Innovative Alternatives to AT Command Sets

Leveraging the simplicity of TiWiConnect LIFT device-to-cloud software for embedded Wi-Fi® product design

LSR
Design. Create. Certify. *Connect.*

IoT White Paper

“Don’t I need an AT Command Set to control my Wi-Fi module?”

With the current white-hot trends of the Internet of Things (IoT) and Machine-to-Machine (M2M) communications, there is a strong push on product developers to convert their current products into “connected products” by integrating Wi-Fi or another wireless technology. Typically, engineering teams look to shorten development time by minimizing the number of changes to the existing design. While the latest generation of Wi-Fi modules and chipsets now provide the ability to run custom applications directly, there are often advantages in keeping the product’s existing microcontroller (MCU) or microprocessor (MPU) in the design. There typically is a significant investment in the design, development and testing of the product’s embedded software, and retaining the original MCU/MPU can allow it to act as a host that controls the Wi-Fi radio. Using this type of host with a pre-certified Wi-Fi module is often a significantly faster and easier means of adding Wi-Fi connectivity to an existing product.

While the technology and capabilities of Wi-Fi modules have progressed significantly in recent years, there remains a common perception that an “AT-like” ASCII command set is the best way for a host to control a Wi-Fi module over a serial interface. However, because there is frequently one command for each parameter that can be set in the Wi-Fi module, an AT-like command set can have tens or even hundreds of commands. Furthermore, these commands may have interdependencies on the order or values, which adds complexity. Fortunately, there are new, innovative software approaches to the Wi-Fi host interface, greatly reducing complexity and shrinking the learning curve for the developer.

“there are new, innovative software approaches to the Wi-Fi host interface, greatly reducing complexity and shrinking the learning curve for the developer”

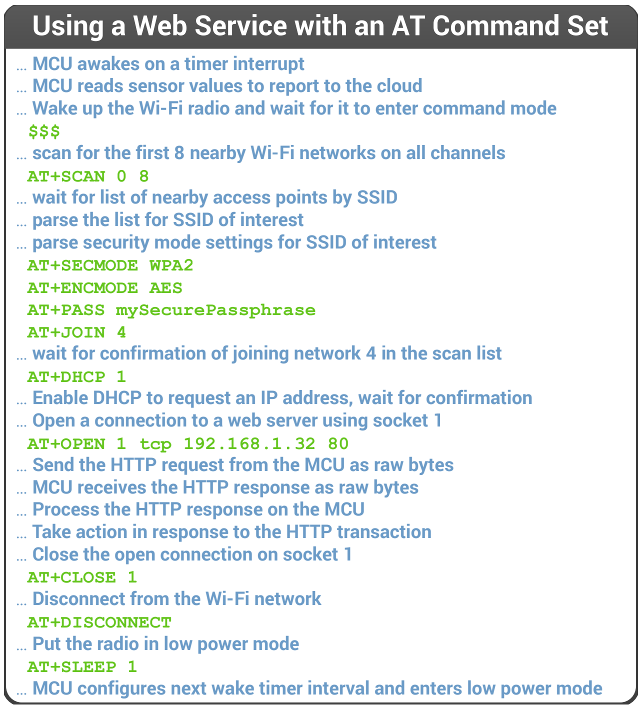
The goal of this white paper is to walk through the benefits and inherent limitations of traditional ASCII-based command sets for integrating Wi-Fi with an existing microcontroller, and detailing how new software approaches can provide significant advantages in development time and efforts for those engineering teams that lack deep experience in Wi-Fi development.

Historical Prevalence on AT Command Sets

AT Command Sets have long been utilized as a means to transmit commands between embedded systems and communication devices using minimally-sized data packets. AT Command Sets, also referred to as Hayes Command Sets after their creator Dennis Hayes, originated back in 1981 as a command language for a 300 Baud computer modem. At the time, this was an elegant solution that overcame the strict memory constraints and availability of a single serial port for commands and data. Short ASCII strings were used to represent detailed high-level commands such as dialing or hanging up. It became commonly referred to as an “AT Command Set” because implementations typically had each command begin with the first 2 characters “AT”, short-hand for “Attention.”

For today’s embedded software developers, the term AT Command Set is still frequently used when referring to any ASCII-based command set that facilitates serial communications between an MCU and another serial device such as a Wi-Fi module. The primary benefit of these AT-Like command sets is that they provide a human-readable interface and enable the developer to experiment with a new module using a terminal and serial port.

This benefit doesn’t come without trade-offs, however. Most notably, it requires the embedded developer to not only learn lower-level details concerning Wi-Fi, but also a proprietary command set that may have tens or even hundreds of terse commands. Depending on the application, a developer may have to search through a large library of commands to determine how to correctly perform common Wi-Fi tasks such as scanning for nearby networks, provisioning, joining networks, configuring the network interface and low-level socket connection management, amongst others. This does not even include the application layer operations such as making HTTP requests, setting up network time synchronization, or communicating with a cloud server, which often requires large combinations of such commands. Figure 1 below provides an example of what an ASCII-based command set approach may look like for a common Wi-Fi function in an IoT application.



Using a Web Service with an AT Command Set

```
… MCU awakes on a timer interrupt
… MCU reads sensor values to report to the cloud
… Wake up the Wi-Fi radio and wait for it to enter command mode
$$$
… scan for the first 8 nearby Wi-Fi networks on all channels
AT+SCAN 0 8
… wait for list of nearby access points by SSID
… parse the list for SSID of interest
… parse security mode settings for SSID of interest
AT+SECMODE WPA2
AT+ENCMODE AES
AT+PASS mySecurePassphrase
AT+JOIN 4
… wait for confirmation of joining network 4 in the scan list
AT+DHCP 1
… Enable DHCP to request an IP address, wait for confirmation
… Open a connection to a web server using socket 1
AT+OPEN 1 tcp 192.168.1.32 80
… Send the HTTP request from the MCU as raw bytes
… MCU receives the HTTP response as raw bytes
… Process the HTTP response on the MCU
… Take action in response to the HTTP transaction
… Close the open connection on socket 1
AT+CLOSE 1
… Disconnect from the Wi-Fi network
AT+DISCONNECT
… Put the radio in low power mode
AT+SLEEP 1
… MCU configures next wake timer interval and enters low power mode
```

Figure 1: Example of an ASCII command set approach for a common device-to-cloud Wi-Fi transaction

The advantages of a human-readable approach to embedded code

One of the pros of ASCII-based command sets (i.e. large sets of short commands) can also be seen as a con. In order to utilize the communications interface, the commands are abstract representations for very specific functions provided by the module. This relies on a sort of "dictionary" to define all the various commands essential for operating the module. This places the burden on the developer working on the attached host MCU to send the proper sequence of commands to the module in order to accomplish higher layer application tasks. For developers not already familiar with the detailed sequences required for a particular technology, such as Wi-Fi, this means hours of additional research to understand how to locate a network and initiate a connection before even considering the overall goal of sending application data to a cloud server for example.

With today's advancements in wireless module feature sets and the greater amount of memory available to embedded processors, there are strong advantages to employing an approach that utilizes a human-readable, object-oriented data format. With a modest increase in on-chip memory, a development team can utilize formats, such as JSON and XML, that are directly compatible across embedded, server, and mobile applications. This can minimize system integration time by reducing the learning curve across developers of different disciplines working on different parts of the system.

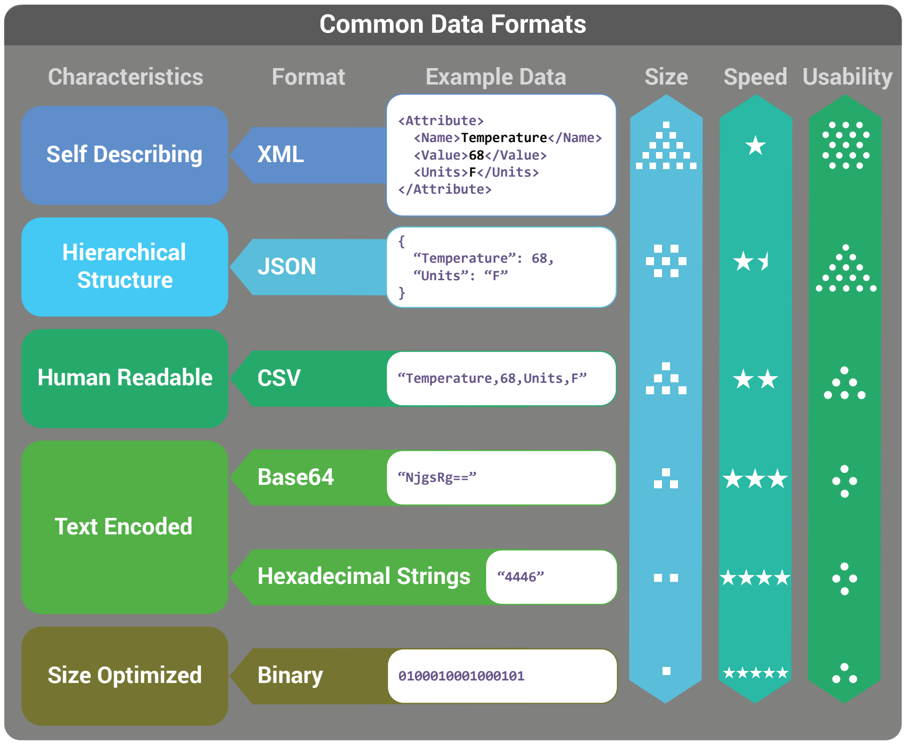


Figure 2: Spectrum of data formats and trade-offs in terms of size, speed, and usability.

Figure 2 above compares several common data formats from a number of perspectives. Looking across this spectrum, there is a clear trade-off between human readability and data size. Usability takes into account how quickly a developer who's new to a project can get up to speed. Choosing formats that balance both performance and maintainability can save time during development, provide flexibility for future feature enhancements, and ease the burden when troubleshooting during system integration.

JSON-RPC (Remote Procedure Call) is an example of a human-readable solution and is employed in the TiWiConnect platform. JSON-RPC is inherently an object-oriented approach to data communication. Rather than having one command per parameter, groups of related parameters are supplied to function calls. The result is a much simpler, cleaner and more intuitive interface for the developer. In addition to the benefits of human readability, JSON-RPC integrates seamlessly into existing server infrastructures and web services platforms. This provides crucial benefits for "connected products" with remote data-sharing requirements.

Extending a serial host interface all the way to the cloud-server

So why did the architects of the TiWiConnect IoT platform choose not to follow the traditional path of creating an ASCII-based command set? It started with the recognition that the solution that it supports, TiWiConnect, is more expansive than just a Wi-Fi module solution. TiWiConnect was created to provide product developers with an end-to-end solution for building cloud-connected IoT products for their customers. That means there is an opportunity to arm the developer with something that does far more than facilitate host-to-module communication, TiWiConnect LIFT enables *host-to-cloud* communication for their product.

"…far more than facilitating host-to-module communication, TiWiConnect LIFT enables *host-to-cloud* communication for their product."

To frame it another way, a Wi-Fi module manufacturer would typically provide a command set to allow the developer to be able to call very basic Wi-Fi and network level commands (e.g. scan for a network, join a network, open a socket, etc.). TiWiConnect LIFT recognizes that the problem to be solved is not just to get the Wi-Fi module on the local network, but get the data to the cloud.

With TiWiConnect LIFT, there is an additional host MCU API library that implements the Wi-Fi control interface as simple C function calls. The interface to the module is human-readable JSON, so developers can easily see the result of making a function call. With this solution, the developer has the tools to define the specific data structures needed by their application, and is provided the software to handle all the serial transactions to perform data sharing with the cloud server. With TiWiConnect LIFT, a product developer doesn’t need to be well-versed in AT commands, JSON, or even how HTTP works. And to the host MCU, sharing data with the cloud looks just like any other API library call.

“With TiWiConnect LIFT … sharing data with the cloud looks just like any other API library call.”

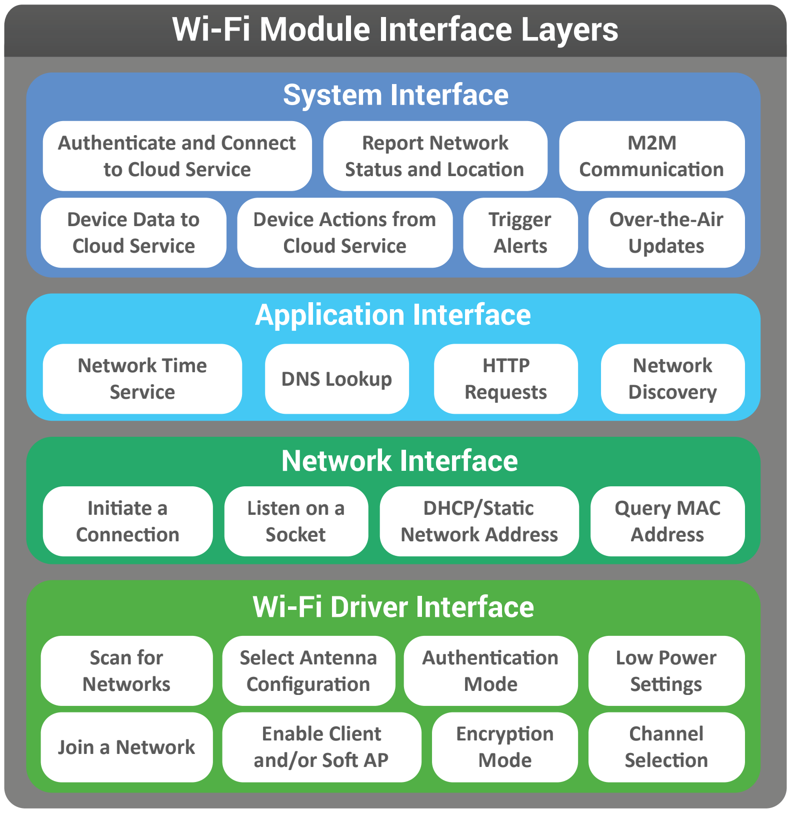


Figure 3: Interface Layers in Embedded Software found in Wi-Fi Modules

Figure 3 above illustrates the varying breadth of software layers that may be available based on the Wi-Fi module selected for a design. Though most modules provide Wi-Fi Driver and Network Interface functionality, some also provide Application Interface layer functionality. The highest-value modules go one step further and include System Interface functionality to integrate directly with other systems such as cloud services and mobile apps.

Ultimately, the problem that needs to be solved for a cloud-connected IoT application is not just getting onto a network via Wi-Fi, but to communicate with something on the “other end” of that network. For this reason, TiWiConnect LIFT bridges the space from the host MCU all the way to the cloud server through the Wi-Fi module, utilizing a data format (JSON-RPC) familiar to data-base and server applications.

The TiWiConnect LIFT device-to-cloud solution

LSR's IoT Platform, TiWiConnect, is powered by the TiWiConnect LIFT software. Its components focus on complete device-to-cloud integration, providing software components across several platforms to link devices to their users through cloud connectivity. TiWiConnect LIFT is comprised of 3 main software elements necessary to provide the 'data tunnel' between the on-board MCU and the TiWiConnect cloud server: **LIFT Client**, **LIFT Agent**, and **LIFT Server**.

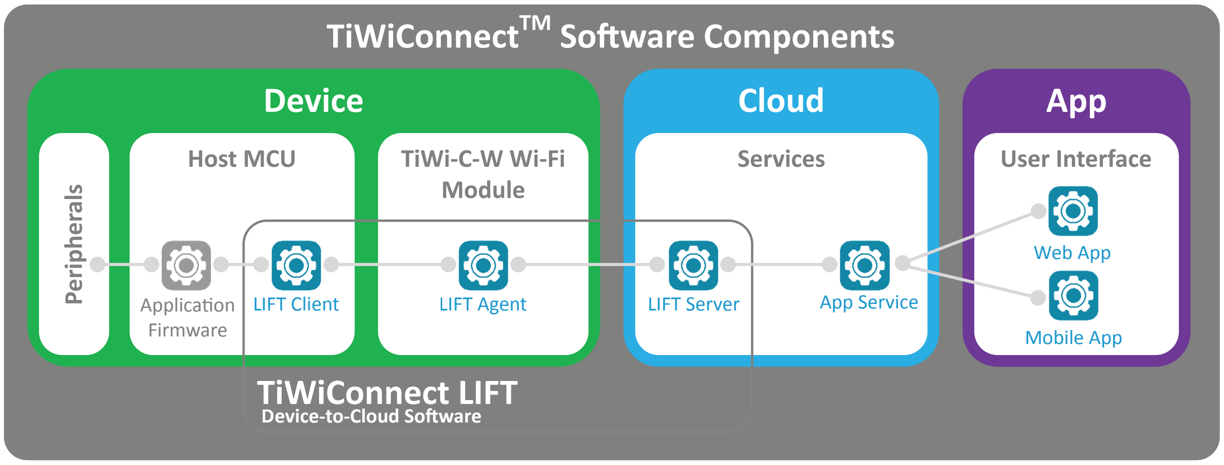
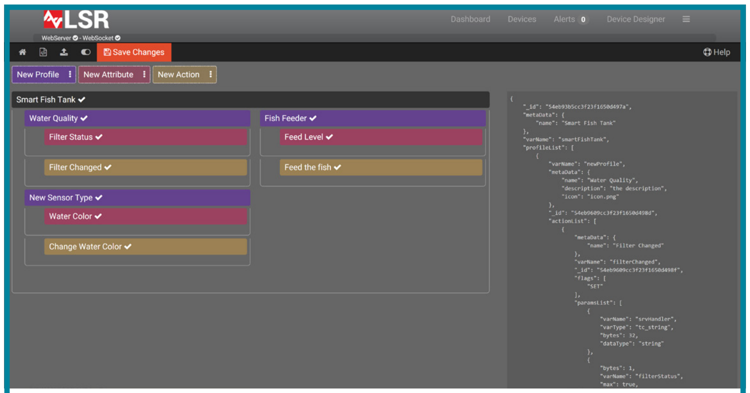


Figure 4: The key software components of TiWiConnect LIFT and how they interact across devices, cloud and apps

LIFT Client refers to the LSR-provided C source code that runs on the host MCU. This code is auto-generated by the **LIFT Device Designer** web-based tool. It is designed to run on any MCU with a serial port and C compiler, allowing developers to continue working in their preferred environment (IAR, etc.). Furthermore, the API documentation is customized specifically to their design's data needs, and embedded directly within the commenting of the C source code.



```
{
    "_id": "54eb93b5cc3f23f1650d497a",
    "metaData": {
        "name": "Smart Fish Tank"
    },
    "varName": "smartFishTank",
    "profileList": [
        {
            "varName": "newProfile",
            "metaData": {
                "name": "Water Quality",
                "description": "the description",
                "icon": "icon.png"
            },
            "_id": "54eb9609cc3f23f1650d498d",
            "actionList": [
                {
                    "metaData": {
                        "name": "Filter Changed"
                    },
                    "varName": "filterChanged",
                    "_id": "54eb9609cc3f23f1650d498f",
                    "flags": [
                        "SET"
                    ],
                    "paramsList": [
                        {
                            "varName": "srvHandler",
                            "varType": "tc_string",
                            "bytes": 32,
                            "dataType": "string"
                        },
                        {
                            "bytes": 1,
                            "varName": "filterStatus",
                            "max": true,
```

LIFT Device Designer tool for TiWiConnect™ IoT Platform (devkit.tiwiconnect.com)

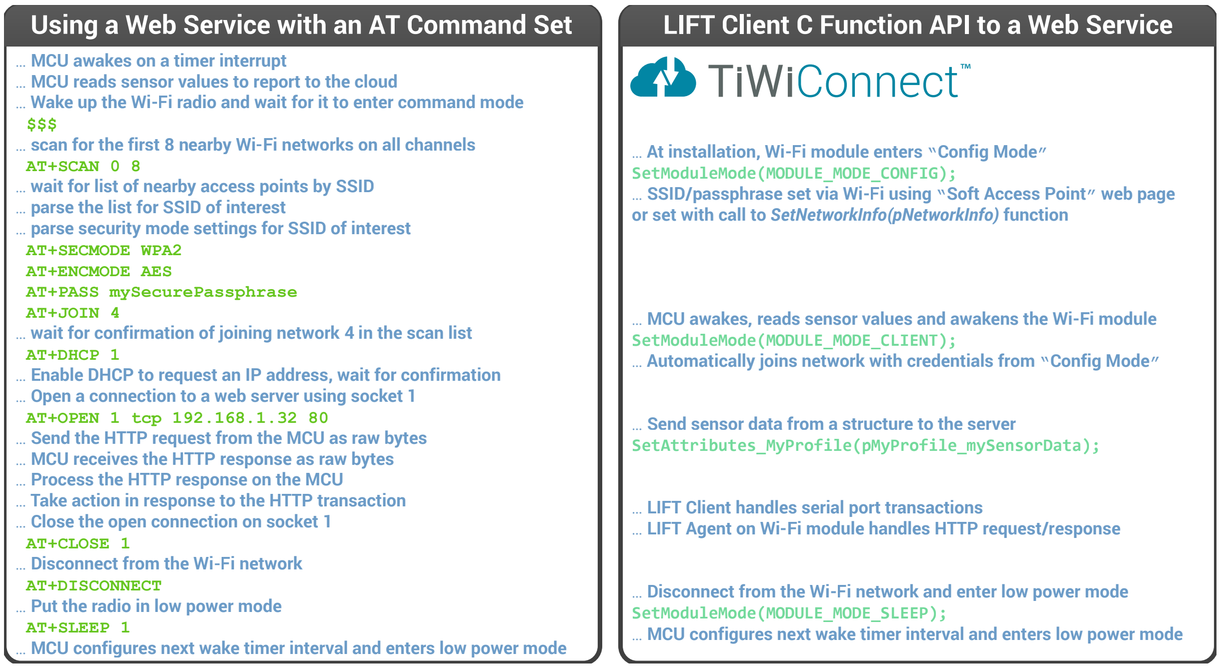
Making simple even simpler: LIFT Device Designer tool

While a JSON-RPC based software approach (as used in the TiWiConnect LIFT software) brings inherent simplicity with its human-readable nature, the task of defining your specific serial data can be further simplified for those without experience handling coding in JSON. The LIFT Device Designer is a web-based, drag-and-drop GUI tool that allows the user to define the data attributes and remote actions that their connected product will need, and auto-generate the necessary embedded code on the fly. The LIFT Device Designer is available to users developing with LSR's **TiWi-C-W Dev Kit featuring TiWiConnect**.

LIFT Agent refers to the LSR-designed embedded software running on the TiWi-C-W™ module alongside a soft AP-based network provisioning feature for the module. The LIFT Agent acts as the bridge between the host MCU and the cloud-server, abstracting lower-level Wi-Fi commands, saving the developer time and effort.

LIFT Server refers to LSR's server-side interface running on the TiWiConnect cloud, communicating with the device through a simple JSON-RPC 'data tunnel.' In cases where a developer wishes to integrate TiWi-C-W communication with their own servers, LSR provides example source code for the LIFT Server.

Going back to the complexity of an AT Command Set approach to Wi-Fi (as detailed in Figure 1 earlier), Figure 5 below contrasts that experience with how the LIFT Client embedded firmware can dramatically simplify a common IoT web-service task by providing direct function calls on the host MCU.



Using a Web Service with an AT Command Set

```
… MCU awakes on a timer interrupt
… MCU reads sensor values to report to the cloud
… Wake up the Wi-Fi radio and wait for it to enter command mode
$$$
… scan for the first 8 nearby Wi-Fi networks on all channels
AT+SCAN 0 8
… wait for list of nearby access points by SSID
… parse the list for SSID of interest
… parse security mode settings for SSID of interest
AT+SECMODE WPA2
AT+ENCMODE AES
AT+PASS mySecurePassphrase
AT+JOIN 4
… wait for confirmation of joining network 4 in the scan list
AT+DHCP 1
… Enable DHCP to request an IP address, wait for confirmation
… Open a connection to a web server using socket 1
AT+OPEN 1 tcp 192.168.1.32 80
… Send the HTTP request from the MCU as raw bytes
… MCU receives the HTTP response as raw bytes
… Process the HTTP response on the MCU
… Take action in response to the HTTP transaction
… Close the open connection on socket 1
AT+CLOSE 1
… Disconnect from the Wi-Fi network
AT+DISCONNECT
… Put the radio in low power mode
AT+SLEEP 1
… MCU configures next wake timer interval and enters low power mode
```

LIFT Client C Function API to a Web Service

TiWiConnect™

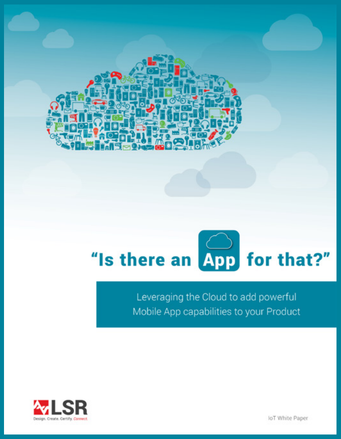
```
… At installation, Wi-Fi module enters "Config Mode"
SetModuleMode(MODULE_MODE_CONFIG);
… SSID/passphrase set via Wi-Fi using "Soft Access Point" web page
or set with call to SetNetworkInfo(pNetworkInfo) function

… MCU awakes, reads sensor values and awakens the Wi-Fi module
SetModuleMode(MODULE_MODE_CLIENT);
… Automatically joins network with credentials from "Config Mode"

… Send sensor data from a structure to the server
SetAttributes_MyProfile(pMyProfile_mySensorData);

… LIFT Client handles serial port transactions
… LIFT Agent on Wi-Fi module handles HTTP request/response

… Disconnect from the Wi-Fi network and enter low power mode
SetModuleMode(MODULE_MODE_SLEEP);
… MCU configures next wake timer interval and enters low power mode
```

Figure 5: Comparison of an ASCII Command Set approach and TiWiConnect's LIFT Client approach for a common device-to-cloud Wi-Fi transaction

Bringing it All Together

The continued expansion of IoT products will be driven in part by the ability for product developers to integrate wireless cloud-connectivity into their product with minimal development time and risk. One key aspect for achieving this is the ease of establishing serial host communications between the wireless module and the product's existing microcontroller.

While 'AT-like' ASCII command sets are historically familiar to many embedded developers, there are superior software approaches developed specifically for cloud-connectivity systems that dramatically reduce complexity and development time. Today's IoT platforms, such as TiWiConnect™, provide a more elegant, efficient, and powerful approach to establish connectivity for data not simply to pass between host controller and Wi-Fi radio, but to have a 'data tunnel' all the way from host to the cloud-server. The TiWiConnect LIFT device-to-cloud software offers a higher level of functionality designed specifically for developing cloud-connected products, and tools like the **LIFT Device Designer** virtually eliminates the learning curve for adding wireless connectivity to your product design.

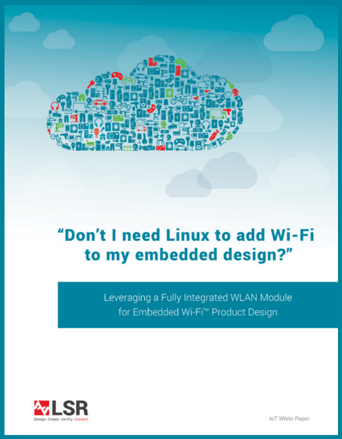


"Is there an App for that?"

Leveraging the Cloud to add powerful Mobile App capabilities to your Product

LSR

IoT White Paper



"Don't I need Linux to add Wi-Fi to my embedded design?"

Leveraging a Fully Integrated WLAN Module for Embedded Wi-Fi™ Product Design

LSR

IoT White Paper

TiWiConnect™

Module | Cloud | App

Elevate your products with TiWiConnect™

LSR introduces the TiWiConnect cloud-connectivity platform, the first true end-to-end IoT solution for wirelessly connecting products to the cloud. This IoT platform enables smartphone apps and web portals that can re-define the product experience for both your customers and service professionals alike.

TiWiConnect simplifies your product development efforts by providing all the components of a comprehensive solution, all built from the ground up to connect seamlessly: Embedded wireless modules, cloud platform and mobile apps.

Learn more at **www.tiwiconnect.com**.

LSR

Inspiring through Wireless Innovations.

Bringing a winning product to market in today's competitive environment requires greater skill, creativity and experience than ever before. More and more, your customers demand intuitive, reliable wireless capabilities that give them the real-time information and controls to be more connected.

Since 1980, our partners, spanning a wide range of industries, have trusted LSR to help develop solutions that exceed their customers' expectations. We provide an unmatched suite of both integrated services and wireless products that improve speed to market and return on your development investment.

Our experienced professionals are passionate and committed to partnering with you, allowing your team to focus on the most important element of product development: the unique needs of your customers.

Visit us at **www.lsr.com** and follow us on **LinkedIn** and **Twitter (@LSResearch)**.