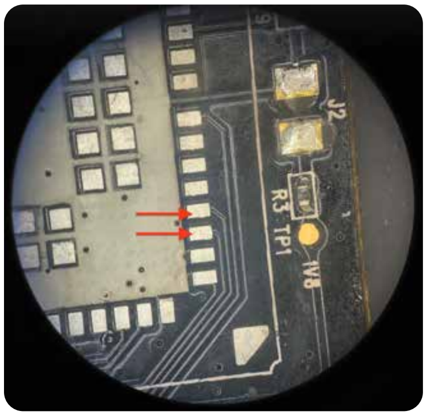


Figure 29: Identified Main UART LGA Pads

Some of the supplies we used for the exercise example included circuit board traces, a pad repair kit, and UV solder mask ink, as shown in Figure 30. We tested several brands of UV ink and found all required a long UV set time to cure. This set time varied from a few minutes to 10+ minutes based on how thick the UV solder mask was applied.

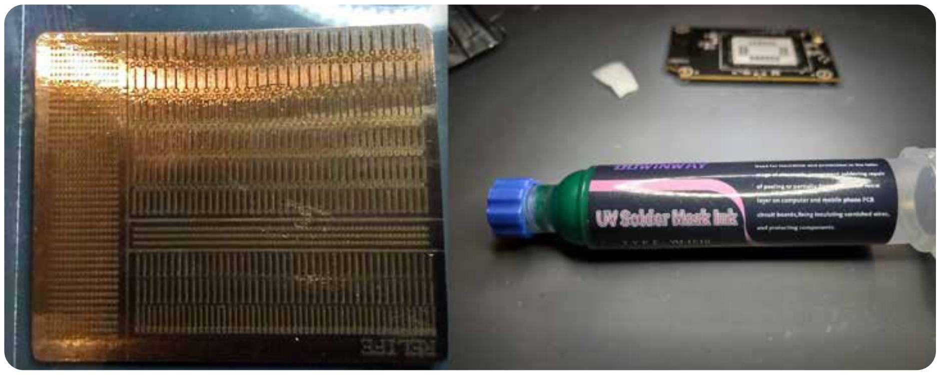


Figure 30: Pad Repair Kit and UV Solder Mask Examples

In the first step, we peeled off the copper repair trace from the kit, which had a sticky back that allowed us to place it on the circuit board to hold it in place. Note: These repair traces are small and fragile and patience should be taken while handling them. In our case, this typically took a couple of tries before we did it correctly without damaging the fragile traces.

Besides using a circuit board repair kit, it was also possible to use thin gauge wire, such as 40 gauge or smaller. Once we had the repair traces properly placed, we tinned the LGA pad and the trace connected to the LGA pad so that proper electrical continuity could be validated between the LGA pad and the repair trace pad. Once that was validated, we applied the UV solder mask to electrically insulate the added trace, and hold it in place. We then set the UV solder mask with a UV lamp, as shown in Figure 31 and Figure 32, respectively.

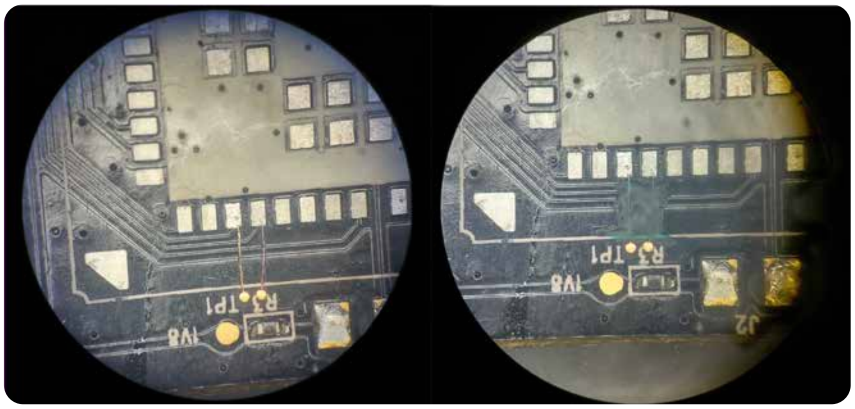


Figure 31: Installed Repaired Traces

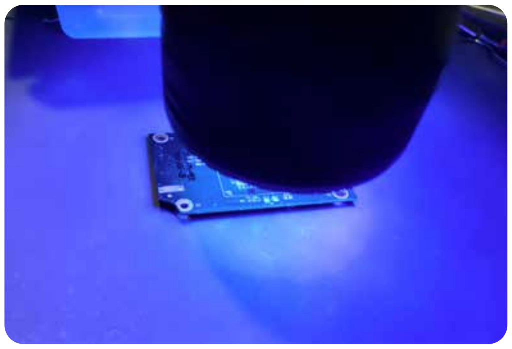


Figure 32: Hardening UV Mask with UV Lamp

Next, we tinned the repair trace pads, added more UV solder mask around the pad, and cured it to strengthen it further, as shown in Figure 33. This was done to avoid accidentally damaging the pads later when connecting them for testing purposes.

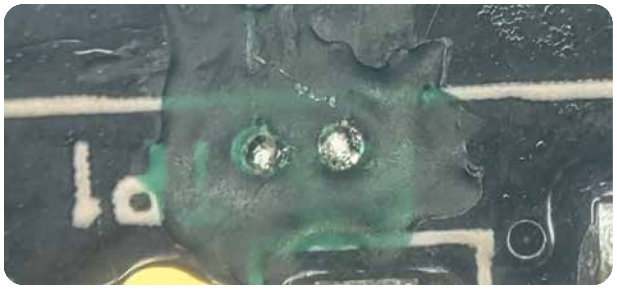


Figure 33: Tinned Repair Trace Pads and Set UV Mask

The final step was re-attaching the cellular module by reflowing it on with an IR oven. To re-attach, there are several methods available. Often the solder paste is applied using a screen that matches the LGA layout for that module. Diagrams of these screens are typically available from the manufacturer. If solder screens are not readily available, using solder paste correctly can often be problematic. In those cases, we have successfully used BGA balls applied to the cellular module with much success.

An example of this process is demonstrated in Figure 34, where BGA balls were used and then reflowed within an IR oven using the manufacturer's recommended temperature curve. The final results after reassembly show the added circuit traces and pads in the right side image of Figure 34.

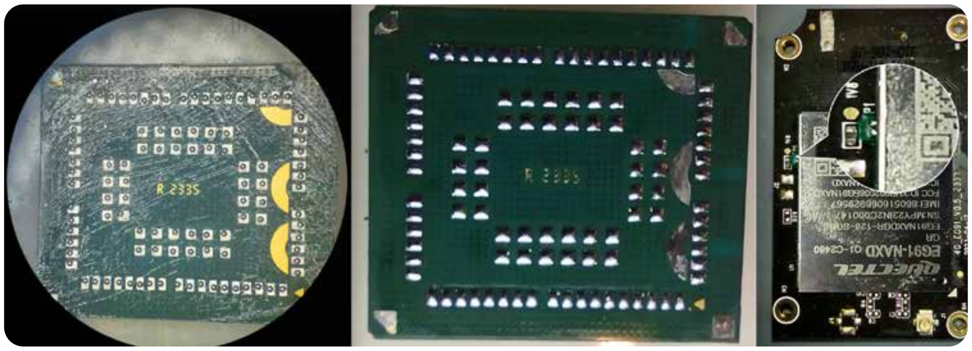


Figure 34: Reflowed BGA Balls with an IR-Oven

AT COMMANDS OVERVIEW

AT commands, short for "Attention Commands," are instructions that allow a host controller to control and debug a modem. AT commands, originally known as the Hayes command set, were developed in 1981 for the Hayes Smartmodem 300. The original Hayes command set allowed for dialing, hanging up, and configuration of the modem.

To support new functionality available with 1200 and 2400 baud modems, a Hayes extended commands set was created with the new commands denoted by including an ampersand (&) prefix. By the early 1990s, with the introduction of the 14.4 and 28.8 kbit/s modems, modern AT commands heavily based on the original Hayes extended set were widely adopted by most modem manufacturers.

The original Hayes extended command set influenced the modern universal AT commands. While AT commands are often common across different modems, AT commands for cellular modems have been defined in Technical Specifications (TS). In particular, the ETSI GSM TS and 3GPP TS 27.007 technical specifications apply to cellular modules that support NB-IoT and CAT-M1/M2. During our research, we also found cellular module manufacturers, such as Telit, U-Blox, Quectel, and Nordic, implemented proprietary AT commands for use in their cellular modules. AT commands in mobile communications can be used to perform many actions including, but not limited to:

- Obtaining device or manufacturer information
- Accessing SIM and network information
- Scanning for and registering with supported networks and radio types
- Receiving updates about network conditions
- Waking and putting the device to sleep
- Receiving and sending SMS and calls
- Sending data via TCP/IP

There are four (4) main AT command types, including test, read, write, and execute. The command type informs the syntax of the command. Each AT command begins with the prefix 'AT' or 'at' followed by a command. The command portion may contain one or more fields or suffixes, which indicate the command mode or 'type' and a data field for commands that support or require data. The table below outlines the command syntax associated with the main AT command types.

Type	Syntax	Function
Test	AT+<COMMAND>=?	Returns parameters and value ranges set by <COMMAND>.
Read	AT+<COMMAND>?	Returns the current parameter values.
Write	AT+<COMMAND>=<User Input>	Sets command parameters to user-defined values.
Execute	AT+<COMMAND>	Executes the command. May function similarly to read returning non-variable parameters.

Note that AT commands can be sent individually, with each AT command terminated by a carriage return or multiple commands placed on the same line with a single AT prefix at the beginning of the line followed by individual commands separated by a semicolon.

The 3GPP TS 27.007 details a standard AT command set for controlling Mobile Termination (MT) functions and network services from Terminal Equipment (TE) through Terminal Adapter (TA). The 3GPP TS 27.007 used commands from ITU T Recommendation V.250 and existing digital cellular standards whenever applicable.

Some examples of common AT commands include:

- **AT: This is the most basic AT command, confirming that the modem and host system can communicate, denoted by an 'OK' response.**
- **AT&F: This command sets all modem parameters to manufacturer defaults.**
- **AT+CPIN: This command can query whether the module requires a PIN, which is necessary for registration, and be used to set the PIN. By issuing this AT command with a question mark (?) at the end, the cellular module will respond with whether a PIN is set, whereas issuing this command with an equal sign (=) followed by alphanumeric characters will set a PIN.**
- **AT+CGSN: This command requests the product serial number identification or the International Mobile Equipment Identifier (IMEI).**
- **AT+CLAC: This command will return a list of AT commands available to the user.**

As mentioned previously, manufacturers can implement their own proprietary commands for use with their cellular modules. Based on AT command documentation and articles, manufacturer proprietary commands can be identified based on the syntax adopted by the manufacturer.

Examples of cellular module manufacturer's proprietary AT command syntax include:

- Quectel: AT+Q<COMMAND>
- U-Blox: AT+U<COMMAND>
- Telit: AT@<COMMAND>, AT#<COMMAND>, AT$<COMMAND>, or AT*<COMMAND>
- Nordic: AT%<COMMAND>

Since AT commands are used to perform various functions on a cellular modem, unauthorized use poses an inherent security risk to cellular-enabled devices. We reviewed previous security research on AT command vulnerabilities and found it focused on the Android ecosystem and smartphones. In particular, a research project called ATtention Spanned released a research paper at USENIX in 2018 on vulnerable AT commands. Additionally, we found a security paper titled "ATFuzzer: Dynamic Analysis Framework for AT Interface for Android Smartphones," which had a corresponding GitHub repository where the researchers published the Python-based ATFuzzer tool.

As part of our research, we wanted to focus on the ability to intercept and identify AT commands sent to the cellular module and attempt to inject AT commands. Since there are both standard and custom AT commands, we used documentation released by vendors to generate a list of possible AT commands. Manufacturers will often include AT commands in cellular module documentation or AT command reference guides that have proprietary AT commands, such as Telit's AT Commands Reference Guide.

To automate testing AT commands against cellular modules, we created a couple of simple Python scripts that can be used to send various AT commands when connected to the Main UART of the cellular module. The scripts we generated are focused on the following functions:

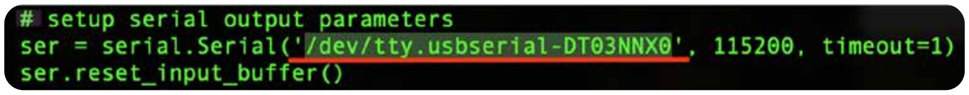
- Read command options
- Retrieve device settings and status
- Fuzz command options

Each script allows the user to configure the local UART interface within the script. Additionally, the ATquery.py script allows users to select from common cellular module manufacturers, including Quectel, Telit, and U-Blox, to read and test against common AT commands. The CellModuleATFuzz.py script enables the user to input individual AT commands for testing and fuzzing the various variables associated with each AT command. The following section describes the syntax for using these various scripts.

The first script, ATquery.py, allows us to query the cellular module via serial connection to carry out two possible options:

1. Identify which commands are functional by using the TEST switch, or
2. Query the available commands using the READ switch to retrieve the current setting for that command, if available.

To initially set up the script, you must add the location path for the attached serial device to this script. Currently, the change needs to be made on line 40, as shown below in Figure 35.



```
# setup serial output parameters
ser = serial.Serial('/dev/tty.usbserial-DT03NNX0', 115200, timeout=1)
ser.reset_input_buffer()
```

Figure 35: Configuration of Serial Connection on Line 40 of ATquery.py

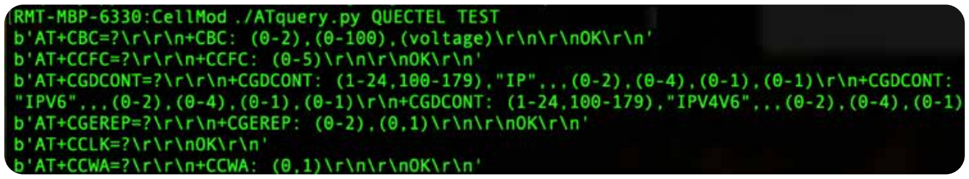
The syntax for this script is simple and takes two arguments. The first argument is for the brand of the device being tested. The script currently supports the following manufacturers:

- CINTERION
- QUECTEL
- TELIT
- UBLOX

Note: The supplied AT commands do not include those that may negatively impact testing, such as those that cause the cellular module to reboot, shutdown, sleep, or delete data, which impacts desired functionality when testing.

The second switch is either READ or TEST. As mentioned above, the TEST switch returns the command with the parameters and available input syntax data. The READ switch will return the current setting under that command, if available. If the command or data are unavailable, the device will return an error message for that AT command.

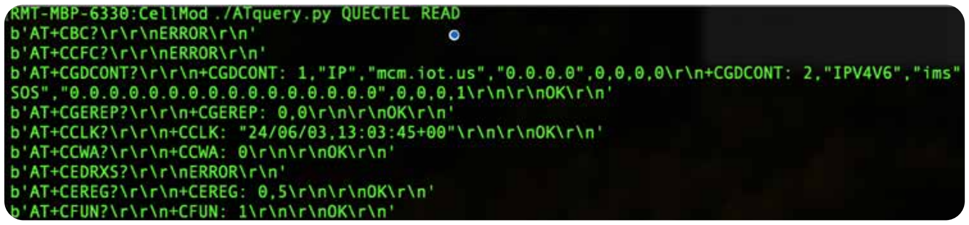
An example of the script ATquery.py being run against an EG91-NAXD cellular module with the TEST switch set, as shown in Figure 36.



```
RMT-MBP-6330:CellMod ./ATquery.py QUECTEL TEST
b'AT+CBC=?\r\r\n+CBC: (0-2),(0-100),(voltage)\r\n\r\nOK\r\n'
b'AT+CCFC=?\r\r\n+CCFC: (0-5)\r\n\r\nOK\r\n'
b'AT+CGDCONT=?\r\r\n+CGDCONT: (1-24,100-179),"IP",,,(0-2),(0-4),(0-1),(0-1)\r\n+CGDCONT:
"IPV6",,,(0-2),(0-4),(0-1),(0-1)\r\n+CGDCONT: (1-24,100-179),"IPV4V6",,,(0-2),(0-4),(0-1)
b'AT+CGEREP=?\r\r\n+CGEREP: (0-2),(0,1)\r\n\r\nOK\r\n'
b'AT+CCLK=?\r\r\nOK\r\n'
b'AT+CCWA=?\r\r\n+CCWA: (0,1)\r\n\r\nOK\r\n'
```

Figure 36: ATquery.py with TEST Switch Set for a Quectel Module

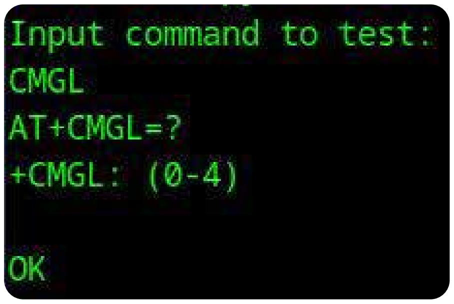
Figure 37 shows the script ATquery.py running against the EG91-NAXD cellular module with the READ switch set.



```
RMT-MBP-6330:CellMod ./ATquery.py QUECTEL READ
b'AT+CBC?\r\r\nERROR\r\n'
b'AT+CCFC?\r\r\nERROR\r\n'
b'AT+CGDCONT?\r\r\n+CGDCONT: 1,"IP","mcm.iot.us","0.0.0.0",0,0,0,0\r\n+CGDCONT: 2,"IPV4V6","ims"
SOS","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1\r\n\r\nOK\r\n'
b'AT+CGEREP?\r\r\n+CGEREP: 0,0\r\n\r\nOK\r\n'
b'AT+CCLK?\r\r\n+CCLK: "24/06/03,13:03:45+00"\r\n\r\nOK\r\n'
b'AT+CCWA?\r\r\n+CCWA: 0\r\n\r\nOK\r\n'
b'AT+CEDRXS?\r\r\nERROR\r\n'
b'AT+CEREG?\r\r\n+CEREG: 0,5\r\n\r\nOK\r\n'
b'AT+CFUN?\r\r\n+CFUN: 1\r\n\r\nOK\r\n'
```

Figure 37: ATquery.py with READ Switch Set for a Quectel Module

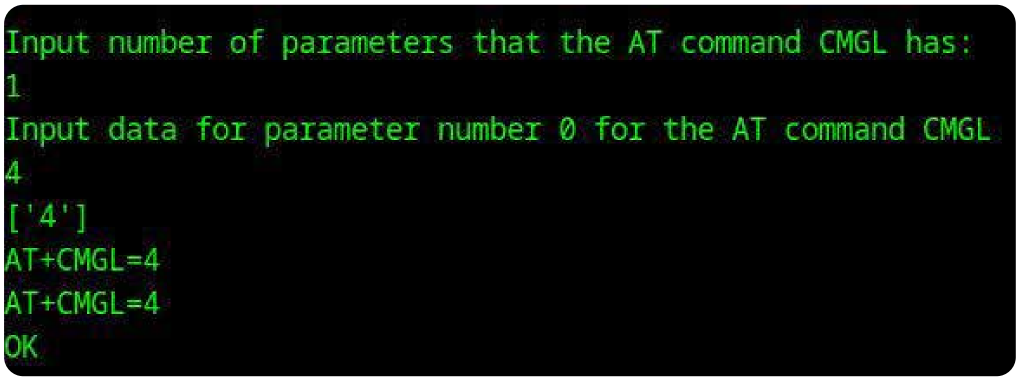
The second script, CellModuleATFuzz.py, allows us to query the cellular module and inject various values into the options available for an AT command. Upon initial execution of CellModuleATFuzz.py, the user is prompted for an AT command. The script runs the TEST version of the AT command and returns the available parameters and values. Figure 38 shows the initial prompt and supplied AT command "CMGL," used to list either new (0), read (1), stored and not sent (2), stored and sent (3), or all (4) SMS messages depending on the value selected.



```
Input command to test:
CMGL
AT+CMGL=?
+CMGL: (0-4)

OK
```

Figure 38: Initial Prompts for CellModuleATFuzz.py

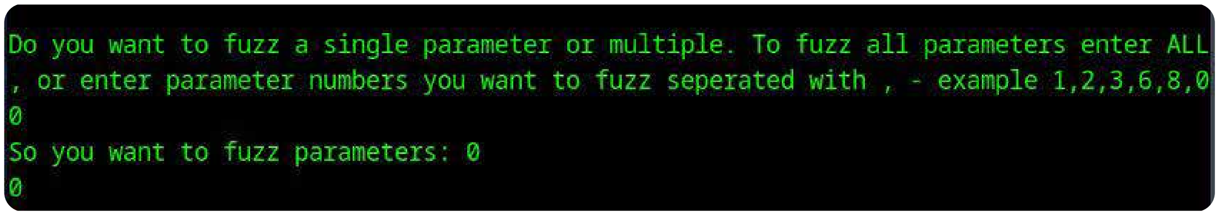
The Python script then requests the number of parameters for the supplied AT command and user input for the parameters with an index beginning at zero (0), as shown in Figure 39.



```
Input number of parameters that the AT command CMGL has:
1
Input data for parameter number 0 for the AT command CMGL
4
['4']
AT+CMGL=4
AT+CMGL=4
OK
```

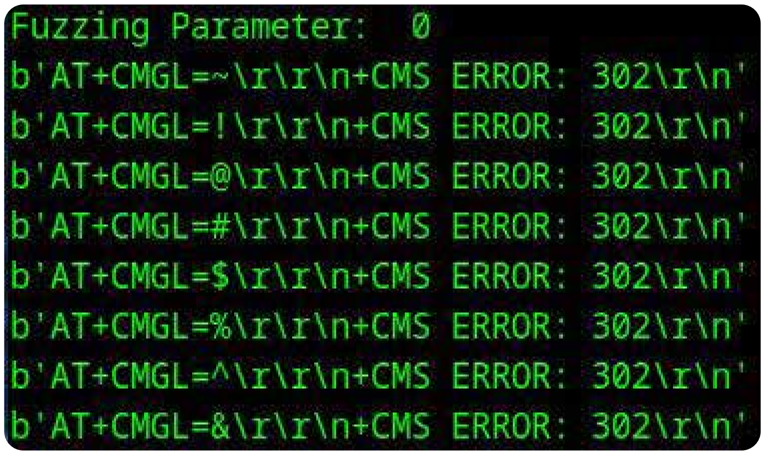
Figure 39: User Input Configured for AT Command Parameters in CellModuleATFuzz.py

Once the parameters are defined, a user can either select to fuzz all the parameters or select individual parameters, as shown in Figure 40.



```
Do you want to fuzz a single parameter or multiple. To fuzz all parameters enter ALL
, or enter parameter numbers you want to fuzz seperated with , - example 1,2,3,6,8,0
0
So you want to fuzz parameters: 0
0
```

Figure 40: Selecting AT Command Parameters to Fuzz



```
Fuzzing Parameter:  0
b'AT+CMGL=~\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=!\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=@\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=#\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=$\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=%\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=^\r\r\n+CMS ERROR: 302\r\n'
b'AT+CMGL=&\r\r\n+CMS ERROR: 302\r\n'
```

Figure 41: Output for CellModuleATFuzz.py

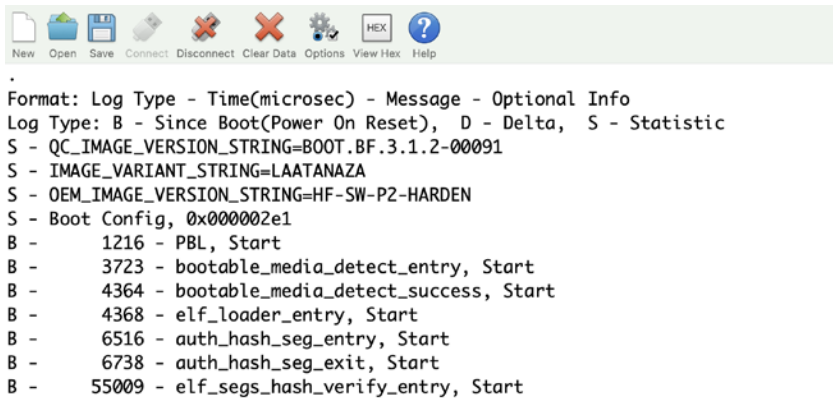
The script will then use the supplied file of fuzzing payloads to test the selected parameters. The output, including the command sent and the response, is printed to the console and saved to a text file for easy review. Figure 41 shows the fuzzing initiated on the AT+CMGL command.

INTER-CHIP COMMUNICATION DATA CAPTURE AND ANALYSIS

In the following section, we focus on capturing data and understanding what is taking place and possibly manipulating that data in some cases to allow for more advanced security testing to take place. To effectively do these we need to first be able to capture the data. Using the method described in the previous section, we can tap into Main UART and USB communication for this purpose. In the following examples, we will be using a cellular-based trail camera, which is using a Quectel EG91-NAXD cellular module to demonstrate these processes.

GENERAL UART EXAMINATION

The first communication data we want to look at on this device is the UART. This cellular module has two UART connections, one for device debug purposes (Debug UART) and one for cellular data and/or AT commands (Main UART). When we attach to the Debug UART we can see the device boot up and a login prompt show up. This boot process shows some detailed messaging that can help reveal some details about the cellular module and its current state. An example of this output of the cellular module device booting up is shown in Figure 42.



```
.
Format: Log Type - Time(microsec) - Message - Optional Info
Log Type: B - Since Boot(Power On Reset),  D - Delta,  S - Statistic
S - QC_IMAGE_VERSION_STRING=BOOT.BF.3.1.2-00091
S - IMAGE_VARIANT_STRING=LAATANAZA
S - OEM_IMAGE_VERSION_STRING=HF-SW-P2-HARDEN
S - Boot Config, 0x000002e1
B -      1216 - PBL, Start
B -      3723 - bootable_media_detect_entry, Start
B -      4364 - bootable_media_detect_success, Start
B -      4368 - elf_loader_entry, Start
B -      6516 - auth_hash_seg_entry, Start
B -      6738 - auth_hash_seg_exit, Start
B -     55009 - elf_segs_hash_verify_entry, Start
```

Figure 42: Debug UART Output

Next, we focused on the MAIN UART port, which handles cellular communication and associated cellular modem commands. Since this Main UART had communication going in both directions (Inter-chip communication) between the cellular module and the CPU, it is important to capture communication in both directions, represented in Figure 43.

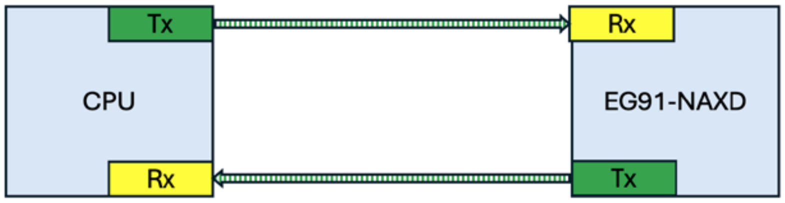


Figure 43: Inter-chip Communications Example

To accomplish this, we used two FTDI serial to USB devices, represented in Figure 44, attached the Rx (receiver) side of the FTDI devices to the device under test, and started two instances of Coolterm, a serial application to the FTDI devices. When choosing FTDI devices, it is important to match the device to the proper voltages of the UART you are attaching to. This voltage is most often either 1.8vdc or 3.3vdc. In the case of the EG91 modules, the voltage being used is 1.8vdc, as identified within the available datasheets.

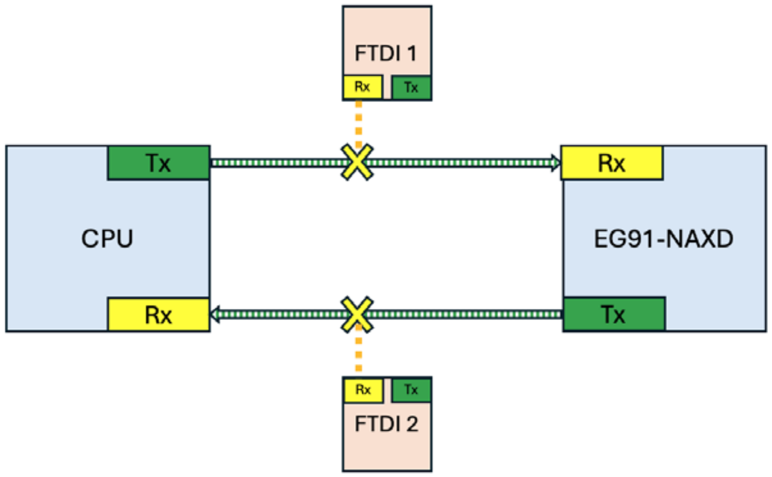
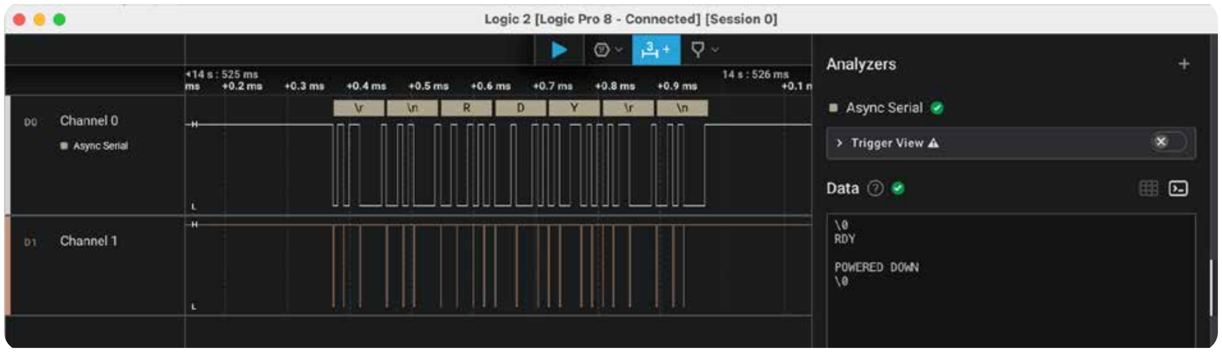


Figure 44: Placement of FTDI Serial Devices

Using the methods described in the section “Circuit Board Layout Analysis,” we identified the location of the Main UART — which is typically used for cellular data and or AT commands — and attached a Saleae logic analyzer to the circuit to identify communication and baud rate. With the Saleae logic analyzer, we determined the baud rate to be 115200 and that the CPU was not using the Main UART for cellular communication nor cellular modem AT commands.

No data or commands were transmitted from the CPU side of the circuit. We only saw the cellular module responding with an ‘RDY’ ready response when it powered up for initial functionality. Additionally, we noted a ‘Powered Down’ response when the cellular module was turned off or was placed into sleep mode. This indicated that the trail camera device was configured for data and AT commands to flow from CPU over USB to the cellular module device. An example of this Main UART communication from the cellular module on the Saleae logic analyzer is shown in Figure 45.



```
\0
RDY

POWERED DOWN
\0
```

Figure 45: Main UART Communication Example from Trail Camera

USB EXAMINATION AND EVALUATION

Next, we focused on examining the USB communications of the sample device (a cellular trail camera). Typical USB communication within IoT technology is the USB 2.0 standard or a subset of that standard. The communication speed of USB 2.0, also known as high-speed USB, is 480Mbps. To sniff and decode this traffic requires a device that can handle communications at that speed. Typical logic analyzers are unable to sniff and decode high-speed USB. Also, many USB sniffers are designed to only be placed in line to be able to capture and decode USB traffic. In the case of high-speed inter-chip USB communication, it is not possible to place a typical USB sniffer device inline. However, after some searching, we found that the Beagle USB 480 product from Total Phase was capable of performing inter-chip communication sniffing.

Using the configuration discussed in the section “Circuit Board Layout Analysis” for tapping into USB inter-chip communication, we attached our USB sniffer to the device being tested, as shown in Figure 46.

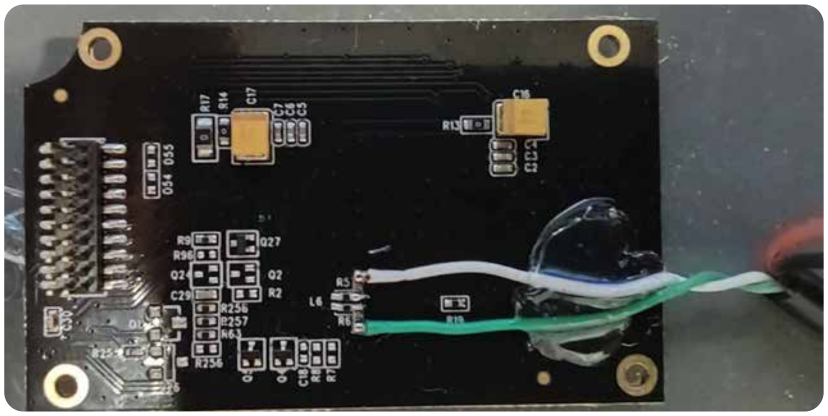


Figure 46: Inline USB Sniffing

According to the manufacturer, to work properly, the Beagle also needs to detect the VBus voltage, which would normally be detected when inline testing of USB devices. However, since this is inter-chip communication and VBus is not always accessible, we tricked the Beagle by supplying the 5v VBus to it via an external power supply, as shown in Figure 47. The vendor also defines settings that can be made to the Beagle on their support site to bypass the required detection of the 5v VBus.

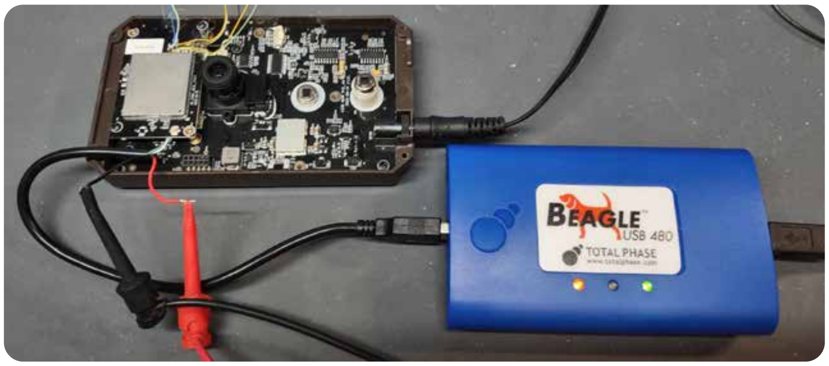
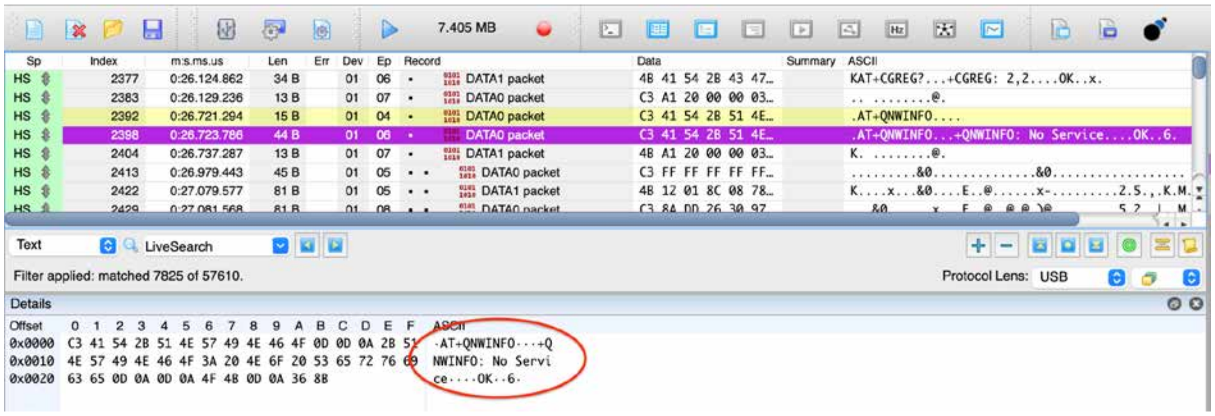


Figure 47: Attachment of an External 5V Power Supply for VBus

The device under test had not yet been registered nor activated for cellular service, since we wanted to capture the entire activation, configuration, and data transfer via the USB sniffer. By doing this, it allows us to gather a full understanding of — from beginning to end — how a hardware device is set up and integrated into a functioning ecosystem.

Once the sniffer application "Data Center", which was provided by Total Phase, was connected and started, we enabled filters to only show data packets. We quickly saw that the CPU was sending AT command queries to the cellular module to make some general settings and to check the status.

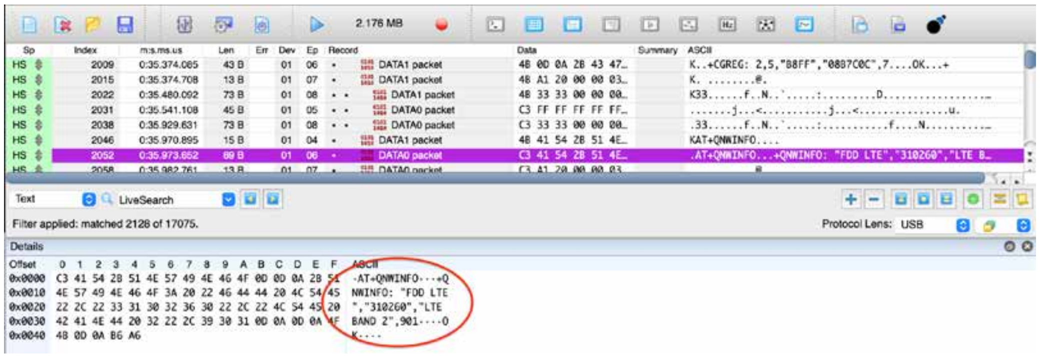
Since we had not requested activation of the product service as of yet, we see a number of queries about setting and status showing that the service has not yet been activated. For example, the CPU continued to make requests for the status of the network using the command AT+QNWINFO to the cellular module, and the cellular module continued to reply 'No Service,' as shown in Figure 48.



```
7.405 MB
Sp  Index  m:s.ms.us     Len   Err  Dev  Ep  Record             Data                  Summary  ASCII
HS  2377   0:26.124.862  34 B       01   06  •    DATA1 packet  4B 41 54 2B 43 47…             KAT+CGREG?...+CGREG: 2,2....OK..x.
HS  2383   0:26.129.236  13 B       01   07  •    DATA0 packet  C3 A1 20 00 00 03…             .. ........@.
HS  2392   0:26.721.294  15 B       01   04  •    DATA0 packet  C3 41 54 2B 51 4E…             .AT+QNWINFO....
HS  2398   0:26.723.786  44 B       01   06  •    DATA0 packet  C3 41 54 2B 51 4E…             .AT+QNWINFO...+QNWINFO: No Service....OK..6.
HS  2404   0:26.737.287  13 B       01   07  •    DATA1 packet  4B A1 20 00 00 03…             K. ........@.
HS  2413   0:26.979.443  45 B       01   05  • •    DATA0 packet  C3 FF FF FF FF FF…           .........&0.............&0................
HS  2422   0:27.079.577  81 B       01   05  • •    DATA1 packet  4B 12 01 8C 08 78…           K....x...&0....E..@......x-.........2.5.,.K.M.

Text   LiveSearch
Filter applied: matched 7825 of 57610.                                   Protocol Lens: USB
Details
Offset  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   ASCII
0x0000  C3 41 54 2B 51 4E 57 49 4E 46 4F 0D 0D 0A 2B 51  ·AT+QNWINFO···+Q
0x0010  4E 57 49 4E 46 4F 3A 20 4E 6F 20 53 65 72 76 69  NWINFO: No Servi
0x0020  63 65 0D 0A 0D 0A 4F 4B 0D 0A 36 8B              ce····OK··6·
```

Figure 48: Example Sniffing of AT Commands

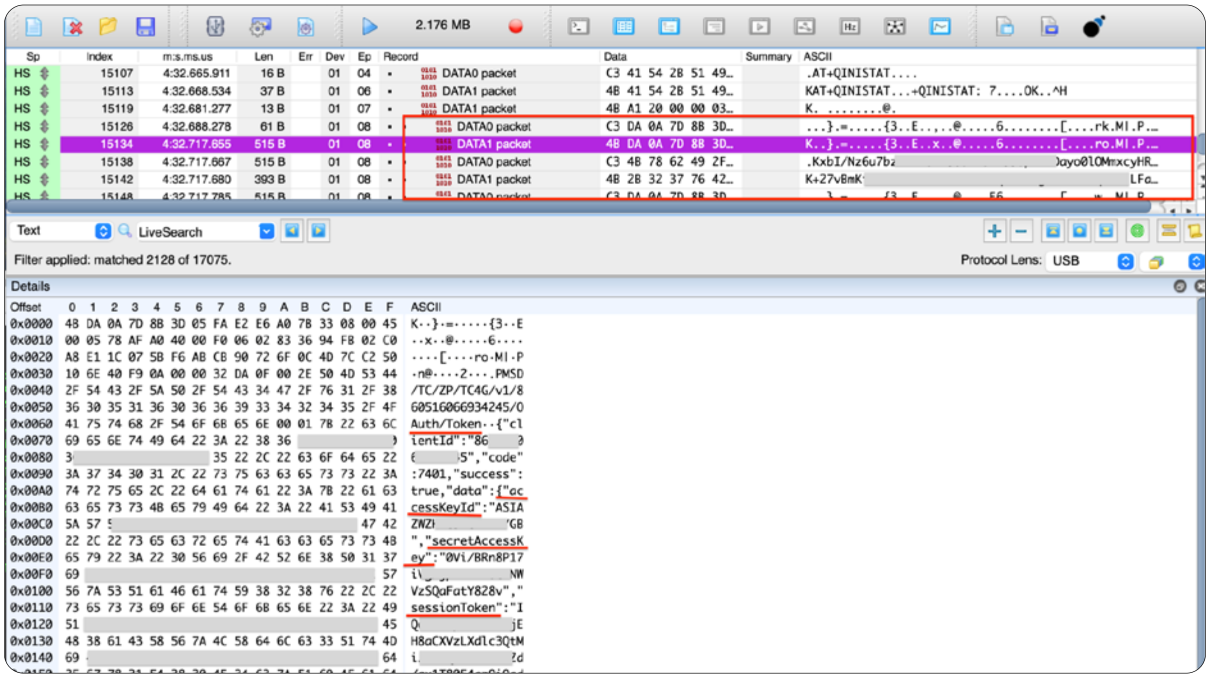
Once the service activation occurs and the cellular module connects to the LTE-M1 cellular services, the response to the AT+QNWINFO returns details on the current network status, as shown in Figure 49.



```
2.176 MB
Sp  Index  m:s.ms.us     Len   Err  Dev  Ep  Record             Data                  Summary  ASCII
HS  2009   0:35.374.065  43 B       01   06  •    DATA1 packet  4B 0D 0A 2B 43 47…             K..+CGREG: 2,5,"B8FF","08B7C0C",7....OK...+
HS  2015   0:35.374.708  13 B       01   07  •    DATA1 packet  4B A1 20 00 00 03…             K. ........@.
HS  2022   0:35.480.092  73 B       01   08  • •    DATA1 packet  4B 33 33 00 00 00…           K33......f..N..`.....:.........D................…
HS  2031   0:35.541.108  45 B       01   05  • •    DATA0 packet  C3 FF FF FF FF FF…           .......j...<..........j...<..............u.
HS  2038   0:35.929.631  73 B       01   08  • •    DATA0 packet  C3 33 33 00 00 00…           .33......f..N..`.....:...........f....N..........…
HS  2046   0:35.970.895  15 B       01   04  •    DATA1 packet  4B 41 54 2B 51 4E…             KAT+QNWINFO....
HS  2052   0:35.973.652  69 B       01   06  •    DATA0 packet  C3 41 54 2B 51 4E…             .AT+QNWINFO...+QNWINFO: "FDD LTE","310260","LTE B…

Text   LiveSearch
Filter applied: matched 2128 of 17075.                                   Protocol Lens: USB
Details
Offset  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   ASCII
0x0000  C3 41 54 2B 51 4E 57 49 4E 46 4F 0D 0D 0A 2B 51  ·AT+QNWINFO···+Q
0x0010  4E 57 49 4E 46 4F 3A 20 22 46 44 44 20 4C 54 45  NWINFO: "FDD LTE
0x0020  22 2C 22 33 31 30 32 36 30 22 2C 22 4C 54 45 20  ","310260","LTE 
0x0030  42 41 4E 44 20 32 22 2C 39 30 31 0D 0A 0D 0A 4F  BAND 2",901····O
0x0040  4B 0D 0A B6 A6                                   K····
```

Figure 49: AT Command Output Showing Connection Information

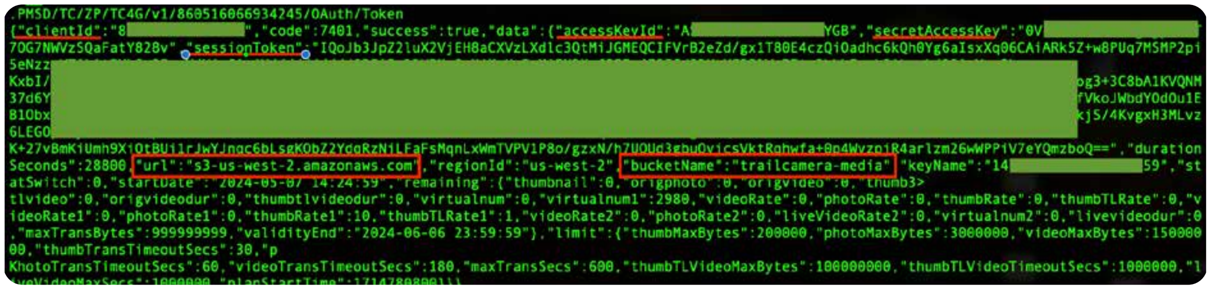
Once the trail camera was configured to capture images and send the data to the available cloud service via cellular communication, we captured key parts of that communication as it flowed between the CPU and the EG91-NAXD cellular modules for transmission. Analyzing this communication helps in properly identifying and understanding the device's end-to-end security and associated security configurations. Figure 50 shows key authentication data being sent for authorization to Amazon S3 buckets and the transmission of captured images and video.



```
2.176 MB

Sp  Index  m:s.ms.us    Len    Err Dev Ep Record         Data                Summary ASCII
HS  15107  4:32.665.911 16 B       01  04 • DATA0 packet C3 41 54 2B 51 49…          .AT+QINISTAT....
HS  15113  4:32.668.534 37 B       01  06 • DATA1 packet 4B 41 54 2B 51 49…          KAT+QINISTAT...+QINISTAT: 7....OK..^H
HS  15119  4:32.681.277 13 B       01  07 • DATA1 packet 4B A1 20 00 00 03…          K. ........@.
HS  15126  4:32.688.278 61 B       01  08 •   DATA0 packet C3 DA 0A 7D 8B 3D…        ...}.=.....{3..E..,..@.....6........[....rk.MI.P…
HS  15134  4:32.717.655 515 B      01  08 •   DATA1 packet 4B DA 0A 7D 8B 3D…        K..}.=.....{3..E..x..@.....6........[....ro.MI.P…
HS  15138  4:32.717.667 515 B      01  08 •   DATA0 packet C3 4B 78 62 49 2F…        .KxbI/Nz6u7bz[redacted]Dayo0lOMmxcyHR…
HS  15142  4:32.717.680 393 B      01  08 •   DATA1 packet 4B 2B 32 37 76 42…        K+27vBmK[redacted]LFa…
HS  15148  4:32.717.785 515 B      01  08 •   DATA0 packet C3 DA 0A 7D 8B 3D…

Text   LiveSearch
Filter applied: matched 2128 of 17075.                    Protocol Lens: USB

Details
Offset  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   ASCII
0x0000  4B DA 0A 7D 8B 3D 05 FA E2 E6 A0 7B 33 08 00 45  K··}·=·····{3··E
0x0010  00 05 78 AF A0 40 00 F0 06 02 83 36 94 FB 02 C0  ··x··@·····6····
0x0020  A8 E1 1C 07 5B F6 AB CB 90 72 6F 0C 4D 7C C2 50  ····[····ro·MI·P
0x0030  10 6E 40 F9 0A 00 00 32 DA 0F 00 2E 50 4D 53 44  ·n@····2····.PMSD
0x0040  2F 54 43 2F 5A 50 2F 54 43 34 47 2F 76 31 2F 38  /TC/ZP/TC4G/v1/8
0x0050  36 30 35 31 36 30 36 36 39 33 34 32 34 35 2F 4F  60516066934245/O
0x0060  41 75 74 68 2F 54 6F 6B 65 6E 00 01 7B 22 63 6C  Auth/Token··{"cl
0x0070  69 65 6E 74 49 64 22 3A 22 38 36 [redacted]      ientId":"86[redacted]
0x0080  3[redacted]                  35 22 2C 22 63 6F 64 65 22  [redacted]5","code"
0x0090  3A 37 34 30 31 2C 22 73 75 63 63 65 73 73 22 3A  :7401,"success":
0x00A0  74 72 75 65 2C 22 64 61 74 61 22 3A 7B 22 61 63  true,"data":{"ac
0x00B0  63 65 73 73 4B 65 79 49 64 22 3A 22 41 53 49 41  cessKeyId":"ASIA
0x00C0  5A 57 [redacted]                            47 42  ZNZ[redacted]GB
0x00D0  22 2C 22 73 65 63 72 65 74 41 63 63 65 73 73 4B  ","secretAccessK
0x00E0  65 79 22 3A 22 30 56 69 2F 42 52 6E 38 50 31 37  ey":"0Vi/BRn8P17
0x00F0  69 [redacted]                                  57  i[redacted]NW
0x0100  56 7A 53 51 61 46 61 74 59 38 32 38 76 22 2C 22  VzSQaFatY828v","
0x0110  73 65 73 73 69 6F 6E 54 6F 6B 65 6E 22 3A 22 49  sessionToken":"I
0x0120  51 [redacted]                                  45  Q[redacted]jE
0x0130  48 38 61 43 58 56 7A 4C 58 64 6C 63 33 51 74 4D  H8aCXVzLXdlc3QtM
0x0140  69 [redacted]                                  64  i[redacted]Zd
```

Figure 50: Authorization and Transmission of Data

Exporting the captured USB communication data also allows for more detailed parsing and examination of the captured information. For example, the above packets were exported and then parsed to extract the ASCII data showing OAuth Authentication and S3 bucket information, as shown in Figure 51.



```
.PMSD/TC/ZP/TC4G/v1/860516066934245/OAuth/Token
{"clientId":"8[redacted]","code":7401,"success":true,"data":{"accessKeyId":"A[redacted]YGB","secretAccessKey":"0V[redacted]
7OG7NWVzSQaFatY828v","sessionToken":"IQoJb3JpZ2luX2VjEH8aCXVzLXdlc3QtMiJGMEQCIFVrB2eZd/gx1T80E4czQi0adhc6kQh0Yg6aIsxXq06CAiARk5Z+w8PUq7MSMP2pi
5eNzz[redacted]
KxbI/                                                                                                               og3+3C8bA1KVQNM
37d6Y                                                                                                               fVkoJWbdYOdOu1E
B1Obx                                                                                                               kjS/4KvgxH3MLvz
6LEGO
K+27vBmKiUmh9XiOtBUilrJwYJnqc6bLsgKObZ2YdqRzNiLFaFsMqnLxWmTVPV1P8o/gzxN/h7UOUd3gbuOvicsVktRqhwfa+0p4Wvzpi R4arlzm26wWPP1V7eYQmzboQ==","duration
Seconds":28800,"url":"s3-us-west-2.amazonaws.com","regionId":"us-west-2","bucketName":"trailcamera-media","keyName":"14[redacted]59","st
atSwitch":0,"startDate":"2024-05-07 14:24:59","remaining":{"thumbnail":0,"origphoto":0,"origvideo":0,"thumb3>
tlvideo":0,"origvideodur":0,"thumbtlvideodur":0,"virtualnum":0,"virtualnum1":2980,"videoRate":0,"photoRate":0,"thumbRate":0,"thumbTLRate":0,"v
ideoRate1":0,"photoRate1":0,"thumbRate1":10,"thumbTLRate1":1,"videoRate2":0,"photoRate2":0,"liveVideoRate2":0,"virtualnum2":0,"livevideodur":0
,"maxTransBytes":999999999,"validityEnd":"2024-06-06 23:59:59"},"limit":{"thumbMaxBytes":200000,"photoMaxBytes":3000000,"videoMaxBytes":150000
00,"thumbTransTimeoutSecs":30,"p
KhotoTransTimeoutSecs":60,"videoTransTimeoutSecs":180,"maxTransSecs":600,"thumbTLVideoMaxBytes":100000000,"thumbTLVideoTimeoutSecs":1000000,"l
iveVideoMaxSecs":1000000,"planStartTime":1714780800}}}
```

Figure 51: Traffic Capture of OAuth Authentication Information

MAIN UART LEVERAGED FOR COMMAND INJECTIONS

In the final part of the section, we want to circle back and further address how Main UART could be used for testing. On this trail camera device, we found that the Main UART, although electrically connected in-circuit, was not leveraged on this device for command or data communication services.

During monitoring for communication, we only observed the cellular module responding with an 'RDY' response when it powered up and became functional and a 'Powered Down' response when the cellular module was turned off or was placed into sleep mode. Through that observation, we determined that the Main UART was functional and could be used to send and receive command data to the cellular module. To test this option, the Main UART TX and RX circuit traces between the CPU and cellular modem would need to be severed to allow external injection of AT commands via serial FTDI, as shown in Figure 52.

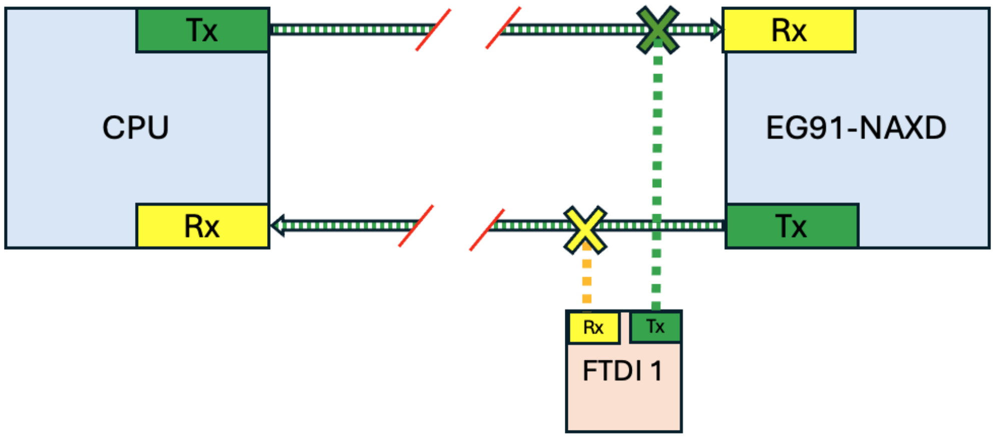


Figure 52: Placement of FTDI Serial to USB Devices for Main UART Channel

To implement the above configuration, we located the Main UART circuit traces between the CPU and the cellular module on the trail camera circuit board, cut the circuit traces, and rerouted each side of the severed connection to a breakout board containing on/off switches with 2.54mm headers and screw terminals, as shown in Figure 53 and Figure 54. This allowed us to connect the device under test and needed test equipment (FTDI, Saleae Logic Analyzer, etc.). This also allowed us to use the on/off switches to open and close the Main UART communication circuit between the CPU and cellular module as needed. Details of this alteration are shown in Figure 53.

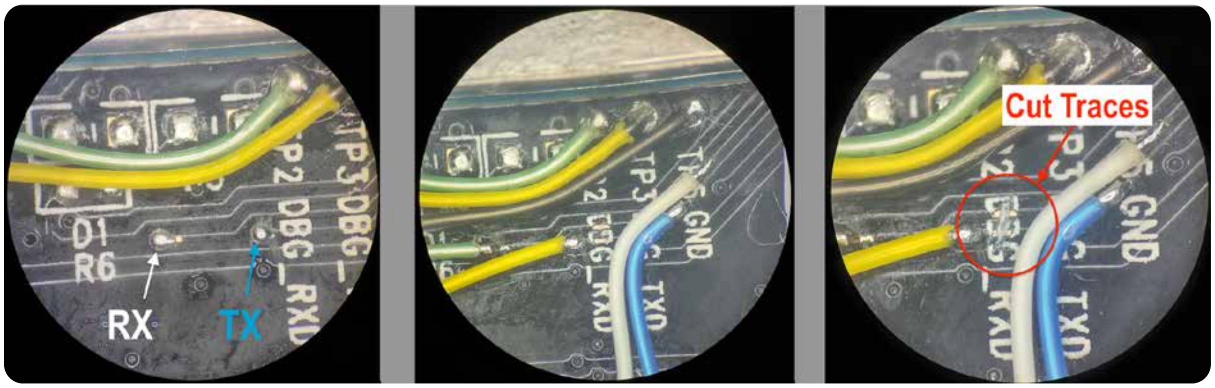


Figure 53: Cut and Tap in Main UART TX and RX Circuit Traces

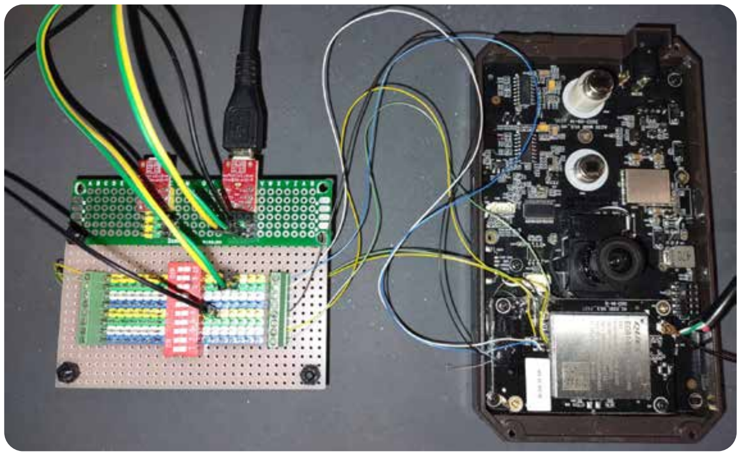
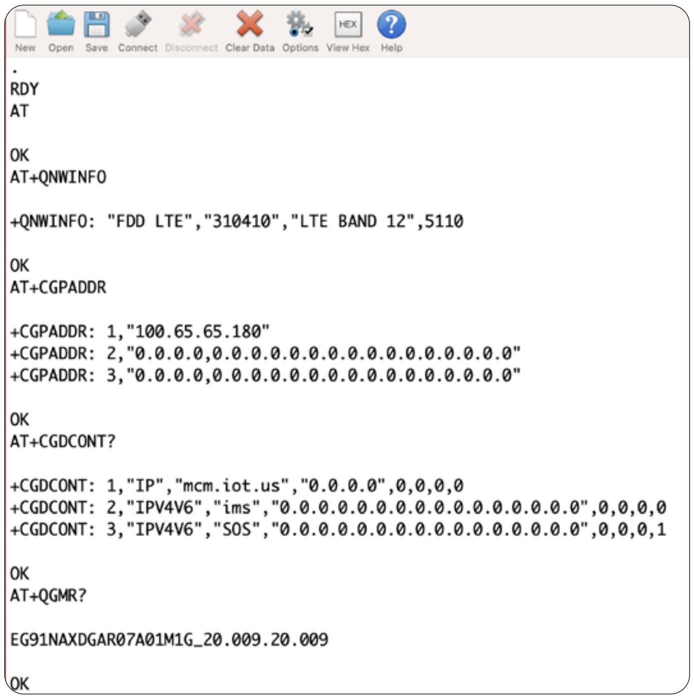


Figure 54: Attachment of Tapped Main UART TX and RX Leads to Switches

Once the trail camera circuit board was prepped and serial FTDI was connected, we powered up the device and opened the on/off switches on the breakout board to sever the Main UART inter-chip communications. Once we received the 'RDY' reply from the cellular module, we were able to then send AT commands, and successfully received responses from the EG91-NAXD cellular module. Figure 55 demonstrates the sending and receiving of multiple AT commands.

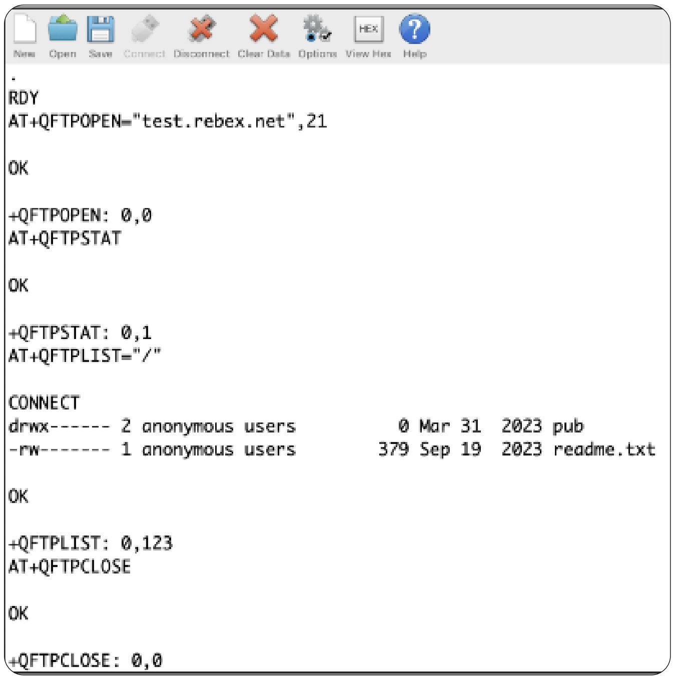


```
.
RDY
AT

OK
AT+QNWINFO

+QNWINFO: "FDD LTE","310410","LTE BAND 12",5110

OK
AT+CGPADDR

+CGPADDR: 1,"100.65.65.180"
+CGPADDR: 2,"0.0.0.0,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0"
+CGPADDR: 3,"0.0.0.0,0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0"

OK
AT+CGDCONT?

+CGDCONT: 1,"IP","mcm.iot.us","0.0.0.0",0,0,0,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,0
+CGDCONT: 3,"IPV4V6","SOS","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0,0,0,1

OK
AT+QGMR?

EG91NAXDGAR07A01M1G_20.009.20.009

OK
```

Figure 55: Successful Execution of AT Commands on Cellular Modules

During this testing, we also injected a full list of AT commands using the ATquery.py scripts described in the AT Commands section and the QUECTEL AT command list, which we created using available Quectel vendor AT manuals we could identify. With this script, we identified that many AT commands were valid despite not being listed within the EG91 AT manual nor being returned by the module when the AT+CLAC command was used. The most interesting of these commands were communications commands for interacting with FTP, HTTP or setting up and communicating over IP sockets.

Next, to test this undocumented functionality on the EG91-NAXD module and validate what level of communication could be established outside of the cellular module's purpose within the trail camera, we used the trail camera's installed cellular module (EG91-NAXD) to establish an FTP connection to the internet and listed the files and folders on that server. This was done by injecting specific AT commands to establish FTP connection to the server over the internet and retrieve listing, as shown in Figure 56.



```
.
RDY
AT+QFTPOPEN="test.rebex.net",21

OK

+QFTPOPEN: 0,0
AT+QFTPSTAT

OK

+QFTPSTAT: 0,1
AT+QFTPLIST="/"

CONNECT
drwx------ 2 anonymous users          0 Mar 31  2023 pub
-rw------- 1 anonymous users        379 Sep 19  2023 readme.txt

OK

+QFTPLIST: 0,123
AT+QFTPCLOSE

OK

+QFTPCLOSE: 0,0
```

Figure 56: FTP Connection to Remote Server

It is also important to note, unlike the device we used in this test example, we have encountered cellular-enabled IoT devices that were configured to only establish private Access Point Name (APN) or private cloud (VPC) connections into backend services and were not accessible directly from the general internet IP address ranges.

This is typically done to add a higher level of security to those services by not having them directly exposed to the internet. In those cases, communication functions on the cellular module could potentially be leveraged to conduct further security testing into those private APN environments using features such as MQTT, FTP, HTTP, and IP socket enablement functions if available on the cellular module of the device being tested. Figure 57 visually represents a high-level diagram of such use cases.

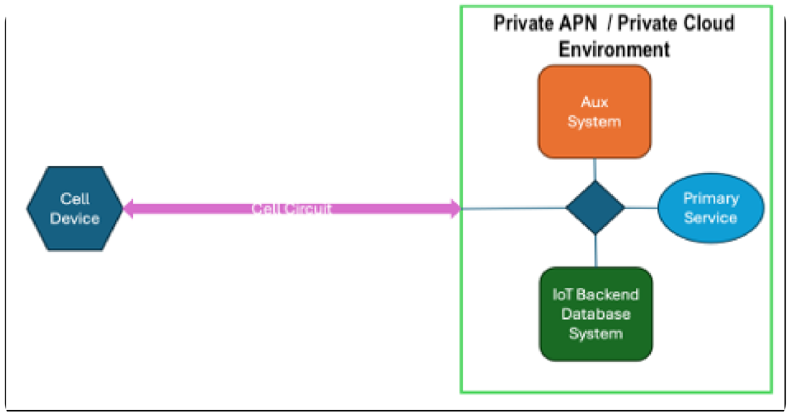

Private APN / Private Cloud Environment
Aux System
Cell Device
Cell Circuit
Primary Service
IoT Backend Database System

Figure 57: Possible Communications Path of Private APN or Private Cloud Environments

CONCLUSION

In conclusion, the main purpose of this paper was to discuss various methods and methodologies for interacting and working with cellular-based IoT technology for the purpose of understanding and being able to effectively test these technologies. Although this paper is only the starting point for being able to truly understand and conduct thorough security testing of cellular IoT technologies, it has helped us build a solid baseline to continue growing.

As with most projects, just the process of sharing this knowledge and working through some of these methods and procedures has helped us further expand our areas of knowledge. It is important to note that even beyond cellular-related device research, the sections "Circuit Board Layout Analysis" and "Inter-chip Communication Data Capture and Analysis" can also be easily applied to other embedded device testing and analysis work.

REFERENCES:

https://i.blackhat.com/USA-19/Wednesday/us-19-Shupeng-All-The-4G-Modules-Could-Be-Hacked.pdf

https://pallavaggarwal.in/2023/11/08/quectel-ec25-cellular-modem-teardown/

https://www.5gamericas.org/wp-content/uploads/2021/01/InDesign-3GPP-Rel-16-17-2021.pdf

https://www.gsma.com/solutions-and-impact/technologies/internet-of-things/wp-content/uploads/2019/07/201906-GSMA-NB-IoT-Deployment-Guide-v3.pdf

https://www.nxp.com/docs/en/application-note/AN2265.pdf

https://www.rapid7.com/globalassets/_pdfs/whitepaperguide/rapid7-leveraging-inter-chip-communication-analysis.pdf

https://github.com/FICS/atcmd

About Rapid7

Rapid7 is creating a more secure digital future for all by helping organizations strengthen their security programs in the face of accelerating digital transformation. Our portfolio of best-in-class solutions empowers security professionals to manage risk and eliminate threats across the entire threat landscape from apps to the cloud to traditional infrastructure to the dark web. We foster open source communities and cutting-edge research–using these insights to optimize our products and arm the global security community with the latest in attacker methodology. Trusted by more than 11,000 customers worldwide, our industry-leading solutions and services help businesses stay ahead of attackers, ahead of the competition, and future-ready for what's next.

RAPID7

PRODUCTS

Cloud Security
XDR & SIEM
Threat Intelligence
Vulnerability Risk Management
Application Security
Orchestration & Automation
Managed Services

CONTACT US

rapid7.com/contact

To learn more or start a free trial, visit: https://www.rapid7.com/try/insight/