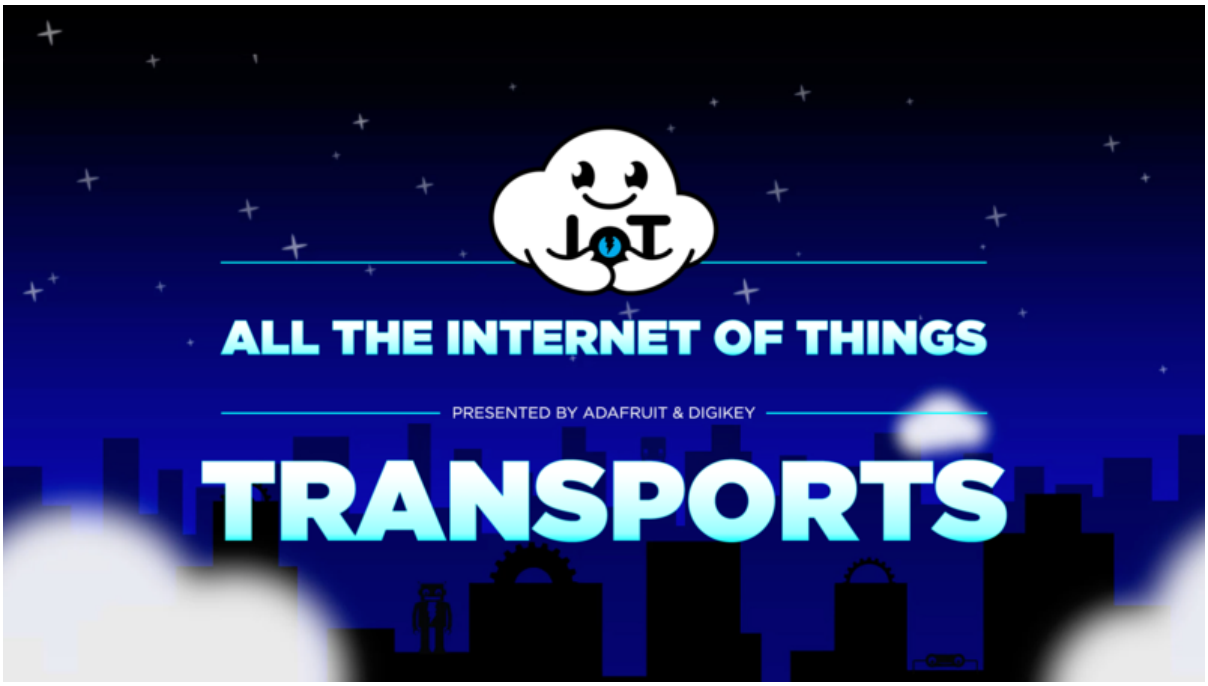
# All the Internet of Things - Episode One: Transports

Created by lady ada




https://learn.adafruit.com/alltheiot-transports

Last updated on 2024-06-03 02:13:23 PM EDT

# Table of Contents

# Introduction

# Hello!

I'd like to welcome you to the first episode of our new series of videos and guides, designed to help you learn about and make your very own connected objects

This is Adafruit and Digi-Key's **ALL THE INTERNET OF THINGS** - a six-part series, covering everything you need to know about the Internet of Things (which we will shorten to **IoT**).

For our first guide, we'll go over the most popular transports used in the IoT industry, as well as the upsides and downsides of each type of transport to help you decide what you'll use to connect your devices to the internet.

# Connecting to the IoT

The Internet of Things is all about **connections** - connecting your electronics design, product, or project to the wider world. We start with the idea that you have a "Thing" that you want to connect to the "Internet of."

How do you do that?

As a maker, engineer, or designer there are a lot of choices to make. And those choices have a big impact on the cost, size, runtime, and usability of your Thing. Thinking and knowing about your options early on, perhaps even before you open up your IDE or CAD tool, will help you save money, time, and make your product the best it can be.

# Transports

Whether you plan on using Ethernet, WiFi, Bluetooth, Cellular. RFID/NFC, satellites, sub-GHz, LoRa or mesh networks, there are advantages and disadvantages to each. The choices are abundant and whichever you choose will have very interesting features, constraints, and considerations.

It's all about trade-offs - do you need lots of power and range? Are you pushing lots of bits? Or will be it small packets, low-power, and short-hop distances?

That's why this guide is all about  transports. Our methods of categorization will be **POWER, DISTANCE, AND BITS.**

Tune in, turn on, and let's connect these "things." :)

---

# Transports

You're very lucky! You have tons of options on how you can connect your Thing to a wider network. As we've mentioned, there are trade-offs and design considerations with each method.

Since the transport you choose has massive implications on your design, you'll want to nail this down first. It's also very hard, if not impossible to change transports after you've started. Pick the right transport and the rest of your design will fall into place.

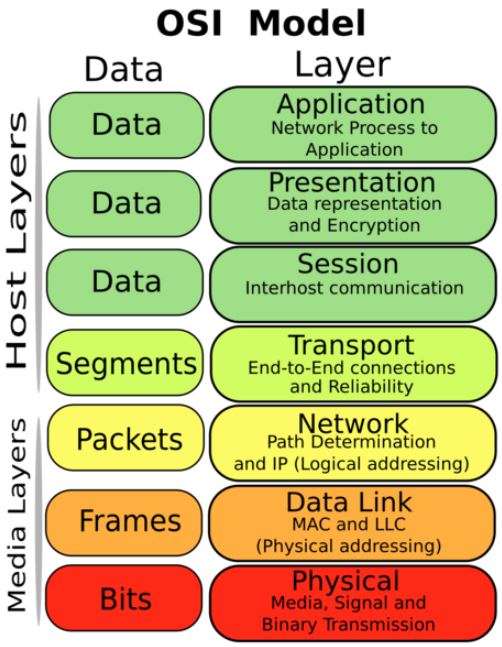
We'll start with the oldest/stable transports first and then branch into the more esoteric ones. This isn't to say you have to stick with the first one we mention - it's just you're most likely to be familiar with it!

# What is a Transport?

What do we mean by transports?

The most common way we 'split up' a datapath is to use the 7-layer OSI model (not to be confused with the delicious 7-layer burrito (https://adafru.it/Bss)).

Below is a generic diagram of the OSI layers:




Now ideally we could pick and choose each layer of our connectivity - so you can swap out any layer to get the perfect balance of power/range/throughput.

But that's not how the real world works. Often times the Transport layer (which, above, is the 4th layer) bleeds through both below and above. While in theory you could run Ethernet over a wireless LoRa link, it would be...strange. We're going to assume that you're using the transport/network/data/physical set together as intended.

Sometimes, these are deeply practical considerations: if you have Ethernet, we assume you've got standard Cat-6 wiring, Ethernet jacks, MAC addressing, and using TCP/IP over it - all those layers are closely tied together and optimized.

If you have a cellular transport, you've got GSM/GPRS/LTE and are using the world-wide cellular network. Doing anything else would require low-level hacking and probably violate the terms of use for the cellular network.

And note that some of the transports don't guarantee end-to-end connectivity, or it may be your responsibility - especially radio.

So, think of transports as a rough sketch of the first 4 layers!

# Ethernet

Good ol' Ethernet. This ‘ancient’ protocol has withstood the test of time. You almost certainly use Ethernet at home and at work. Ethernet has a standardized connector, the venerable RJ-45. When you need something to "just work," often a wire will do that. (Just because it's "IoT" doesn't mean it's wireless, just that many Things happen to be wireless.)

**Here are some of the good things about Ethernet:**

- World wide universality, completely open and free standard - no patents or licensing!
- Just about every hotel/home/office has an Ethernet port for connecting to the internet.
- High speed, 1 GBPS Ethernet is available on many routers and on some single board computers.  But even the ‘slowest’ 10 Base T has better high-speed throughput than legacy WiFi (802.11b/g)
- No interference from other Wireless protocols, no drop-outs. A reasonably designed NIC won’t suffer from flakiness
- Can go up to 100 meters on a single cable
- No passwords/usernames/pairing. Plug in and go!
- Fairly inexpensive to implement - many chips have built in Ethernet MAC, so you only need a PHY and plug.
- Can be used as a private intranet, or connect directly to the Internet

**Some downsides**

- Requires a permanent cable connection

- High current draw - expect ~200-300mA all the time
- Large and chunky connector
- Watch out for default passwords on Things that plug in directly to the Internet!

# Ease of Use!

The biggest benefit of Ethernet is the high speed and plug-and-play. No SSID configuration or pairing.

There are some security considerations (we'll get into security later), but there are benefits of just having a wire since someone needs physical access to tap or connect to Ethernet. If you have a wire and MAC-assigned IP addresses you generally have more control over what and who can access something. Just don't use default passwords on your IoT devices and expose them to outside internet directly without being smart about security.

# Common Uses

Ethernet is generally used when you do not need a lot of range (you're stuck with the length of the cable after all) and you need to move around a lot of bits.

Common Ethernet-connected IoT devices are:

- Cameras - video, especially lately, has gone up to 720p which can strain WiFi connections
- Voice (VoIP)
- Set-top boxes - video/audio streaming and storage
- Game systems
- Industrial equipment that is permanently installed
- Devices that need ‘air gapped security’ and cannot use wireless connectivity
- High-reliability control like industrial control, robotics or medical etc.


Ethernet is low cost and highly reliable, so its perfect for voice/VoIP applications. This simple telephony box has pretty much 3 plugs - power, Ethernet and phone.

Security and video cameras also often come with Ethernet so they cannot be jammed, and so they don't have to be updated when the password changes. You can see on this one the Ethernet jack is at the end of a cable

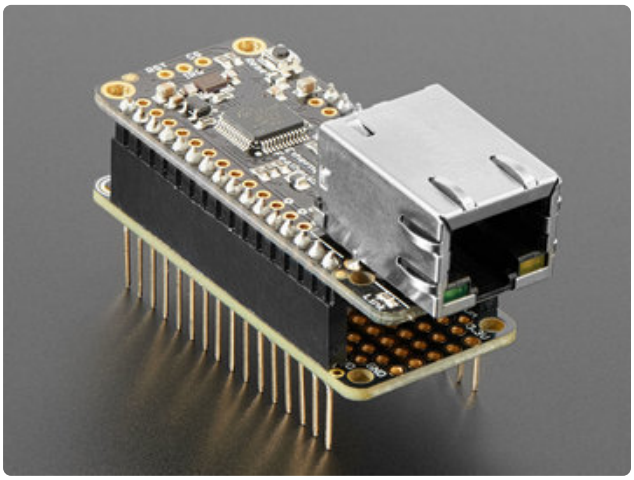

You'll get way fewer drop-outs / hiccups with Ethernet, and the throughput and 'lag' tend to be better than WiFi, so it's perfect for gaming consoles or set-top boxes where you have a lot of video/audio/data

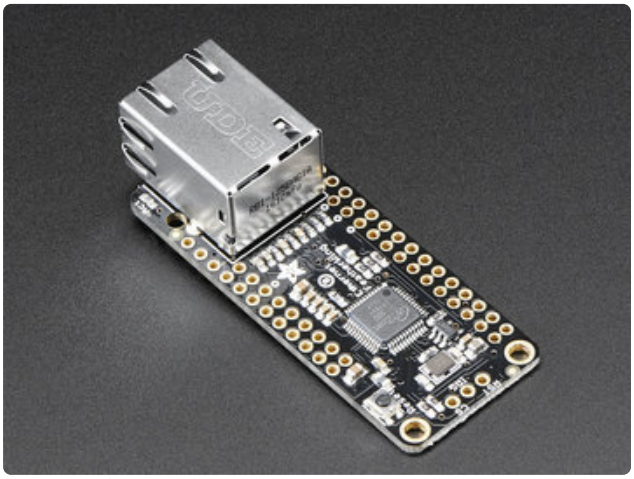
# Very High Speed

Ethernet can range from 10 Mbps to 1 Gbps (1000 Mbps). 10-base-T is what you'll find on small microcontrollers, 100-base-T (100Mbps) is found on many higher-power microcontrollers or single board computers, 1 Gbps is what you'll find on high end equipment that needs to stream a lot of data. Compare that with future transports we'll discuss, that are sometimes specified in 'bps or 'kbps!

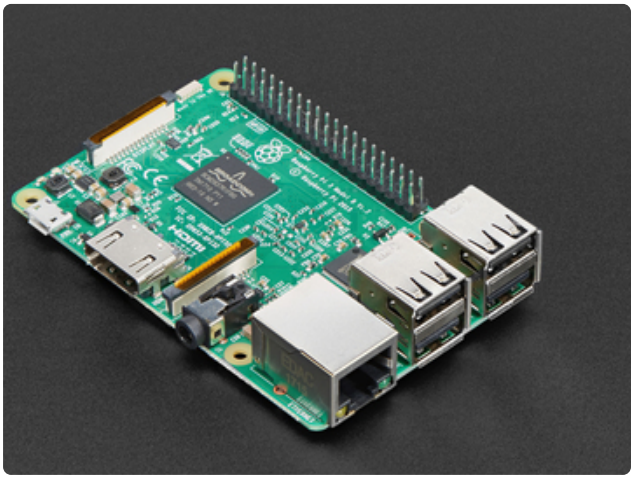
# Adding Ethernet to your IoT project

Since Ethernet is royalty-free, universally used, and reliable, it's common to find ways to add Ethernet on a device. That is to say, you will not be the first person who wants to Ethernet-connect.



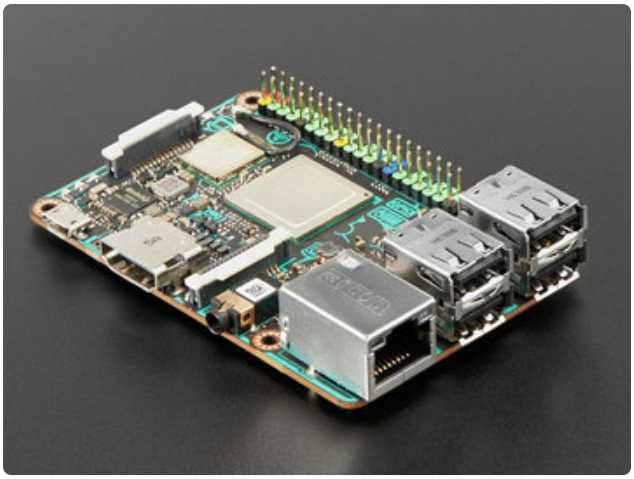


If you're using our Feather series of boards (https://adafru.it/zDk) you can add Ethernet very easily with our Ethernet Featherwing (https://adafru.it/zDl). It is 10-base-T only, and uses common SPI to send and receive data. It works well, and has nice link/activity indicator lights on the plug. There's an Arduino library for the Wiznet 5500 chip, and the Wiznet chip handles all the transport-and-below layers.



Raspberry Pi computers like the Pi 3 (https://adafru.it/zDf) have 100-base-T Ethernet built in and default work when plugged in. Literally no effort from the development side required!

100-base-T is the most common speeds you'll find above the most simple microcontrollers, and can handle video/audio streaming just fine.



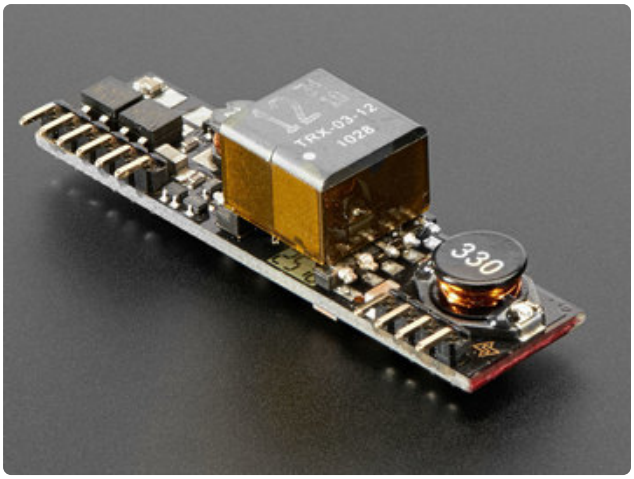
Other higher-end single-board computers like this ASUS Tinker may come with 1 GBPS Ethernet. Because gigabit Ethernet can't run over USB 2.0 at full speed, you can't use a common USB-to-Ethernet converter. If you need the speed, the board needs to have native hardware support or a Thunderbolt/USB 3 class port

# Watch out for...

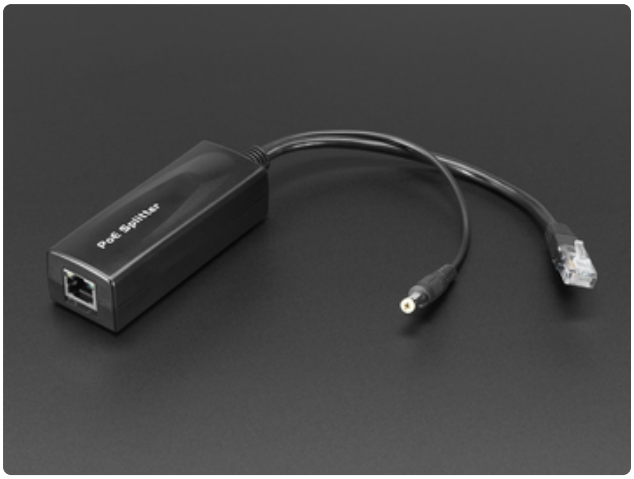
One down-side of Ethernet is that it has a fairly high current draw, and you'll need to budget 200-300mA of constant current available during listen/transmit. However, since you're running a wire anyway for an Ethernet device usually you can plug in for power and/or get Power over Ethernet PoE.

PoE is a network standard that puts 48V on the unused data wires (or, with Gigabit Ethernet, shared with data wires). (https://adafru.it/Bsu) This allows the network cables to carry data and power. IP cameras, VoIP phones, lighting, and wireless access points are where you'll see most of this action. PoE routers are becoming fairly common and they're very affordable.

PoE routers have big power supplies but can provide multiple devices with power. It's a little more investment up-front but if the end-devices support it, makes cabling waaaay easier.




You can add PoE to your design with extra chips and hardware (hopefully the device you have can do the transport 'negotiations' to turn on PoE). The technical name for PoE is **IEEE 802.3af** or **IEEE 802.3at** so look for that labeling.



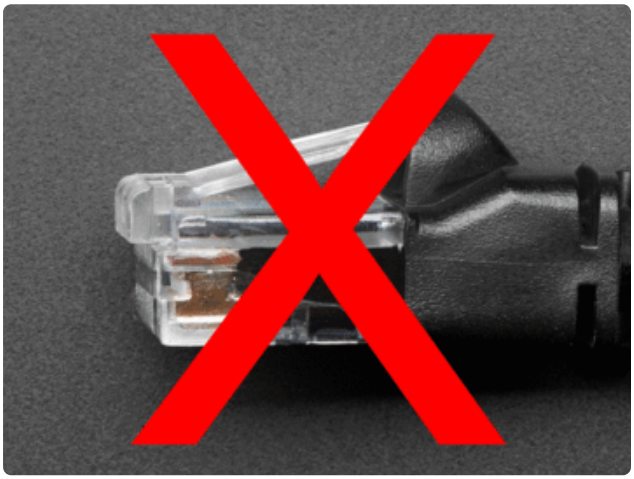

Or you can use a PoE converter which will split the Ethernet+Power into two plugs - there are both injectors and splitters.

Check out our PoE splitter at Digi-Key (https://adafru.it/zDm)


They also have a wide range of other 802.3af devices here (https://adafru.it/zDn)

# WiFi

You know WiFi - you use it all-day every-day. It’s the standard wireless protocol for connecting to the Internet. It’s available not only at home and work but also in stores, cafes, trains, planes and in most major cities. The worldwide universality of WiFi makes it a common first choice for IoT.



## Wire-less!

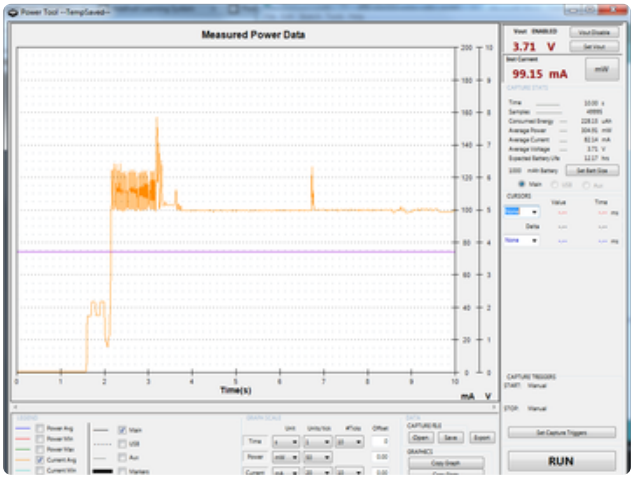
WiFi has a lot of the benefits of Ethernet but it's wireless - and you’ll often find that devices that support one support the other. WiFi has gone through many iterations (802.11b, g, n...) and the speed and throughput has increased in time. Like Ethernet, it can transport lots of data and connect to the wider Internet with ease. And like Ethernet, WiFi can require a significant power budget if not managed carefully.

# Power Management

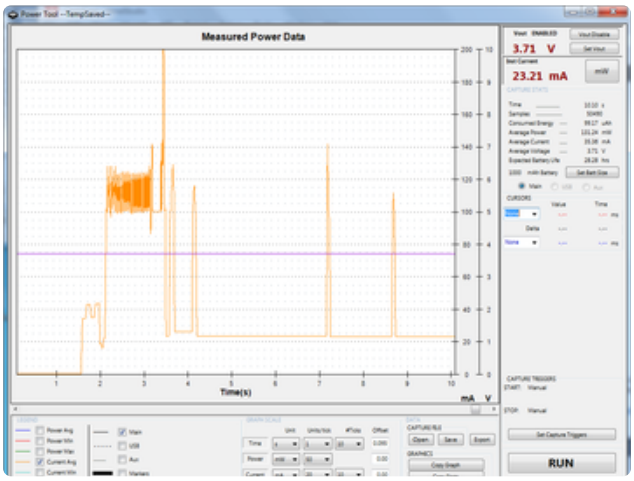
WiFi removes the need for wire for the bits portion, but you'll still need to power it somehow. So many IoT devices, from sensors to cameras, use WiFi for the data but are still plugged in for power.

If you’re indoors, you may find that wall outlets are plentiful but dragging an Ethernet cable is unnecessary.

For mobile applications, WiFi can go hours and maybe even days on battery, you'll just need to make sure you're in range of the access point and have a plan to charge or power.




For example, here's a WINC1500 WiFi module Feather power trace. When its time to scan and connect to the access point, current draw is approximagely 120mA (that's with 3.3V power regulation). Keeping the WiFi module on all the time to listen for connections draws about 100mA.




On the other hand, if you're able to put the WiFi radio to sleep, you can send up drawing power only on receive and transmit. You can see on the left the much lower idle-current and then spikes when the radio is on.

You'll get slightly slower performance since you have to bring it in and out of sleep mode. And, not all radios support sleeping! So you have to really try out your code and measure the draw to know what your run time will be.

# Range & Speed Ratings

With WiFi every-day 802.11g or n, you can get about 150 feet / 46 meters. Outside, about 300 feet / 92 meters. But as you are probably aware, the further you get from the access point, the flakier the connection will get and the more power you will need to amplify. You can save power with strategies like putting the device to sleep, waking when needed or just setting up specific times to turn on WiFi and do your thing - but that will all require a lot of engineering time.

Just as Ethernet comes in a few speed-ratings (10, 100 and 1Gbps), WiFi also comes in various ratings, there's also 2.4GHz and 5.8GHz flavors: a, b, g, n, ac. (although, honestly we have never seen 5 GHz WiFi on a microcontroller/microcomputer scale

project). With each flavor, you'll get more bits but need more power and get less range. There are also different encryption standards. **802.11b** is a little old but still available, **g** and **n** are most common, just keep in mind this is a spectrum that anyone can use so expect some interference from other devices like wireless keyboards, microwaves, other WiFi devices, etc.

| Protocol | Frequency | Data rate (Mbit/sec) | Approx Range (indoor) | Approx Range (outdoor) |
|---|---|---|---|---|
| 802.11b | 2.4 GHz | 1, 2, 5.5 or 11 | 35 meters | 140 meters |
| 802.11g | 2.4 GHz | 6, 9, 12, 18, 24, 36, 48, 54 | 38 meters | 140 meters |
| 802.11n | 2.4 GHz | up to 288.8 | 70 meters | 250 meters |
| 802.11n | 5 GHz | up to 600 MBit/s | 70 meters | 240 meters |

Table adapted from https://en.wikipedia.org/wiki/IEEE_802.11 (https://adafru.it/Bsx)

Note that just because your chipset says it can do 802.11g doesn't mean you'll get the max 54 MBit/s, just that it's the max you can get. And of course, the range varies wildly with antennas, power, chipsets, wireless traffic, and more...

# Good Things about WiFi

- Wireless!
- Worldwide universality, completely open and free standard - no licensing!
- Networks/routers anywhere and everywhere
- Some good security/encryption built-in
- Can be used as an ad-hoc network
- Fairly good range, high throughput
- Popular implementations abound, including some low-cost versions
- Direct internet access

One thing that's new is that WiFi has become pretty cheap! Wifi modules are only a few $ per unit. Sometimes you can get a microcontoller + WiFi chip all in one to save space and money.

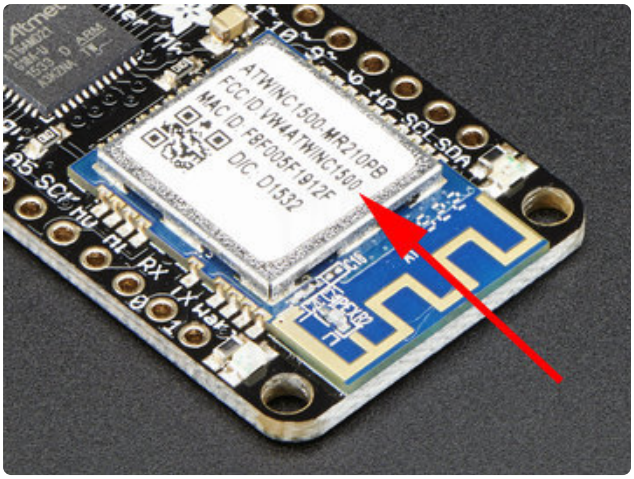
# Challenges

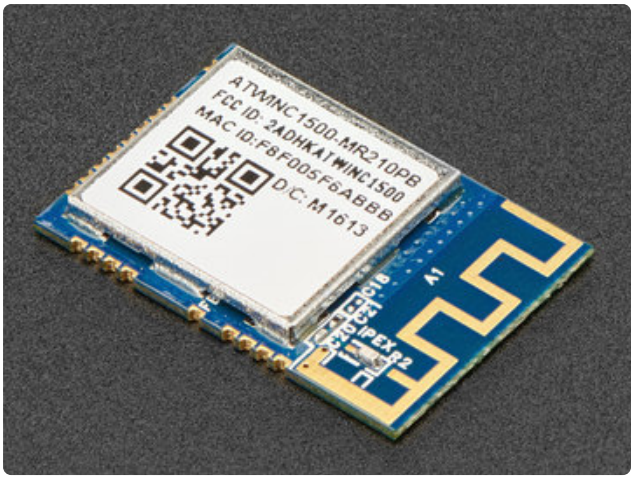
- Requires authentication, which can be frustrating to set-up, new authentication every time
- High power usage without a lot of work
- Wide range of embedded-access chipsets, have to choose which to go with
- Flakiness, drop-outs - challenging to debug
- A really good WiFi stack sometimes requires a full RTOS or kernel to ‘kick’ the network
- Watch out for default passwords on Things that plug in directly to the Internet!
- Radio-emitter certifications

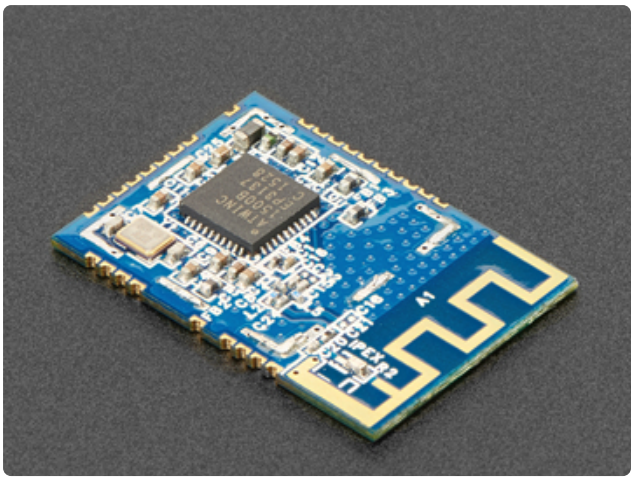
Especially if you have plug-in or easy recharging, WiFi is a clear winner: good range, good speeds, well understood. You can be certain that WiFi is a protocol that is ‘here to stay’ so it's a safe bet. For portable items, it's not always the best choice as the battery management can take up a lot of your engineering time. Check your chipset and use a power monitor to compare current draw. Use eval boards to quickly ‘sketch out’ your bandwidth needs and figure out what size battery you can get away with.

# Radio Emitter Certifications

As a radio emitter, and a powerful one at that, you may need to spend some time on your wireless certification process. E.g. in the USA you’ll need FCC, in Europe CE, in Japan you’ll need TELEC, etc.

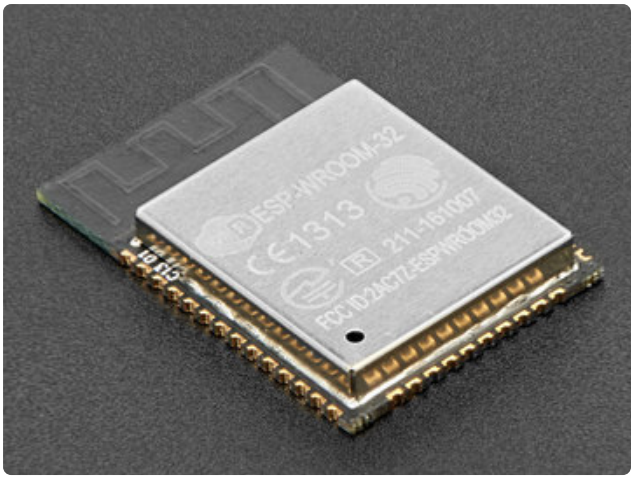


You can save time by going with a precertified module, these often have pretuned antennas that work out-of-the-box - less certification is required. But you'll pay more! That said, almost every design we've seen with WiFi has gone with a pre-certified module

For example, the WINC1500 shown here has a tin that covers a ton of RF parts for you




The ESP32 is an up-and-coming chipset that contains both WiFi and dual core processor, also available in a pre-certified module.

You can see the FCC ID engraved at the bottom.

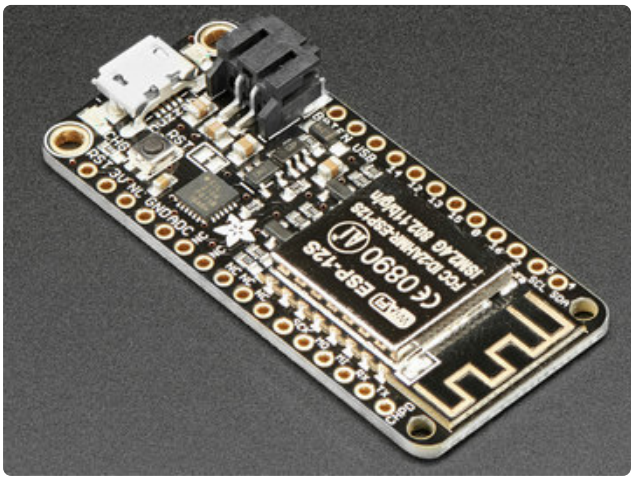
# Authentication & Security

Some of the big frustrations with WiFi are authentication - you absolutely must have security, but that same security can make it annoying to set the SSID and password. A common method these days is to set up an ad-hoc network so the device can be configured. (It's annoying but common). Another option is to go with BTLE for configuration, if you get BTLE 'for free' with your WiFi. It can make portability annoying too if there's no UI for setting the SSID and password directly on the device.

Also **some schools/offices have 'enterprise' wifi where a ToS is clicked through** - that may not be possible for you to do on your small device in which case you'd need the administrator to allow the MAC address through...as you can tell, it gets hairy fast.

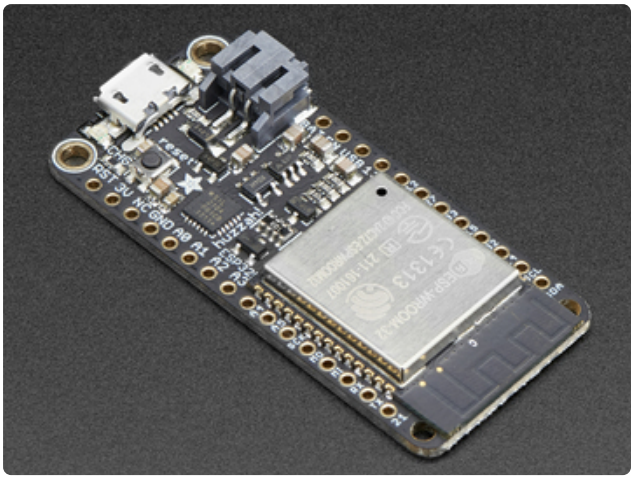
# Adding WiFi to your Thing

The prices for embedded modules and chips have dropped drastically in the last few years. What used to be $20/per module is now trending towards $5/per module. We're also starting to see modules where the WiFi component is controlled by an internal RTOS so you may be able to get away with a single-chip solution rather than a bi-chip solution - that will do wonders for your BoM costs! But watch out to make sure you aren't getting stuck with a chip core that hamstrings you. We expect to see dozens more of these fully integrated WiFi chips in the next few years.



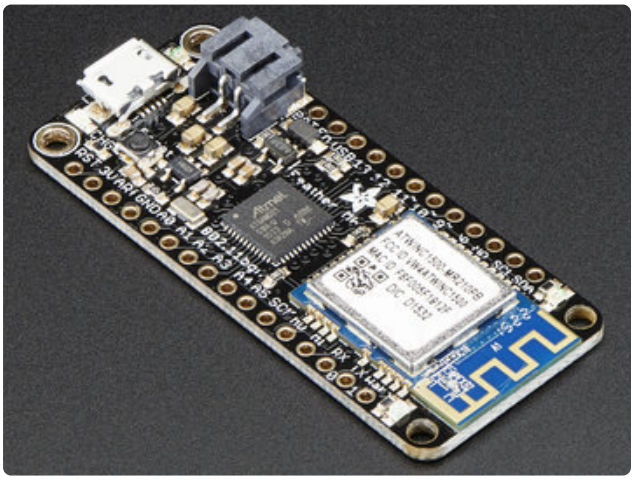
At the Feather HUZZAH's heart is an ESP8266 WiFi microcontroller clocked at 80 MHz and at 3.3V logic. This microcontroller contains a Tensilica chip core as well as a full WiFi stack. You can program the microcontroller using the Arduino IDE for an easy-to-run Internet of Things core. The ESP8266 is very very low cost, but has an RTOS running that you can't control, which can sometimes make programming high-reliablity projects a little difficult. It's great for low cost Things and maker projects.

Pick up an Adafruit ESP8266 Feather at Digi-Key (https://adafru.it/zDb)



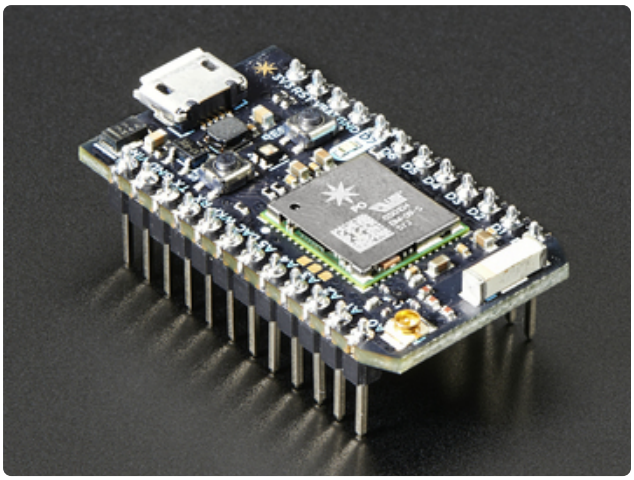
The ESP32 is a perfect upgrade from the ESP8266 that has been so popular. In comparison, the ESP32 has way more GPIO, plenty of analog inputs, two analog outputs, multiple extra peripherals (like a spare UART), two cores so you don't have to yield to the WiFi manager, much higher-speed processor, etc. etc! We think that as the ESP32 gets traction, we'll see more people move to this chip exclusively, as it is so full-featured.

Pick up an Adafruit Feather ESP32 at Digi-Key (https://adafru.it/zDc)



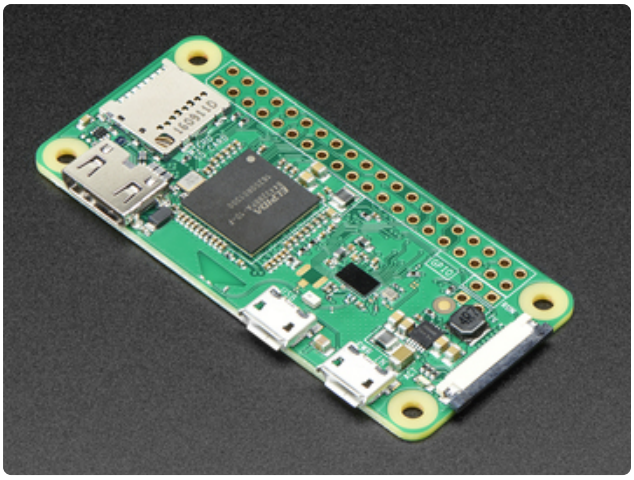
The Feather M0 + ATWINC1500 is a pairing of chips: there's a main processor (the Feather M0 part) and the wifi processor module (the ATWINC1500 part). As such, these Feathers are more expensive than all-in-one WiFi solutions. But, as a positive, they have a really powerful and well-documented main processor that runs separately from WiFi which can give you more control.

Pick up an Adafruit Feather M0 WINC1500 from Digi-Key (https://adafru.it/zDd)



The **Particle** series of WiFi boards has a built in RTOS, single-chip solution and is pre-certified. You also get a cloud-based coding and deployment system, which is aimed towards enterprise customers and speed-to-market

Pick one up at DigiKey (https://adafru.it/zDe)



Raspberry Pi computers like the Pi 3 (https://adafru.it/zDf) and Pi Zero W have built in WiFi and BTLE which makes them very easy to get started with WiFi. The kernel support seems good and stable and you get a full Linux computer so you can do very fast connects and transfers


You can also easily add WiFi to other single-board computers with a simple USB WiFi stick. These tend to do 802.11b/g/n and are low cost. Just make sure your OS has driver support for the chipset!

## Bluetooth & BTLE

Bluetooth is a newer protocol but one you're likely familiar with because it has gained a ton of popularity with gadgets and small devices. Like WiFi, Bluetooth is wireless, and like WiFi, it operates in the 2.4GHz band so pretty much anyone can use it, and does. In fact, the frequency-sharing of WiFi and BT is why just about every mobile phone, tablet, computer, and laptop that has WiFi also has Bluetooth.

# Bluetooth Classic and Bluetooth Low Energy

Watch out! There are TWO Bluetooths, and these really trip people up. There is Classic and Bluetooth LE / BTLE/BLE (Bluetooth-Low Energy) BLE is sometimes referred to as "Bluetooth Smart" but honestly we just hear people say B.L.E., BT 4.0 or just (sigh) BT. That confusion trips people up the first time they work on Bluetooth.

# Range

Range is about the same for both classic and low energy. Technically the max range is about 300 feet or 100 meters, but that's very generous and assumes a powerful radio. In reality, you will find over 10 or 20 meters can be challenging.. about 2 Mbps max for Bluetooth Classic and less than a Mbps for BLE.

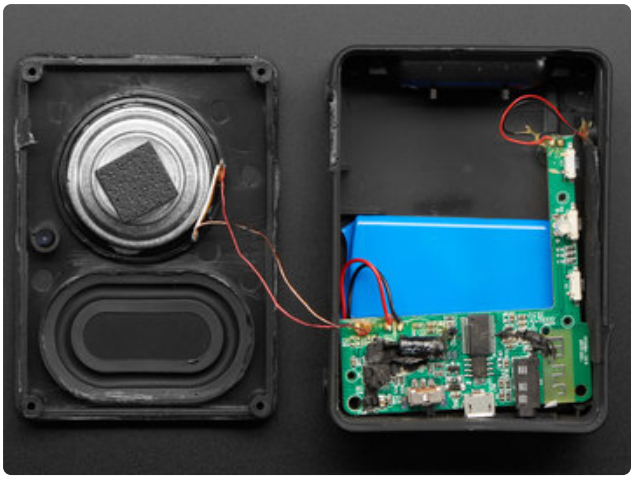
Compare that with WiFi's bandwidth and range! **But** remember! BT classic and BTLE are completely different protocols and are not compatible. **There are some radio chips that can do both but be sure to check before quoting the parts.**

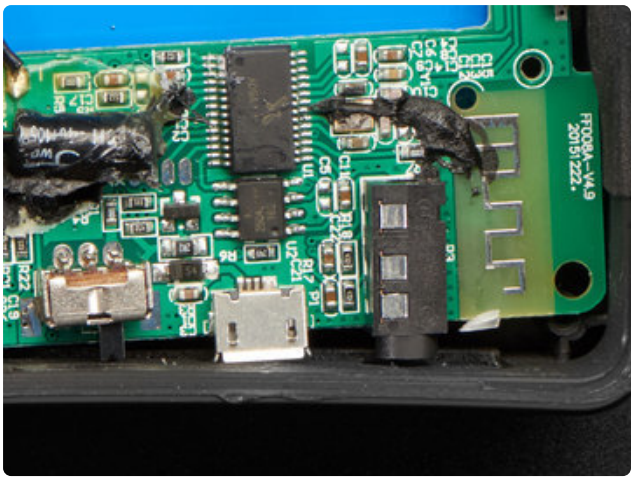
# Classic & BTLE Similarities

There are some overlaps between classic and LE. Both use 2.4 GHz and are intended for short-hop wireless communication. For example, wireless keyboards and mice, headsets, or smart watches. The idea behind BLE was keep the range of Classic, use a lot less power (e.g. the Low Energy part), and add some more features. For example, one of the newer features of BLE are beacons that broadcast information, so when you fire up your phone the beacon will tell you about itself - These are often set up with apps to do building-scale location services.

# Classic Use Cases: Audio & Keyboards

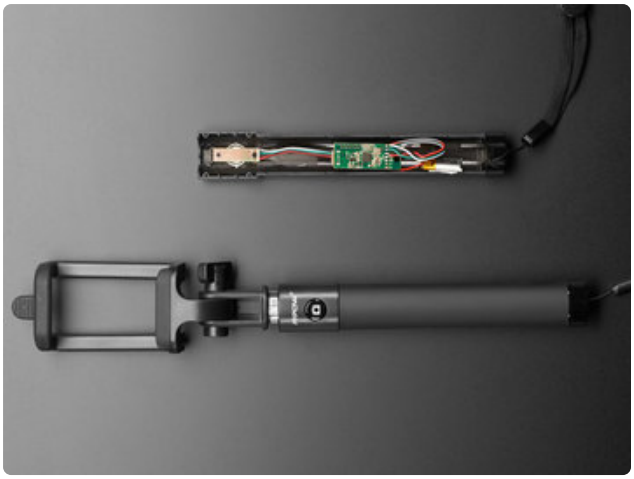
Even though Classic is no longer being actively developed you'll still see it used for some purposes.

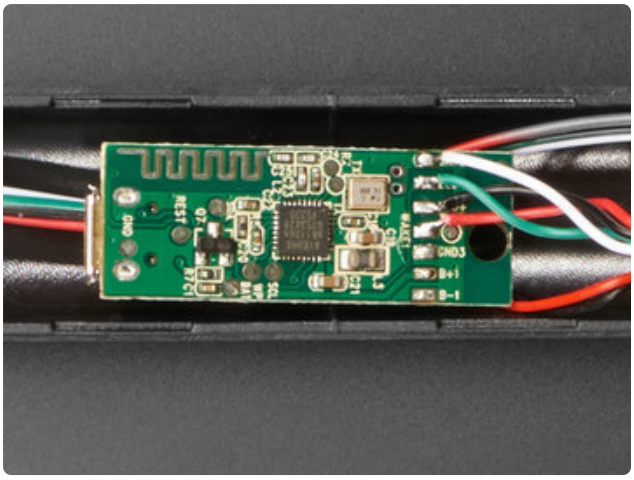


For example, audio devices such as wireless headsets and speakers are almost always Classic (there is a push to get audio devices moved to BLE but it hasnt happened yet due to the lower data rates specified in BT 4, version 5 will fix that).

In this Digi-Key bluetooth classic speaker, when we open it up we can see the control circuit board and a few chips, on the right you can see the BT trace antenna



Keyboards and mice can be either Classic or BTLE, it depends on when it was made but many are still Classic because production is already in progress.



For example this selfie stick uses a small Classic chip to send a single keypress.

We can barely make out the chip in the center of the selfie-stick PCB and a trace antenna in the top left corner

Classic can also do 'generic' data transmission using the SPP serial port service but SPP has fallen out of favor, mostly because it is unavailable to iOS developers without a lot of Apple-side licensing efforts. **Basically, If you want to do short-hop data transmission, go with BTLE.** Classic BT's current draw is maybe 20-30mA, and classic chips don't sleep well.

# Bluetooth LE - The New Hotness

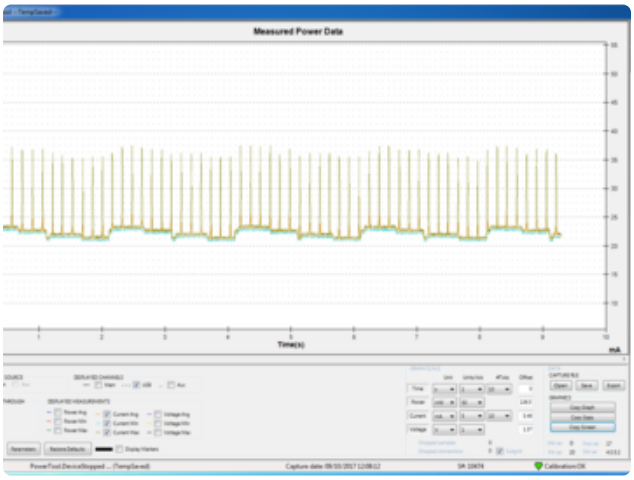
BLE is the newer Bluetooth. As we mentioned, it is not back-compatible, but adds a few things:

- Lower power
- Can use sleep modes for even lower power
- Much faster pair/connection times (classic is 30 seconds, BTLE cane be 3 seconds)

But...

- It has less bandwidth (the coming-soon version 5.0 changes this)

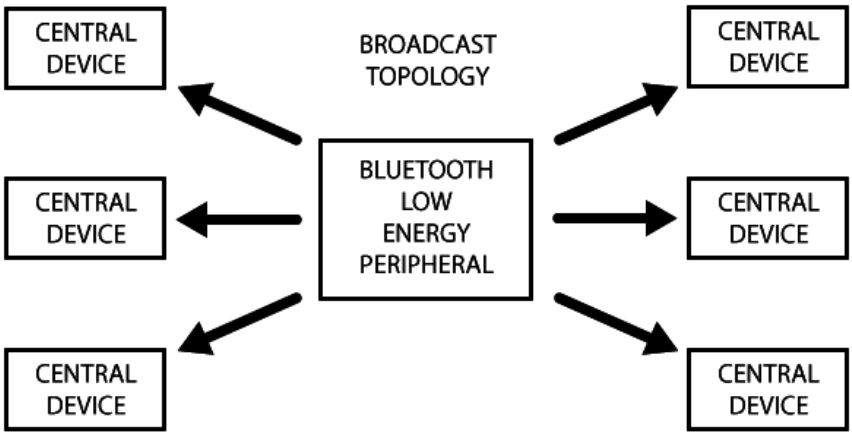
Because BLE pairs fast, you can sleep until it's time to connect and transmit data and the user often can't even tell. For IoT, **BLE devices can easily run for weeks, months or even a year on battery**. Your fitness monitor, for example, runs on BLE. It only has to sync wirelessly once a day so it sleeps until then and then does the data sync fast.

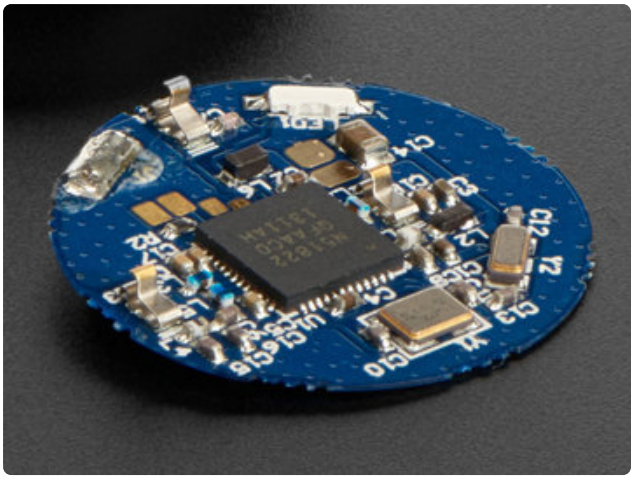



Here's a power trace showing a microcontroller with a BLE radio turning on just to transmit/receive data. The microcontroller isn't in sleep mode here, so you can see there's 20mA of baseline quiescent.

## Ultra low power Beacon mode

In beacon mode, where it is only transmitting a short burst of data once every few seconds, a year's lifetime is easy to budget. You can transmit a couple dozen bytes in a beacon burst which may be enough for your sensor data.



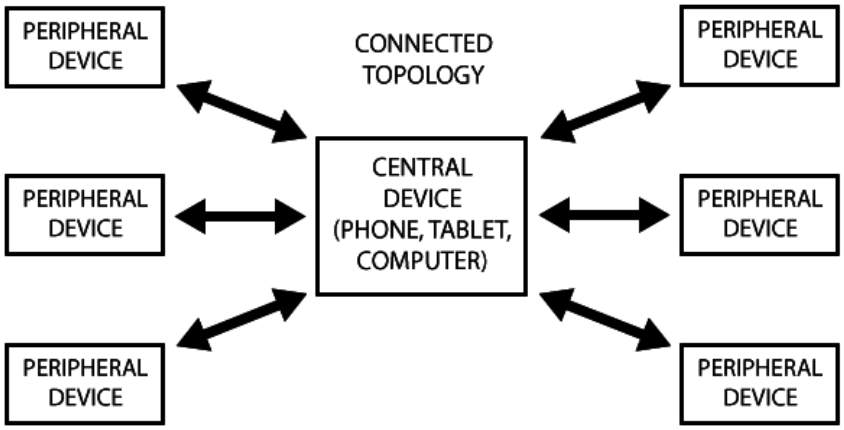
Commonly found BTLE 'lost and found' tags contain a small BTLE chip and a single coin cell. They send a beacon once a minute or so and can last for a year or more on the battery thanks to the sleep modes afforded.

# iOS Support

The biggest reason BLE has taken off is that Apple allows anyone to create a BLE-capable app and pair with their hardware. It's pretty much the only way you can connect hardware to iOS devices without licensing, special programs or an NDA. This has pushed vast numbers of developers over to BLE and has created a massive ecosystem of chip manufacturers, stacks, and code examples. Again, no such thing for Bluetooth Classic other than generic BT keyboard support

# Server to Multi-Client Connection

Note that for both classic and low energy, you have a 'controller' (a.k.a. central) and 'clients' (often called peripherals) so its a point-to-multipoint setup, just like WiFi's single access-point router and multiple clients.




It is possible to set up some BLE chipsets to be in central and/or peripheral mode for point-to-point links. Check the documentation for your device & ask the manufacturer - sometimes you can flip between the two, sometimes you can do both simultaneously - but you need the software stack support.

# Which to Choose? Classic or BLE?

One of the nice things about BLE is that the number of chips that support it have grown a lot so there's a lot of great options for BLE whereas for BT classic, the development environment, documentation, and options were very limited. **If you have to make a choice between the two, consider supporting BLE only!**

# Good Things about BLE

- BLE is very low power
- Supported by just about every phone/tablet/laptop and most PCs.
- Good for short-hops, up to ~20 meters
- BLE is fairly reliable for pairing, and fast
- Beacon mode may be all you need (but it is clear-text)
- All-in-one chipsets have improved a great deal
- Latest BLE 5.0 chipsets can do multipoint/mesh networking (check your chip docs!)

# Challenges...

- You may need to add and verify your own security - unlike WiFi many BLE stacks are often unencrypted by default!
- Not connected directly to the Internet, may need a gateway
- You may need to pay a BT SIG Fee per device.

- Some OS support is weak, especially pre windows-10 and linux

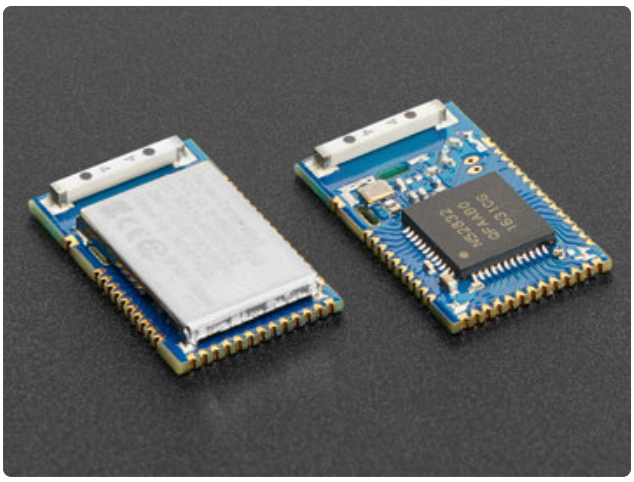
# Gateway to the Internet

Keep in mind, Bluetooth doesn't easily directly connect to the internet like WiFi can. (If you think about it, WiFi goes through a router to Ethernet) For Bluetooth, you'll need to set up your own router to send data via WiFi, ethernet, etc. Since almost every mobile device has both BLE and cellular/WiFi, often times those handheld devices are used as a bridge.

# Multi-Platform support

**The biggest downside we've seen with BLE is that desktop OS support for everything but audio/keyboard lags behind mobile support.** Because so many BLE devices connect to mobile, the Android/iOS support may be easier to implement and better supported than Linux/Windows/Mac. Note that this is software/OS support we're talking about. Hardware support for BTLE is pretty much standard issue now.

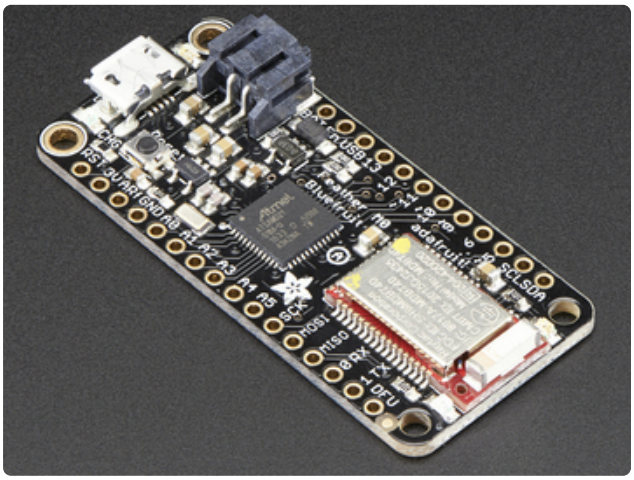
Often times, developers just decide that the only way to communicate with their gadget is through a cell phone. There is a slow-but-steady push to have WebBluetooth as a cross-platform wireless interface but it isn't finished yet (https://adafru.it/Bsy). This may not matter to you depending on your use case, but if you expect users to interact using their PCs, be sure to allocate plenty of time to develop a multi-platform solution!



You can save time by going with a precertified module, these often have pretuned antennas that work out-of-the box - less certification is required. But you'll pay a little more!

More final product designs do not use a module - the low power output of BLE and protocol-simplicity (compared to WiFi) makes it a lot easier to pass certification requirements

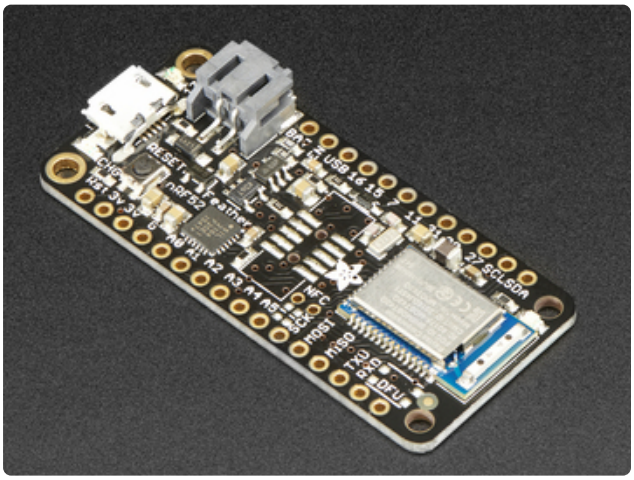
Here's an nRF52832 based module and the internals, as you can see there's not a lot going on here



Our original Bluefruit Feathers are the 32u4 and M0-based. These pairs are very similar looking, and have the same basic idea behind them: there is a **main processor** which is an ATmega32u4 or ATSAMD21 and a **co-processor module** which is the red and silver rectangle, containing an nRF51 which can do **Bluetooth Low Energy only**.

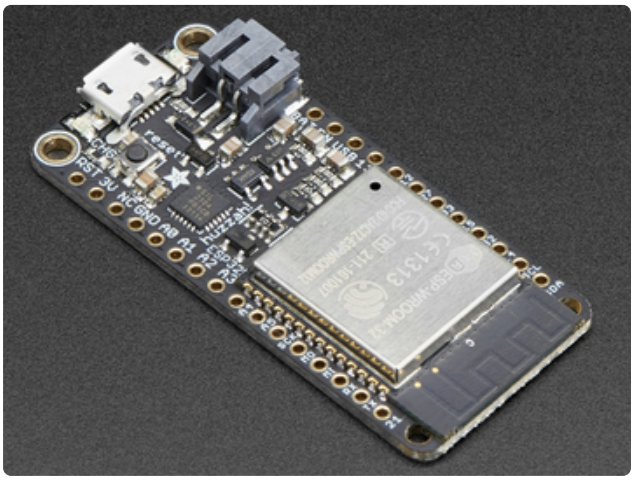
The nRF51 is programmed with our Bluefruit firmware and can be controlled with AT commands over SPI connection. When the main processor (32u4 or M0) wants to send or receive BLE data, it sends commands to the co-processor module

Pick up a Feather M0 or 32u4 Bluefruit at Digi-Key (https://adafru.it/zDo)



The Feather nRF52 is a new direction compared to our 32u4 or M0 Bluefruit boards. This Feather has only one chip on it - and that chip is both the processor you program and also the **Bluetooth Low Energy radio**. What's nice about this is you can do more powerful stuff, and faster too, because you don't have to manage two chips. It's also lower price and lower power since there's only one processor, and easier to put into sleep modes.

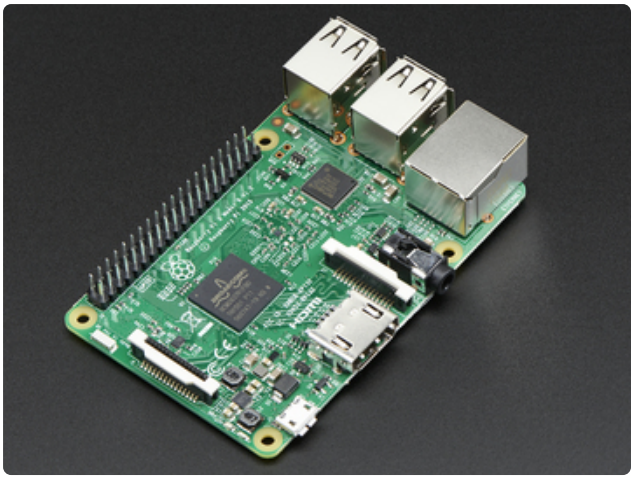
Pick up an Adafruit Feather nRF52 at Digi-Key (https://adafru.it/zDp)

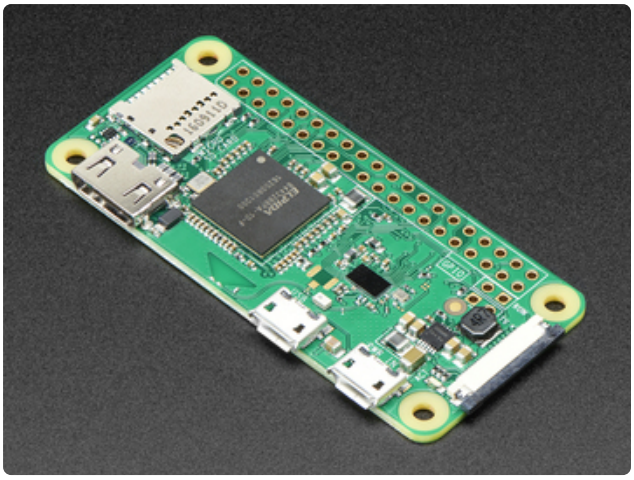


Even though this is primarily considered a WiFi Feather, the ESP32 does contain a BT LE and BT Classic radio! That's right this is the only Feather that can do BT classic. It's also the only one that can do WiFi and BT (altho, as of this writing, it cannot do both at once)

Sounds great, right? Well, there's some caveats. As of this writing, October 2017, the ESP32 Arduino Bluetooth core is still under development and there's only one basic beacon example. The Espressif IDF has more examples but some are not fully documented, so we put this one at the bottom of the list. If there's an example for what you want to do, then you're in luck!

Pick up an Adafruit Feather ESP32 at Digi-Key (https://adafru.it/zDc)





Raspberry Pi computers like the Pi 3 (https://adafru.it/zDf) and Pi Zero W have built in Bluetooth Classic **and** BTLE and BTLE can be central or peripheral. That's the good news. The bad news is Linux BT & BTLE support is notoriously difficult and variable. It's certainly possible to get it working, but it isn't as easy as something using a Nordic chipset, for example.

That said, the fact that the Pi has WiFi, BT/BTLE, and Ethernet makes it a very good candidate for a gateway device

---

# Cellular & Satellite

Of the three transports we've discussed so far, they have one thing in common - they require a hub/router/central that is nearby. So even if they don't have physical wires, there's a range tether. They are best used indoors or in a controlled environment. But if you need ultimate range, cellular, & satellite is where it's at!

# Cellular Communications

As you're well aware, cell phones can go just about anywhere - there are towers across every country, in almost any place there are people. And cellular towers can be very powerful. When coupled with a high power transceiver on your device, range can easily hit miles away.

But, as you probably know from your cell phone's battery life, power management is a big concern. In order to reach those towers, cellular radios can draw **Watts** of power and you can see current spikes of 2 Amps or more.

Historically, cellular was designed for audio use only, but cellular technology moves fast. Whereas a decade ago we could only really send emails and SMS messages, it's now commonplace to transmit full videos and large files. You may not need that kind of bandwidth but it's good to know its there.

# Cell Generations

There are a few choices for cellular.

The current generation is LTE (Long Term Evolution 4G and 5G) and is what new designs should target.

However, you'll still see some 3G chipsets and the ancient 2G GSM standard is still kicking around.

The biggest downside to cellular apart from the high current draws is cost: **Cellular modules are more expensive than WiFi or BTLE, and you have to use a module in order to get cell network-certified.** And no matter what you'll need to pay a provider, usually monthly for access and data usage.

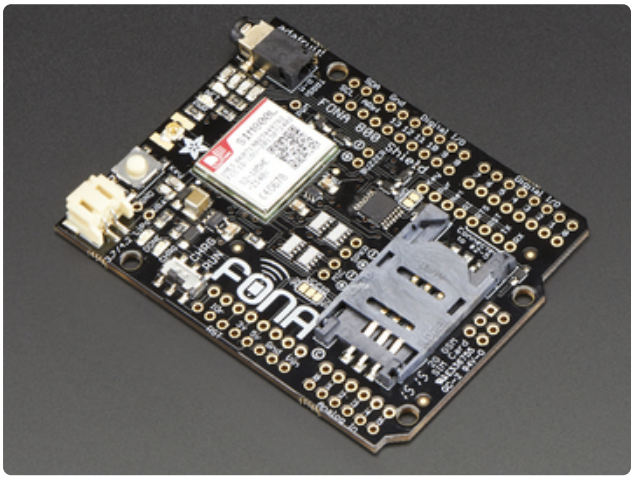
# The Past - 2G / GSM

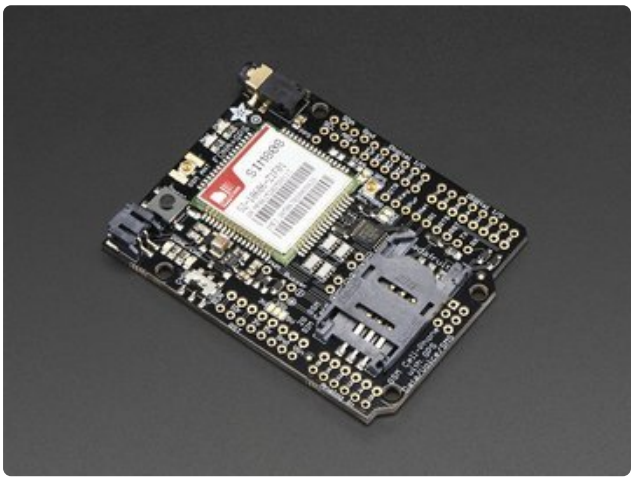
Watch out for low cost GSM cellular modules. GSM is also called "2G" or "GPRS" provides a 200-or-so kbps datarate and it is being shut down or already has been in many countries.

In the USA, AT&T has already re-assigned their bandwidth. **T-Mobile, the only other GSM provider in the US - has indicated that we only have until 2020 guaranteed**. But, there are millions of old GSM devices, so you may see that deadline inch along for a bit more.

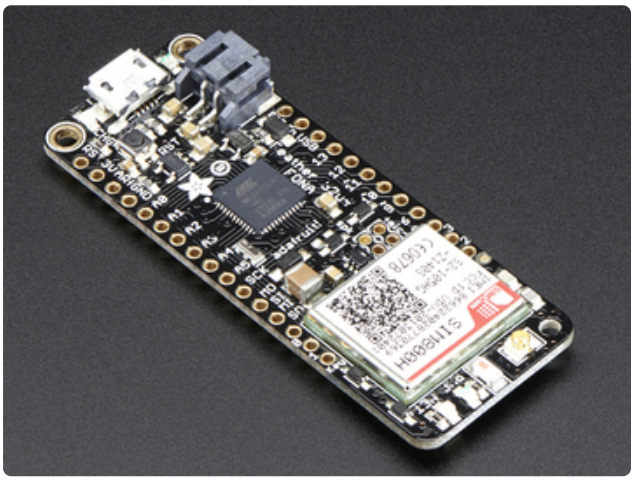
That said, if you have a project that operates in another country and that country is still on primarily GSM, the modules for GSM are smaller, lower power, and the data/ rates are pretty inexpensive. One nice thing about GSM chips is they tend to be

‘quad-band’ and will work in any country so you can develop in the USA and then send it abroad, or vice versa.





For 2G support, you can add the Adafruit FONA 800 (http://adafru.it/2468) or FONA 808 Shield (http://adafru.it/2636) to your Arduino-compatible. The 808 adds GPS in addition to GPRS



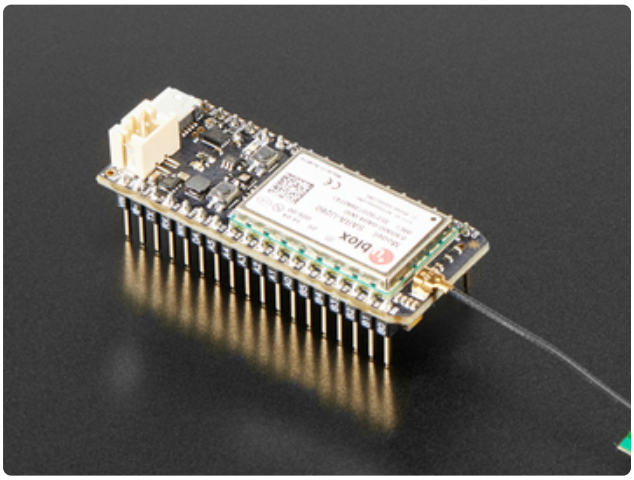
If you want Feather-format, we also have a Feather FONA 800 with an ATmega32u4 (http://adafru.it/3027)

# The Present - 3G

There are also many 3G cellular modules available now, these tend to be larger, higher priced and higher power, but still very reasonably priced. 3G will get you a few Mbps datarate. Major carriers have not announced any plan to shut down 3G services

so you should be good for at least 5-10 years. One thing to watch for is most modules are based on technology that can only be used in one region at a time - so you may have to pick different modules if you are using in the Americas, Asia or Europe.

For 3G usage, we have a breakout for the SIM5320 3G module (http://adafru.it/2696) that can do voice and/or data.



Particle uses the U-Blox SARA chipset, they also have a data management service. One nice thing about the SARA modules is they are upward/downward compatible with 2G and LTE versions of the same modules.

# The Future - 4G LTE

But, if you're starting a design now, you may want to consider LTE (sometimes called '4G').

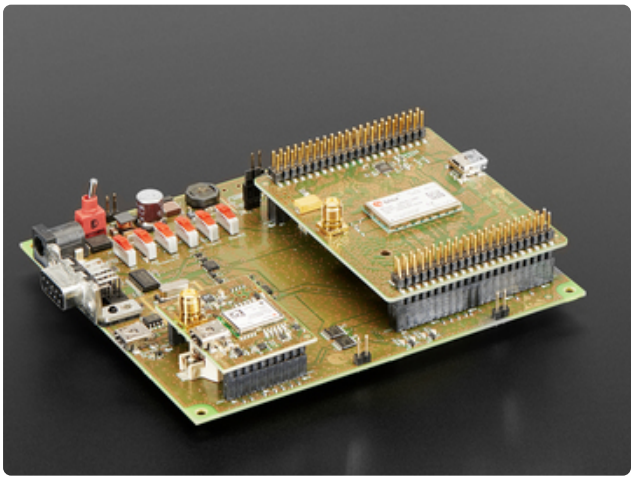
One nice thing about LTE is it has been thought through with respect to IoT so rather than sort of 'grafting' onto GPRS or 3G data and sharing the same technology stack as cell phones, there are different 'categories' to LTE.

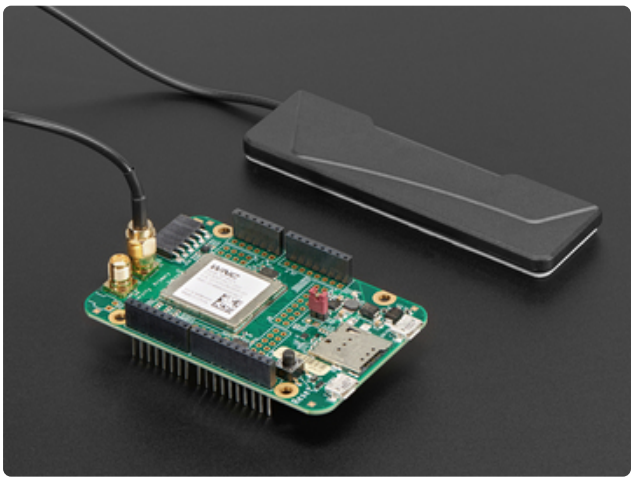
The categories basically range from 0 to 9. 0 is slowest/lowest power and 9 is highest speed. To make it a little more confusing, what would have been CAT 0 technology is referred to as **CAT M** - M is for machine.

What this means is that if you want to send sensor data once a week from a remote farmhouse you can go with a low category modem (say CAT M or CAT 1) and don't have to spend the same amount of money and power on a cellular modem that you would for streaming high speed video (say CAT 6).

Right now Cat 2+ is deployed world-wide wherever 4G is available. Cat 1 is available in North America and is rolling out world-wide. Cat M, which is best for low-bandwidth machine-to-machine is being rolled out right now, and has a few different sub-categories with varying bandwidths and ranges.

Since this technology is advancing so fast, you'll want to check with your favorite module manufacturers and carriers to see what is available!





We're still seeing the low-category modules filter into the market, there's maybe half a dozen available but there isn't enough data yet to form a strong opinion. (That said, the UBox SARA LTE modules work well and they are well documented)

# Cellular Pros

- Around-the-world off-grid range
- Direct internet connectivity
- Direct SMS/Voice connectivity (unique compared to other transports)
- Fairly reliable, cellular network is maintained by others
- Security built in - 3G/4G cellular network well encrypted, SSL when going to Internet
- Can do some rough geolocation

# Cellular Cons:

- Modules can be expensive and large
- Very high power usage
- Frequency ranges change in different countries
- Carrier required to setup access
- Monthly / per MB charges
- Network coverage varies
- Often have to use somewhat-archaic AT command set over UART
- Commitment to network: 2G cheap but discontinuing, 3G common but more expensive, LTE on the horizon, but not fully deployed for all M2M uses
- Some security issues with 2G, and phone #'s can be spoofed

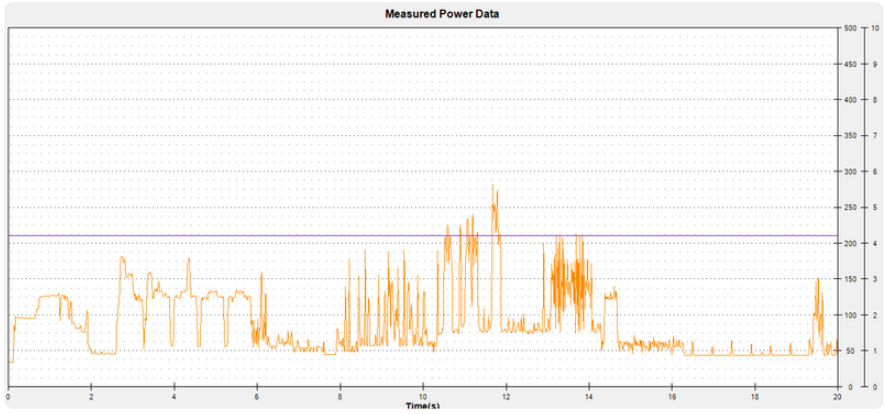
# Power Usages

If you think WiFi is power hungry, you will be surprised at how much power draw you'll need to manage with a cellular module.

Here's some power traces for common events with a cellular module:

## Turning on the FONA Feather

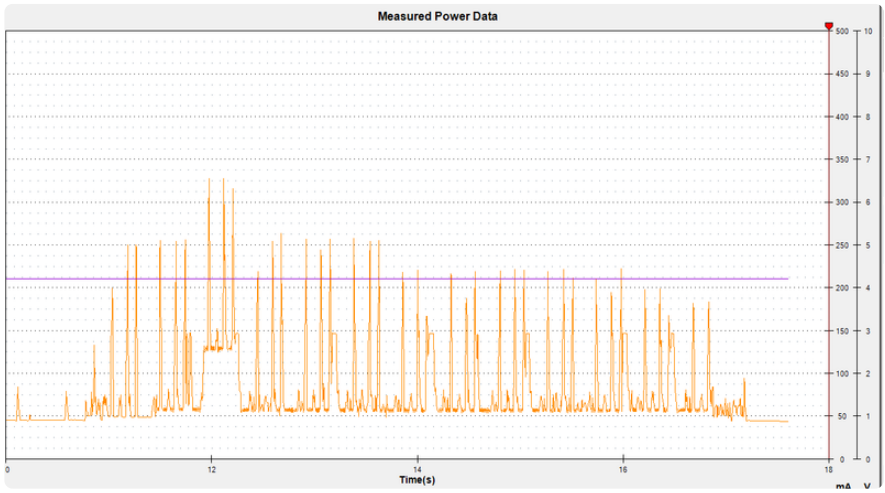
Booting cell module + connecting to network




## Sending an SMS

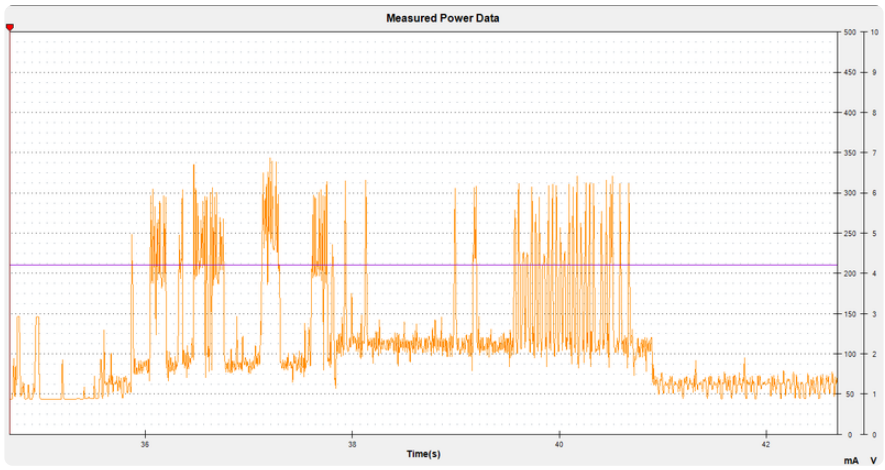
send SMS: 6.5s, 150uAh, 300mW, 52mA

recv sms: 6.5s, 140uAh, 330mW, 78mA



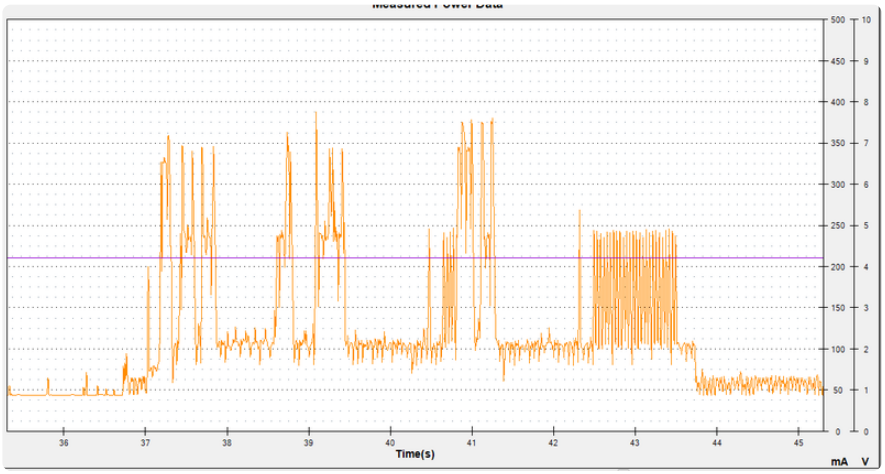

## Enabling GPRS

enabling GPRS: about 8 seconds, 850uAh, 300mW, 70mA avg

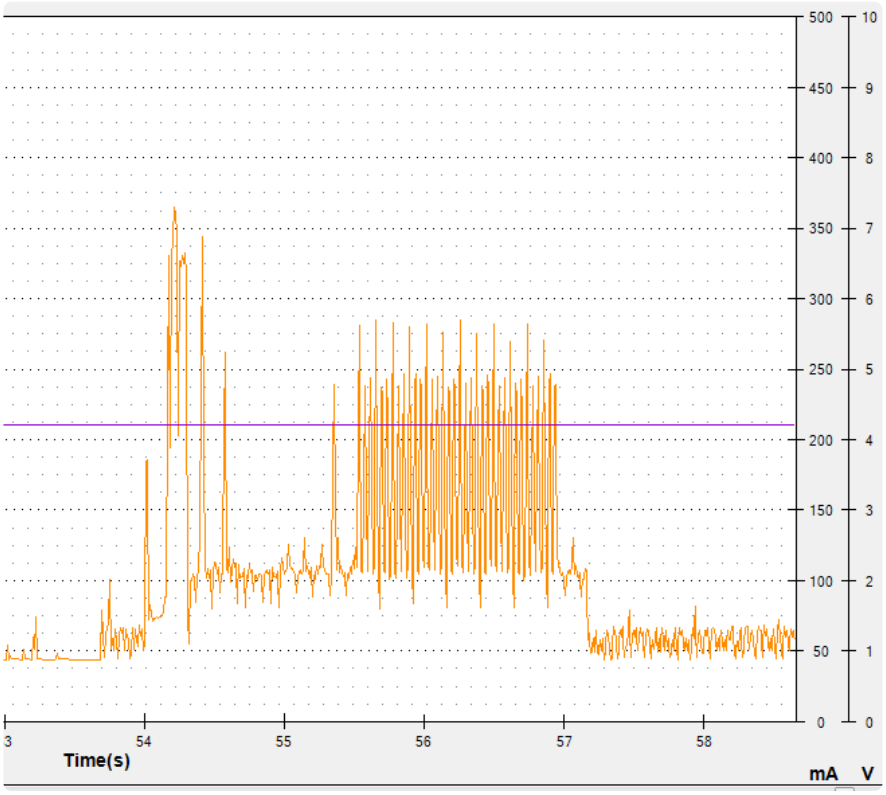


## TCPIP connection

grab mini webpage: 4.5 sec, 203uAh, 650mW, 150mA avg

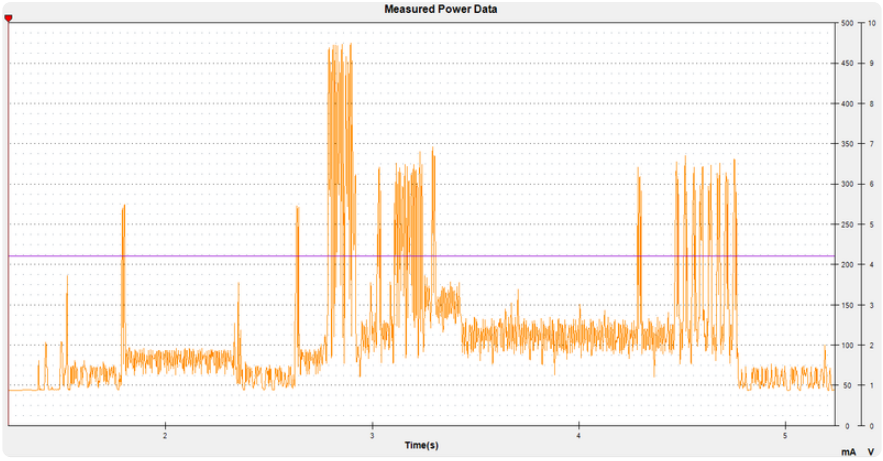



## Sending an MQTT packet (about 200 bytes)

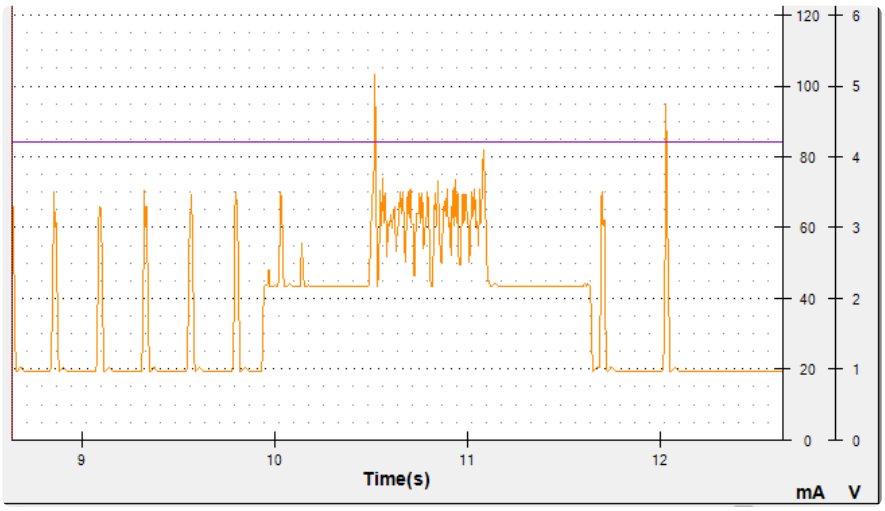



## Disabling GPRS

disabling GPRS: about 4 seconds, 120uAh, 480mW, 113 mA avg




You can put the FONA into sleep mode (with the **AT+CSCLK** command) which will drop the current draw but keep the cellular connection open so you can still receive an SMS and/or wakeup quickly.

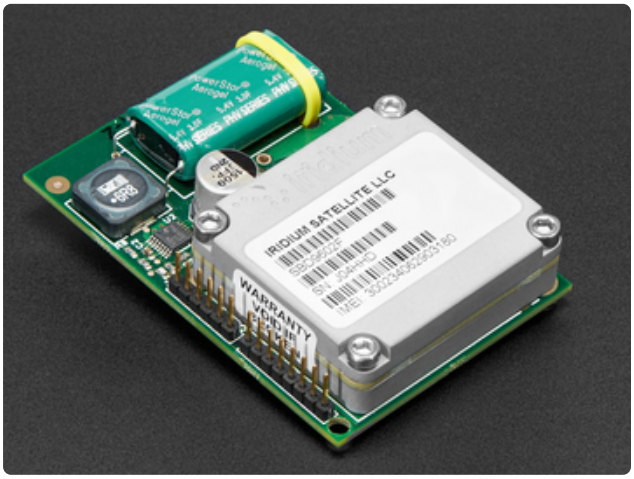



Note that the quiescent current drops from 40mA to 20mA and of that 20mA, about ~12mA is the microcontroller

# Satellite

This is sorta like cellular so we will cover it briefly here. When you're not near a cell tower, like the middle of the ocean, in the wilderness, far from civilization, or where world-wide functionality is required, how can you get your thing communicating? Sensors, transportation, scientific data capture, search and rescue - these situations might stick you in the middle of nowhere. For those situations where you have **very little data** you have to manage and transport, satellite link-ups may work well. Note that this is a very constrained transport. For civilian uses, you will be using the Iridium constellation. You'll pay a lot for the technology, it's huge, takes a lot of power, and you'll also pay per month and per message.

Of course, there are a lot of downsides but if you absolutely need to communicate from anywhere in the world, without needing to set up your own bridge, relay or network - this is the only option you've got.


The RockBlock is an easy-to-use all-in-one module that also has a message purchasing system. Its expensive, but you get both modem and GPS unit and it works anywhere in the world!

---

# ZigBee & Z-Wave

# 'Home Area Network' Radios

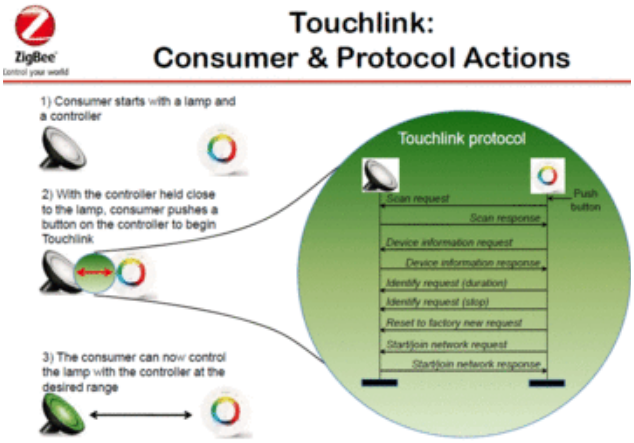
Now we're getting into set-ups where you have to run your own network. Remember, with Ethernet, WiFi, Cellular, and Satellite you've got an existing network you're joining as a client, and with BTLE your phone or computer can act as the network manager.

ZigBee and Z-Wave (as well as 'similar' low power radios such as XBees) are often used for home or office-scale networks and automation. These are low power radios, so great for battery usage. But more importantly, they are usable in mesh and high-density point-multipoint or multipoint-multipoint networks - something BTLE does not yet do very well.

These two transports, despite both starting with Z and both used for home/ industrial automation, have nothing to do with each other.


# ZigBee

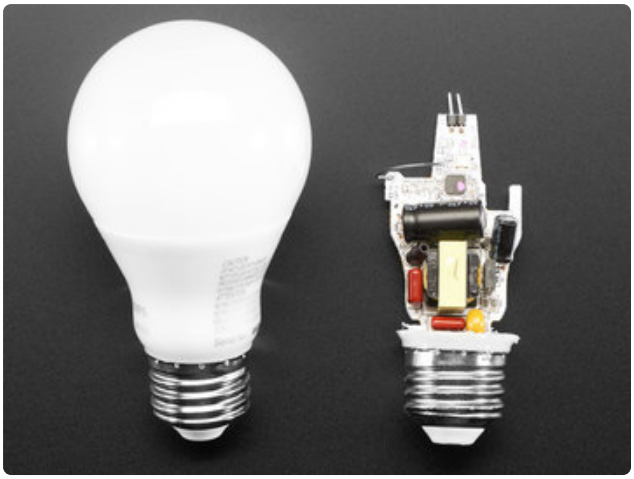
ZigBee/802.15.4 is available in both ~900MHz and  2.4GHz. You'll see it most often as 2.4GHz though, so it shares the same frequencies as WiFi/BT. Zigbee and is a freely available standard with dozens of chips available and a flexible protocols. You can use an existing 'profile' like the ZLL (Zigbee Light Link) or just make your own.



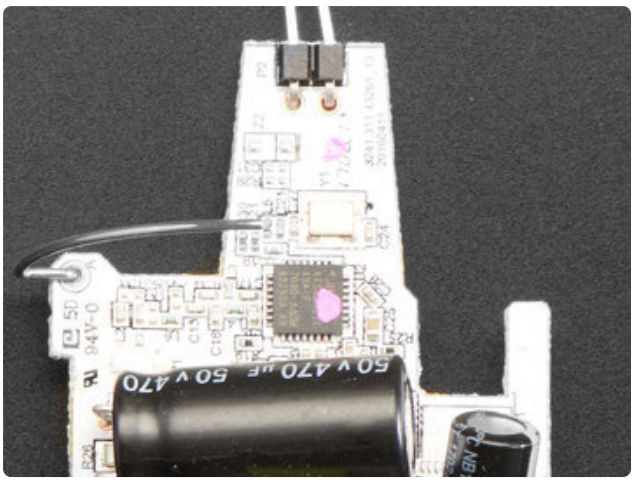

For example, here's a slide from the ZigBee Alliance presentation on Zigbee Light Link (https://adafru.it/zDq) showing the defined protocol for a lamp and switch to 'pair'. It's complicated, sure, but it's well defined and interoperable.

## Gateways

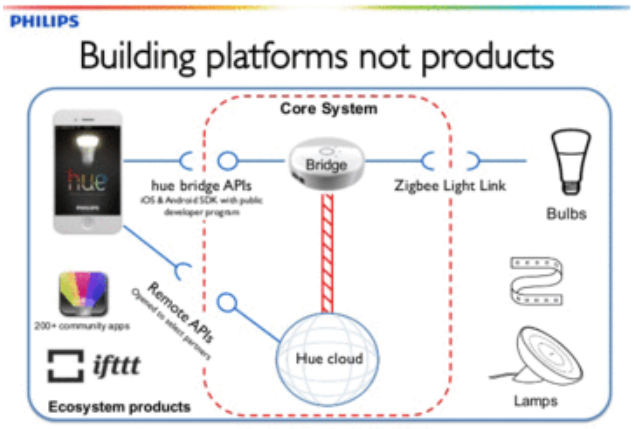
ZigBee is cheap, cheaper than WiFi, but to connect it to the Internet you'll need a gateway.



So, for example, Phillips Hue bulbs have a ZigBee chipset (The ATSAMR21 series of chip) inside that does both the radio and LED control.



They could have gone with WiFi or Bluetooth in each bulb but it would have greatly increased the cost and complexity. When you need it to be, well, as cheap as a light bulb, ZigBee radio's $1-per-chip radio probably helped a lot.




All the lightbulbs talk to a gateway bridge/ hub which as both Zigbee and WiFi or Ethernet. All of the switches and controls all talk to the same hub. That hub then connects to the internet or just is used from within the home without external access.

Slide from Philips' technical presentation (https://adafru.it/zDr)

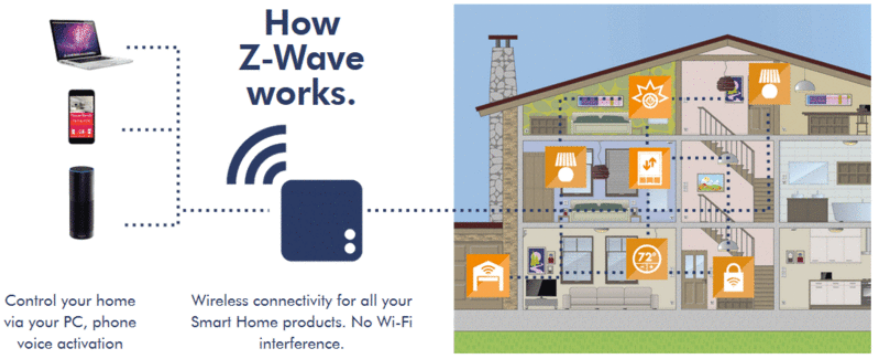
# Z-Wave

Z-Wave is a more constrained-use protocol, its design for home use and to make equivalent devices you have to buy chips from the single-supplier. It's a 900 MHz standard so you get a little more range than 2.4GHz ZigBee. especially in an industrial or urban environment since Sub-1GHz penetrates brick and concrete better than 2.4GHz.

The big benefit to going with Z-Wave is you're joining 100+ other companies that provide cross-compatible support.

The big downside is you are stuck with a very proprietary protocol - even though some parts have been open sourced, you're still marrying a single company - compare that to the dozen suppliers of ZigBee chips and multiple ZigBee stacks available.

Like ZigBee, you need to have a Z-Wave gateway device to get to the Internet and enable BTLE/WiFi access




# Mesh Networking

ZigBee and Z-Wave do have one thing that they do well that Ethernet/WiFi/Cellular/ BTLE do not - that's 'mesh' networking. Mesh, when done right, lets you increase network range with every new device added because each radio acts as a transceiver/router. But, that's when its done well. And it's not easy to do well. In particular, mesh doesn't work nicely with low power networks because every device has to be ready to wake up at any moment and start to do a bunch of packet routing.

That said, BTLE 5.0 will have mesh capabilies, so you may want to compare BTLE 5.0 and ZigBee when designing your low power radio device. In general, BLE is still being

improved and developed and is gaining more Z-Wave/ZigBee-like capabilities than the other way around.

# Reasons To Use ZigBee or Z-Wave

- Very low cost and complexity (compared to WiFi, cellular)
- Low power
- 802.15.4 stacks are popular and available from multiple vendors, many even bundle a microcontroller with it
- ZigBee profiles may make your design easy - e.g. ZLL for light control or HA for home automation
- Interoperability: May be able to join existing home area network if on a compatible profile
- Can be mesh-networked (just watch out for power usage)

# Watch out for...

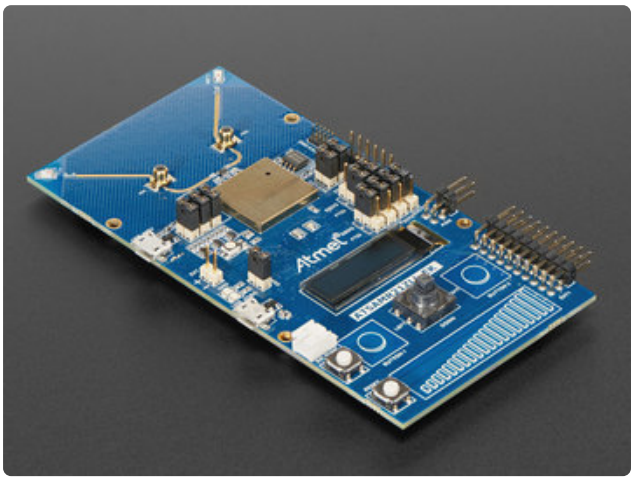
- Need a gateway for internet access
- Need to manage your own network
- Z-Wave chips are only available from one supplier.
- May have to join an ‘alliance’ or pay into a proprietary network
- DIY authentication, some security built in

# Adding ZigBee to your Project


There are vast numbers of available 900 Mhz & 2.4GHz ZigBee modules and chipsets. If you want plug+play, the XBee series use a simple AT command set that you can drive from a secondary micocontroller.

There are also dozens of chips and modules that have a microcontroller core + ZigBee radio. You can get these chips from TI, Microchip, Atmel, NXP, SiLabs...pretty much everyone. So if you have a favorite core that you use a lot already, it's pretty easy to find one in the family with ZigBee built in!

Check out the huge offerings at Digi-Key (https://adafru.it/zDs)




If you want to use ZigBee Light Link, there are some nice existing dev kits such as the ATSAMR21ZLL-EK from Atmel (https://adafru.it/zDt) available from Digi-Key

# Adding Z-Wave

For Z-Wave, there aren't as many options - remember that there's only one chip available - but you can get various modules and dev-kits built on the ZM chips.

You can see the full range of Sigma Z-Wave offerings at Digi-Key (https://adafru.it/zDu)

---

# LoRa & SigFox

So far we've covered plenty of short-hop radios (BT, ZigBee, ZWave), and next up are their big sisters, the medium-hop radios. These radio protocols take the same idea of a self-controlled wireless network, and give you much longer ranges. Since they have wide range, and low power, they are called LPWANs for "low power wide area networks"

# Low Power Wide Area Networks (LPWAN)

LPWAN devices are not new technology, in fact, cellular is essentially a national-sized version of one. But for people who don't want to tie into the Cellular network, and don't need the bandwidth, LPWANs can give you excellent range for very little power and not very much money!

These networks can reach for many miles outdoors, even in high-density cities. They're perfect in country-sides, rural and semi-urban spots, especially if you have the ability to set up a directional antenna in a high spot. What's great about all of them is there's no complex pairing, connection or authentication overhead (unless you want it) so you can wake up your setup and start transmitting within milliseconds.

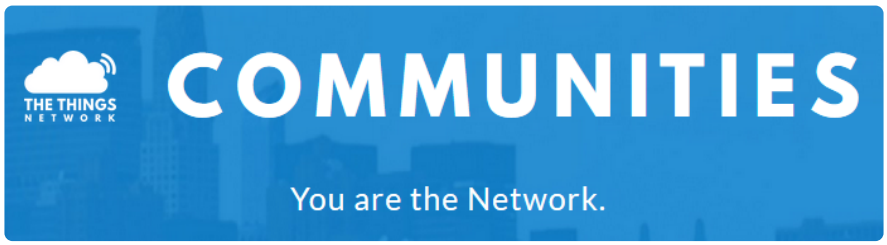
There are three common LPWAN solutions - two commercial and one generic.

# LoRa & LoRaWAN

The first commercial one is LoRa (sometimes called LoRaWAN when a controlling protocol layer is added on.) LoRa uses any frequency, it's more of a frequency management protocol - but tends to be used at 315, 433, 868 and 900 MHz ISM bands. LoRa has very little structure, it's just for sending and receiving packets of data - retransmission, link management, etc is all on you.

You can have more network management control if you add LoRaWAN on top. Like ZigBee, you're on your own to create the network and manage routers, bridges etc. But, there are some cities that have networks set up, much like the radio relays that Amateur HAMs use.

For example, The Things Network (https://adafru.it/BsB) is a social network for LoRaWAN gateways and communities




LoRa is patented, and radio chips are only manufactured via SemTech but are bundled into modules by many companies. They're much cheaper than cellular, but a bit more expensive than BTLE or ZigBee (maybe around $4 each) and need to be controlled by a microcontroller. You can also get full LoRaWAN modules that make connecting up to a LoRaWAN network easy.

# LoRa pros

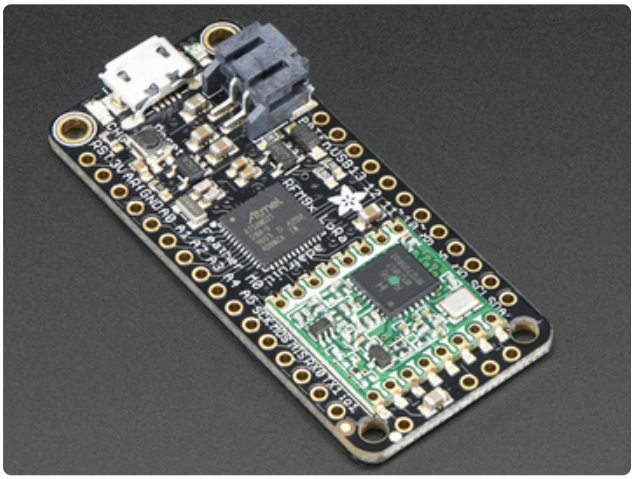
- Free to use, can set up own network
- Send as many messages as you like, at good speeds up to 50 Kb/s

- Very long range (a few km in cities, up to 40km in rural areas with directional antennas)
- Fairly low power, depending on your radio amplification, not as low as BTLE but much better than cellular
- Pick and choose any frequency you are legally permitted to use

# LoRa cons

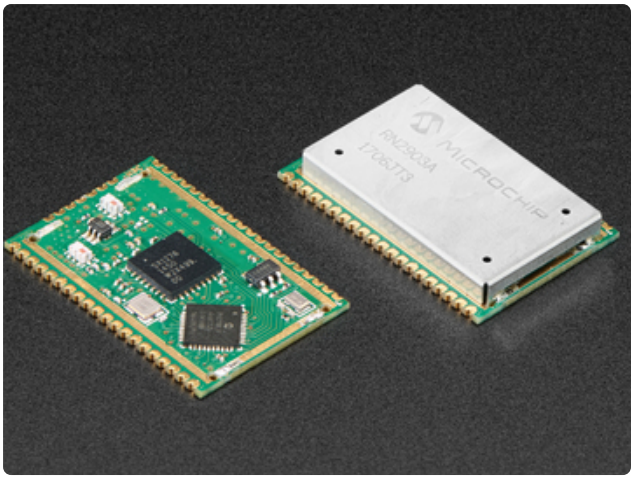
- Must manage your own network/gateway
- Chips only available from SemTech, and under patent

# Add LoRa to your project



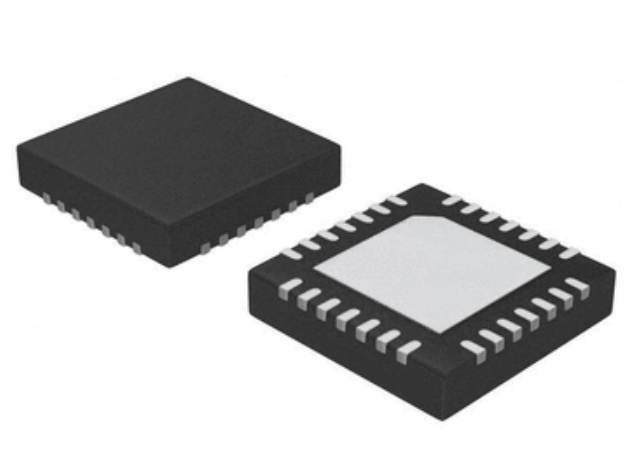
We have Feather boards with ATmega32u4 (8-bit) or ATSAMD21 (32-bit Cortex M0) chips and Semtech LoRa chipsets ready to go. You can get either ~433 MHz or ~900 MHz variants, the code is the same, its just the antenna/radio part that is tuned.

You can pick up an Adafruit Feather LoRa from Digi-Key! (https://adafru.it/zDM)




The Microchip RN2483 is a 'fully integrated' LoRaWAN module. You get the Semtech chip plus a microcontroller with a stack on it. It's popular with people who don't want to implement the LoRaWAN parts and want to get right into it!

You can pick up the RN2483's from Digi-Key (https://adafru.it/zDN)

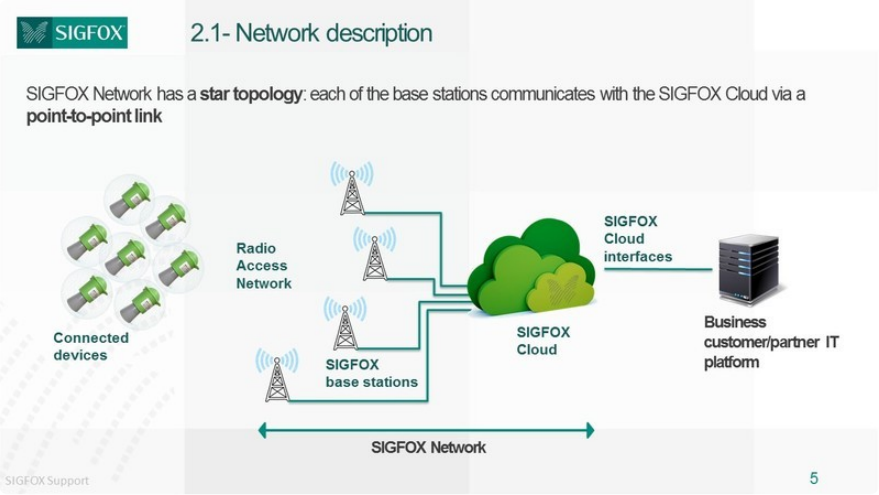


If you want to, you can also just DIY your own setup by picking up a 'raw' Semtech SX127x chip from Digi-Key (https://adafru.it/zDO)

# sigfox

sigfox is a similar radio protocol, but closer to the cellular network in style. Rather than set up your own LoRa network, sigfox is an existing network set up by the self-same company. You can only use sigfox chips in a location where there is an existing network set up, but many cities have one. After you've registered and paid, you can join the network, send and receive data.

For example, as shown below, the **SIGFOX Network** portion is taken care of for you, you just do the Connected Devices part and then send/receive data via the cloud interface.




The nice thing about SigFox is you don't have to create a network, manage gateways, etc. Since the receivers are permanently installed, they're very fancy and powerful and let you have less expensive and less powerful chips.

**The big constraint is the very tiny message bandwidth**: only 140 x 12 byte message per day upload, and 4 x 8 byte per day download. Like, really limited. But maybe for some sensor data it's enough?

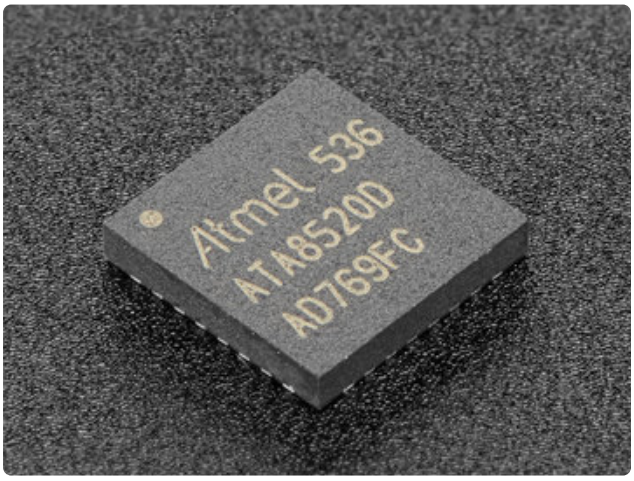
# sigfox pros:

- Backend network taken care of
- Very long range (a few km in cities, up to 40km in rural areas with directional antennas)
- Very low power

# sigfox cons:

- Paid subscription service
- Ultra slow: 100 bytes/sec
- Not available everywhere - check if there's a network provider where you are deploying
- 140 x 12 byte upload and 4 x 8 byte download messages a day
- Fixed frequency per location

LoRa and Sigfox are fairly new but fill an important need where you need very low bandwidth and very high ranges, even with SigFox pricing, it's cheaper than managing a SIM card for each device. And compared to cellular the power draw is minimal. Ethernet, WiFi, Bluetooth, Cellular & Satellite were not made for millions of little low power devices whispering around the clock - but LPWANs are.




For SigFox integration, Microchip/Atmel makes the ATA8520 which is a all-in-one SigFox transciever chip on 868 MHz (https://adafru.it/A05)

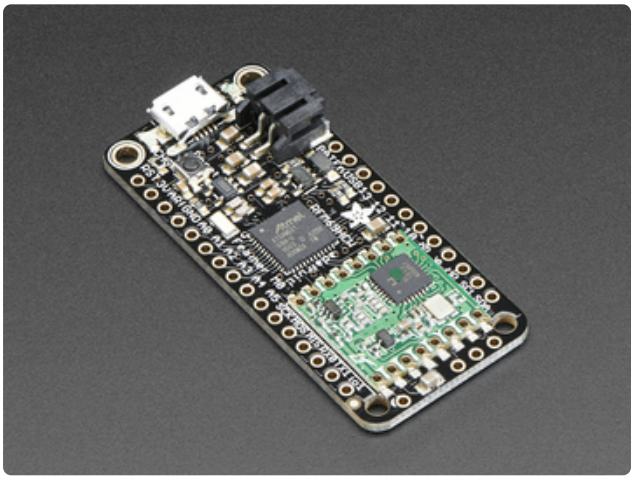
# DIY Radio!

The third option is for those who want long range and maybe don't want or need to use LoRa or SigFox at all - just roll your own!

E.g. GoTenna used Silicon Labs Si4460 Transceiver (https://adafru.it/BsC) to create a 100 MHz radio network. The low frequency allowed them to get awesome range for point-to-point links. Some makers like using modules that use FSK radios such as SX1231 (or any number of others).

Managing these is done a lot like LoRa - you set up your own network and gateway. But with generic FSK-y radios you aren't even required to pay out for the LoRa patent licensing, so they're very inexpensive for the range you get.

Some radios even come with built-in encryption and link management assistance so you get stuff like CRC, retransmission, and node addressing. The simplicity of these radios let you stay in sleep mode for a long time until you need to wake up and transmit.



We have Feather boards with ATmega32u4 (8-bit) or ATSAMD21 (32-bit Cortex M0) chips and Semtech SX FSK-encoding 'RFM69' chipsets ready to go. You can get either ~433 MHz or ~900 MHz variants, the code is the same, its just the antenna/radio part that is tuned.

You can pick up an Adafruit Feather RFM69 from Digi-Key! (https://adafru.it/zDP)