

WasteScan



Innhold

Prosjektets målsetning	2
Kort om løsningen	2
Maskinvare / Teknisk løsning	2
Beskrivelse av fremgangsmåte	3
Microsoft Azure IoT Hub	4
Opprette IoT Hub	4
Opprette device	5
Arduino IDE	6
Ekstra programbiblioteker	6
NodeMCU	7
Microsoft Azure SQL Database	8
Opprette SQL server og database	8
Opprette tabeller	9
Opprette Stored Procedure	10
Microsoft Azure Logic App	11
Step 0 – Opprette en Logic App	12
Step 1 – When a HTTP Request is received	14
Step 2 – Parse JSON	15
Step 3 – Insert row	16
Step 4 – Execute stored procedure	17
Step 5 – Get row	17
Step 6 – Request response	18
Step 7 – POST request	18
Barcode Scanner Terminal	19

Oppsett.....	19
Skjermbilder	20
Testing.....	20
Beskrivelse av programkode	21
Spesielle problemer	22
Fremtidig arbeid	22
Litteraturliste	22

Prosjektets målsetning

Målet er å gjøre det lettere å sortere avfall for gjenvinning, og potensielt unngå å sortere avfall. Det kan også motivere mennesker og føre til mer og bedre kildesortering. Dette kan spesielt være tilfelle for offentlige søppelkasser og større lokaler som USN. Ideen er å veilede hvilken søppelkasse brukeren skal bruke. Løsningen vil gi forbrukerne veiledning om avfallssortering på gjenvinningsstasjoner rundt for eksempel på Campus Vestfold.

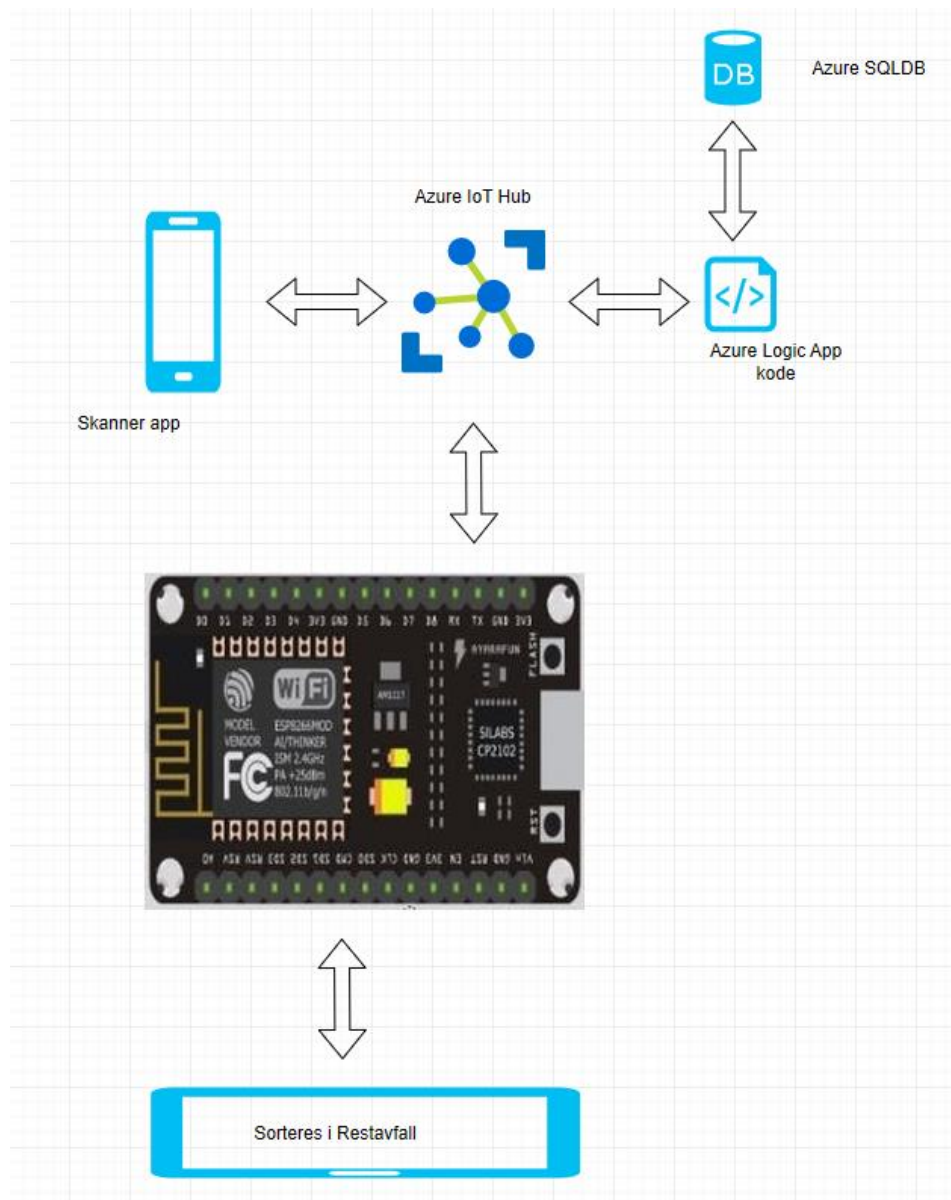
Kort om løsningen

Prototypen består av en mobil enhet med en app som skanner strekkoder. I tillegg vil det være montert en IoT-enhet på gjenvinningsstasjonen. Denne IoT-enheten vil, gjennom et LCD-display, vise hvilken avfallskategori et element du vil resirkulere er i. Løsningen vil logge bruken og vil kunne trekke ut statistikk. Se attemnet for et systemskjema. (att, 1)

Strekkoder registreres i en database av kategoriene plast, papir, restavfall, glass og metall og bio. Ved hjelp av mobilappen "Barcode Scanner Terminal" kan du bruke et mobilkamera til å skanne en strekkode. Appen gjenkjenner strekkodenummeret og gjør det til en tekststreng. Denne strengen sendes til en skytjeneste, som sjekker strekkodenummeret mot databasen og returnerer informasjon om hvilken kategori strekkoden tilhører. Denne informasjonen vises i både mobilappen og LCD-skjermen (iot).

Maskinvare / Teknisk løsning

- Mobiltelefon med android 4.0 eller høyere
- Barcode Skanner Terminal for Android
- NodeMCU microcontroller med LCD-skjerm
- Microsoft Azure SQL Database
- Microsoft Azure Logic App
- Microsoft Azure IoT Hub
- Enhetene kommuniserer via HTTP POST og GET Requests



Beskrivelse av fremgangsmåte

Beskrivelsen under er i logisk rekkefølge. Dvs. at en del av stegene ikke vil la seg gjennomføre uten at foregående steg er gjennomført.

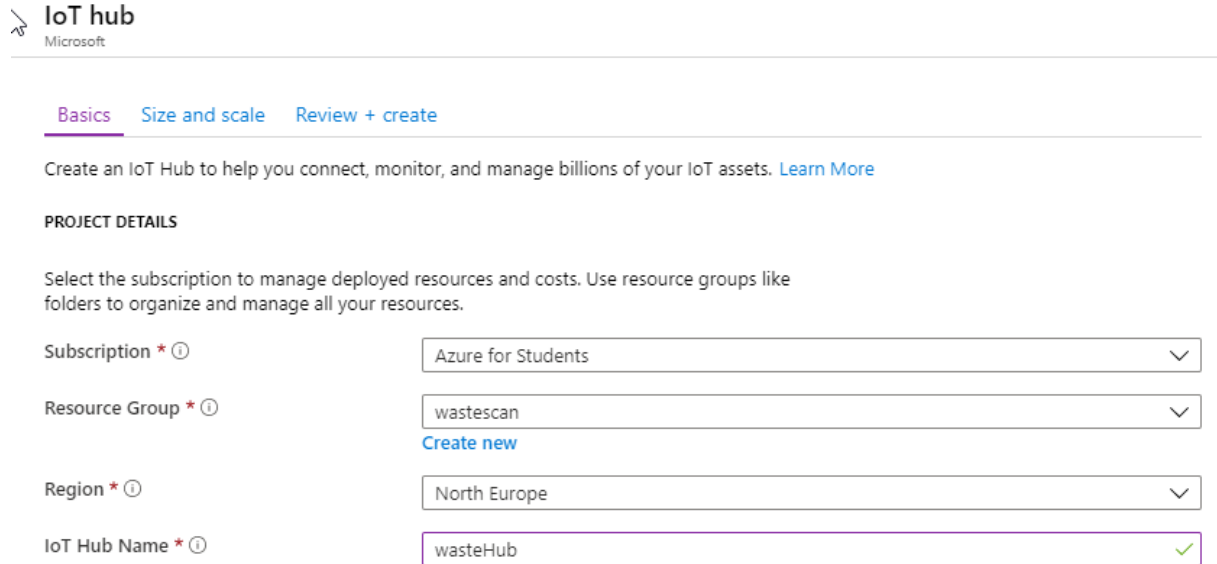
For eksempel vil det ikke gi mening å opprette et Logic App steg for å hente en rad fra databasen hvis ikke databasen er opprettet og tabellen har et innhold og er definert på riktig måte.

Det vil heller ikke være noe poeng i å sette opp IoT-devicen uten connection-string fra Azure IoT Hub.

Microsoft Azure IoT Hub

Opprette IoT Hub

- I MS Azure: Gå til Dashboard / All resources og klikk på "Add"
- "Search the Marketplace" etter "IoT Hub", klikk på " IoT Hub " og velg "Create"
- Gjør valg som under:



The screenshot shows the 'Basics' tab of the IoT Hub creation wizard. It includes a header with the IoT hub logo, navigation tabs for 'Basics', 'Size and scale', and 'Review + create', and a description of IoT Hub. The 'PROJECT DETAILS' section contains four fields: 'Subscription' (Azure for Students), 'Resource Group' (wastescan), 'Region' (North Europe), and 'IoT Hub Name' (wasteHub). A 'Create new' link is visible under the Resource Group field.

IoT hub
Microsoft

[Basics](#) [Size and scale](#) [Review + create](#)

Create an IoT Hub to help you connect, monitor, and manage billions of your IoT assets. [Learn More](#)

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure for Students ▼

Resource Group * ⓘ wastescan ▼
[Create new](#)

Region * ⓘ North Europe ▼

IoT Hub Name * ⓘ wasteHub ✓

Figure 1 - Oppretter IoT Hub

- Gå til "size and scale" og velg prismodell. Prismodell "F1: Free tier" kan håndtere 8000 meldinger om dagen og holder fint til å teste et prosjekt som dette.
- Velg "Review + create" og "Create"

Opprette device

- Inne i IoT-huben, gå til IoT devices og klikk "New"

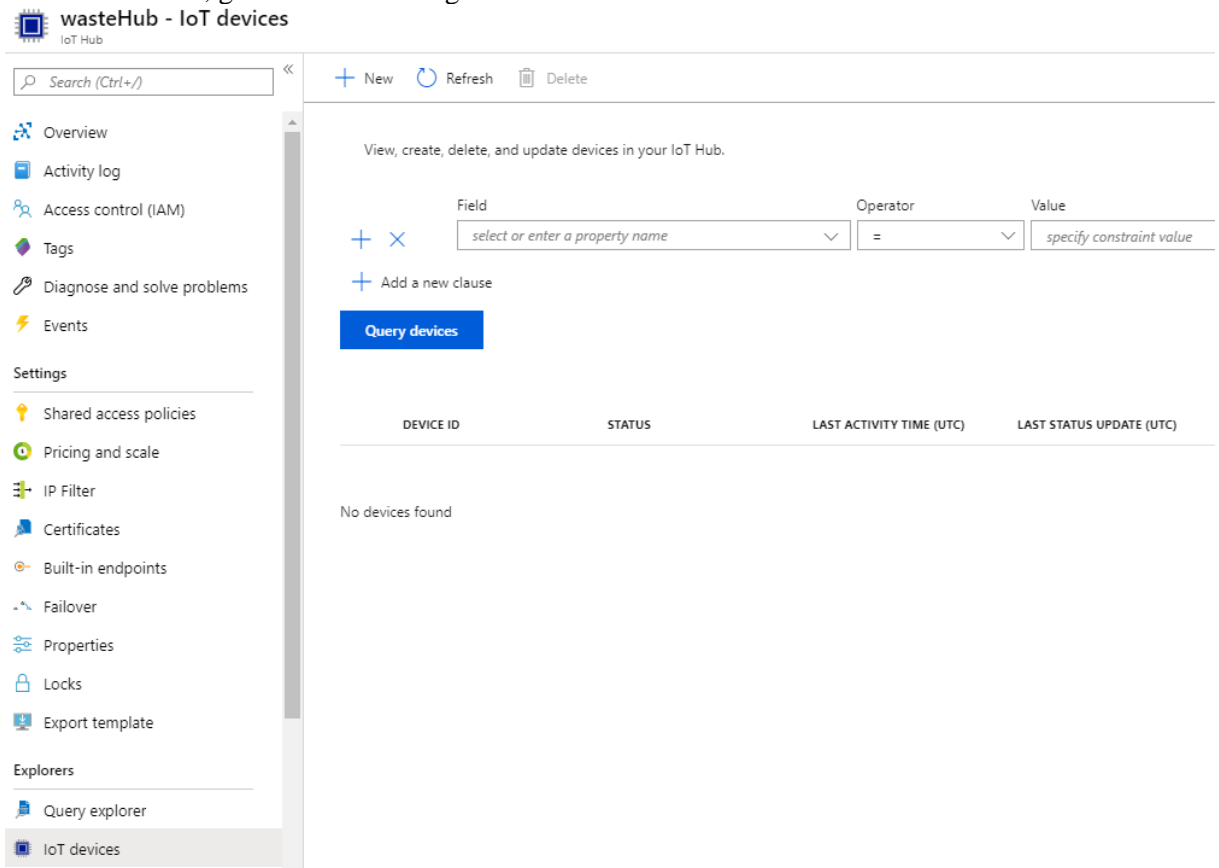


Figure 2 - Opprett IoT-device

- Velg navn og la resten av innstillingen stå som default og klikk "Save"

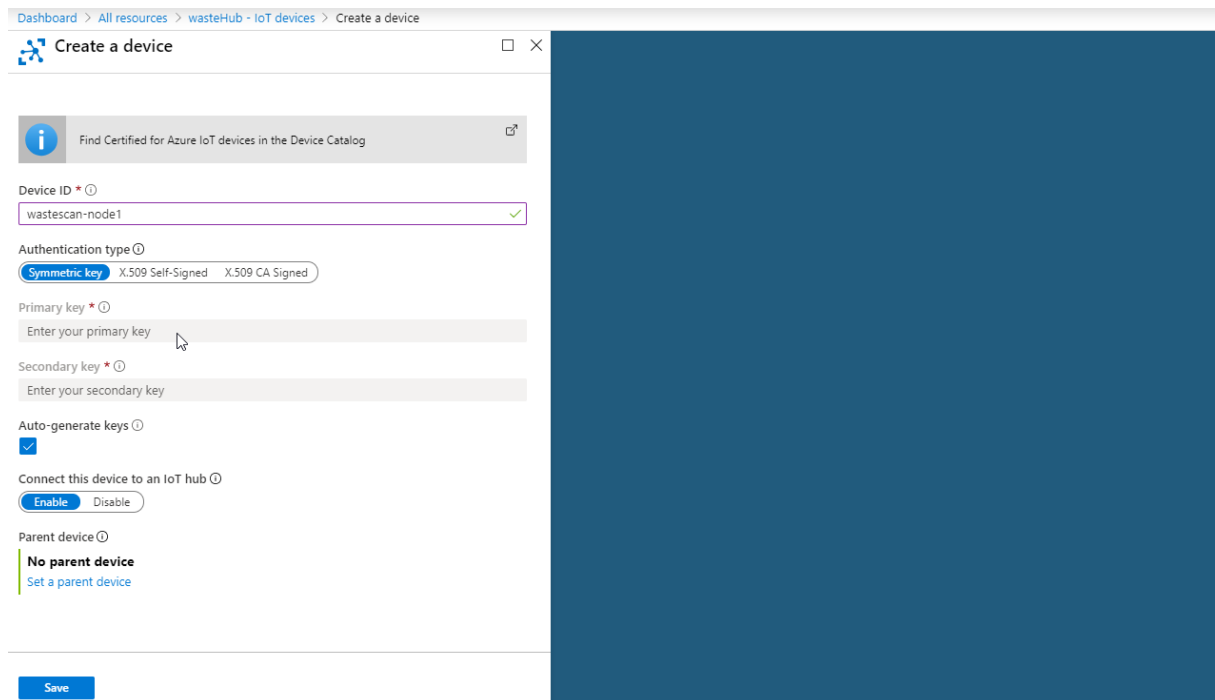


Figure 3 - Create device

IOT 3000 – Eksamen H19 – WasteScan av IoT Drømmeteamet

- Fremdeles i IoT-hub / IoT devices, klikk på den nye devicen.
- Kopier "Primary Connection String". Denne skal senere inn i device-koden og i siste step i Logic App.

Dashboard > All resources > wasteHub - IoT devices > wastescan-node1

wastescan-node1

wasteHub

Save Message to Device Direct Method Add Module Identity Device Twin Manage keys Refresh

Device ID	wastescan-node1
Primary Key
Secondary Key
Primary Connection String	HostName=wasteHub.azure-devices.net;DeviceId=wastescan-node1;SharedAccessKey=tsdZ25vIselFUjjiNYg9rmcjj8Oaw61HnScnaJ6vC4jY=
Secondary Connection String
Enable connection to IoT Hub	<input checked="" type="radio"/> Enable <input type="radio"/> Disable
Parent device	No parent device
Distributed Tracing (preview)	Not configured

[Learn more](#)

Arduino IDE

For å programmere IoT-devicen bruker vi Arduino IDE.

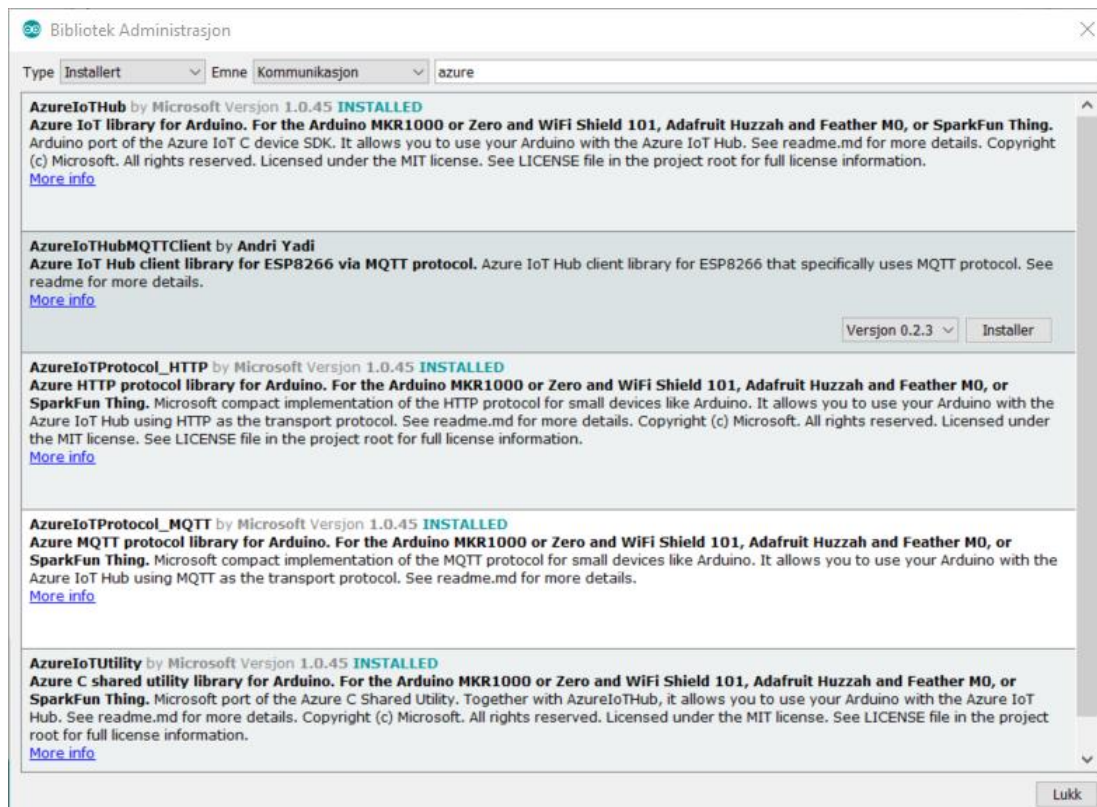
Siste versjon av denne kan lastes ned fra <https://www.arduino.cc/en/main/software>.

Ekstra programbiblioteker

For å kunne realisere dette prosjektet har vi lagt til følgende biblioteker i Arduino IDE.

- AzureIoTHub
- AzureIoTProtocol_MQTT
- AzureIoTUtility
- Time
- ArduinoJson
- LiquidCrystal

Bibliotekene installeres ved å, i Arduino IDE, velge Verktøy / Bibliotek Administrasjon



NodeMCU

NodeMCU er egentlig navnet på en IoT-plattform og en firmware som kan kjøre på ESP8266 WIFI SoC (System on a Chip). I vårt tilfelle snakker vi om selve enheten som vi har programmert som en helt vanlig arduino*.

Enheten er koblet sammen med LCD-skjermen som vist under. Dette kan gjøres mer kompakt ved å koble til via en IC2-bus i. På den måten ville man bare hatt behov for 2 pinner på enheten i tillegg til strøm.

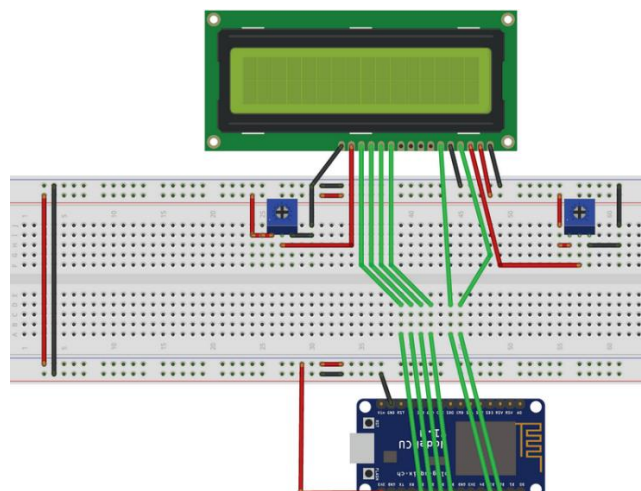


Figure 4 - Koblingskjema for IoT-device

Programmet installeres med Arduino IDE men før det gjøres må koden endres slik at WI-FI brukernavn og passord og connection-string blir riktig i forhold til prosjektet vårt. Se under.

```
#define IOT_CONFIG_WIFI_SSID          "*****"  
#define IOT_CONFIG_WIFI_PASSWORD     "*****"  
#define IOT_CONFIG_CONNECTION_STRING ""  
#define DEVICE_ID                     "wastescan-nodel"
```

I vårt tilfelle er connection-string hentet fra IoT-devicen vi opprette tidligere (Primary Connection String:

HostName=wasteHub.azure-devices.net;DeviceId=wastescan-nodel;SharedAccessKey=tsdZ25vIselFUjiNYg9rmcj8Oaw61HnScnaJ6vC4JY=

Microsoft Azure SQL Database

Her beskrives oppsett av server og database samt generering av tabeller og en stored procedure. Til administrasjon av databasen har vi brukt MS SQL Server Management Studio. Vi går ikke inn på bruken av denne men inkluderer bilder av tabellene, SQL-query for den lagrede prodyren og beskrivelse av hvordan systemet tar de i bruk. Det forutsettes av leseren som vil følge denne beskrivelsen har grunnleggende kunnskap om relasjonsdatabaser og SQL. Det er ellers viktig å merke seg at alle tabellene må settes opp med primær-nøkkel for at SQL-triggerene i Logic App skal fungere som ønsket.

Opprette SQL server og database

- I MS Azure: Gå til Dashboard / All resources og klikk på "Add"
- "Search the Marketplace" etter "SQL Server", klikk på "Azure SQL" og velg "Create"
- Under "How do you plan to use the service", velg "SQL databases", "Single database" og klikk "Create"
- Gi databasen et navn, velg Subscription, Resource Group.
I vårt tilfelle: wastescandb, Azure for Students og wastescan.
- Velg "Create new" under "Server" og legg til servernavn, admin, passord og lokasjon i popup-vinduet.
I vårt tilfelle: wastescandbserver, wasteadmin, ***** og North Europe
- Klikk "OK"
- Klikk "Review + create"

IOT 3000 – Eksamen H19 – WasteScan av IoT Drømmeteamet

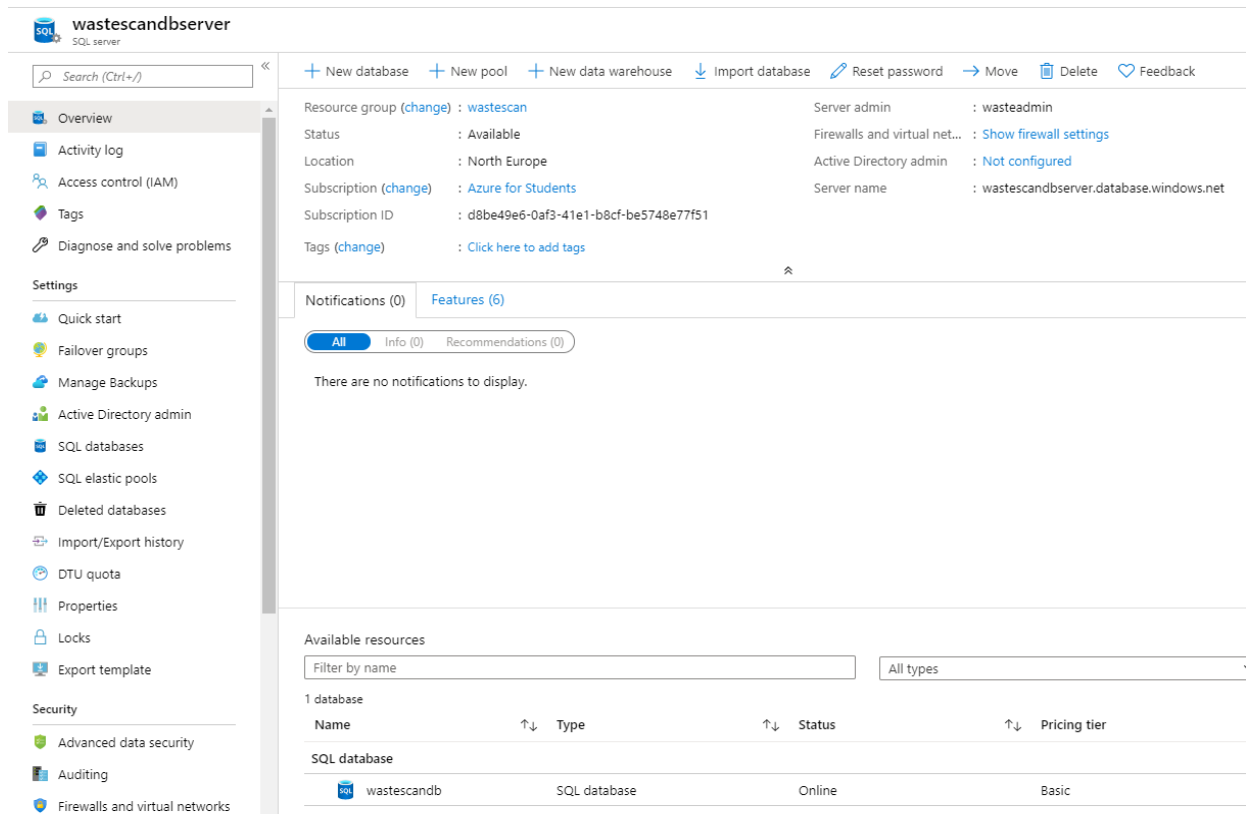


Figure 5 - Opprettet SQL Server og database

Opprette tabeller

- Opprett tabellen scanInput. Denne skal oppdateres av Logic Appen med data fra Barcode Scanner Terminal.

	id	data	scannerId	latitude	longitude	time
▶	1	5012345678900	bio			1574673199405
	2	7010001234567	alfScanner			1574673542540
	3	12195125	alfScanner			1574673549872
	4	610762569327	alfScanner			1574673557431
	5	5012345678900	alfScanner			1574673567227
	6	7040880216309	alfScanner			1574673575463
	7	7010001234567	alfScanner			1574673621463
	8	682430611744	alfScanner			1574673634927
	9	610762569327	alfScanner			1574673649938
	10	5012345678900	alfScanner			1574673660339
	11	7040880216309	alfScanner			1574673673009

Figure 6 - Tabellen scanInput

- Opprett tabllen category. Denne tabellen er statisk og skal inneholde 4 forhåndsdefinerte strekkoder og kategorityper som vist under.

	id	data	catType
▶	1	7010001234567	plastikk
	2	682430611744	papir
	3	5012345678900	glassMetall
	4	7040880216309	bio

Figure 7 - Tabellen category

- Opprett tabellen categoryresult. Denne inneholder en rad med kategoritype og kategori output som blir oppdatert av den lagrede prosedyren hver gang systemet mottar en scannerdata. Innholdet i feltet catType vil alltid samsvare med navnet på en av de kallbare metodene på IoT-devicen. Feltet catOutput vil inneholde teksten vi ønsker å skrive til LCD-displayet og sende tilbake til telefonen / skanner-appen.

	id	catType	catOutput
▶	1	restavfall	Restavfall

Figure 8 - Tabellen categoryresult

Opprette Stored Procedure

Den lagrede prosedyren sjekker om data-feltet i raden med høyest id-nummer i "scanInput" tilsvarer et av data-feltene i "category" og oppdaterer så "categoryresult" med aktuell kategori-type og kategori-output.

SQL-query for lagret prosedyre

```
CREATE PROCEDURE catupdate
AS
DECLARE @v_catType VARCHAR(50);
DECLARE @v_catOutput VARCHAR(50);

SET @v_catType = (Select catType from category where category.data =
(select data from scanInput where id = (select max(id) from scanInput)));

IF (@v_catType = 'plastikk')
BEGIN
UPDATE categoryresult
SET catType = @v_catType, catOutput = 'Plastikk'
WHERE categoryresult.id = 1;
END
```

IOT 3000 – Eksamen H19 – WasteScan
av IoT Drømmeteamet

```
ELSE IF (@v_catType = 'bio')
BEGIN
UPDATE categoryresult
SET catType = @v_catType, catOutput = 'Matavfall'
WHERE categoryresult.id = 1;
END

ELSE IF (@v_catType = 'papir')
BEGIN
UPDATE categoryresult
SET catType = @v_catType, catOutput = 'Papp og papir'
WHERE categoryresult.id = 1;
END

ELSE IF (@v_catType = 'glassMetall')
BEGIN
UPDATE categoryresult
SET catType = @v_catType, catOutput = 'Glass og Metall'
WHERE categoryresult.id = 1;
END

ELSE
BEGIN
UPDATE categoryresult
SET catType = 'restavfall', catOutput = 'Restavfall'
WHERE categoryresult.id = 1;
END
GO
```

Microsoft Azure Logic App

WasteScan sin Logic App består av syv steps:

- Step 1 er en lytter som venter på en HTTP GET request,
- Step 2 gjør query-delen av requesten om til objekter som kan brukes videre i logikken
- Step 3 oppdaterer databasetabellen "scanInput" med objektene fra step 2
- Step 4 starter en lagret prosedyre i databasen som oppdaterer tabellen "categoryresult" med riktig kategori i forhold til strekkode-dataene som ble mottatt i step 1
- Step 5 oppretter objekter av innholdet i databasetabellen "categoryresult".
- Step 6 sender en request-response med en tekst og avfallstype tilbake til enheten som initialiserte step 1. I vårt tilfelle, Barcode Skanner Terminal.
- Step 7 sender en POST request til IoT-enheten. Requesten inneholder avfallskategori og trigger en metode (funksjon) på enheten som viser en tekst med avfallstype på LCD-skjermen

Rekkefølgen her er viktig. Bytter man om step 6 og 7 og IoT-enheten av en eller annen grunn ikke svarer så vil ikke step 7 triggres. Hvert step er altså avhengig av at det foregående har kjørt.

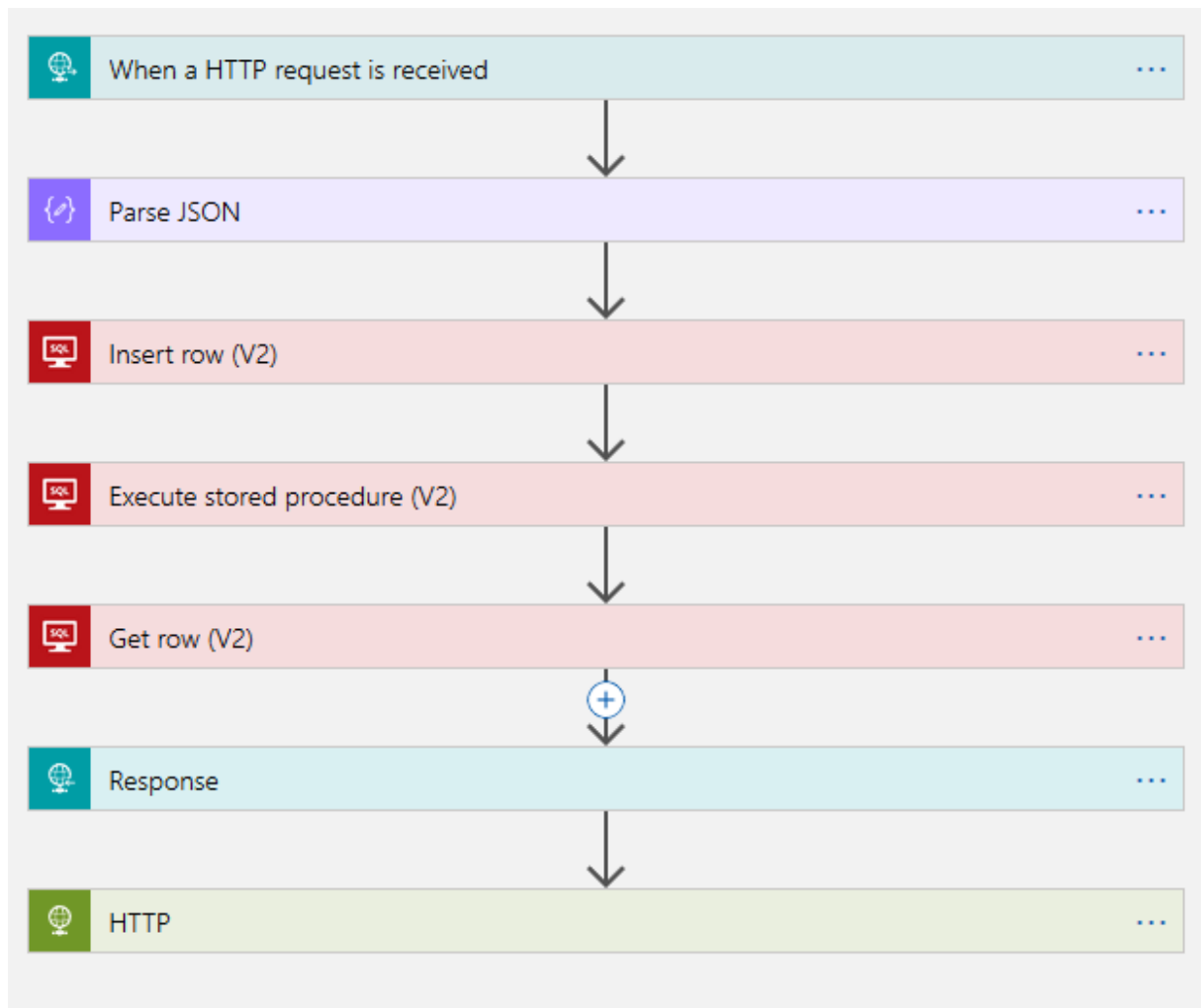


Figure 9 - Logic app når alle stepene er laget

Step 0 – Opprette en Logic App

- I MS Azure: Gå til Dashboard / All resources og klikk på "Add"
- "Search the Marketplace" for "Logic App". Klikk på "Logic App" og velg "Create"
- Gi appen et navn, velg Subscription, Resource Group og location. I vårt tilfelle: WasteScanLogic, Azure for Students, wastescan og North Europe
- Klikk "Create"

Home > Logic Apps > Logic App

Logic Apps
USN

+ Add Edit columns More

Filter by name...

☐ Name ↑↓

☐ scantest ...

Logic App
Create

Name *
WasteScanLogic ✓

Subscription *
Azure for Students ▼

Resource group * ⓘ
☐ Create new ☒ Use existing
wastescan ▼

Location *
North Europe ▼

Log Analytics ⓘ

[Automation options](#)

Figure 10 - Opprettelse av Logic App

- Når appen er klar (dette kan ta opptil ett par minutter), klikk på "Refresh" og velg appen som akkurat er laget (WasteScanLogic)

Home > Logic Apps

Logic Apps
USN

+ Add Edit columns Refresh Assign tags Enable Disable Delete

Subscriptions: All 2 selected – Don't see a subscription? [Open Directory](#) + [Subscription settings](#)

Filter by name... All subscriptions ▼ All resource groups ▼

2 items

<input type="checkbox"/> Name ↑↓	Status	Resource group ↑↓
<input type="checkbox"/> scantest	✓	alf
<input type="checkbox"/> WasteScanLogic	✓	wastescan

Figure 11 - Oversikt over logic apps tilknyttet vår Azure konto

- I skjermbildet som kommer opp, velg "When a HTTP request is received"

IOT 3000 – Eksamen H19 – WasteScan av IoT Drømmeteamet

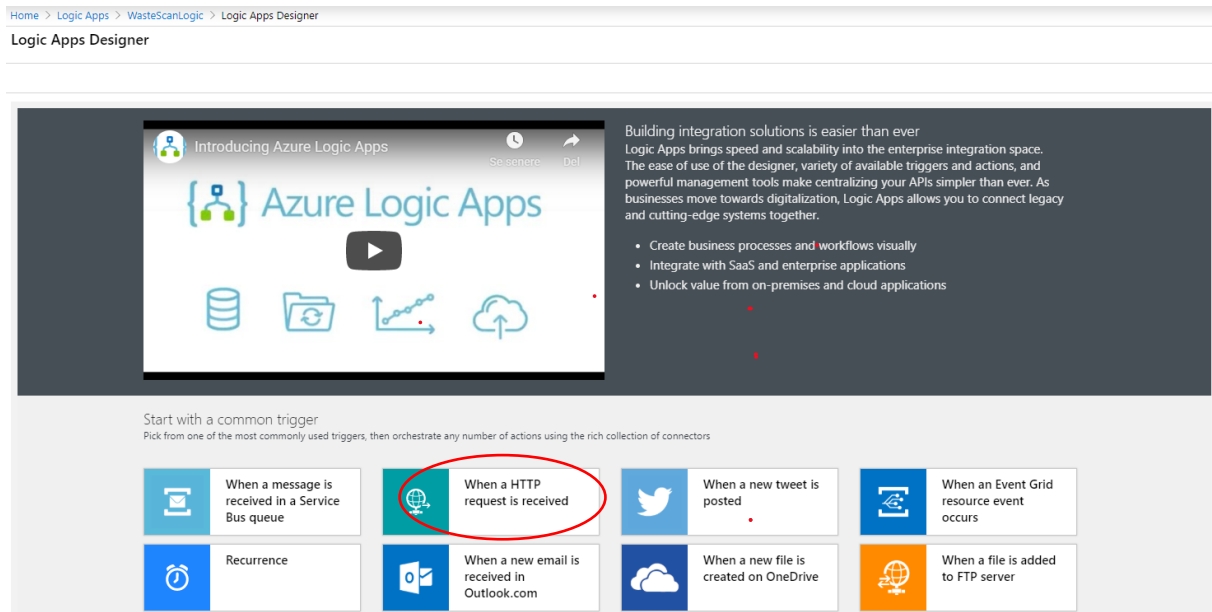


Figure 12 - Logic app designer startside

Step 1 – When a HTTP Request is received

- Klikk "Add new parameter", huk av for "Method" og velg "GET" i nedtrekksmenyen som kommer opp automatisk.
- Klikk på "Save" øverst i venstre hjørne
- Kopier innholdet i "HTTP GET URL". Dette skal senere inn i "Settings / Send to URL" i Barcode Scanner Terminal".

Logic Apps Designer

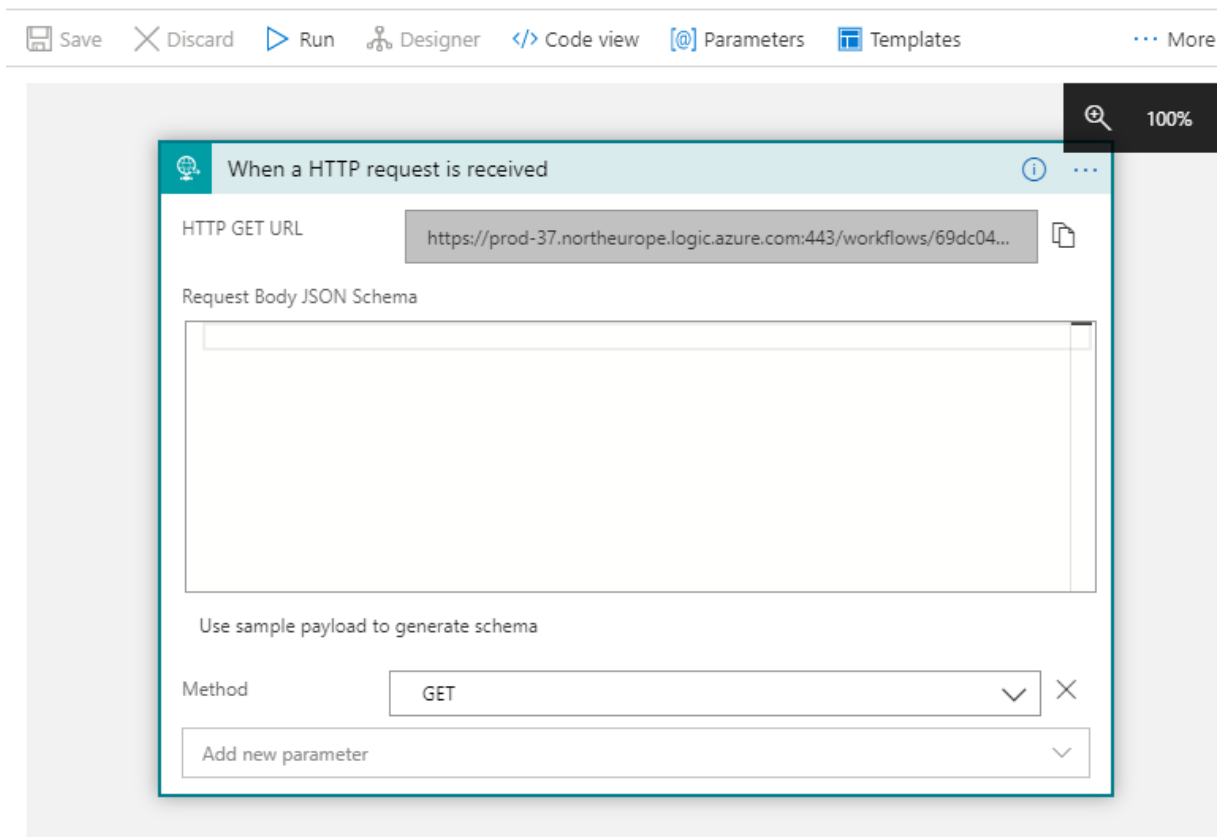


Figure 13 - Step 1: When a HTTP request is received

Step 2 – Parse JSON

- Klikk "New Step", søk etter "Parse JSON" og klikk på "Parse JSON – Data Operations".
- I steppet som akkurat ble laget, sett cursoren i feltet "Content", klikk "Add dynamic content" og velg "Queries" i listen over dynamisk innhold.
- Klikk "Use sample payload to generate schema, lim inn:

```
{ "data": "7090018952661", "id": "alfScanner", "lat": "", "long": "", "cont": "0", "ver": "3.8", "time": "1573993568433" }
```

Dette er eksempeldata fra Barcode Scanner Terminal og brukes for å fortelle "Parse JSON" hva slags data den kan forvente.
- Klikk "Done" og save.
Dette er eksempeldata fra Barcode Scanner Terminal og brukes for å fortelle "Parse JSON" hva slags data den kan forvente.

Logic Apps Designer

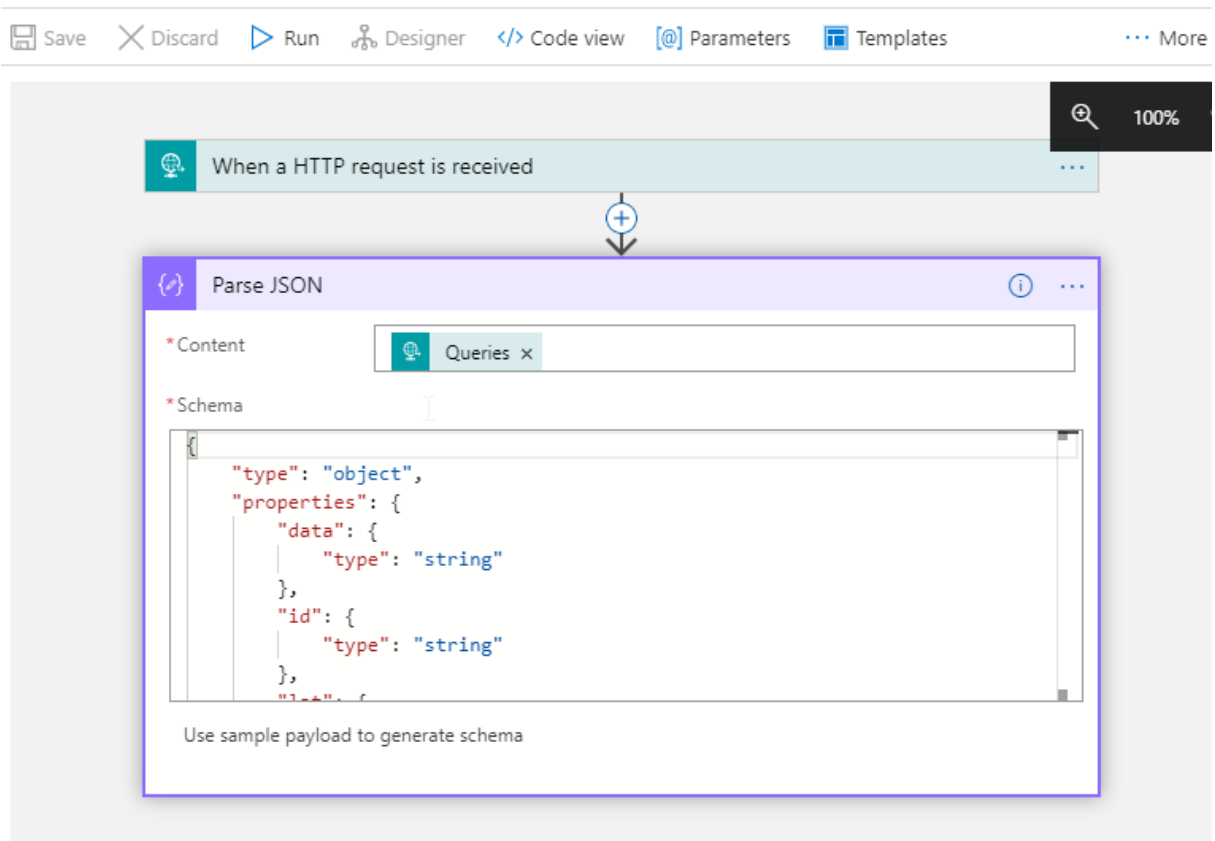


Figure 14 - Step 2: Parse JSON

Step 3 – Insert row

- Klikk "New Step", søk etter "SQL", klikk på "SQL Server" og velg "Insert row (V2)".
- Sett inn Server name, Database name og Table-name. I vårt tilfelle: wastescandbserver.database.windows.net, wastescandb og scanInput.
- Klikk "Add new parameter" og velg alle.
- I steppet som akkurat ble laget, sett cursoren i første feltet, klikk "Add dynamic content" og velg aktuelt objekt under "Parse JSON". Feltnavnet indikerer hvilket objekt som bør velges. data = data, scannerId = id osv.
- Gjenta siste instruks for alle feltene.

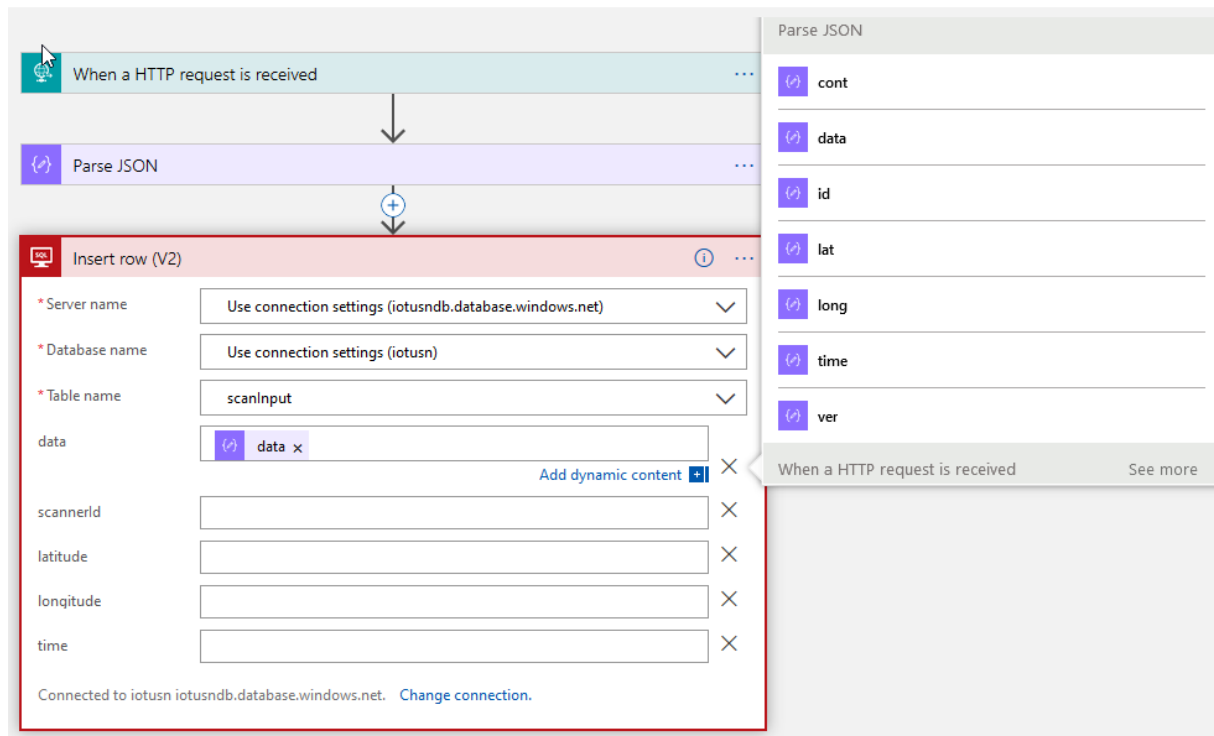


Figure 15 - Step 3: Insert row

Step 4 – Execute stored procedure

- Klikk "New Step", søk etter "SQL", klikk på "SQL Server" og velg "Execute stored procedure (V2)".
- Sett inn Server name, Database name og Procedure name.

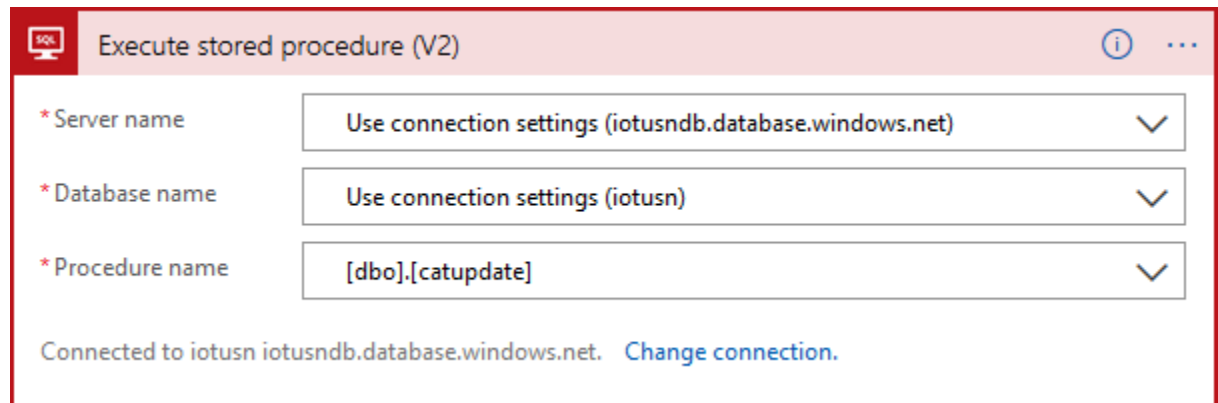


Figure 16 - Step 4: Execute stored procedure

Step 5 – Get row

- Klikk "New Step", søk etter "SQL", klikk på "SQL Server" og velg "Execute stored procedure (V2)".
- Sett inn Server name, Database name, Table name og Row id.

Get row (V2)

* Server name: Use connection settings (iotusndb.database.windows.net)

* Database name: Use connection settings (iotusn)

* Table name: categoryresult

* Row id: 1

Connected to iotusn iotusndb.database.windows.net. [Change connection.](#)

Figure 17 - Step 5: Get row

Step 6 – Request response

Denne delen sørger for tilbakemelding til telefon og skanner-app.

- Klikk "New Step", søk etter "request" og velg "Response request".
- Gjør valg som vist under.

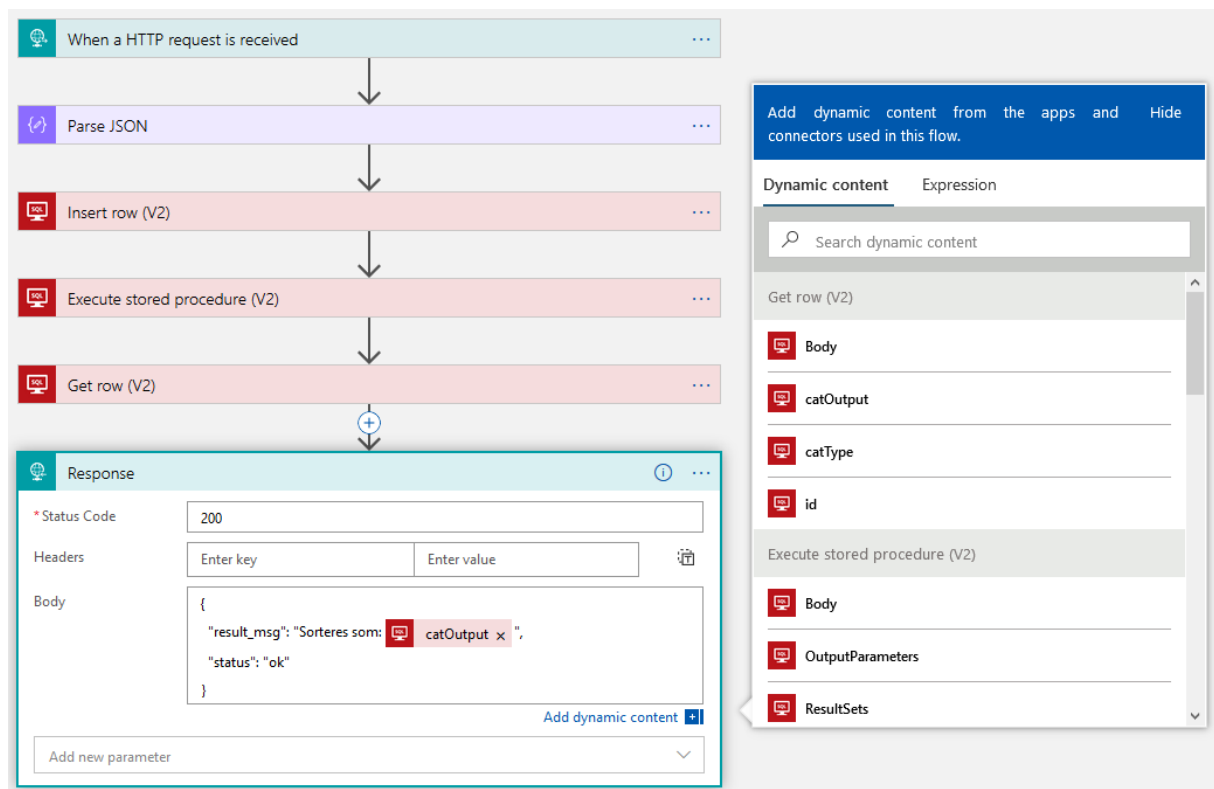


Figure 18 - Request response

Step 7 – POST request

Denne delen sender en HTTP POST request til IoT huben som på sin side aktiverer aktuell metode på IoT-devicen.

- Klikk "New Step", søk etter "http", klikk "HTTP" og velg "HTTP".
- Gjør valg som vist under.
deviceID, hostname og key er deviceinfo fra Azure IoT-hub

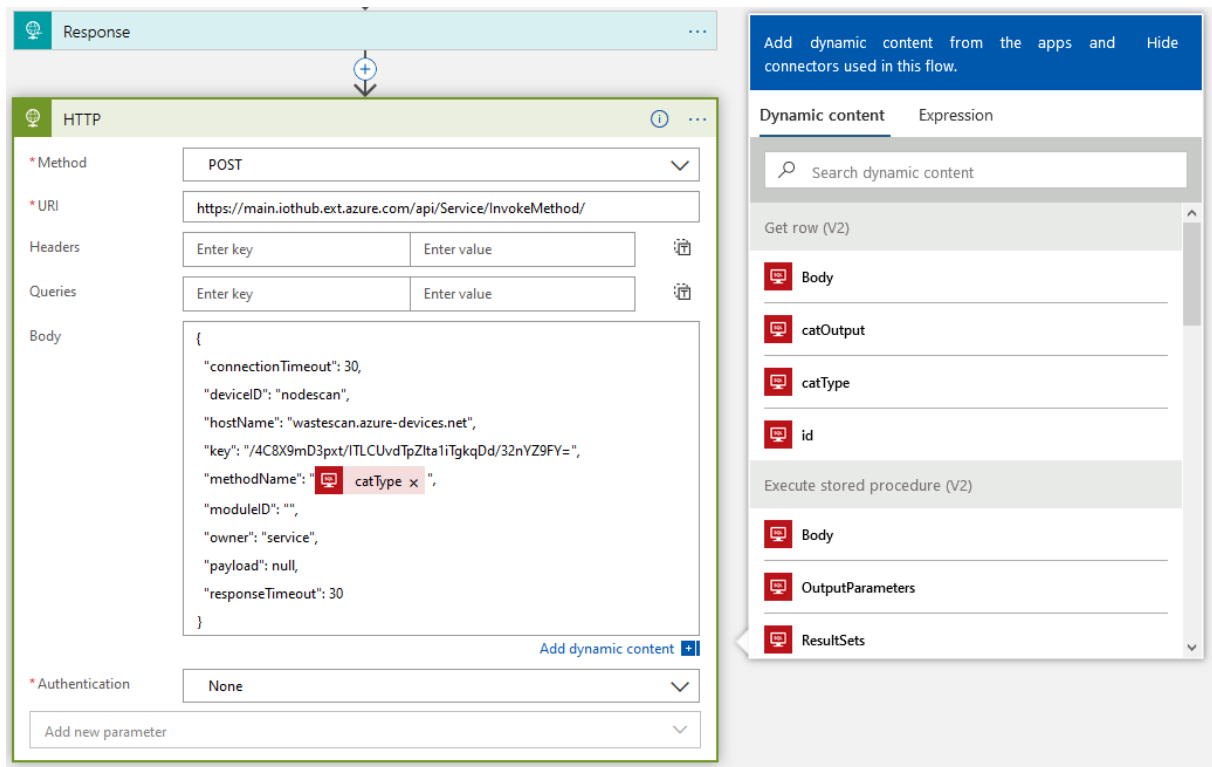


Figure 19 - Step 7: POST request

Barcode Scanner Terminal

Barcode Scanner Terminal kan installeres fra Google Play på adressen:

<https://play.google.com/store/apps/details?id=no.winternet.barcodescannerterminal3>

Appen har en del forskjellig funksjonalitet. Bla. kan den lagre strekkoder lokalt og teste de mot Excel-filer, plukke opp GPS-koordinater og sende scannerdata til Google Docs.

Til vårt prosjekt har vi brukt "Send to URL" og Scanning Device = Zxing Barcode Skanner. Det siste for å få litt større skanne-skjerm, som etter vår mening gir en bedre brukeropplevelse.

ZXing Barcode Scanner må eventuelt installeres separat. Fra Google Play på adressen:

<https://play.google.com/store/apps/details?id=com.google.zxing.client.android>

Oppsett

- Åpne Barcode Scanner Terminal, gå til Settings og oppdater "Scanner ID", "Processing mode", "Send to URL" og "Scanning device".
 - o "Scanner ID" og "Scanning device" er valgfritt. dvs. at konseptet vil fungere uansett hva som velges.
 - o "Processing Mode" skal være "Always only send to server"
 - o "Send to URL" skal inneholde connection-string fra første ledd i Logic App + strengen: "&data=%barcode%&id=%scannerID%&lat=%lat%&long=%long%"

I vårt tilfelle:

[https://prod-](https://prod-37.northeurope.logic.azure.com:443/workflows/69dc04b5d0da4cfd9068cb6b0a7a3546/triggers/manual/paths/invoke?api-version=2016-10-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=Q1UZ3TOmGBkrbFSWsTDNWuq8tOEEo2WBkZ3hesu49XQ&data=%barcode%&id=%scannerID%&lat=%lat%&long=%long%)

[37.northeurope.logic.azure.com:443/workflows/69dc04b5d0da4cfd9068cb6b0a7a3546/triggers/manual/paths/invoke?api-version=2016-10-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=Q1UZ3TOmGBkrbFSWsTDNWuq8tOEEo2WBkZ3hesu49XQ&data=%barcode%&id=%scannerID%&lat=%lat%&long=%long%](https://prod-37.northeurope.logic.azure.com:443/workflows/69dc04b5d0da4cfd9068cb6b0a7a3546/triggers/manual/paths/invoke?api-version=2016-10-01&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=Q1UZ3TOmGBkrbFSWsTDNWuq8tOEEo2WBkZ3hesu49XQ&data=%barcode%&id=%scannerID%&lat=%lat%&long=%long%)

Skjermbilder

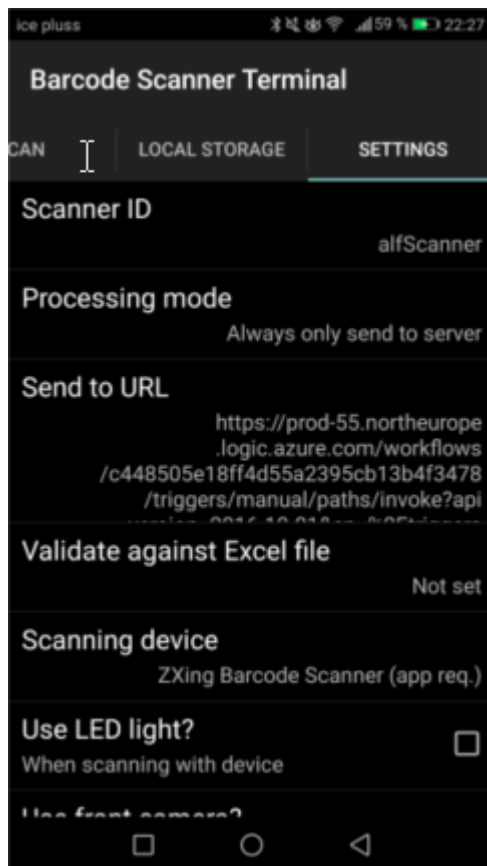


Figure 20 – Barcode Scanner Terminal: Settings

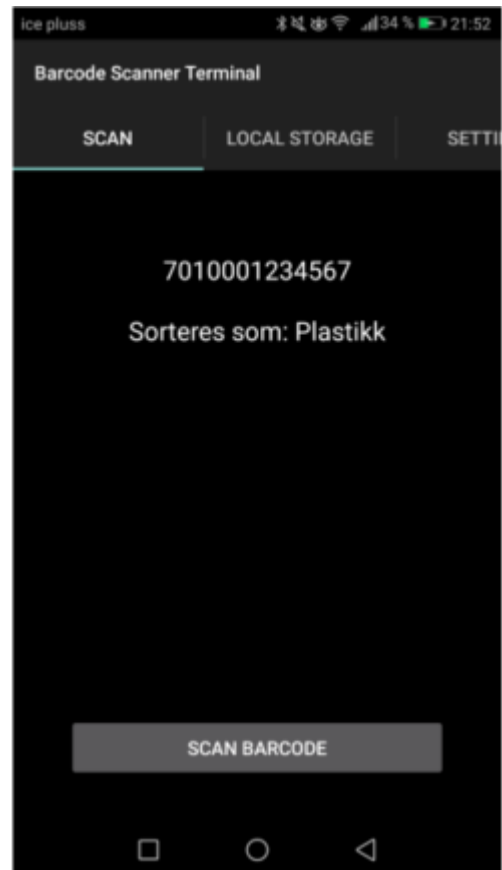


Figure 20 - Barcode Scanner Terminal: Resultat

Testing

Konseptet kan testet ved å skanne strekkodene under. Alle strekoder som ikke er definert i databasen vil returnere "Restavfall".



Beskrivelse av programkode

Prosjektet bruker en del Azure IoT tjenester som man trenger egne programbiblioteker for å kunne nyttegjøre seg av på en NodeMCU.

WasteScan-devicen har en ESP8266 chip som kan brukes til trådløs kommunikasjon og det er lett å finne mange kodeeksempler man kan benytte seg av.

Videre er det behov for å behandle JSON data samt håndtere et LCD-display. Biblioteker for dette må derfor også med. Våre biblioteker er som følger:

- ESP8266WiFi
- AzureIoTHub
- AzureIoTProtocol_MQTT
- AzureIoTUtility
- ArduinoJson
- LiquidCrystal

ESP8266WiFi brukes i koden vår for å sette opp trådløskommunikasjon.

AzureIoT bibliotekene brukes til kommunikasjon med Azure Iot Hub.

MQTT brukes til verifisering og er en kommunikasjons protokoll.

LiquidCrystal håndterer kommunikasjon mellom devicen og en LCD-skjerm.

JSON data håndteres med ArduinoJson.

Se ellers vedlegget Wastescan.ino.

Spesielle problemer

Vi opplevde i grunn få problemer under utviklingen av dette prosjektet men det ble litt prøving og feiling.

Vår første versjon hadde all logikk i Logic App. Det ble uoversiktlig, og vanskelig å vite hva som feilet når vi for eksempel ikke mottok return til skanner appen.

Vi startet også uten database. Når den kom på plass flyttet vi mye av logikken til en Stored Procedure, noe som løste de aller fleste utfordringene.

De første databasetabellene ble opprettet uten primærnøkler. Det førte til at Logic App-stepet "Get Row" returnerte uønskede verdier. Etter litt lesing av dokumentasjon oppdaget vi at Logic App sine SQL-triggere ikke fungerer som de skal uten primærnøkkel i tabellene.

Vi forsøkte oss også på en løsning med en SQLtrigger for å oppdatere tabellen "categoryresult". Det fungerte ikke i det hele tatt. Dvs. triggeren fungerte når vi aktiverte den direkte i database men av en eller annen grunn ble den ikke aktivert av Logic App-stepet "Insert Row". Igjen ble det en tur i dokumentasjonen, som kunne fortelle at nettopp dette ikke støttes av Logic App SQL-triggere. Løsningen ble å bruke Stored Procedures i stedet.

Fremtidig arbeid

- Stepmotorer til å låse opp søppelkassene alt ettersom type metode som sendes til devicen (dette var det egentlige oppdraget).
- Lamper som indikerer hvilken kasse som skal brukes.
- IoT-devicen kunne hatt sensorer for volum og vekt, som kunne fortalt avfallsselskapet / renholdere om når kassen burde tømmes og / eller hvor mye som skal betales hvis for eksempel en kunde betaler for vekt eller volum.
- Skaffe tilgang til, og kjøre spørringer mot EPD (database for produktinformasjon)
- Posisjonsdata kan vært tatt i bruk
- Webservice (Power BI e.l.) med statistikk av forskjellig slag
- Bruke timestamp til statistikk.

Litteraturliste

Bosu K., & Choudhuri R. (2017). *Learn Arduino Prototyping in 10 days*.
Birmingham: Packt Publishing Ltd.

Barcode Scanner Terminal Documentation

Hentet nov 2019 fra <http://winternet.no/barcodescannerterminal/#toc10>

Microsoft Azure IoT Device SDK for C

Hentet nov 2019 fra <https://docs.microsoft.com/en-us/azure/iot-hub/iot-c-sdk-ref/>

Connecting NodeMCU to Microsoft Azure IoT Hub

Hentet nov 2019 fra <https://www.thingforward.io/techblog/2018-05-22-connecting-nodemcu-to-microsoft-azure-iot-hub-part-1.html>

Connecting NodeMCU to Microsoft Azure IoT Hub - Part 2

Hentet nov 2019 fra <https://www.thingforward.io/techblog/2018-06-04-connecting-nodemcu-to-microsoft-azure-iot-hub-part-2.html>

Interfacing Nodemcu ESP8266 with Microsoft Azure Event Hub and using stream analytics

Hentet nov 2019 fra <https://www.youtube.com/watch?v=K7KEkEHNGPs>

Understand Different Connection Strings in Azure IoT Hub

Hentet nov 2019 fra <https://devblogs.microsoft.com/iotdev/understand-different-connection-strings-in-azure-iot-hub/>

HTTP Messages

Hentet nov 2019 fra <https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

Interfacing LCD with NodeMCU without using I2C

Hentet nov 2019 fra <https://circuitdigest.com/microcontroller-projects/interfacing-lcd-with-nodemcu>

Programming ESP8266 NodeMCU using arduino IDE

Hentet nov 2019 fra <https://www.youtube.com/watch?v=UdMEySEQCQk>

Powering the ESP-12E NodeMCU Development Board

Hentet nov 2019 fra <http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips-and-more/powering-the-esp-12e-nodemcu-development-board/>

Alternatives to Barcode Scanner for all platforms with Open Source License

Hentet nov 2019 fra <https://alternativeto.net/software/barcode-scanner/?license=opensource>