My IoT-button: de knop voor het net

deel 1: IoT-architectuur

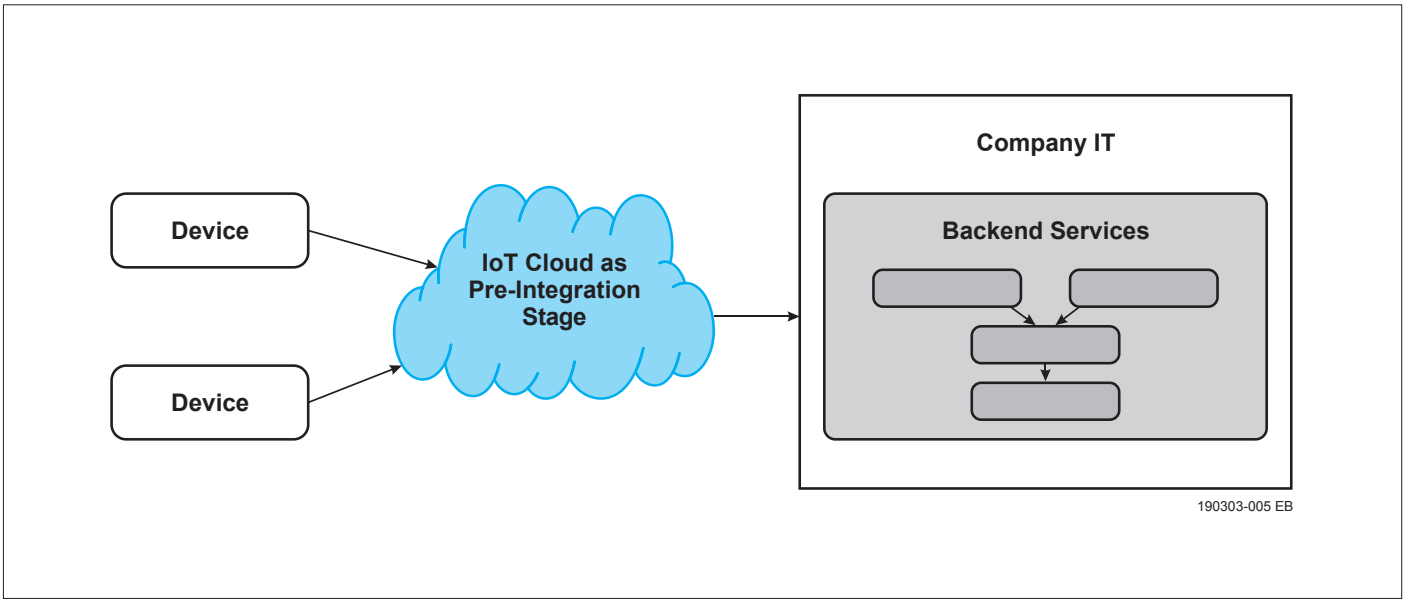
Dr. Veikko Krypczyk (Duitsland)

Er komen steeds weer nieuwe toepassingen rondom het Internet of Things (IoT). Eén daarvan is de IoT-button, een drukknop die naar eigen inzicht kan worden geprogrammeerd en gebruikt. Hij heeft geen netwerkverbinding nodig en haalt zijn intelligentie uit de cloud. In het eerste deel van deze artikelserie bespreken we de IoT-architectuur op basis van *serverless cloud computing* en het samenspel tussen soft- en hardware.

Steeds weer krijgen we ideeën voor nieuwe toepassingen met het Internet of Things (IoT). Veel ideeën zijn experimenteel en/of zullen wel niet snel hun weg naar de echte IoT-wereld vinden. Het visioen van de zichzelf vullende koelkast strandt waarschijnlijk al vanwege het feit dat we toch het liefste zelf die levensmiddelen kopen waar we trek in hebben en dan liefst bij een leverancier waar we ons prettig bij voelen. Een interessant idee, dat ergens op de grens tussen een serieuze toepassing en een speeltje zweeft, is de zogenaamde IoT-button. Een IoT-button kan bijvoorbeeld:

- iets starten of stoppen;
- iets bestellen;
- iemand opbellen;
- iets tellen.

Er zijn talloze toepassingen en die kunnen zowel in het privé- als in het zakelijke domein liggen. Het idee is om met hardware in de vorm van een drukknop (button) een signaal te versturen. Waar dat signaal naartoe gaat, welke actie ermee samenhangt en welk systeem eraan gekoppeld is, dat hoeven we in dit stadium helemaal niet vast te leggen. En ook niet of het om één drukknop gaat



Figuur 1: Basisarchitectuur van een IoT-toepassing met een cloud-backend.

of om meerdere.
Voorlopig pakken we dit thema op om moderne IoT-architectuur te bekijken en te bespreken. In dit eerste deel gaat het over het samenspel tussen de verschillende componenten en de mogelijkheden van clouddiensten. Het is gemakkelijk om met het IoT te experimenteren en dat geldt ook voor het gebruik van clouddiensten. Met (minimale) hardware, een klein beetje software en wat configuratie gaan we in het tweede deel onze eerste experimenten uitvoeren en daarmee de voorwaarden scheppen om eigen IoT-ideeën uit te proberen.

IoT-architectuur

De architectuur van een moderne IoT-oplossing ziet er, onafhankelijk van het toepassingsgebied en de gekozen provider, ongeveer uit zoals in **figuur 1**. De individuele apparaten (*devices*) aan de linkerkant kunnen sensoren of actuatoren zijn, of combinaties daarvan. Sensoren meten bijvoorbeeld omgevingsparameters zoals de temperatuur, ze bewaken het functioneren van een apparaat, of ze sturen een signaal als de toestand van een systeem verandert. Actuatoren voeren acties uit, ze openen of sluiten bijvoorbeeld het ventiel van een gasbrander. In sommige IoT-toepassingen komen sensoren en actuatoren voor die samen als een soort regelkring werken, onder besturing van een toepassing op afstand. In andere IoT-toepassingen gaat het alleen maar om registratie en bewaking van een toestand. Onze simpele IoT-button doet niet anders dan controleren of er op gedrukt wordt.
De informatie dat de toestand van de button is veranderd, wordt overgedragen via een netwerk (meestal het internet). De ontvanger van de signalen is een dienst op een server. De speciaal voor IoT-toepassingen opgezette diensten noemen we bij elkaar de IoT-cloud. Deze IoT-cloud speelt een centrale rol, Hij werkt als een server en is dus de communicatiepartner voor de IoT-devices. Omdat hier alleen gebruik wordt gemaakt van geselecteerde functies en niet van een complete server, spreek men van *Serverless Computing*. Natuurlijk draaien de diensten op de server van een provider, maar met de inrichting, de administratie en dat soort dingen hoeven we ons niet bezig te houden. Bij een grotere belasting, bijvoorbeeld als er heel veel verkeer naar de dienst gaat, worden extra resources beschikbaar gemaakt (*horizontal scaling*). Ook daar hoeven wij als gebruikers ons niet druk over te maken. Centrale functies van de IoT-cloud zijn:

- Veilige communicatie: biedt een veilig communicatiekanaal voor de uitwisseling van data met de IoT-apparaten.
- Doorgeleiding van de data: sommige gegevens kunnen meteen in de IoT-cloud worden verwerkt, andere moeten worden doorgestuurd naar verschillende diensten voor verdere verwerking. In de afbeelding zijn deze diensten weergegeven als backend-services van de IT-systemen van een bedrijf. In dit geval zou het om een industriële IoT-oplossing gaan. Eenvoudiger toepassingen sturen de data alleen door naar software die zorg draagt voor evaluatie en visuele weergave.

Bij de IoT-button moet worden vastgelegd, welke reactie het indrukken van de knop tot gevolg moet hebben. Dat kan bijvoorbeeld zijn dat een ander IoT-device, een actuator, moet worden aangestuurd.

Naast deze taken moet de IoT-cloud zo ‘communicatief’ mogelijk zijn, dat wil zeggen dat hij met veel verschillende apparaten overweg moet kunnen. Hij moet zo flexibel zijn, dat hij programmatuur voor verschillende systemen van de IoT-devices in de aanbieding heeft. In het ideale geval wordt deze interface beschikbaar gemaakt met behulp van gemakkelijk te integreren SDK’s (bibliotheken) in verschillende programmeertalen (bijvoorbeeld C, Python, Java). Ook moet generieke communicatie via een REST-interface (zie **tekstkader** “Wat is

REST?") mogelijk zijn. De IoT-cloud moet ook verschillende protocollen, bijvoorbeeld HTTPS, AMQP of MQTT ondersteunen voor de communicatie met de devices.

IoT-cloudservices

De grote cloud-aanbieders stellen ook functies beschikbaar die zijn aangepast aan de specifieke eisen van het IoT. Marktleiders op het gebied van IoT-cloudoplossingen zijn de platformen van Windows Azure, Amazon Web Services, Q-Loud en Oracle Cloud. De keuze van het juiste platform is niet eenvoudig en is sterk afhankelijk van de toepassing [1]. De volgende criteria moeten bij de keuze in overweging worden genomen:

- Platform-functies: welke functionaliteit wordt er precies aangeboden? Denk aan device-management, authenticatie en autorisatie en doorsturen van gegevens.
- Platform-eigenschappen: dat zijn aspecten als databescherming, versleuteling en gebruikersinterface (dashboard).
- Koppeling: ondersteunde programmeertalen, SDK's en interfaces.
- Overig: bijvoorbeeld community en documentatie, ondersteuning, kosten en prijsmodellen.

De meeste IoT-cloudservices kunnen in beperkte mate gratis worden uitgeprobeerd. Op de internetsites van de aanbieders zijn meestal gedocumenteerde voorbeelden te vinden in de vorm van broncode, compleet met informatie over de inrichting (configuratie).

IoT-device en koppeling

De verbinding tussen de IoT-devices en de clouddienst loopt via het openbare internet. Daarbij maakt het niet veel uit op welke manier de verbinding tot stand komt. Naast een directe, bedrade verbinding (LAN) van stationaire devices kan de koppeling ook draadloos via het lokale netwerk (WLAN) of via het GSM-netwerk (LTE) worden gerealiseerd; dat is afhankelijk van de toepassing, de beschikbare infrastructuur, de gekozen hardware en het energieverbruik. Bij onze IoT-button gaat we uit van een zo simpel mogelijke integratie in een lokaal beschikbaar WLAN. Omdat de IoT-button flexibel inzetbaar en de behuizing compact (bijvoorbeeld in de vorm van een lichtschakelaar) moet zijn, zijn er duidelijke eisen voor wat betreft de hardware. Alleen hardwareplatforms (boards) met een eenvoudige WLAN-connectiviteit en een zo gering mogelijk energieverbruik, vooral in standby-modus, komen dus in aanmerking. Op die aspecten gaan we in het tweede deel nog nader in, als we prototypes gaan bouwen en gaan experimenteren.
Als die beperkingen qua grootte, voeding en draadloze koppeling er niet zijn, dan is er veel meer keus in het te gebruiken platform. Als de IoT-button bijvoorbeeld op een vaste plek komt te staan, dan kan hij uit het lichtnet worden gevoed, het energieverbruik in standby-modus is dan minder belangrijk en een bedrade LAN-verbinding is ook mogelijk.

Aanbieders van IoT-clouds

Nu moeten we nog een aanbieder van IoT-cloudservices kiezen. De dienst *Azure IoT Hub* van Microsoft [2] is bedoeld voor bidirectionele communicatie tussen IoT-apparaten en het cloudportaal Azure. Het is een cloud-hosted oplossings-backend, waar allerlei apparaten aan kunnen worden gekoppeld. De integratie van de devices werkt met SDK's, die voor verschillende platformen en programmeertalen beschikbaar zijn. Ook biedt de IoT-hub functies voor het beheer van de aangesloten apparaten. Daarbij heeft elk apparaat een eigen identiteit met inloggegevens of certificaten, elk apparaat kan onafhankelijk van de andere via het dashboard rechtstreeks in de browser worden geactiveerd of gedeactiveerd. Deze functionaliteit is ook beschikbaar via API's, zodat ontwikkelaars de IoT-devices in eigen applicaties kunnen beheren. De online documentatie bevat uitgebreide stap-voor-stap-handleidingen: onder [3] wordt bijvoorbeeld beschreven, hoe we een Raspberry Pi met een IoT-hub kunnen koppelen en data van het device kunnen ontvangen. Het gebruik van de Azure IoT-hubs is zowel voor prototyping als voor professioneel en privégebruik gratis, zolang een beperking in het aantal te verwerken calls geen probleem is. De abonnementsvorm *Standard* is kosteloos, mits er niet meer dan 8.000 berichten per dag worden uitgewisseld en de berichten per stuk niet groter zijn dan 0,5 kB.
Een vergelijkbare dienst wordt door Google aangeboden onder de naam *Cloud IoT Core* [4]. Het gaat om een volledig beheerde dienst, waarmee we devices via het internet kunnen beheren en data kunnen uitwisselen. Cloud IoT Core kan bovendien samenwerken met andere

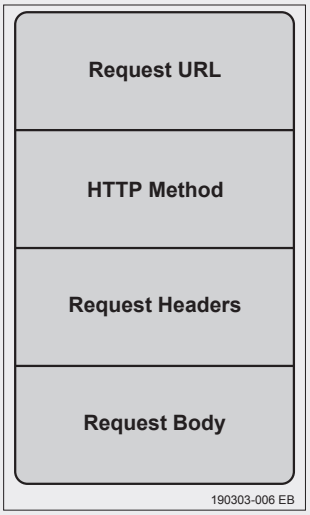
cloudservices van Google, bijvoorbeeld met Google Big Data Analytics en ML-diensten zoals Dataflow, BigQuery, BigTable, ML, Data Studio of BI-tools van partners. Daardoor is efficiënte evaluatie, verwerking en visualisatie van de IoT-data in real time mogelijk. De dienst ondersteunt de gebruikelijke MQTT- en HTTP-protocollen, zodat er goede compatibiliteit is met veel devices. Een kernfunctie van Cloud IoT is de geïntegreerde apparaatmanager. De beheerde devices kunnen vanuit een console worden beheerd, ook programmatische besturing is mogelijk. De apparaatmanager bepaalt de identiteit van een apparaat en zorgt voor eenduidige authenticatie. De veiligheid van de datatransfer wordt gegarandeerd met behulp van *end to end encryption*. Cloud IoT Core is technisch gezien ook een serverloze dienst, die bij groei van het dataverkeer naadloos horizontaal opschaalt. Het systeem is gebaseerd op een REST-interface, zodat het vergaand onafhankelijk is van de systemen van de devices. De kosten van de dienst zijn afhankelijk van het gebruik en zijn gestaffeld. Tot een maandelijks datavolume van 250 MB is Cloud IoT Core van Google gratis te gebruiken. Zodoende is deze dienst heel geschikt voor experimenten, prototyping en private of kleine commerciële projecten. Vergelijkbare diensten, maar vrijwel alleen voor industriële klanten, bieden de IoT Cloud Services van Oracle [5], AWS IoT Services van Amazon [6] en het platform Q-Loud IoT [7].

Ook het aanbod van *ThingsBoard* [8] mag hier niet onvermeld blijven. Zij bieden een Open Source-cloudplatform voor het IoT. Het gebruik ervan is volledig gratis, als u de software installeert op uw eigen computer. In feite installeert u een eigen IoT-backend, dat noemen ze *"On Premise"*: U bent zelf verantwoordelijk voor installatie, beheer en beschikbaarheid. Interessant is de verscheidenheid aan ondersteunde systemen, Het geheel loopt onder meer op diverse Linux-distributies (Ubuntu, CentOS, Red Hat), op Windows of op een Raspberry Pi 3. Dat laatste kan juist voor maker- en community-projecten en voor prototyping interessant zijn. De software van ThingsBoard wordt ook aangeboden als een beheerde cloudoplossing, waarbij de kosten gestaffeld zijn afhankelijk van de hoeveelheid dataverkeer, de tarieven beginnen bij $10 per maand. De functionaliteit van deze server- of clouddienst omvat de typische taken van een IoT-backend, dus beheer van apparaten,

Wat is REST?

Als client en server via het netwerk (internet) met elkaar communiceren, dan wordt er meestal gebruik gemaakt van een REST-interface [9][10]. REST is een afkorting van *REpresentational State Transfer*. RESTful API's werken met gestandaardiseerde protocollen, zoals HTTP/S, URI, JSON of XML. Hierbij gelden de volgende principes:

- Client/server-model: de communicatie verloopt op basis van een client/server-model. Het doel is een flexibel en generiek gebruik van de diensten over de grenzen van de platforms heen.
- Statelessness: de communicatie is altijd toestandsloos. Elk verzoek van de client aan de server moet volledig zijn. De server kan niet teruggrijpen op data uit voorgaande requests.
- Caching: clients kunnen de antwoorden van de server bewaren (cacheable). Daarmee kan een periode van uitval van het netwerk worden overbrugd en kan tijdelijk offline worden gewerkt. De bewaarde gegevens kunnen dan bij een herhaalde aanvraag worden gebruikt als een alternatief voor een nieuw antwoord van de server.
- Uniformiteit: de services gebruiken een uniforme interface, die ontkoppeld is van de geleverde dienst.



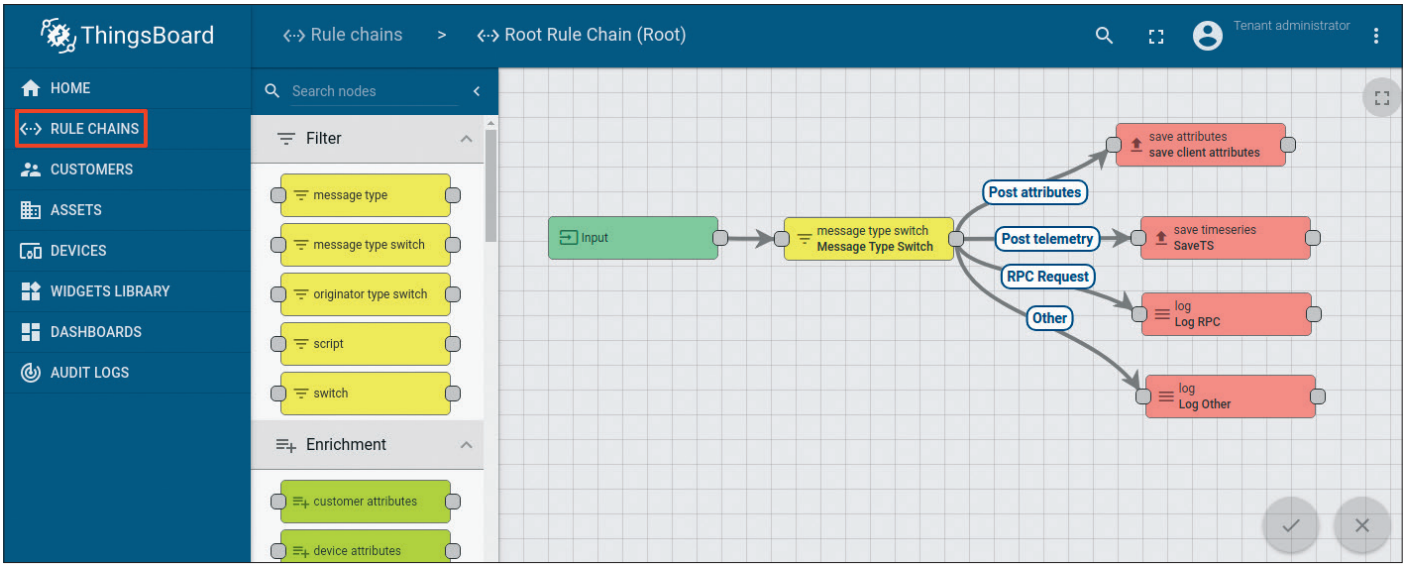
Aufbau einer REST-basierten Anfrage.

Een REST-API wordt over het algemeen met http of https gerealiseerd. De diensten worden benaderd via een URL en http-methoden zoals GET, POST, PUT, waarbij de client een aanvraag (*request*) naar de server stuurt en een antwoord (*response*) terug krijgt. Een aanvraag bestaat uit vier bestanddelen: Eindpunt (*endpoint*), methode (*method*), header (*header*) en data (*data*). Het eindpunt bestaat uit een *root-endpoint*, een *path* en eventuele parameters. Zo is bijvoorbeeld *https://api.github.com* het root-endpoint van GitHub en */users/veikkoef/repos* het pad naar de repository van de auteur. Samen wordt dat *https://api.github.com/users/veikkoef/repos*. Er kunnen nog parameters volgen, bijvoorbeeld *?query1=value1&query2=value2*.

Bij de methoden hebben we keus tussen GET, POST, PUT/PATCH en DELETE. GET staat voor leesoperaties, dus het lezen van data. Omdat GET-requests alleen lezen van de server, kunnen ze onbegrensd worden verzonden. GET is de meest gebruikelijke methode. POST wordt gebruikt voor create-operaties, dus voor het aanmaken van een datastructuur. POST is niet vrij van neveneffecten. Door een POST-request kunnen velden in een databank worden veranderd of er kunnen processen worden gestart op de server. PUT/PATCH wordt gebruikt voor update-operaties en DELETE om gegevens te verwijderen.

Het derde element van een aanvraag is de header. De header dient voor het doorgeven van informatie voor de client en de server. Hij kan voor veel doelen worden ingezet, bijvoorbeeld voor authenticatie en voor het doorgeven van informatie over de inhoud die volgt.

Het laatste element van een aanvraag is de data (Body), die we naar de server willen sturen. De body is alleen relevant voor de operaties POST, PUT, PATCH en DELETE.



Figuur 2: De verwerking van events kan grafisch worden gedefinieerd (bron: ThingsBoard).

verzamelen en visualiseren van data, *event handling* en *remote procedure calls* voor het besturen van de devices. De functies worden ook aangeboden via een REST API en zijn dus systeem- en apparaatneutraal te gebruiken. Met de zogenaamde *rule engine* kan comfortabel op een grafische interface worden gedefinieerd, hoe de binnenkomende events moeten worden verwerkt (**figuur 2**).

Conclusie en vooruitblik

We hebben u een overzicht gegeven van de architectuur van een IoT-oplossing; het centrale bestanddeel daarvan is een clouddienst voor het beheer van IoT-devices en het ontvangen en distribueren van data. In het volgende deel gaan we met die kennis praktisch aan de slag. We bespreken dan het ontwerp van een prototype voor een IoT-device in de vorm van een IoT-button. Ook al zijn dat soort oplossingen al kant-en-klaar te koop, toch is het nuttig om te experimenteren, want dat maakt veel beter duidelijk hoe alles in elkaar grijpt en dat maakt het gemakkelijker om ook eigen oplossingen te bedenken.

(190303-04)

Weblinks

[1] Vergelijking van platformen: www.informatik-aktuell.de/betrieb/virtualisierung/iot-in-der-cloud-erkenntnisse-und-erfahrungen-eines-plattformvergleichs.html
[2] Microsoft Azure: https://azure.microsoft.com/de-de/services/iot-hub/
[3] RPi in Azure: https://docs.microsoft.com/de-de/azure/iot-hub/iot-hub-raspberry-pi-kit-c-get-started
[4] Google-Cloud: https://cloud.google.com/iot-core/
[5] Oracle-IoT-Cloud: www.oracle.com/internet-of-things/
[6] Amazon-IoT-Cloud: https://aws.amazon.com/de/iot/
[7] Q-Loud-Cloud: www.q-loud.de/
[8] Thingsboard: https://thingsboard.io/
[9] REST: www.codecademy.com/articles/what-is-rest
[10] REST API: https://restfulapi.net/