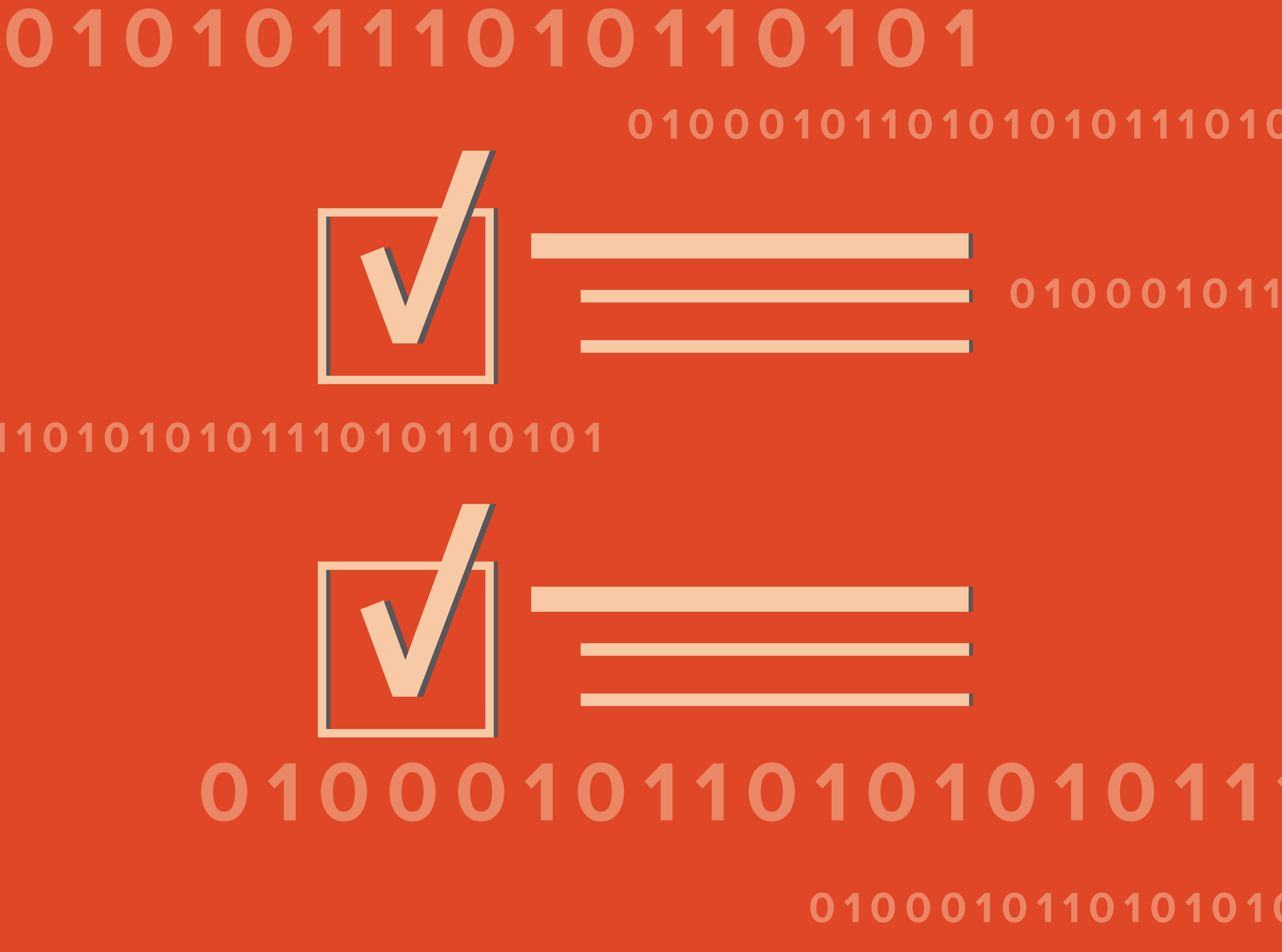
9 TIPS FOR EMPLOYING THE INTERNET OF THINGS IN BUILDINGS



Deep Thoughts by Deepinder Singh

A series on the future of cloud computing, big data and buildings

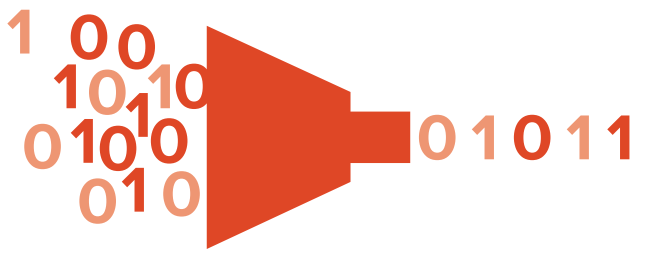
My motto in every endeavor is to simplify operational complexity and make products intuitive. Today this remains at the heart of my values and in the spirit of Internet of Things (IoT) month, I've chosen to address these topics holistically. The creation of intuitive products and the deployment of IoT have the opportunity to work flawlessly in unison, if executed and engineered properly. Employing IoT does not require a product to be complex on a consumer level. In fact, if executed mindfully it can open windows to a wealth of knowledge and efficiency without overwhelming the user. My goal here is to shed light on some important factors when considering IoT.

1. PINPOINT THE PROBLEM YOU ARE TRYING TO SOLVE.

When considering IoT as a business tool, consider if it is appropriate to use IoT by asking if it will provide a value to the end consumer. Once you've determined IoT will help you reach your goal and benefit your consumer, it's important to have a plan that, if deployed, is going to scale. IoT is sometimes confused with the early prototyping stage. For example, individuals use platforms such as Arduino or Particle.io (previously Spark.io) and find they are able to turn on a tie rack remotely using a phone app. That's great, but if one were to market this idea, they would be doing a disservice to their end consumer by increasing the complexity in that customer's life. While it's exciting to pull out your phone and be able to automate something, it's crucial to ask if there is true value in employing IoT. Becoming enamored with IoT for the sake of IoT is something to be cognizant of, so keep your eye on the problem you are solving.

2. ENSURE YOUR DEVICES GENERATE DATA, NOT NOISE.

It has taken years for the car industry to become more robust. For example, in most cars, you'll find many sensors. A common issue in any model occurs when your check engine light continuously comes on. You respond by calling your service tech who says, "As long as it runs, ignore it." This response defeats the purpose of the sensor in the first place. On that same note, when was the last time you heard a car alarm going off? Honk, honk, honk. Besides perhaps looking in that direction, did anyone attempt to do anything about it? Even if someone appears to be struggling near the alarm, it's unlikely that anyone takes action. In fact, if the car alarm going off belongs to you, then you instinctively grab the keys and click, click, turn it off before going out to even check on the problem. The false positives are simply so high that we have become attuned to ignoring the alarm, which then becomes more of a nuisance than a value.



The key here is to build in reliability and also ensure you obtain the tools to sanitize the data, because real world data is noisy and ugly. It's important to have methods that exclude any outliers and if you need to investigate those outliers, make sure that you have a way to do so easily. This may require the ability to drill down to specific subsets. If the system is too ubiquitous then it needs to be resilient and you have to filter out the noise! The more sensors that you have, the more chances of noise that incur. Part of the filtering process comes with understanding the problem you are solving, because only if you have that knowledge can you pinpoint the typical range of values that you would see. When looking at the sensor data coming in, you can constrain and sanitize it so the noise is automatically filtered out. This will prevent unnecessary alarms from deploying in the first place.

3. THE MORE INSTRUMENTATION YOU INSTALL, THE LESS RELIABLE YOUR SYSTEM BECOMES.

There's a concept called mean time between failures (MTBF). Much like it sounds, MTBF is simply the average time from one failure to the next. The MTBF of all the individual components contributes to the MTBF of the overall system. Say you have two sensors which have a failure rate of 100,000 hours, meaning that after 100,000 hours they are expected to fail. Now let's say you have ten of those same sensors in your solution. Now the failure rate becomes one in 10,000, because each of those sensors could fail causing the entire system to fail.

Essentially, you've decreased the reliability of the system ten-fold by installing all that instrumentation. With this in mind, it's imperative that if you're considering deploying thousands to millions of IoT devices for solving your problems, you need to ensure that the MTBF of each IoT device is so high that the overall system MTBF remains acceptable. This means that each device has to fail not in 100,000 hours, but in one million hours and this is a hard problem. At the end of the day, don't over engineer products and ensure your system is reliable.

A good example of well-engineered devices comes from the general aviation industry, which handles planes for private individuals. Most of these planes use very simple engines that were developed in the 1950s with no computers. The engine manufacturers have made the deliberate choice of not adding complex electronic sensors or engine control units, because these would adversely affect reliability. It is after all, really hard to pull over and call a towing service when your engine quits while flying in the sky! The commercial aviation industry needs complex engines and sensors, so they spent hundreds of billions of dollars in research to tackle the problem. Even then, all commercial planes have at least two engines for redundancy.

4. THE MORE DEVICES YOU HAVE, THE LESS OF A USER INTERFACE (UI) YOU WANT TO PROVIDE.

If you have too many interfaces, the user gets confused. People often try to add too many components into simple devices. Now with smart phones being more ubiquitous, if you don't have a high density screen, you should not be putting in a rich user interface. So you really have to figure out where you want the complexity. Do you want it in a single gateway device? Or do you want it in all of the small devices?

You can also have a combination of interface levels, having a complex, but intuitive interface (right) where you need it and a simple interface (left) where you don't.

On the flip side, if you provide too little of an interface, people can end up overloading functions. This can create a poor user experience, leaving users crying for something more intuitive. Let's take pairing your Bluetooth headset, for example. There is just one small LED light and a few buttons. Every time you have to re-pair it, you basically have to open the manual to figure out which one of those buttons does what. If it's blinking too fast, does that mean the device is pairing or has it lost pairing? Does a red blink and a blue blink mean the battery is low or is the device out of range?

This is a case of trying to oversimplify things or cut costs. If you truly need to convey that information, then you need a richer user interface. It would be so much easier if you had a small LCD display that scrolled, "press pairing button," so you didn't have to go back and reference the user guide, which you may or may not have kept at this point! If it's simple, then it doesn't need anything at all. But if it's complicated, it would behoove you to have a display indicating what is happening.

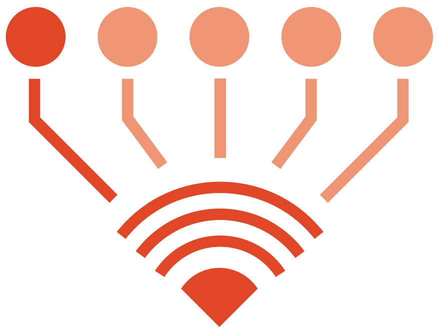
5. HAVE A WAY OF MANAGING YOUR DEVICES.

On its own, IoT is very broad, as it expands to include any device which can connect to the internet. In today's world, you can have billions and billions of IoT devices, but that also means that there are billions of components that have the potential to go wrong. Therefore, it's important that you have a way of managing these devices. Device management can range from monitoring the power status, or network signal strength, to actually pushing firmware updates remotely. Remote firmware updates are a critical piece because they allow systems to be patched as security vulnerabilities or other bugs to be found. It would really be a bit too utopian to say that a system will never have bug – hackers over a long history of computing devices have proven otherwise!

6. CONSIDER SELF-CONFIGURING IOT DEVICES.

One of the most common methodologies for IoT is to have the end device connect directly to Wi-Fi. That is, however, by far the most expensive solution since it requires each end device to have a full Wi-Fi stack. Or you often end up employing small aggregation networks like Bluetooth BLE 4.0 or ZigBee. These networks require a gateway which communicates data out to the internet. In each of these cases, the devices are not self-configuring, so you have to provide a simple way of reconfiguring them in an intuitive manner.

For example, consider the Amazon *Dash* button that connects to your Wi-Fi and allows you to order laundry detergent, paper towels, or a number of other pre-set products via Amazon Prime at the push of a button. This is a good example of an interface that doesn't normally need to be repaired or set up. Or if you have a wireless thermostat like the Honeywell *Lyric*, it will go through the pairing process once because it is meant to be paired out of the box. But if you change your Wi-Fi password (a similar issue exists if the gateway device for a Zigbee or other aggregation network needs to be replaced) or you change to a new Wi-Fi router with another security protocol, then you're likely headed for a lot of consternation.



By then, you've thrown out the user manuals and frustration quickly ensues, as the new device infatuation has worn off. Now you have to go through an arcane process to figure out how to reconfigure your devices to communicate with the new Wi-FI router again. You will likely have a bunch of these devices and you have to reconnect every one, without having a rich enough user interface. This is a seriously tedious process. It was one thing to do this one at a time when you were setting these things up, but now this single change has impacted a multitude of devices.

One solution is to have self configuring IoT devices that connect to a gateway out of the box. The gateway can provide a rich interface (maybe via a touchscreen) to allow easy changes to system parameters such as the Wi-Fi connection.

7. ENSURE YOUR PRODUCT CAN SCALE.

It may be easiest to use third party platforms which accelerate adoption in the early stages. But at some point, a company has to solve the hard problem of taking it in-house to ensure that the system runs efficiently and can scale. That is not to say that we shouldn't be using platforms. They can be quite valuable, but we need to be cognizant that platforms are not necessarily what will get companies to scale when they truly have billions of devices.

A good example is Facebook. They used a lot of scripting languages in the beginning. As a result, the company has had to completely rework their backend a number of times. So even while they preserve the look, the backend has been developed without scaling in mind.

8. WHEN LEVERAGING CLOUD COMPUTING, HAVE A BACKUP PLAN FOR LOCAL INTELLIGENCE.

Cloud computing enables storing and accessing data via the internet. It's a great tool to use, but beware of relying solely on the cloud.

The flip side of cloud computing is that some have taken this to the extreme, where every decision has to be made on the cloud and then sent back to each device. This can result in a lot of things going wrong. Basically, the entire data chain has to be perfect before you can accomplish even the most trivial of tasks. Ultimately, IoT devices need the ability to make localized decisions. They must to be able to function when wireless connectivity goes down, because as we all know this can happen!

Surprisingly, many companies have no backup plan, meaning they rely solely on getting instruction back from the cloud. Millions of people use web based logic from IFTTT (https://ifttt.com/) to automate their lives, like opening their garage doors when they arrive at the house. It sounds cool and it works great, but sadly, something this simple requires the use of a smartphone with a GPS triangulation connected to the internet via cellular data. The garage widget connects to the home Wi-Fi router, the home Wi-Fi router connects to the Internet Service Provider (ISP). All these connections depend on the home ISP service working, the garage widget companies' servers working and IFTTT servers working. If any of the elements in the chain are broken, then you'd better have your garage door remote handy. For this reason, having a fail-safe plan that will allow the system to degrade gracefully is a must.

9. CREATE AN END-TO-END SOLUTION.

As it stands today, IoT basically provides just the tools of gathering big data. While it's excellent progress to have access to so many data points, what good is it if you don't have the ability to efficiently analyze this big data? Therefore, on its own, IoT is somewhat overrated and it's much the same for cloud computing. While storing great amounts of data and information on the cloud is useful, it's important to think about how to best leverage this tool to increase efficiencies and improve solutions.

Interestingly enough, when you pair IoT with cloud computing, something truly beautiful happens. You end up with massive amounts of aggregated data, without having to manage the servers yourself! So while we sometimes treat IoT and cloud computing as separate entities, they really belong together. In fact, the true value chain lies in having an end-to-end solution. This includes overseeing the hardware, software, managements tools and the analytics at the backend. I think the winners are going to be those who figure out a core problem and employ IoT in a pointed fashion.

ABOUT THE AUTHOR

Deepinder Singh founded 75F in 2012 after he designed some of the world's fastest core networks for Tier 1 service providers like AT&T, NTT and Verizon. With almost 25 years experience in electronics and computing, he's brought a wealth of embedded products to the market. His key goal in every endeavor is to simplify operational complexity and make products intuitive.

That's why he created 75F, an intelligent building solution that utilizes the Internet of Things and the latest in cloud computing to create systems that predict, monitor and manage the HVAC needs of light commercial buildings.