

IoT Use Cases

IoT Deployments for Fun and Profit

1. Why IoT? Why ESP?
2. Common IoT Fails
3. Backends
4. Architectures:
 - a. Device to Cloud
 - b. Cloud to Device
 - c. Device to Device
5. Concluding Thoughts





What is the Internet of Things?

Internet of Things (IoT) technologies are very poorly defined -- but we usually mean very inexpensive devices (“Things”) that send or receive data from the Internet. Unlike a computer or smartphone, they tend to have a very specific function.

In recent years, low-power hardware that can connect directly to the Internet via Wi-Fi has become available for under USD 4\$, and understand Arduino C, Lua, and Python.

Not long ago, the cost was over USD 100\$, and required specialized knowledge of embedded systems.

Internet connected sensors have become so cheap to design and deploy, that advanced data collection and remote control are accessible to any company or hobbyist.

Why Internet of Things?

Data Matters. IoT technologies let you collect a lot of data very cheaply.

Automation is the future. IoT technologies allow AI and Big Data direct agency over things that matter in the analog world.

Decentralization adds value. IoT technologies allow device-to-device messaging, data processing, and local decision making. You can build systems without a single point of failure.

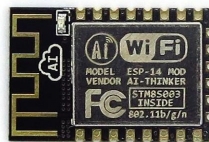


What Hardware is Used

There are many platforms. The cheapest and most versatile one is the ESP8 and ESP32 chips from Espressif.

They are Arduino compatible, although support frameworks more suitable for commercial deployment too.

Our examples today will use the ESP8266, available in Vietnam for under VND 100.000





Things to Look Out For

No hardware security: The whole flash contents of the chip can be trivially downloaded

Limited memory: Encryption and HTTPS are challenging

Power consumption: Can consume up to 350mW, use sleep modes to bring it down to 1 mW or less

(Arduino only) libraries need work: Often the pin assignments or other details are wrong in the libraries and you'll have to fix that yourself. Other times they work fine.

Default clock is set to half maximum: It defaults to 80Mhz, but can run at 160Mhz

Backends

ThingsBoard: Free version you can host it yourself! Really slick.

Ubidots, Cayenne , Adafruit IO: Limited free access, and no need to install

Google Cloud IoT: Not as easy to use somehow! Very difficult to get data into. Good for commercial deployments.





Where to Start?

The easiest way to divide and conquer those issues is to define 3 IoT architectures with a unique set of challenges and solutions. We will cover a working example of each.

Device to Cloud: IoT devices collect data from their area and send it to a server on the Internet. Security is usually less of a concern here. Common examples are internet connected weather stations.

Cloud to Device: IoT devices receive data (e.g. control signals) from a server on the Internet. Security is usually a bigger concern here. Common examples are access or climate control systems controlled by your smartphone or a website.

Device to Device: IoT devices send and receive data only with each other. Security is usually a moderate concern. Common examples are physical remote controls for other IoT devices, or offline sensor networks.

Device to Cloud -- A WiFi Panopticon

A panopticon is a device invented 200 years ago that lets you monitor an area invisibly. This one uses Wi-Fi!

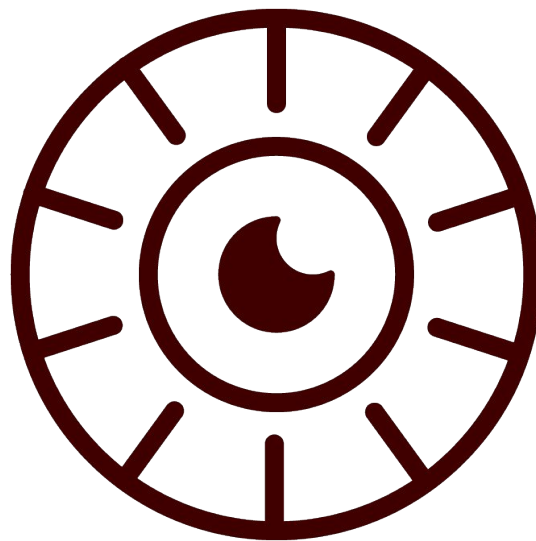
BOM:

ESP8266 (VND 60.000)

Cellphone LiPo (30.000)

USB Charger (10.000)

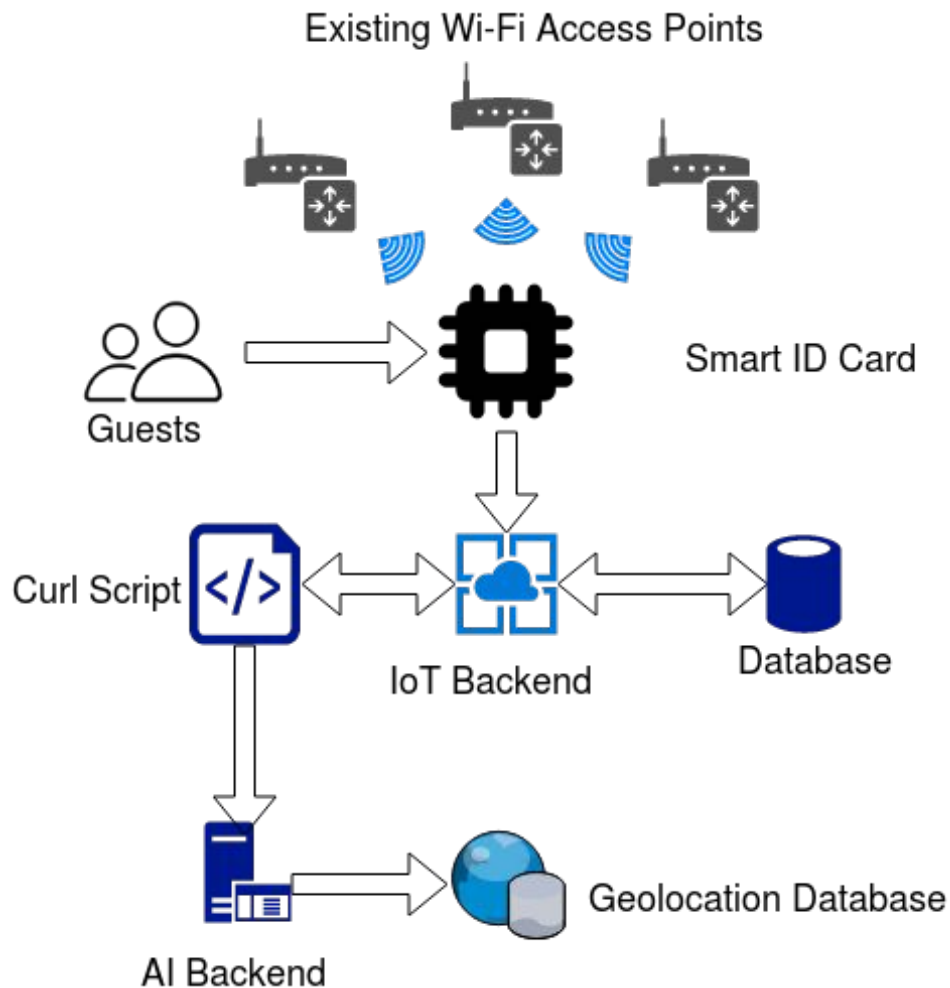
Total: VND 100.000 (less in volume)



Architecture

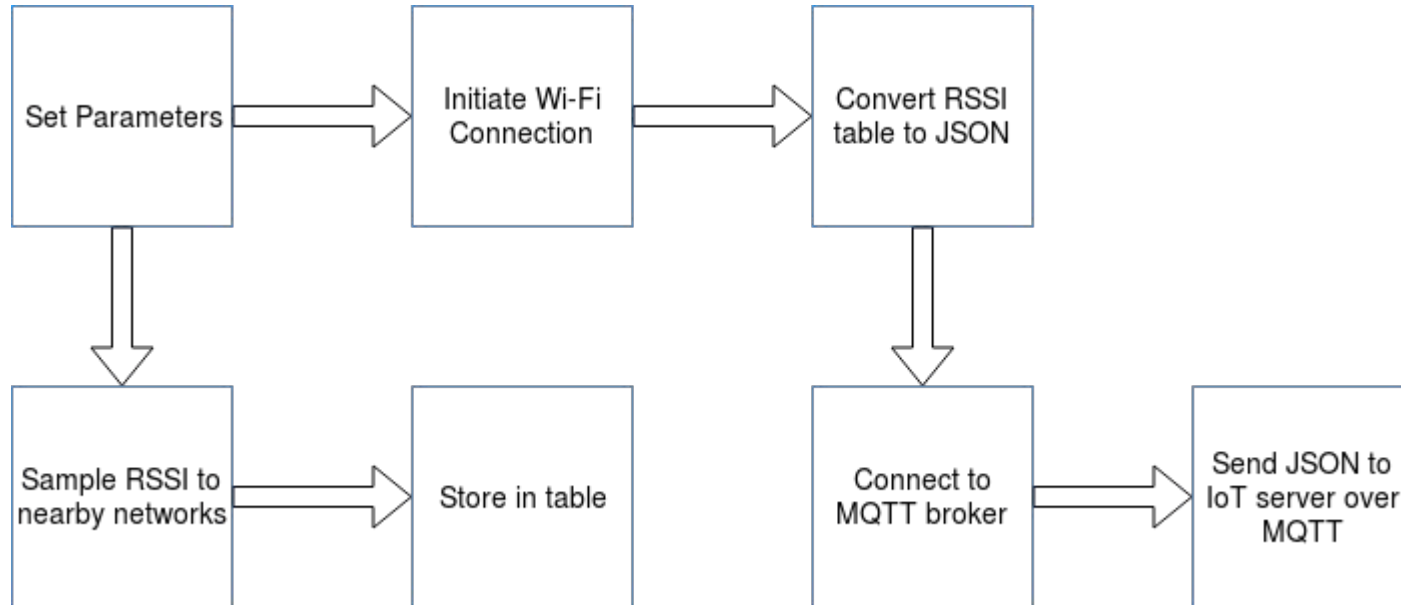
The self contained IoT access card has tight restrictions on power, size, and cost!

The smart ID card can tell the backend what room it's in based on signal strength from the different routers.



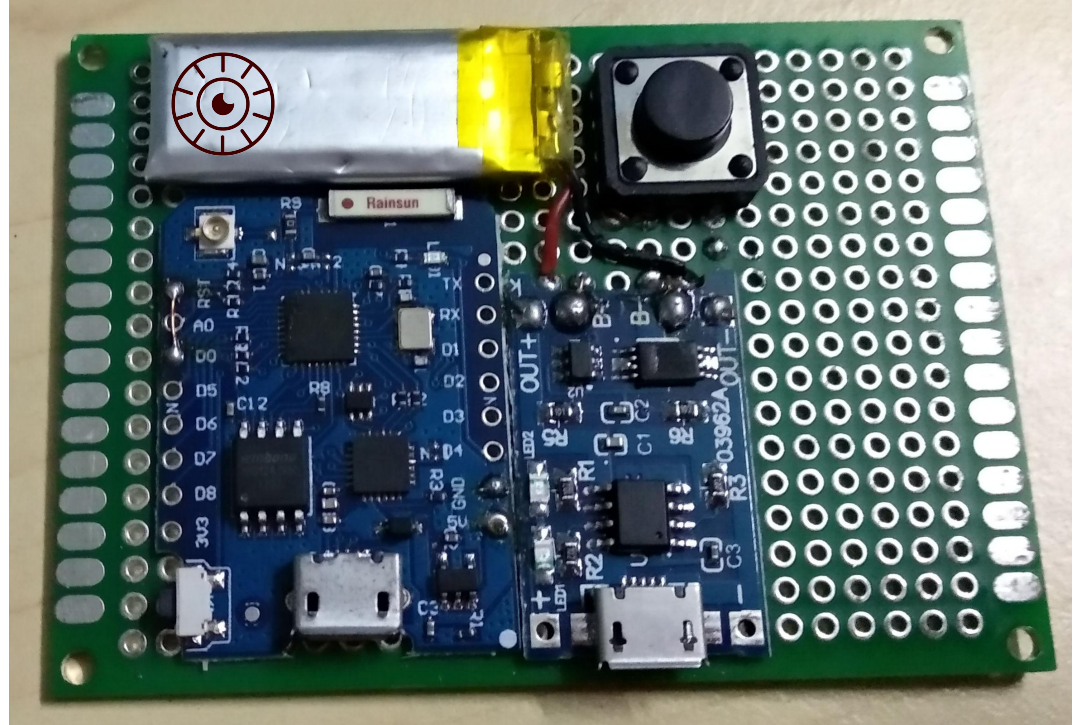



Program Flow



Finished Hardware

When you press the button, the chip powers on, scans for local routers, and send the data to the Internet.





Device to Cloud -- A Multi Sensor Board

This prototype multi sensor board was designed to monitor the environment in a datacenter. It includes:

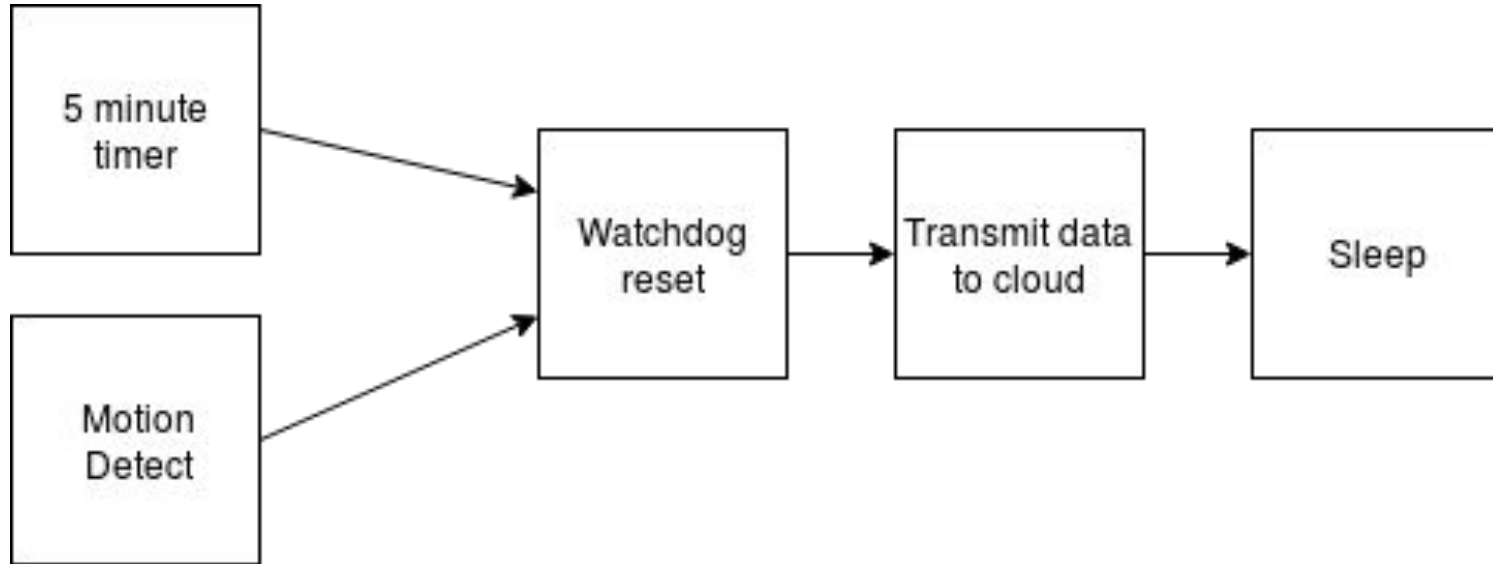
- Pressure plate to detect if any equipment is tampered with
- Humidity sensor
- Air temperature sensor
- Dust sensor
- Penetrating radar (6m sphere) for intrusion detection

The total BOM was around VND 800.000

A commercial IoT data center dust sensor (same sensor model) starts at around VND 17.000.000 and requires that you buy a dedicated router for another VND 22.000.000!

Clearly the price point of some of these products needs to be reconsidered.

Architecture / Program Flow



Finished Hardware



Cloud to Device (low security) -- An Ethereum Wallet Display

Need to check how much cryptocurrency you own?

Logging in to your wallet every time requires your password, creating a security hazard. Since data on the blockchain is public, why not just request your balance without logging in?

We can do that using an API (e.g. Coingecko, Etherscan).

We can alternatively display the current Ethereum price.



How it Works

It's just a HTTP request to:

[http://api.etherscan.io/api?module=account&action=balance&address=0xde0b295669a9fd93d5f28d9ec85e40f4cb697bae&tag=latest&apikey=\(your API key\)](http://api.etherscan.io/api?module=account&action=balance&address=0xde0b295669a9fd93d5f28d9ec85e40f4cb697bae&tag=latest&apikey=(your API key))

It returns JSON format, easy to decode and display on a screen.

Note that most APIs work like this, from weather, to blockchain, to stock markets so this approach lets you build devices that can be aware of rich data.

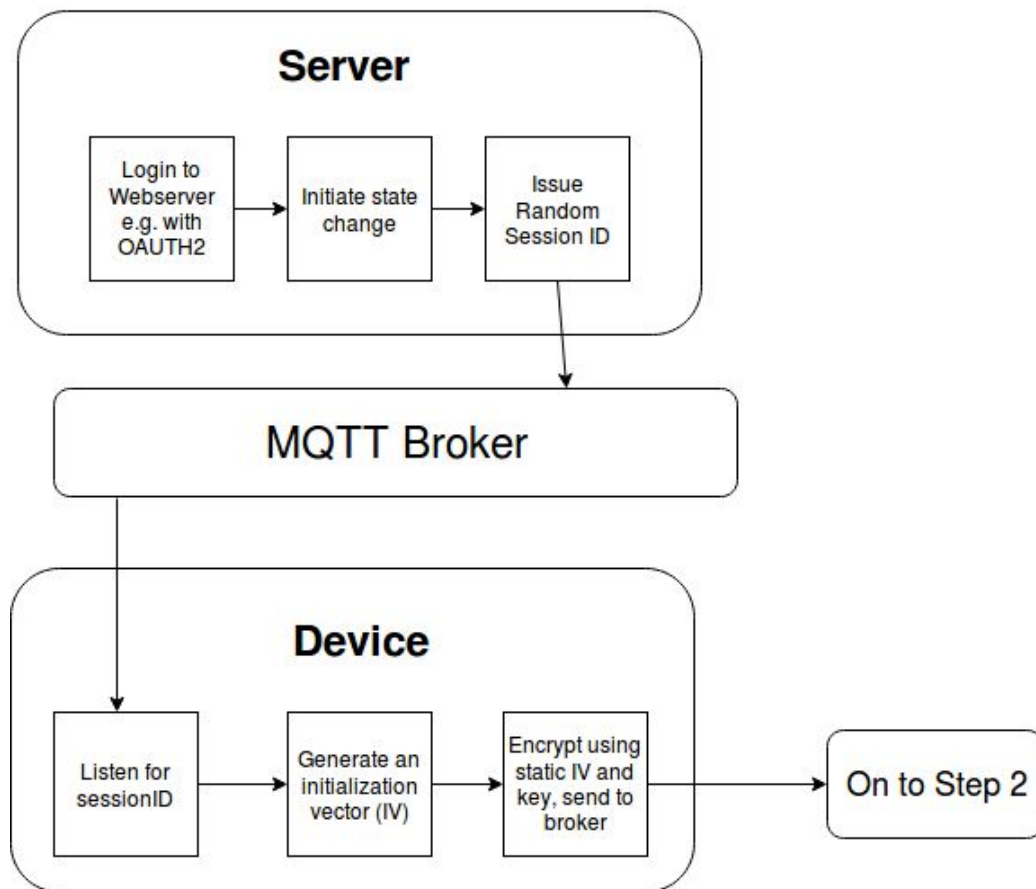
Cloud to Device (high security) -- Remote Kill Switch

What if we need a switch we can control from the Internet?

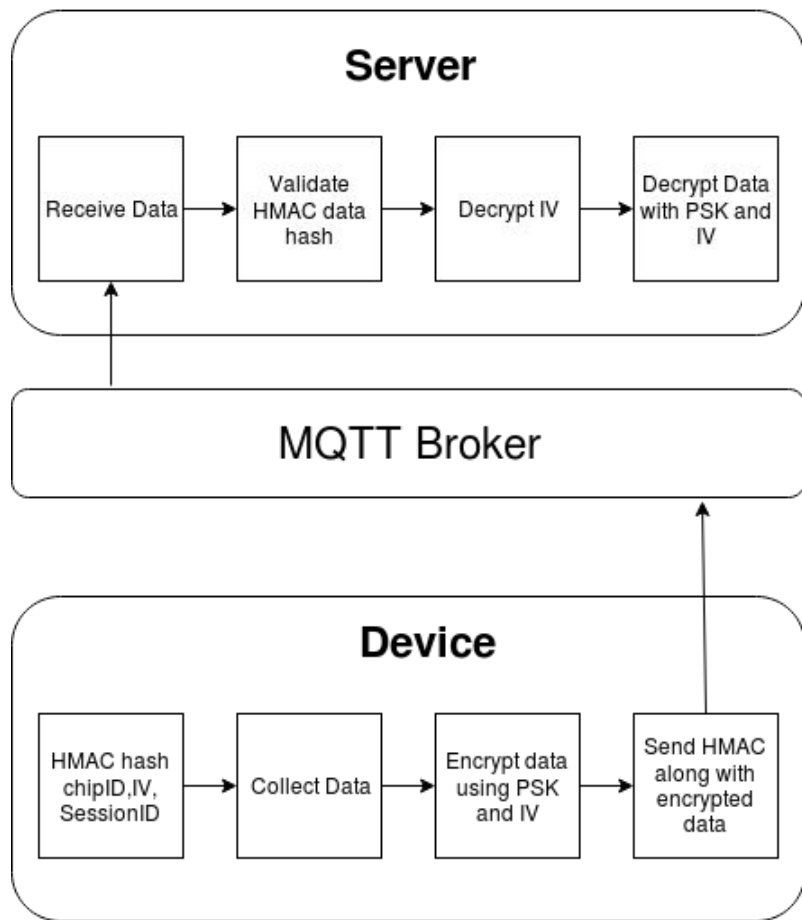
There are other people on the Internet, so now we need security. How would we do that?



Step 1: Initialization



Step 2: Data Transfer



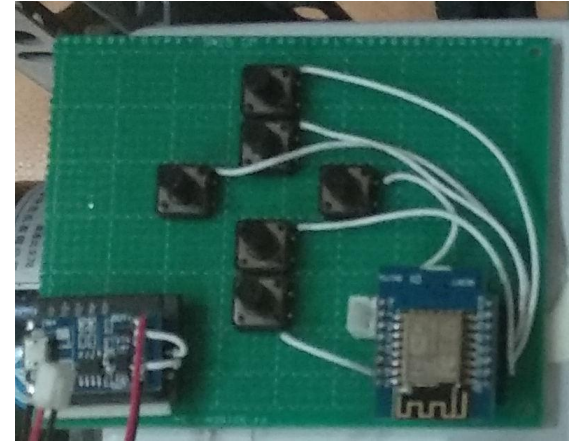
Device to Device -- Robotics Control System

Sometimes all we need is a remote control for something else in the room.

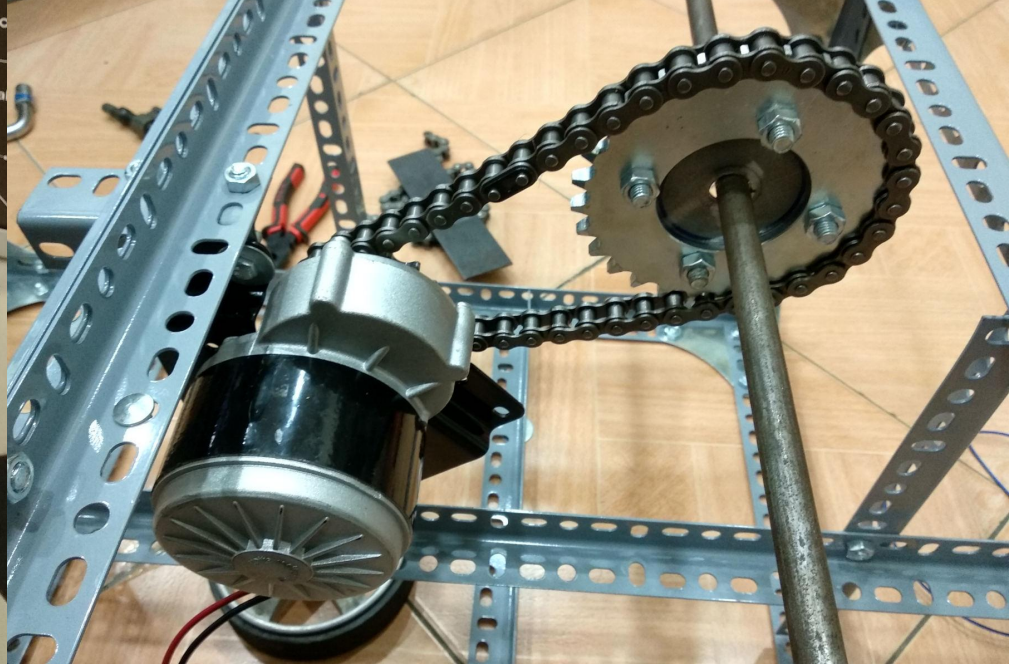
Sometimes that thing is a huge robot that can crush you, and you need to drive it around a room full of people.

This is where we think of device failure states, e.g. in case of an error or accident, we plan out how the system must react.

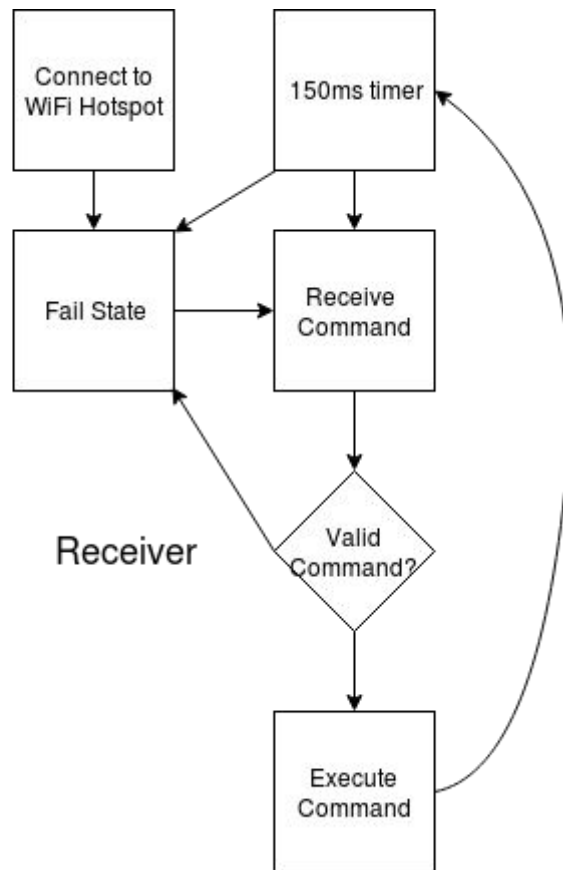
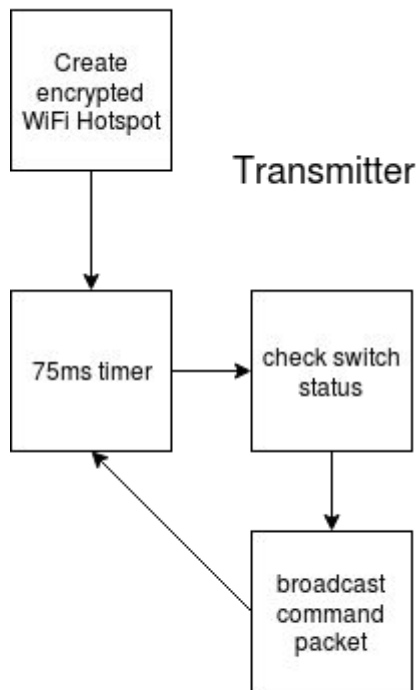
The simplest example is a 'deadman's switch'.



What Robot? H.E.M.

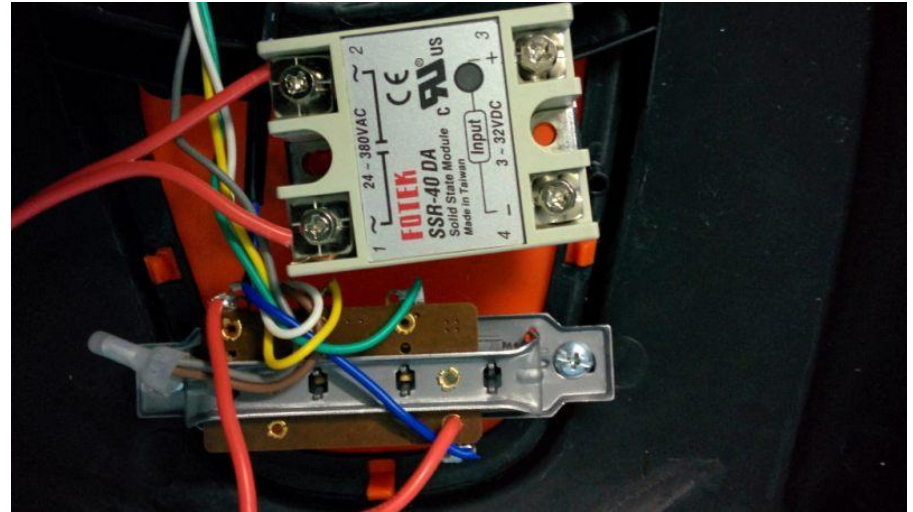


Program Flow

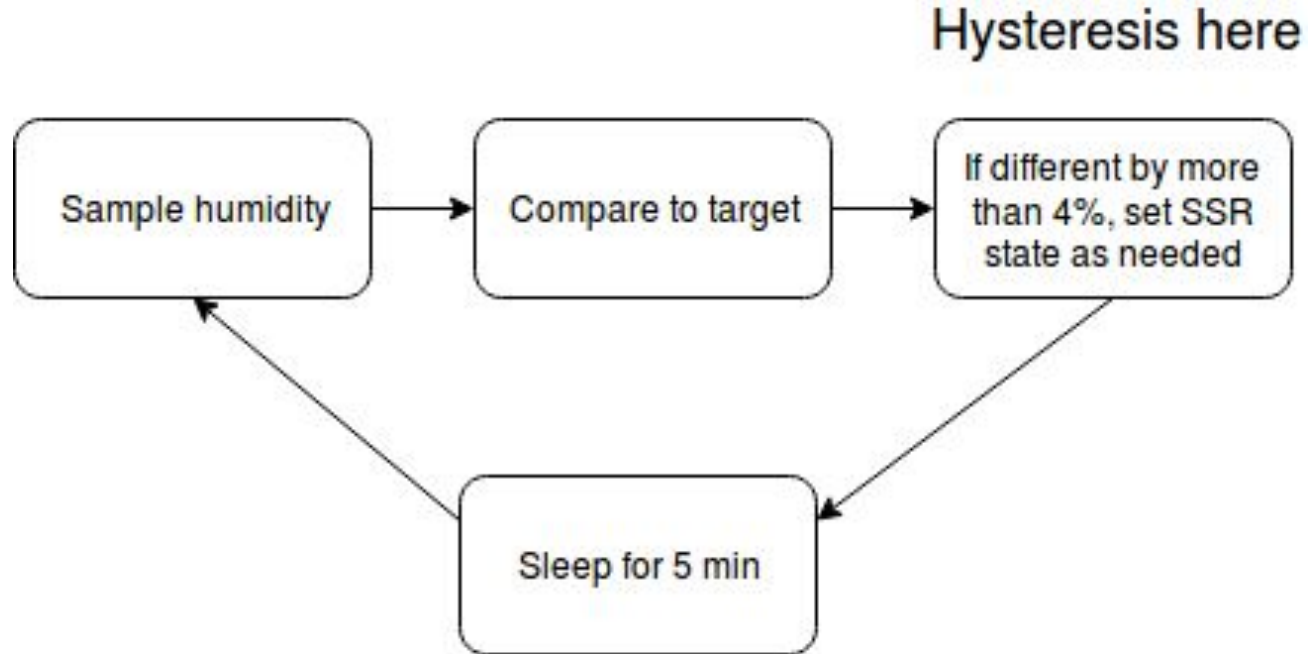


Device to Device -- Humidistat Fan

The IoT chips are pretty good microcontrollers in their own right -- it's OK to turn off the WiFi and just use the chip.



Program Flow -- Hunting Problem!





Conclusions

Consumer IoT devices are mostly terrible and security is hard -- why learn IoT?

Lesson 1: It doesn't matter if we automate our homes, because our homes don't produce anything and there are limited opportunities for labor saving. IoT is for the workplace where human time is most valuable.

Lesson 2: Data is the new oil, except it's renewable, carbon neutral, and costs nothing to transport. IoT is the best way to collect a lot of data. Also AI burns a lot of data-oil.

Lesson 3: The two most important skills an engineer can have for the next decade are automation and communication with stakeholders. IoT is at the center of both.

IoT devices are only VND 100.000, support Arduino and other frameworks, and are a key part of tomorrow's most exciting technologies.



**Build Everything.
Tell Everyone.**

