Lessons learned after developing 10+ IoT apps

For hardware developers

COFFEE IT

coffeeit.nl | contact@coffeeit.nl | +31 30 737 1093

Lessons learned after developing 10+ IoT apps

As you’re the hardware expert, you’ll know everything what it takes to make a beautiful product. As a software expert, we at Coffee IT know all about creating an app for hardware product. We’ve seen many pitfalls, have been through difficulties, but have learned from all of them. Learnings that help you and us with future projects.

Coffee IT has developed more than ten IoT apps in the past nine years. We reveal **six of the biggest hurdles in developing an app for your hardware.**

Read further to discover our findings.

Learnings

01 Choosing the right connectivity/ radios is crucial

02 Have a dev/test-kit ready as soon as humanly possible

03 Do you really need Cloud?

04 Be ready to ship hardware to Apple for the app-review

05 From a UX perspective the most optimized hardware might not be the best

06 What controls the system?

01 Choosing the right connectivity/radios is crucial

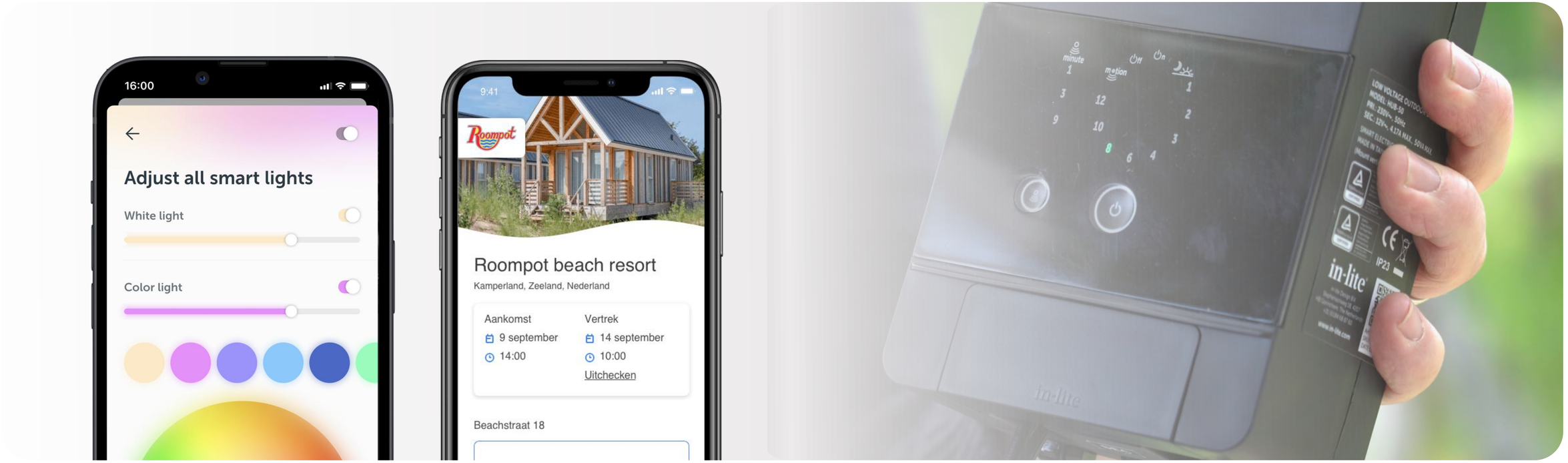
In the earliest stages of the project, choices have to be made. One of those choices is choosing a connectivity-stack, a tough choice to make as this heavily impacts the costs of production. There are many stacks to choose from, most common are Bluetooth, Wifi, Zigbee, Thread (Matter) and cellular. It’s good practice to have an early look together with the software developer to check if the chosen stack makes sense, there is a lot that software can fix, but there are always things it can’t. Every connectivity form has pros and cons. Therefore, it's important to let the software developer check in from a mobile perspective. For example, the in-lite stack with ‘just’ Bluetooth perhaps wasn’t the logical choice for a mesh network, but it did work out in the end and now they have super scalable mesh network, with over 300 meters of range.

Client: In-Lite Beheer B.V.

02 Have a dev/test-kit ready as soon as humanly possible

To help software developers build for your hardware, it's essential to give them a dev-kit while your soldering work is still cooling down. The earlier the better, that way the software developer can start creating a PoC (Proof of Concept). Usually we start on one platform (either iOS or Android) to explore if everything that was envisioned is possible.

At this stage it's still easy to make changes to the hardware, firmware or API. As soon as the PoC is completed, the hardware and software is both ready to take it to the next level, UX and design of the actual user-facing app.



03 Do you really need Cloud?

While the Cloud brings wonderful opportunities and some see it as the holy grail of IoT, there is a (growing) group of users who actually want to keep their data stored local. And that group has some good reasons, with security breaches being one of the biggest. IoT devices are able to know a lot about its user and can obtain very private information, so complying with GDPR guidelines can become a headache (but very much needed when going through Cloud). Keeping the data connection local will make things easier, both for you and the end-user, because you still want to control your devices if the internet is not working? Might be handy with a smart oven.

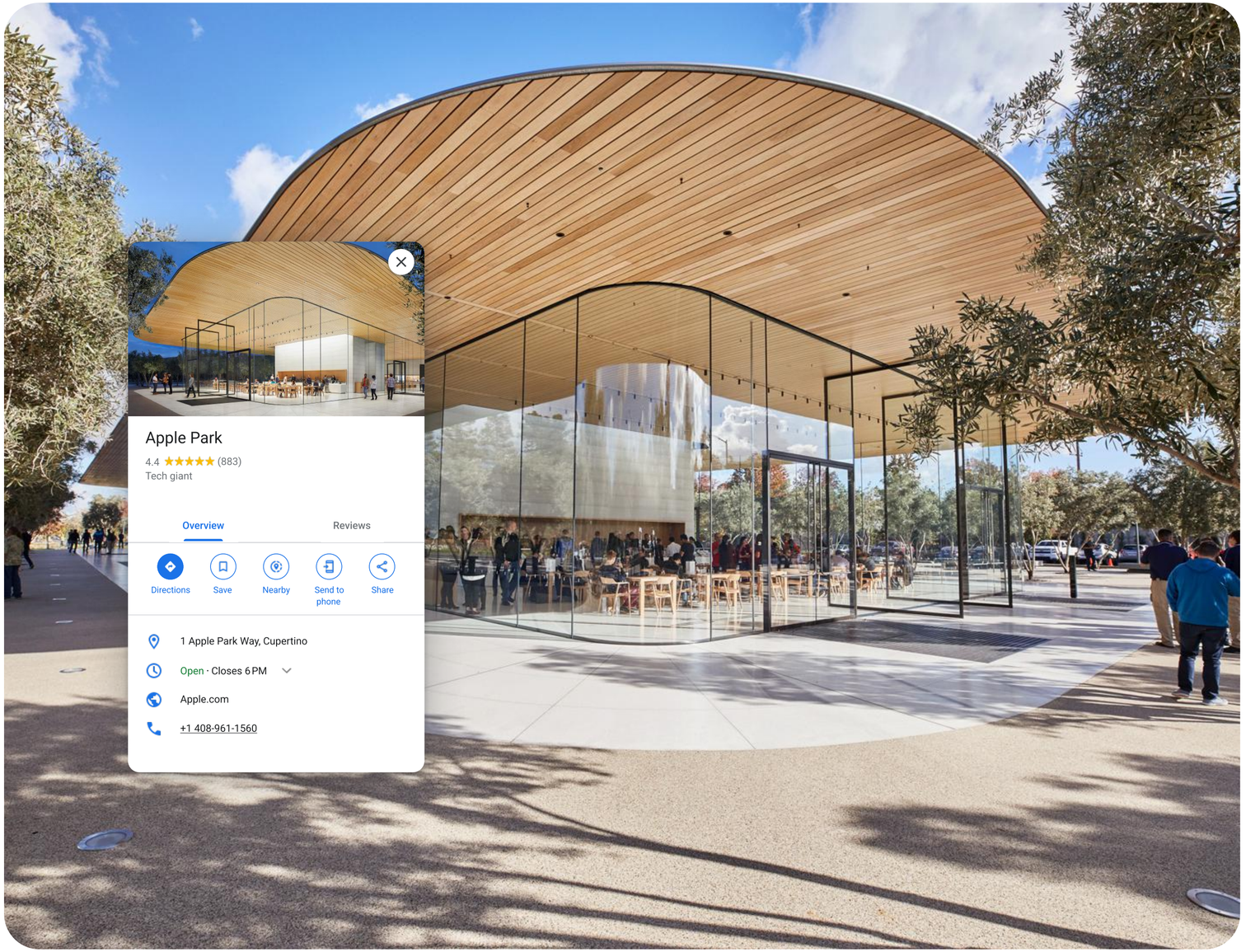
If there is a want/need for releasing an app in China (as one might want, since it's a huge market), keep in mind that Google doesn't exist there. As many choose Google Cloud/Firebase for Cloud based connection, this will become troublesome when releasing in China. Therefore, a local connection between app and device(s) will assure that it will connect no matter where the user and device(s) are located.

If Cloud is definitely needed, choosing a good Cloud provider (such as Google Cloud/Firebase, Amazon's AWS or Aliyun) for your needs is crucial.

04 Be ready to ship hardware to Apple for the app-review

Apple likes to have total control over whatever happens in their App Store, so every app (even the app updates) go through a manual review process. About 25% of all apps submitted are rejected.

A reviewer needs to be able to use and test the app fully, with the app's functionality being dependent on the hardware this can slow things down. Once a reviewer encounters any issue it will stop and report back, if this happens a new app needs to be submitted and the whole process starts over. Not being able to test the app fully due to missing hardware is a reason to stop. This will result in a two day delay (probably longer), to keep that from happening it's a good idea once the app is nearing completion and gearing up for release to the App Store to send hardware to Cupertino in advance.



05 From a UX perspective the most optimized hardware might not be the best

Optimizing the sh*t out of things in your hardware makes it very efficient and awesomely engineered, however this can take a wrong turn for the user experience (UX). This is best explained with an example.

Let's say you're building a smart garage door to open and close the door of the garage automatically. To optimize hardware, the choice is made to only have a Wifi chip to keep the PCB design simple and cheap. What this means for the UX of the app is that users have to connect to a Wifi network that the device is starting for setting the connection up at onboarding, so the user has to switch Wifi networks and they're losing internet connection temporarily. Onboarding is one of the most important parts as that is the first time users start using the app and product, 45 percent of uninstalls within 30 days occur within the first 24 hours, indicating that the seamless onboarding process is crucial. As you can imagine this onboarding/setup process is quite tedious (specially for a-technical people). This can be easily solved by adding a Bluetooth chip for discovery and setup, user scans for the device and connects to it all done within the app.

06 What controls the system?

Last, but certainly not least.

It’s good practice to have a good thorough strategy made before working on your hardware, is a connection directly to the device in question enough? Or should it be able to run a routine? Or should it be able to control multiple devices simultaneously? These are the questions that lead to the bigger question, is there a need for a so-called ‘hub’?

With a hub between app and device, allows for running automations with certain triggers e.g. turn light on at sundown while the app is not running. Knowing this before starting to work on an app is a huge help, since the UX of the app will be totally different.

Any questions?

Let’s connect!

Reach me directly at contact@coffeeit.nl or
+31 6 80 22 60 64.
Or call our Utrecht office on +31 30 737 1093.

COFFEE IT