GUIDE

# Over-The-Air firmware opdatering

Af Joachim Fisker, Specialist, FORCE Technology

# Introduktion

Formålet med denne guide er at introducere en række praktiske overvejelser og anbefalinger om arkitekturen i- og omkring et elektronisk produkt, som skal kunne foretage Firmware-Over-The-Air opdatering (FOTA).

Overvejelserne går på den praktiske implementering af løsningen og de tilhørende IT-sikkerheds-aspekter. Produktet i fokus vil typisk indeholde en enkelt mikrocontroller med Internet Protokol base-ret kommunikation via internettet. Det har typisk færre ressourcer end f.eks. en Raspberry PI eller et Beaglebone, og det vil sjældent have et filsystem. Guiden er tiltænkt udviklingsingeniører, som er i det kommende produkts tidlige tekniske planlægnings- eller designfase.

# Firmwareopdatering

FOTA gør det muligt at forlænge et produkts levetid ved bl.a. løbende at kunne tilføje og forbedre funktionalitet, øge IT-sikkerheden ved at holde teknologien opdateret, udbedre fejl i firmwaren og f.eks. lave work-arounds af evt. hardwareproblemer, så produktet ikke medfører en reklamationssag eller ender som elektronikskrot.

Under udviklingsarbejdet er opdatering af firmware for det meste en simpel og overskuelig proces, som består i at forbinde en ledning (f.eks. USB, UART, in-circuit-programmer etc.) fra ens arbejdscomputer til prototypen for herefter at aktivere programmeringsprocessen i det integrerede udviklingsmiljø. Efter noget tid er den gamle firmware overskrevet med den nye, og skulle overførslen fejle, kan kredsløbet nemt fejlfindes og skrivningen nemt gentages. For et lanceret produkt, som kan befinde sig så godt som overalt i verden, er processen mere omsiggribende, idet produktet nu vil være eksponeret til omverdenen via internettet og dermed cyberangreb. Dette medfører potentielt en større risiko for uønsket, udefrakommende, påvirkning, hvis risici og konsekvenser bør overvejes i forbindelse med design- og udviklingsarbejdet.

Typiske mål med et angreb kan være:

- Tyveri af intellektuel ejendom (få fat i firmware).
- Tyveri af personhenførbare data.
- Aflytning via produktets sensorer eller det netværk produktet befinder sig på.
- Overtage styring af produktet f.eks. for at indlemme det i DDOS botnet eller obstruere det.
- Ændre funktionalitet (kan have el- / brandsikkerhedsaspekter).
- Få adgang til cloud-infrastruktur med henblik på datatyveri, obstruktion, randsomware-angreb m.f.

Idet trusselsbilledet afhænger af produktet, vil det være svært at komme med fuldstændigt altdækkende råd. En risikoanalyse vil kunne bidrage til at kortlægge angrebsveje, følgerne af et angreb, risiko for at disse finder sted, alvorligheden af angrebet, og hvad der kan gøres for at undgå dem. Analysen vil kunne indikere evt. svagheder i designet både af hardware og firmware. Ved også at udføre en firmware FMEA, som inkluderer firmwarens reaktion på hardwarefejl og anskuer angrebsmål som fejltilstande, vil svagheder i firmwaren, og samspillet mellem den og hardware, kunne afdækkes og prioriteres efter alvorlighed.

For at belyse de praktiske aspekter af FOTA er de næste sektioner opdelt efter FOTA-processens overordnede trin hhv. forudgående planlægning, advisering, distribution og migrering. På tværs af disse er IT-sikkerhedsmæssige overvejelser såsom datafortrolighed, dataintegritet og datatilgængelighed, som vil blive belyst. Det antages, at produktets IT-sikkerhed er moderat prioriteret på line med niveauet, som er defineret i standarden ETSI 303 645.

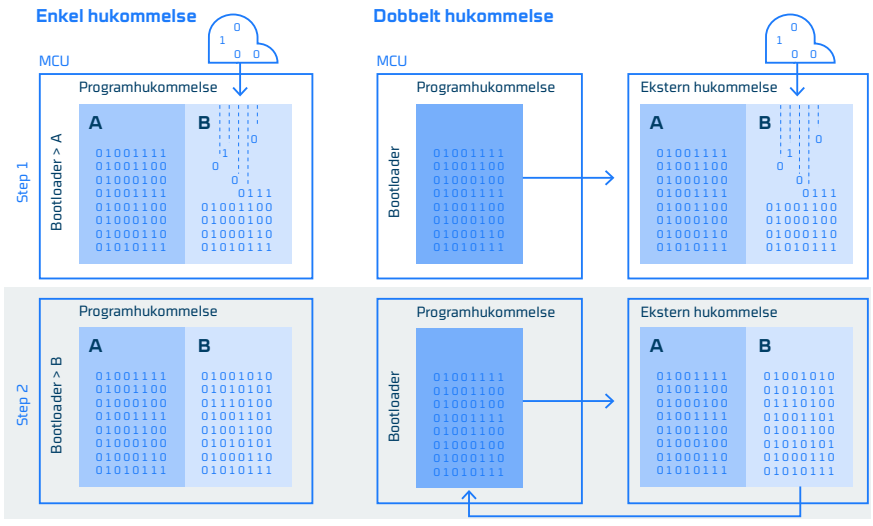
# Planlægning

Inden BOM'en (Bill of Materials) kan låses, er der nogle essentielle spørgsmål, som skal besvares f.eks. hvor meget programhukommelse er der brug for i produktets levetid? Skal produktet kunne modstå et angreb direkte på sin hardware?

### Mikrocontrollerens (mcu) programhukommelse

Programhukommelsens størrelse afhænger af den nuværende applikations omfang, hvor meget den kommer til at vokse i produktets levetid, bootloaderens funktionalitet, migreringsmetoden, ønsket om mulighed for at kunne foretage tilbagerulning af firmware til en tidligere udgave og antallet af tidligere versioner. Flere IT-sikkerhedsstandarder bl.a. ETSI 303 645, EN62443-4-2 anbefaler tilbagerulning som en funktionalitet til håndtering af en forfejlet firmwareopdatering, så der medregnes her plads til én tidligere version.

Hvis mcu'en faciliterer afvikling af firmware, uafhængigt af hvilken startadresse den har i programhukommelsen, så kan teknikken kaldet "dual bank" (DB) benyttes. Med DB opdeles programhukommelsen virtuelt i to områder (banker), hvorfra firmwaren kan afvikles. Bootloaderen vælger, ved genstart, hvilken bank der anvendes. Er firmwaren lagret i- og afvikles fra bank A, så gemmes den nye firmware, der hentes i bank B. Når den nye firmware er klar, sættes et flag i bootloaderen, hvorefter mcu'en genstartes. Den nye firmware afvikles nu fra bank B. Ved tilbagerulning sættes et flag igen i bootloaderen, nu pegende på bank A, hvorefter mcu'en genstartes for derefter at være returneret til den tidligere version. Idet muligheden for tilbagerulning kun er nødvendig i tidsrummet fra første opstart af den nye firmware, og til denne har bestået sin selvtest, så kan banken med den tidligere version også fungere som download buffer. Selvtesten bør være omfattende og som minimum sikre at samtlige vitale funktioner, herunder kommunikation til FOTA-servicen, fungerer korrekt, inden overskrivelse af den tidligere firmware tillades. Størrelsen af programhukommelsen skal med dual bank metoden som minimum være: 2 x den planlagte firmwares størrelse x tilvækst i produktets levetid [%]. F.eks. 2 x 74 kB x 1,25 = 185 kB.




Er dual bank teknikken ikke mulig, bliver hukommelsesstørrelse: 3 (aktive, tidligere, buffer) x den planlagte firmwares størrelse x tilvækst i produktets levetid [%]. Den "tidligere" firmware og download bufferen kan godt gemmes i en for mcu'en ekstern hukommelse f.eks. en EEPROM, men det kan være nødvendigt at kryptere dataene, da en angriber ellers ville kunne få fat i firmwaren ved at udlæse fra den eksterne hukommelse med et simpelt værktøj.

## Hærdning af hardware

På OWASP's top-10 over de mest hyppige årsager til et kompromitteret IoT-produkt ligger manglende hærdning af hardware nr. 10 i forhold til dårlige kodeord, som er nr. et. Med andre ord kan det måske virke mindre kritisk, og samtidig kompliceret, at skulle hærde sin hardware. Niveauet af hærdning har her til formål at beskytte følgende: firmware, personhenførbare data, enhedens identifikations- og autentifikations-oplysning f.eks. krypteringsnøgler, brugernavn, certifikater etc. mod at blive ekstraheret af en angriber med moderat teknisk indsigt og begrænsede ressourcer, som har fysisk adgang til produktet. Det betyder typisk, at dataforbindelser (serielport, USB etc.) som ikke er en del af applikationen, skal være afbrudt permanent, at firmwaren ikke må facilitere udlæsning af beskyttet data, og programhukommelsen skal være læsebeskyttet.

Så de overordnede spørgsmål kan være: Yder de elektriske komponenter, som lagrer- og processerer dataene, tilstrækkelig modstand mod udlæsning? Og bliver dataene kommunikeret tilstrækkeligt sikkert mellem de komponenter?

## Tommelfingerregler

- Undgå lagring af kritisk data i ekstern hukommelse. Hvis det er uundgåeligt så krypter data med en nøgle, som kun er lagret i mcu'en, og som er unik for hver produktenhed.
- Er mcu og kommunikationsmodul (modem) separate enheder, skal al sikkerhed (kryptering/dekryptering) håndteres i mcu pga. mulighed for aflytning af databussen mellem dem.
- Vælg en mcu med god beskyttelse mod firmwareudlæsning (mere end almindelig fuse bit læselås).
- Vælg en mcu med hukommelsesfirewall (områder som kun en specifik del af firmwaren har adgang til).
- Vælg en mcu med indbygget root-of-trust. Er det ikke muligt benyt et eksternt secure element (crypto ic).
- Overvej om der er fordel ved at indbygge tamper detection hardware i produktet, som opdager, hvis det åbnes fysisk.

## Advisering

Når den nye firmware er klar til udrulning, og den er gjort tilgængelig for distribution, så kan advisering af "flåden" begynde og dermed sætte gang i dennes opdatering. Dette afsnit giver eksempler på svar af de følgende spørgsmål: Gennem hvilke kommunikationskanaler kan adviseringen foregå? Hvilken information kan den indeholde? Hvordan kan udrulningen afvikles, og hvordan styres det i praksis?

### Kommunikationskanal

Kanal refererer her, til hvilken IP-baseret protokol og service som faciliterer kommunikationen mellem produktet og FOTA-servicen. Kravene til kanalen bør som udgangspunkt være, at det med rimelighed kan siges, at den opretholder datafortrolighed og dataintegritet. Dette gøres ved anvendelse af kryptering samt identifikations- og autentifikationsmekanismer, så produktet og FOTA-servere kan få vished om, at de hver især er, hvem de udgiver sig for at være. Dette er nødvendigt for at sikre, at serveren f.eks. ikke deler den nye firmware med uvedkommende, eller at en produktenhed snydes til at hente en falsk firmware fra en ondsindet server.

For IoT-produkter er protokollen MQTT kombineret med TLS pt. den mest udbredte løsning til udveksling af nytte- og driftsdata. Den er som sådan ikke designet med henblik på FOTA, men da den har egnet funktionalitet (uddybes senere), vil det spare produktet for implementering af yderligere protokoller. MQTT med TLS er en standardarkitektur til både dataudveksling og FOTA hos f.eks. Amazon, Google og Azure, men den kan godt være for ressourcekrævende (program- og arbejdshukommmelse samt regnekraft) for prissensitive produkter. En alternativ løsning kan være MQTT-SN og DTLS. Det er en simplere implementering af MQTT og TLS, som er baseret på UDP frem for den tungere TCP. Et letvægtsalternativ til MQTT/TLS-arkitekturen kunne f.eks. være det UDP-baseret client/server system Nabto. Det krypterer med AES og giver samme garantier som et TCP-baseret system. Det muliggør P2P kommunikation, som bl.a. ville kunne aflaste en evt. cloud løsning, ved at lade brugeren og produktenheden kommunikerer direkte, fremfor at have cloud'en som mellemmand. Uanset hvilken løsning der vælges, kan den evalueres med følgende tommelfingerregler.

### Tommelfingerregler

- Krypteringen skal være på linje med AES-128 eller bedre.
- Krypteringsnøgler og identifikationsparametre (herunder certifikater) skal være unikke for hver enkelt produktenhed.
- Krypteringsnøgler skal fornyes for hver session, (når produktenheden forbinder til server). F.eks. rod-nøgle anvendes til at genererer midlertidig sessionsnøgle.
- Både produkt og server skal kunne autentificere deres identitet f.eks. med en challenge-respons mekanisme.
- En mekanisme skal sikre at data ikke modificeres undervejs f.eks. ved brug af en kryptografisk checksum (CMAC).
- Datapakkertæller el.lign. skal anvendes for at undgå playback angreb.

## Udrulning

Hvordan udrulningen af den nye firmware afvikles, kan bl.a. afhænge af det samlede antal involverede produktenheder, hvorvidt de er online periodevis eller kontinuerligt, om deres netværk er stabilt, og hvilken datarate de kan hente / lagre den nye firmware med. Er der tale om et stort antal enheder, som altid er online, vil udrulningen typisk skulle ske i etaper for ikke at overbelaste FOTA-servicen. Etaperne kan f.eks. etableres ved at inddele produktenhederne i grupper f.eks. efter deres serienummerinterval, produktionsbatch-id el.lign. Har produktet begrænsede ressourcer såsom; bufferhukommelse, regnekraft, båndbredde, begrænset energi, eller befinder det sig på et ustabilt netværk - kan det være en fordel med en pull orienteret udrulning. Her styrer modtageren slagets gang frem for at reagere på direkte kommandoer fra FOTA-servicen. Dette gør det muligt at tilpasse processen til produktenhedens tilstand, f.eks. ved at den selv vælger opdateringstidspunktet f.eks. på baggrund at sin batteristatus, og ved at lade FOTA-servicen facilitere at firmwaren kan hentes i mindre bidder. Så det kan ske uden at kompromittere produktenhedens hovedopgaver.

En pull orienteret strategi på MQTT kan implementeres, således at produktenhederne, på baggrund af tre parametre såsom: produkt-serie-id, produktions-batch-id og nuværende firmware version, løbende kan validere om deres nuværende version er den nyeste, eller om de skal begynde opdateringsprocessen. Ved at lade produktenheden abonnere på et enkelt MQTT topic: "/produkt-serie-id/produktions-batch-id/fwver" hos FOTA-servicen, kan den s ammenholde sit versionsnr. med det sidste nye og dermed blive adviseret, så snart det bliver opdateret. Ved at benytte MQTT's hierarkiske topic-struktur kan produktenhederne inddeles i grupper f.eks. på baggrund af produktions-batch-id, og udrulningsforløbet styres enkelt ved gruppe efter gruppe at opdatere fwver værdien i takt med, at den forrige gruppe bliver færdig.

## Tommelfingerregler

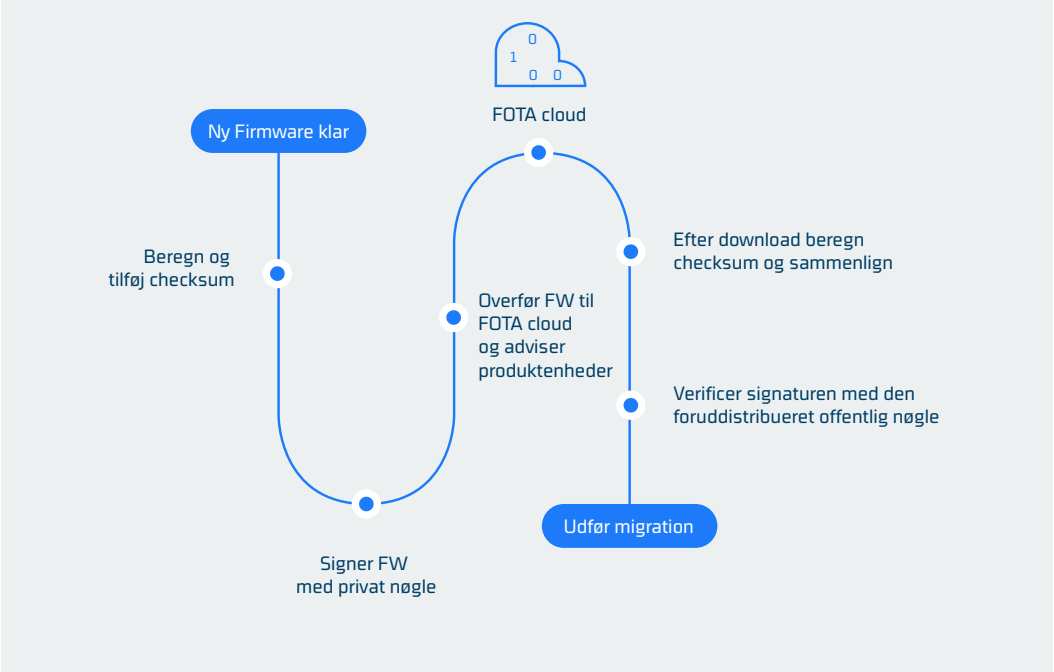
- Foretag først en udrulning af ny firmware på en for producenten intern "testflåde" så fejl og mangler kan opdages og rettes, så ikke de når ud til brugerne.
- Adgangskontrollen til FOTA-servicen skal operere på produktenhedsniveau og være granuleret, således at produktenheden kun kan få adgang til, hvad der for denne er relevant.
- Produktenhederne bør kvittere for modtagelse af det nye firmware versionsnummer, så produktgruppens status kan følges.

## Distribution

Når produktenheden er blevet adviseret om en ny firmware, skal den hentes ned. Til det er der en række muligheder såsom (S)FTP, MQTT, Websocket og RESTful API på HTTP m.f. De vil skulle kunne opfylde de samme IT-sikkerhedskrav som kommunikationskanalen til advisering. Ved at "genanvende" MQTT til distribution begrænses firmwarens generelle kompleksitet på bekostning af mindre fleksibilitet i distributionskanaler.

En pull orienteret MQTT distributionsløsning, med mulighed for blokvis hentning, kan implementeres så enheden på baggrund af to parametre fra adviseringen: produkt-serie-id og fwver kan hente den nye firmware. Ved først at abonnere på topic: "/produkt-serie-id/fw/fwver/info" modtager produktenheden parametre, der er nødvendige for at hente og validere den nye firmware. Hhv. antallet af blokke firmwaren er opdelt i, samt en checksum som skal sikre, at den samlede overførsel er sket uden fejl. Herefter vil produktenheden skiftevis og i kronologisk rækkefølge abonnere på topic: "/produkt-serie-id/fw/fwver/block/0..n", indtil samtlige blokke er hentet. Begyndelsen af hver blok indeholder ligeledes en checksum, så produktenheden kan verificere, hvorvidt den aktuelle blok er korrekt overført, før den kan gå videre til den næste. Ved fejl eller afbrydelse kan blokken hentes igen. Efter endt overførsel ophører produktet med at abonnere på topic: "/produkt-serie-id/fw..", og produktenheden kan efter verifikation af checksum begynde migreringsprocessen.

I nogle situationer kan det være nødvendigt at autentificere den nye firmware, før migrationen kan begynde. Det kan f.eks. være, hvis den blev distribueret via en usikker kommunikationskanal (modsat de tidligere eksempler). Firmwaren vil i så fald skulle være krypteret og en MAC være inkluderet. For at autentificere skal den nye firmware først dekrypteres. Dette gøres med produktenhedens unikke private krypteringsnøgle. Herefter beregnes en ny MAC med producentens offentlige krypteringsnøgle. Er den nye MAC lig med den medfølgende, kan det sluttes, at afsenderen af den nye firmware må være producenten, eftersom den har adgang til dennes private krypteringsnøgle. Derudover har afsender også haft adgang til produktenhedens offentlige krypteringsnøgle, som i denne situation ikke nødvendigvis ville være kendt af andre end producenten. Migreringen kan ny begynde.



### Åben- eller lukket distributionskanal

Med en lukket distributionskanal kan den nye firmware med et gøres tilgængelig som "rå" hex-fil for alle produktenheder, der har adgang til FOTA-servicen. Det er kommunikationskanalens sikkerhed og FOTA-servicens adgangskontrol, der er garant for, at alt data kommunikeres i fortrolighed og opretholder integritet.

Med en åben kommunikationskanal er der ingen garantier om fortrolighed, integritet eller adgangskontrol. F.eks. hvis firmwaren blev distribueret som en hex-fil via et offentligt website. Skal firmwaren fortsat være utilgængelig for uvedkommende mm. vil det være nødvendigt, at den "rå" hex-fil skærmes med kryptering, MAC og evt. digitalt signeres. Det vil gøre det muligt for den tilsigtede modtager at verificere, at afsenderen er korrekt, og at der ikke er uønskede ændringer i firmwaren. Derudover vil krypteringen sikre, at kun tilsigtede modtagere kan få adgang. For at dette kan fungere, skal afsender og modtager have hinandens offentlige X509 certifikater distribueret på forhånd. For at modtagerne kan verificere afsenderens identitet, forudsætter det ét nøglepar (offentlig- & privatnøgle) per afsender. For at afsender kan sikre, at kun tilsigtede modtagere får adgang, skal der enten bruges et fælles nøglepar, som deles af samtlige produktenheder eller et individuelt nøglepar til hver enkelt produktenhed. Benyttes den sidstnævnte arkitektur, vil afsenderen skulle generere lige som mange individuelle firmwarepakker, som der er produktenheder, hvilket kan blive en betydelig proces i takt med, at flåden vokser. Deles et enkelt nøglepar med samtlige produktenheder, vil det ikke være muligt at udelukke en kompromitteret enhed fra at få den nyeste firmware på et senere tidspunkt, da det ville medføre udelukkelse af samtlige produktenheder.

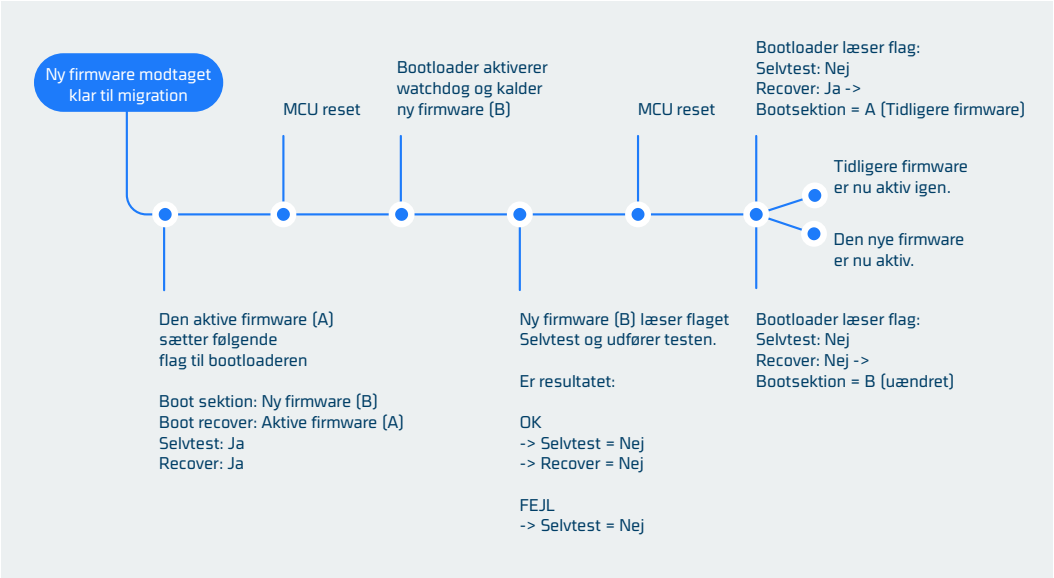
### Tommelfingerregler

- Distributionskanalen skal opfylde samme krav som kommunikationskanalen til advisering.
- Checksummens længde skal muliggøre detektion af enkelt bit fejl.

## Migrering

Når den nye firmware er overført til produktenheden, og den er blevet checksum verificeret og evt. autentificeret, kan migreringen begynde. Alt efter hvad mcu'en understøtter, vil bootloaderen enten begynde afviklingen af applikationen fra den anden hukommelsesbank eller først overskrive den nuværende firmware for derefter at genstarte. Bootloaderen styres med flag sat af applikationen i en ikke-volatil hukommelse.

Udover at facilitere applikationens opstart, så er det bootloaderens opgave at sikre, at produktenheden kan genskabe den tidligere applikation, hvis den nye firmware fejler. Det kunne f.eks. være ved ikke at kunne genforbinde til FOTA-servicen efter genstart. Dette betyder, at den nye applikation skal være i stand til at foretage en fyldestgørende selvtest, når den afvikles første gang og videregiver resultatet til bootloaderen så den evt. kan foretage en tilbagerulning.




## Tommelfingerregler

- Checksum validering af ny firmware når den er fuldt downloadet, samt før og efter den er blevet overført til programhukommelsen for afvikling.
- Firmware har altid versions nr. og produktserie-id etc., så enheden kan sikre sin hardware-kapabilitet, skulle en forkert firmware være blevet distribueret.
- Produktenheden må kun hente en firmware med et højere versionsnr. end dennes nuværende. Tilbagerulning kan kun ske på foranledning af en forfejlet selvtest ikke pga. en advisering fra FOTA-servicen.
- FOTA-servicens offentlige nøgle skal være inkluderet i den nye firmware.
- Bootloader flag skal være redundante.
- Produktenhed kvitterer online efter firmwareopdateringen både ved succes og fejl (tilbagerulning).

Der er mange forskellige spørgsmål der med fordel kan afklares inden et internetforbundet produkt lanceres. Denne guide har beskæftiget sig med nogle af dem og givet eksempler på simple løsninger som inspiration til videre overvejelser. En stor forskel fra traditionelle produkter til de nu internet-forbundet er, at andelen af produktets livscyklus, som producenten er involveret i, kan være blevet betydelig længere. Produktet er ikke længere færdigt når udviklingsafdelingen har overleveret designet til produktion som tidligere og der kan potentielt blive tale om kontinuerlig udvikling eller i hvert fald vedligeholdelse. Det kan være en ny måde at tilgå produktudvikling på, og den kan f.eks. betyde, at udviklingsafdelingen og operationsteamet med fordel kan blive tættere integreret. F.eks. fordi designbeslutninger på den ene side kan påvirke driftsomkostninger, og real-world driftsdata kan informere kommende designbeslutninger osv. Der findes et væld af FOTA-arkitekturer som hver især har fordele og ulemper. Hvad der passer bedst til produktet, afhænger bl.a. af ønsket om redundans og skalerbarhed. Samlet for dem er at de alle kan evalueres med retningslinjerne i denne guide. FORCE Technology har en rådgivningsservice som kan bidrage med sparring og praktiske løsninger.

Har du spørgsmål, brug for sparring eller information om praktisk løsninger er du velkommen til at kontakte FORCE Technology.
Joachim Fisker
jofs@forcetechnology.com
www.forcetechnology.com/rk06

## Forkortelser og termer

| | |
|---|---|
| **FOTA** | Firmware Over The Air opdatering |
| **FMEA** | Failure mode and effects analysis |
| **AES** | Advanced Encryption Standard |
| **TLS** | Transport Layer Security |
| **DTLS** | Datagram Transport Layer Security |
| **UDP** | User Datagram Protocol |
| **TCP** | Transport Control Protocol |
| **MCU** | Micro Controller Unit |
| **MAC** | Message authentication code |
| **MQTT** | Message Queuing Telemetry Transport |
| **DDOS** | Distributed Denial Of Service angreb.<br>(Mange enheder "oversvømmer" ofret med så meget data det står af). |
| **P2P** | Peer-To-Peer kommunikation |
| **Broker** | Central server i MQTT terminology |
| **CRC** | Cyclic Redundancy Check |
| **EMS** | Electronics manufacturing service |
| **Buffer** | Midlertidigt datalager |

5683.2

**FORCE Technology**
Venlighedsvej 4
2970 Hørholm
Danmark
+45 43 25 00 00
info@forcetechnology.com
forcetechnology.com