THE KEY

TO

SUCCESSFUL WIRELESS FIRMWARE UPDATES

WHAT ENGINEERS NEED TO KNOW BEFORE DESIGNING IOT CLOUD CONNECTED EMBEDDED SYSTEMS

Datakey®

The power of memory. Secured.

PRESENTED BY:

NEXUS INDUSTRIAL MEMORY

EXPERTS IN RUGGED INDUSTRIAL MEMORY SOLUTIONS

Tel: +44 (0) 1794 301439

E-mail: info@nexusindustrialmemory.com

Web: nexusindustrialmemory.com

Datakey products are exclusively distributed by Nexus in Austria, Denmark, Finland, Germany, Ireland, Norway, Sweden, Switzerland and the UK.

Datakey

The power of memory. Secured.

White Paper

The Key to Successful Wireless Firmware Updates

What Engineers Need to Know Before Designing IoT Cloud Connected Embedded Systems

By: Lynn Linse, IoT Senior Design Engineer
ATEK Access Technologies

Table of Contents

INTRODUCTION

A well-known principle of embedded products states that the more complex your device firmware is, the higher the probability that you'll need to replace your firmware in the field. More code means more bugs. More function means your sales team and customers will want still more features. With modern cloud-connected IoT (Internet of Things) devices, this is triply true. Firmware sizes have grown from tens of thousands of bytes, to tens of millions of bytes. Applications expect devices to perform more sophisticated data analysis in the field. IoT cloud upload introduces more protocols which change over time. Simply put, IoT devices must have a rock-solid method for in-field firmware updates.

Since IoT devices have internet access, it's logical to expect to update firmware Over-The-Air (OTA). Updating firmware OTA introduces many challenges – especially when paying for mobile data plans. For example, how much will it cost to move 4,000 kilobytes of firmware on a mobile data plan where overage fees are charged if the IoT device moves more than 250 kilobytes of data per month. Be sure to discuss this scenario when engaging with data providers.

This paper is focused on a different challenge – how to successfully update the firmware of an entire network of IoT devices. Let's start this discussion with a simple quiz. You have 10,000 IoT devices in the field. You need to update your device firmware. Your cloud server is all ready to go. You push the GO button and start the update process. How long will it take to update the firmware in all 10,000 devices?

A) A few days to update all 10,000 devices.

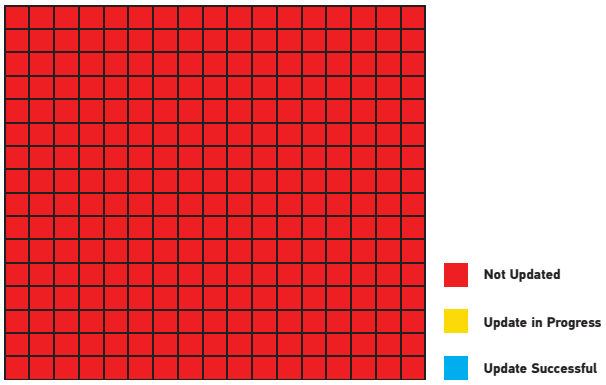
B) It could take months before the last device is updated.

Those who have done this before will know that the correct answer is B. It will likely take months to hit the 100% success goal. Assuming the firmware bug isn't affecting cloud access, then the update of the vast majority (let's say 98%) of devices will go quickly. But what about those last 200 stubborn devices? Worse, what if a firmware bug puts 35% of the devices offline? What if the bug puts 99% of the devices offline?

How can one address these challenges when designing an IoT device so that all (that's 100%) of devices can be updated in the most cost-effective and efficient manner?

ANATOMY OF AN OTA FIRMWARE UPDATE:

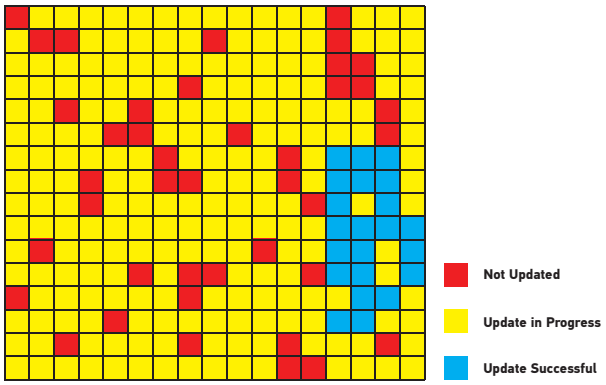
PHASE 1: ALL OF THE DEVICES HAVE THE OLD FIRMWARE:



Here is the initial state – every device is in need of the new firmware update.

The map above shows that 100% of the remote devices are in the undesired state. They are reporting the old firmware and are in need of a firmware update.

PHASE 2: MOST DEVICES HAVE RECEIVED THE FIRMWARE AND HAVE STARTED UPDATING:



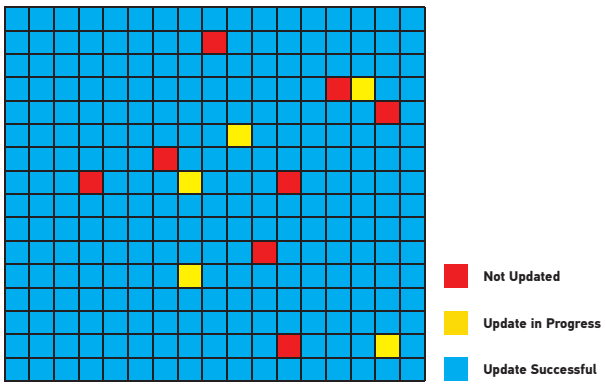
The map above shows that most of the remote devices are in a progressing state (yellow). The yellow devices have accepted the need to reflash and have successfully obtained the new firmware image. A few devices have reached the goal state (blue), where they have reported to the cloud that they have updated their firmware and 100% function has been restored. Still, there are some devices which remain in the old state (red). These devices might be offline or their mobile bandwidth may be so weak that they have not been able to complete the firmware image file transfer.

A requirement of this two-step process (where the device successfully downloads the new firmware image and then switches to the new image), is that all IoT devices which support OTA firmware update, must be able to hold

an entire new firmware image locally. The older firmware paradigm, where a small boot-strap program erases the previous firmware, then downloads the new firmware image OTA , will be problematic. Too many things can go wrong when relying on mobile connectivity! Also, a boot-strap program that handles secure routed SSL/TLS in IP won't be small.

In Phase 2, most of the remote devices have the new firmware. This was the easy part. Ideally, this phase included the appropriate security and authentication steps, which would prevent hacked firmware from being installed. Target Corporation suffered a huge data breach in December of 2013. In the now infamous hacking of the cash registers/point-of-sale terminals, the **IT System** trusted that the infrastructure (network security) would prevent fake firmware from reaching the devices. The infrastructure failed and allowed hacked firmware to reach and be accepted by the trusting terminals. The results? Target's Christmas shopping season was ruined. The breach cost the retailer hundreds of millions of dollars in lost revenue and expenses[1].

PHASE 3: MOST DEVICES HAVE SUCCESSFULLY UPDATED THEIR FIRMWARE:



The map above shows that most of the devices are reporting that they are now running the updated firmware, but NOT 100%. A few devices (red) still have not even begun the firmware-update process, and a few devices (yellow) have failed to complete the update after at least starting the download process. Unfortunately, the job is not done.

Now begins the most painful phase, which is the 3++ phase (three-plus-plus phase). How to get from 98% to 100% success? There are three clear paths to fix the yellow and red nodes:

1. **Send a skilled technician to the site:**
   The technician would diagnose the problem and manually update the firmware. He/She should be equipped with the necessary tools and be trained to perform the required steps. If the device is outdoors or in a hazardous location, it might be necessary to remove the device, fix it somewhere else, then re-install the device, all while NOT breaking anything in the process.

2. **Device replacement:**
   Have the customer return the device and send out a new (or reconditioned) device with the new firmware. The device that failed to update will need to be swapped out and returned, where it may be repaired and updated in-house.

[1]https://www.thesslstore.com/blog/2013-target-data-breach-settled/

3. **Ask a person at the site to help:**
   Depending on the application, at some point it is likely that someone will be onsite as part of their normal activities. This could be a delivery driver, a sales rep, or a customer who is willing to help. Leveraging these resources may be convenient, but it is important to remember that they DO NOT have special tools, cables, or training of the skilled technician mentioned earlier.

Which is the best option to address the non-responsive devices?

Option #1 – Send a skilled technician to the site
In most cases, this option will succeed, but will cost the most money. Unless equipped with a large staff of skilled technicians, an unusually large spike in mandatory site visits will overwhelm most service departments. Certainly, the number of technicians and their geographic location will affect the cost to send them onsite to update the remaining devices. If the task of updating the devices is well enough defined, it may be possible to use subcontractors to complete the updates. Another item to consider is the possibility that removing and reinstalling the device could break something else. So, while Option #1 is low-risk, it could cost, in terms of cash and customer goodwill, an order of magnitude more than the other options mentioned.

Option #2 – Device replacement
In most cases, this option should cost less than Option #1. Shipping a device is relatively inexpensive (in most cases), but the real question is who will perform the removal and replacement of the device? While this option may seem like a good compromise solution, if the customer is tasked with performing the swapping of devices, they may opt to instead discard the offending device and then replace it with a competing device. So, while the tangible costs are less than Option #1, the cost in terms of lost customers may be much greater.

Option #3 – Ask a person at the site to help
This will likely be the lowest-cost option and should be the least likely to fatally break the customer's installation, that is, if the process can be as painless and straight-forward as possible. The optimal question is will customers, delivery drivers or sales reps be willing to help? For most applications, the answer to this question is usually yes.

So assuming availability of willing helpers, what is the best way to manually update the firmware on the units that did not complete the OTA update? The answer is to leverage a well known and proven solution – removable memory.

USING REMOVABLE MEMORY FOR UPDATING FIRMWARE IN IOT DEVICES.

When updating the firmware of cellular IoT devices via an OTA update, there will always be that (hopefully) small percentage that fails to update. As discussed above, the solution for manually updating these devices that has the lowest cost, risk and inconvenience to the customer is to utilise those people who are near the devices as part of their regular duties. But since these people are likely not trained technicians and not experienced with updating the firmware of embedded systems, the manual firmware update process must be as simple and straight-forward as possible.

The solution is to use a removable memory device. A removable memory device can be sent to the person who has agreed to update the device. All he/she needs to do is insert the memory device into the receptacle, hit the RESET button and update process begins. It doesn't get much easier than that!

But for this process to work, the IoT device must be designed to include a removable memory device connector/receptacle, and the system must also support a manual firmware update process using this memory device. The most common types of removable memory include USB flash drives, SD or microSD cards, as well as a variety of proprietary memory devices.

No matter which type of memory device is used, the firmware update process will likely be similar. The user simply inserts the memory device and presses a reset button. The existing firmware will recognise the contents of the memory device as new firmware, validate the new firmware is authorised and compatible, and will then initiate an update of the device's firmware. Because the complete firmware image file exists on the memory device, the update is unlikely to fail.

WHICH MEMORY DEVICE TO CHOOSE

When contemplating which type of removable memory device to use, engineers have a variety of choices. Three of the most-common types of memory devices include USB flash drives, memory cards and proprietary portable memory devices. Are there benefits to using one type over another?

USB Flash Drives

USB flash drives (aka USB sticks, thumb drives, pen drives, etc.) are widely available for purchase and most people already own one. They are low-cost devices and provide gigabytes of storage (not that one would need so much space for a typical firmware update). With a ubiquitous memory device like a USB flash drive, it's not even necessary to physically ship out the drive. The firmware update file can be emailed to the on-site person, where he or she would then copy the file from their computer to their own flash drive. The flash drive could then be inserted into the USB connector of the IoT device and the firmware update would commence. While this sounds simple enough, there are a few problems that may eliminate the USB flash drive from consideration.

1. The first thing to consider is the mentioned delivery method. While emailing the firmware update file to the onsite volunteer sounds convenient, it adds both complexity and risk. How much support will be needed, instructing the volunteer how to copy the file to the flash drive? Will the file reside in the root directory? What if it is stored in a folder? Some organizations even have policies that prohibit (or even prevent) using USB flash drives on their computers out of virus/malware concerns.

2. One must also think about compatibility issues. If the volunteer will be using his own flash drive or buying one, what are the range of capacities and file system formats that the IoT device must support? Will it read older FAT16-formatted devices, current FAT32 devices or newer exFAT devices? How about oddball formats that have Windows-specific compression, for example?

3. Before even getting into compatibility concerns, there are embedded design items to consider. In order for the IoT device to read data from a USB flash drive, it must act as a USB host. Many smaller microcontrollers may not have USB host support built in. While adding support is possible, this increases the cost and complexity of the design. Also, for devices requiring ultra-low power for long battery life, supporting USB host functionality will increase current consumption.

Memory Cards

In microcontroller-based designs, the most popular memory card is the SD card, or its smaller form-factor equivalent, the microSD card. Like USB flash drives, these popular memory devices are affordable, widely available and offer more than enough capacity to store a firmware update image. As above, using an SD card would allow the firmware update file to be emailed to the onsite volunteer. Getting the file from the computer to the SD or microSD card might be a little more challenging. While nearly all PCs have USB ports, not all have built-in SD card slots. Using a card reader is not difficult, but does add another layer of complexity. The SD card does have one distinct advantage over the USB flash drive in that it supports communication over SPI. Many microcontrollers come with SPI ports, so there is no need for the microcontroller to function as a USB host. That being said, SD cards do share some of the same challenges as USB flash drives.

1. As previously mentioned, emailing the update file and getting it successfully transferred to the memory device will add complexity and risk. If the user doesn't have an SD card reader (either built-in to a PC or as a separate USB adapter), the convenience factor has all but evaporated.

2. With USB flash drives, there is the challenge about having to support a variety of mainstream and possibly some less-common file formats. SD cards will have the same challenge.

3. Another possible concern would be long term compatibility. The first recommendation would be to support the latest version of the SD card standard. Supporting SDXC cards and hence providing support for the previous SDHC and SD versions will help ensure compatibility with older, current and future cards to come (for the next couple of years at least). SDXC supports capacities up to 2 TB. 1 TB SD cards are currently available, so in the not too distant future there will be a new SD standard to support SD cards with capacities of greater than 2 TB (not that you would ever need something that big for a firmware update).

Of the two options mentioned so far, SD cards would be the device of choice thanks to its compatibility with SPI ports, but both options introduce security risks. Imagine receiving an email stating that there is an important required update to your IoT devices. It instructs the user to download the update file and transfer it to the IoT devices using either a flash drive or an SD card (whichever the IoT device used). Barring other security features, there would be nothing to prevent the malicious firmware from being installed.

Proprietary Memory Devices
Along with SD cards, another removable memory device to consider would be a proprietary memory device. An example of a proprietary memory device would be Datakey Serial Memory tokens (pictured right). These memory tokens utilise industry-standard non-volatile memory, like serial EEPROM and serial NOR flash. These memory ICs are over-moulded into a variety of proprietary shapes. These proprietary memory devices provide several advantages over the consumer flash drives and SD cards mentioned above.

The Datakey memory token (left) is an example of a proprietary memory device.

1. Their proprietary shape and controlled availability provide added security. Putting other security features aside, these tokens are not available to the public. They are only sold to qualified OEMs. This would make it much harder for a hacker to try to get malicious firmware installed on IoT devices if neither they nor their intended victims have access to these memory devices.

2. Because the OEM controls the memory device, any compatibility questions go away. File formats, folder structure, file location—concern about all of these aren't applicable with a proprietary memory device programmed and shipped out by the OEM.

3. Because proprietary memory devices leverage industry-standard EEPROM and NOR flash memory ICs that have extremely long product life cycles, these removable memory products have excellent long-term product availability—far exceeding that of consumer, NAND flash-based memory devices like USB flash drives and SD cards.

4. Because these memory devices use memory ICs that are commonly used with industrial embedded applications, integrating these into an embedded design is straight-forward. Most use an SPI interface. These devices are also rated for operation over the entire industrial temperature range (-40°C to +85°C), so manual firmware updates in extreme northern (or extreme southern) latitudes in the middle of winter won't be a problem.

5. Since the memory device uses a simple memory access interface, basic strong encryption can be added, providing FIPS-class security, while not overtaxing a small low-power embedded microcontroller.

One of the only downsides to the proprietary memory devices, compared to the consumer memory devices previously mentioned, would be that it would eliminate the solution where the update could be emailed to the customer who would then transfer the update image to the IoT device using their own memory device. So, the OEM would have the added cost of purchasing the proprietary memory devices and sending them to the customer sites. The thing to keep in mind is that this would only be done to those devices that didn't successfully update via the over-the-air update, and the cost to do this is orders of magnitude less than the cost of the two other options for manually updating the firmware.

One other benefit to designing a removable memory device into a cellular IoT device is that the memory device connector can now be used as a removable memory port, where it can support other applications besides firmware updates. For example, a proprietary SPI NOR flash memory token may be used for firmware updates, but an SPI EEPROM memory token might be used for other functions. Some of these other functions are discussed in detail in the section that follows.

OTHER USES FOR REMOVABLE MEMORY IN IOT DEVICES

Besides being used for firmware updates, removable memory has many other potential uses in IoT devices.

Security Functions
Securing IoT devices is a major concern for manufacturers and users of these products. Often, a security solution will include the use of security keys, which may need to be securely loaded or updated. Using a removable memory device, especially if the memory device has its own security features, can be an ideal way of **loading or updating security keys in a way that prevents internet hackers from changing the keys remotely.**

IoT security also includes protecting cloud IoT functions and revenue streams. Any settings available within an IoT device, even if they are hidden, might be used to move devices to competing cloud services. A removable memorydevice can be used like a device access credential that prevents changes to sensitive settings unless the propercredential is physically present.This can be used to enforce a support policy, **which allows only a qualified person,who is on-site and been assigned an authorised access key, to make sensitive settings changes.**

It is important to remember that any IoT device can end up sitting on the workbench of a hacker, where it can be disassembled. While adding a access control key cannot 100% protect your device from hardware reverse-engineering, it can make the process much harder. One of the easiest way to hack IoT devices is to open the enclosure and locate the JTAG, SWD, or serial console port. Most such ports offer privileged ("root" or "super-user") low-level access to the running device. To prevent this type of hack, the firmware can be designed to **disable these low-level ports unless a properly credentialed access control key is present.**

Data Logging
IoT devices remotely monitor and analyse device data, triggering action where appropriate. In most IoT applications, it makes business-sense to only send small amounts of data to the cloud. However, in certain circumstances, it may be advantageous to log much more data locally. For example, an IoT device is monitoring the health of a piece of industrial equipment in a remote location, like a mine or oil field. Small amounts of health data are sent up to the cloud periodically, but during an abnormal event additional data could be logged to the removable memory device, almost like a "black box." The data sent to cloud indicates that the device needs servicing and a technician should be sent into the field to service the equipment. On site, the technician can remove the memory device that contains the **detailed event data** for full analysis. The removable memory allowed for the logging of mass amounts of data, without the large cellular data charges that would have occurred if that data was sent to the cloud.

What if the IoT device is having functional issues, even a potential firmware issue? A portable memory device could be used to **log vast amounts of low-level diagnostic/debug information – all without affecting cloud performance or mobile data costs.** The memory device could be sent in to the manufacturer for troubleshooting.

Another use for a removable memory device would be as a **backup for data** that would normally be sent to the cloud over the modem. In the event that the mobile connection was temporarily or permanently lost, the data could be logged to the removable memory device, such that no data is ever lost. If the mobile signal is restored, the data could be retrieved from the memory device. If the cellular signal was not restored, the information could be retrieved manually from the memory device.

Enable Custom Functions
If an IoT device has been deployed but is not connecting with the local mobile mast, it could be that the device was configured with the wrong mobile parameters. Instead of returning the IoT device, which may be undesirable if it has already been installed, a **memory device with the correct parameters** can be sent to the field.

In the manufacturing process, the removable memory port can be used to configure the IoT device. For example, a memory device could be used to **convert a standard model into a custom OEM model**, with that company's specific feature set.

Also during the manufacturing process, a memory device could be inserted that would put the IoT device into a diagnostic mode. It could supply the device with temporary credentials, thereby enabling self-testing with local cloud resources.

CONCLUSION

While over-the-air (OTA) may be the primary method for updating the firmware of cellular-connected IoT devices, there will always be a small percentage of devices that don't update for one reason or another. IoT designs should account for this reality and have a "Plan B" for delivering updated firmware. Integrating removable memory into the design of the IoT device and adding support for updating the firmware via this memory is a low-cost way to update the firmware of the devices that failed the OTA update. Additionally, once removable memory is part of the IoT design, it can be used for a host of other useful features.

ABOUT THE AUTHOR

For 33 years, Lynn Linse has worked with data collection from field devices. He started in computer-aided-manufacturing, then spent a decade doing multi-vendor integration in oil and gas installations around Southeast Asia. He has worked for networking device-makers Lantronix, Digi International and Cradlepoint. For over 13 years, he has been using mobile data networks for remote device monitoring and control. Over those years, he has experienced the challenges of attempting to update the firmware of thousands of remote devices via an Over-The-Air (OTA) process.