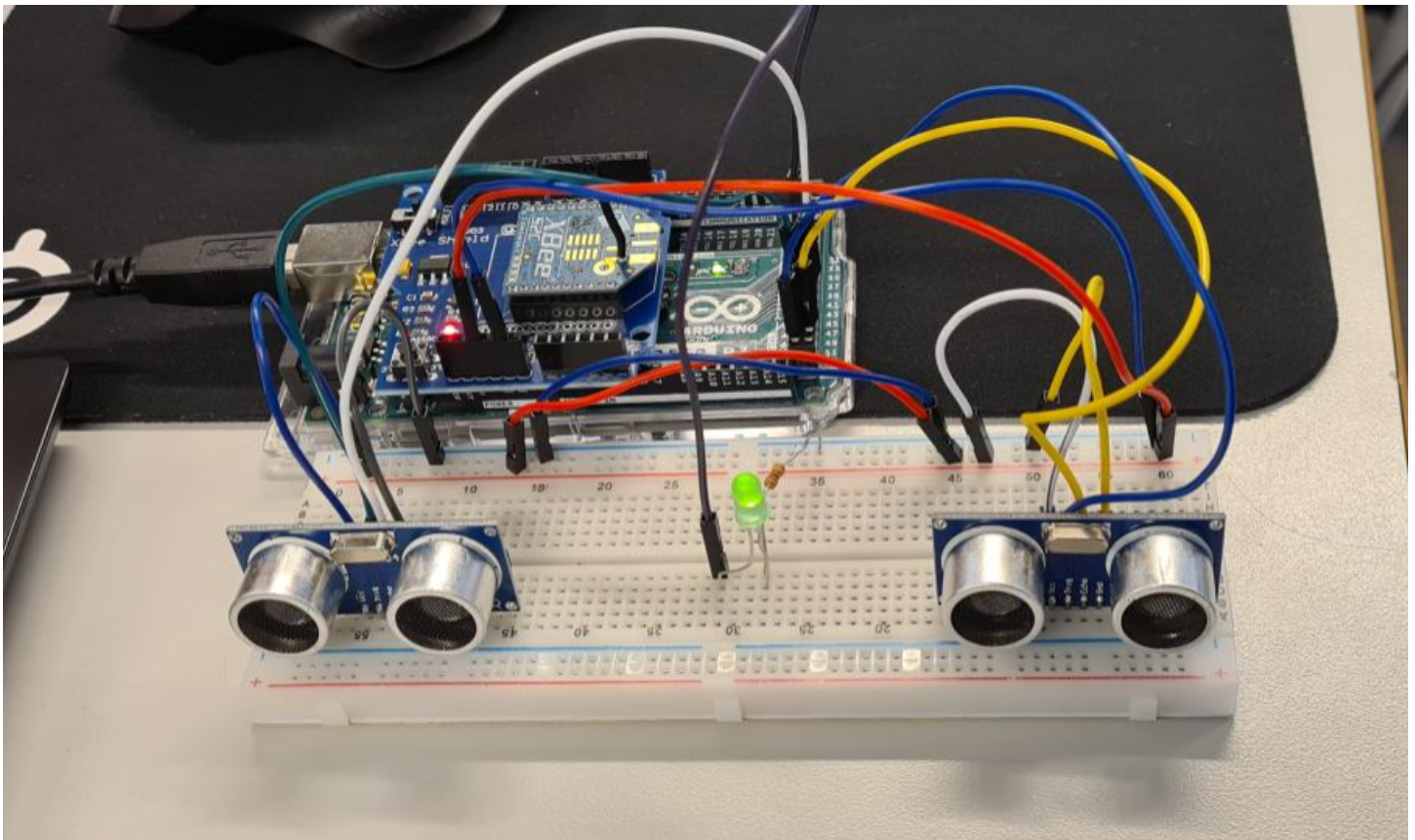


Michael Aggerholm
24-06-2022

H4 IOT Case

Dataoverførsel til MQTT broker via Xbee wifi.



Elev:

Michael Aggerholm

Projekt:

Registrering af butiks besøgs tal med ultralydssensor

Uddannelse:

Datatekniker med speciale i programmering

Projektperiode:

21/06/2022 – 24/06/2022

Afleveringsdato:

24/06/2022

TECHCOLLEGE

Techcollege Aalborg,
Struervej 70,
9220 Aalborg

Forord:	4
Case beskrivelse:	5
Kravspecifikation:	1
Kravoversigt:	1
Accepttestoversigt:	2
Hardware opstilling:	4
Beskrivelse af hardware opstillingen:	4
Liste af komponenter:	4
Program dokumentation:	5
Det har jeg forsøgt at beskrive til grunde herunder:	5
Arduino arbejdet:	5
Router Xbee:	6
Raspberry Pi opsætning:	7
Mosquitto MQTT Broker:	8
Coordinator Xbee:	8
Python script:	8
Fuld program kørsel:	9
Konklusion:	9
Diskussion:	10
Fordele / ulemper:	10
Reflektering:	11
Hvad har jeg lært?	11
Valgte jeg de rigtige teknologier?	11
Referencer:	12 - 13
Bilag:	14 - 21

Forord:

Rapporten her er skrevet til en case vi har fået som afsluttende opgave på hovedforløb 4 på "Datatekniker med speciale i programmering" uddannelsen som foregår på teknisk skole i aalborg.

Opgavens mål er at vi som elever skal vise og beskrive at vi igennem IOT faget har fået forståelse, indblik og kendskab til teknologierne og redskaberne der benyttes i embedded programmering.

Projektet er lavet delvist i samarbejde på tværs af eleverne i klassen, dog afleveret personligt som separate projekter.

Det vil sige at vi i klassen sammen har arbejdet os hen imod løsningerne som fremgår i rapporten, såsom data afsendelse og modtagelse via Xbee modulet, og opsætning af Raspberry Pi enheden.

Formålet med rapporten er ud fra min case at kunne sætte ord på hvorfor følgende valg er taget, samt at kunne beskrive hvorfor jeg har valgt at kode / udføre casen som jeg har gjort og vise forståelse for faget samt værktøjer og løsninger inden for IOT embedded programmering.

Case beskrivelse:

Tøjbutikken “HuddiFrutti Tøj” med placering i kernen af Aalborg har bedt om at få lavet en løsning til deres fysiske butik, som skal kunne registrere antallet af dagligt besøgende kunder i butikken, samt sørge for at lyset er tændt på overetagen så længe der er kunder i butikken.

I denne opstilling vil enhver besøgende være betragtet som kunde, selv hvis de ikke køber noget.

Opgaven er heller ikke opstillet som et tilbud, men som en opgave der skal laves uanset pris, derfor er der ingen dokumentation for pris samt tidsmæssig opstilling for erhvervsbrug.

For at kunne registrere kunder i butikken har jeg valgt kun at benytte to stks ultralydssensor som kan registrere om en kunde kommer ind eller forlader butikken.

Ét registreret kundebesøg skal registreres med ultralyd igennem et setup lavet på en Arduino MEGA2560.

Mit MEGA2560 board vil være udstyret med en Xbee som efterfølgende skal sende de registrerede kundebesøg via Wifi til en anden Xbee som er opsat i en Raspberry Pi, dennes opgave er at modtage dataen via Xbee, som et python script skal lytte på, herefter publicere registreringen til MQTT brokieren som er opsat i et docker miljø på Raspberry'en.

Se bilag for [opstart af docker miljø](#).

Kravspecifikation:

Kravsoversigt:

KravId	Kategori	Krav	Prioritering	Kilde	Type
A1	Arduino	Registrering af bevægelse til optælling af kundebesøg.	Høj	Case Opstartsevisions fasemodul 1	Funktionelt
A2	Arduino	Dataoverførsel til Raspberry Pi Xbee enhed.	Høj	Case Opstartsevisions fasemodul 1	Funktionelt
R1	Raspberry	Styresystem med mulighed for installation af python og diverse libraries.	Mellem	Case Opstartsevisions fasemodul 1	Praktisk
R2	Raspberry	Modtagelse af data fra Arduino MEGA2560 Xbee enhed.	Høj	Case Opstartsevisions fasemodul 2	Funktionelt
R3	Raspberry	håndtering af modtagen data samt. distribuering til MQTT Broker.	Høj	Case Opstartsevisions fasemodul 2	Funktionelt
B1	Broker	Opsætning i Dockermiljø	Lav	Case Opstartsevisions fasemodul 2	Praktisk
B2	Broker	Modtag data fra Raspberry pi.	Høj	Case Opstartsevisions fasemodul 2	Funktionelt
B3	Broker	Visning af modtaget data via. Command Line fra SSH adgang.	Mellem	Case Opstartsevisions fasemodul 3	Praktisk
S1	Sikkerhed	Pakker som overføres via Xbees i systemet skal være krypteret.	Høj	Case Opstartsevisions fasemodul 3	Sikkerhed

Accepttestoversigt:

KravId	Testbeskrivelse	Testkriterier	Bemærkninger	Status
A1	Der undersøges hvorvidt registrering af forbipasserende genstande måles og registreres.	Tælling skal forhøjes med 1 ved hvor forbipassering.	Delay time mellem registreringer bør præciseres yderligere ved tests.	Godkendt
A2	Der testes med XCTU om en pakke modtages.	Overførsel registreres med statuskode 200, samt at pakken indeholder læselig/brugbar data.	Pakken modtages som HEX værdi.	Godkendt
R1	Installation af styresystem samt samt netværksopsætning.	Enheden skal kunne tilgås via SSH med wifi adgang, der skal også kunne opdateres samt installeres Docker, python og libraries.	name/pass: (RASP-MICH AEL, Password) Bilag: Ping Raspberry Pi	Godkendt
R2	Modtagelse af data fra Arduino Xbee enhed.	At en pakke registreres modtaget til Xbee, det kan testes igennem XCTU.	Bilag: Coordinator og Router opsætning.	Godkendt
R3	Håndtering af dataen samt videredistribuerig.	Dette testes igennem python scriptet som opsætningen er foretaget i, ved at direkte printe modtagende pakker inden videresendelse til MQTT broker.	ingen	Godkendt
B1	Opsætning i Docker.	Efter opsætning køres docker containeren og der testes om porten 1883 som MQTT brokern kører på er i brug samt at docker containeren er kørende.	port opsætning: 1883:1883. Bilag: Publish og bilag: Subscribe for dokumentation af tests	Godkendt

B2	Modtag data fra python scriptet som håndtere indkommende packages.	Testes ved at subscribe til subjektet som modtages, og se om der bliver published beskeder ved modtagelse.	ingen Bilag af script kaos fra uendelige tests	Godkendt
B3	Visning af modtagne data.	Ved at subscribe til det publicerede subjekt, og på denne måde se indkomne beskeder.	ingen	Godkendt
SI	Der testes med Wireshark.	Pakkerne som forlader Arduino Xbee modulet er i krypteret tilstand.	ingen	Godkendt

Hardware opstilling:

Beskrivelse af hardware opstillingen:

Jeg har forbundet mit breadboard til spænding og jord på mit Arduino board, for let tilgængelighed på breadboard. Efterfølgende har jeg tilsluttet to ultralydssensorer i hver sin ende af breadboard, den sensor jeg vil bruge til indgangs registrering har jeg sat til echo pin 51 og trigger pin 52 i Arduino boardet.

Ultralydssensoren til registrering af udgang er sat til echo pin 41 og trigger på 42 på Arduino boardet.

I midten af breadboardet har jeg min LED som i mit scenarie repræsenterer lyset i butikken, den har en 12 Ohm's modstand til Jord, og en trigger til pin 30 på Arduino boardet.

På min Arduino har jeg et Xbee shield hvor min Xbee S2C sidder monteret.

Bilag [Se Opsætning af Arduino enhed](#).

I min Raspberry Pi har jeg én Xbee S2C koblet på via en Sparkfun Xbee USB enhed.

Se bilag med [Raspberry Pi som har Xbee forbundet](#).

Liste af komponenter:

- Arduino MEGA2560 R3.
- 2x Xbee S2C.
- Arduino Xbee Shield.
- Grøn LED.
- Sparkfun Xbee board USB.
- Raspberry PI.
- 2x Ultrasonic sensor HC-SR04.
- 12x han/han wire.
- Breadboard 830 huller.
- 12 Ohm modstand.

Program dokumentation:

Programmet er opdelt i to dele, som hver har flere opdelinger af funktionaliteter.

Der er Arduino delen, som blandt andet indeholder to stks. ultralydssensorer til registrering af hvornår en person går ind og ud, samt en pære som kun lyser når der er personer i butikken, samt en Xbee som trådløst sender data ved adgangsregistrering til vores Raspberry Pi del.

Raspberry Pi delen er opsat med et headless ubuntu, her registrere vi modtagende pakker med endnu et Xbee modul, det hele sker igennem et python script som lytter på Xbee modtagelse, derefter processere vi dataen og publisher til vores MQTT broker, som i dette tilfælde er opsat i sit helt eget docker miljø her på Raspberry'en. Ved at subscribe til de/det topic der publishes via python scriptet.

Det har jeg forsøgt at beskrive til grunde herunder:

Arduino arbejdet:

Her definere jeg to stks ultralydssensorer, den ene er registreret som indgangs sensor og den anden som udgangs sensor. Disse er separeret ved delay, det vil sige at når enten udgang eller indgang registreres, kommer der et delay som forhindre modparten i at registrere oveni.

Jeg har herefter defineret en LED og en entrance counter, som starter på 0, derefter tæller den op med én hver gang indgang registreres, og reducerer med én ved udgangs registrering.

Se bilag for [billede af ind og udgangs registreringer samt optælling](#).

LED lampen lyser så længe entrance counteren ikke er 0.

For at forhindre at der kommer flere tællinger med per ultralydssensor registrering, har jeg sat begge sensorers distance til 1000 i enden af hvert registrerings loop.

Koden for registrering af indgang har jeg indsat herunder for hurtig reference:

```
if(distanceIn <= 20){  
    digitalWrite(LED_PIN, HIGH);  
  
    payload[0] = 1;  
    payload[1] = 2;  
  
    zbTx = ZBTxRequest(addr64, payload, sizeof(payload));  
  
    xbee.send(zbTx);  
    distanceIn = 1000;  
    entrance++;  
    delay(1000);
```

Koden for registrering af udgang har jeg indsat herunder for hurtig reference:

```
else if(distanceOut <= 20){
    digitalWrite(LED_PIN, HIGH);

    payload[0] = 1;
    payload[1] = 2;

    zbTx = ZBTxRequest(addr64, payload, sizeof(payload));

    distanceOut = 1000;

    if(entrance >= 1){
        entrance--;
    }

    delay(1000);
```

Router Xbee:

Xbee modulet jeg benytter i min arduino er opsat som en Router med MAC adressen "0013A2004155B980", opsætningen er sket i programmet "XCTU".

Se bilag for [afsendelse af pakker som Router](#).

I arduino pro IDE'en benytter jeg biblioteket Xbee af "Andrew Rapp", herefter definere jeg et nyt object af instancen Xbee.

Se bilag for [Xbee library pakken af Andrew Rapp](#).

Så definere jeg et array kaldet payload, som kommer til at indeholde min data som skal sendes.

Herefter sætter jeg MAC adressen på min Coordinator Xbee, som er modtager Xbee modulet, denne skal sendes som to stks 8 bit adresser.

Koden her for hurtig reference:

```
// XBee object
XBee xbee = XBee();

uint8_t payload[] = { 0, 0 };

XBeeAddress64 addr64 = XBeeAddress64(0x0013A200, 0x4155B959); // Coordinator MAC
Addr.
ZBTxRequest zbTx;
```

Når en indgang registreres af min ultralydssensor, kan jeg sætte distance eller lignende som min payload, og sende dette til min Coordinator, i dette tilfælde har jeg dog sat payload til altid at indeholde hex værdierne 1 og 2, blot for simplicitet.

Se bilag [Coordinator modtager Zigbee data](#).

herefter benytter jeg Xbee bibliotekets prædefineret metode ZBTxRequest til at erklære adressen min payload skal sendes til, samt payload der skal sendes. Herefter har jeg et 2 sekunders delay der forhindre dobbel registrering.

Raspberry Pi opsætning:

Min Raspberry Pi er opsat med et headless ubuntu OS, det er gjort igennem den officielle arduino image installer, her opsætter jeg navnet, koden samt netværket for enheden, for efterfølgende at kunne tilgå via SSH. Når enheden er installeret og startet tilgår jeg den via SSH med mit brugernavn og kodeord (RASP-MICHAEL, Password). Bilag [Putty adgang til Raspberry Pi](#). og Bilag for [success af forbindelse via SSH](#).

Herefter kører jeg en række opdateringer og installationer, for at sikre mig at enheden er klar til opsætningen af mit script samt MQTT.

Til at starte med, sørger jeg for at mit OS er opdateret, derefter installerer jeg følgende pakker:

- Docker
- mosquito-clients
- Python3.8+
- Pip
- Git

Efter installationen ved jeg at jeg skal bruge to python biblioteker, dem installerer jeg ved brug af pythons pip installatør:

- digi-xbee
- paho-mqtt

Nu er enheden klar til at få opsat mosquitto MQTT broderen i et docker miljø, samt at kunne køre vore python script til modtagelse og distribuering af Xbee data modtagelse.

Mosquitto MQTT Broker:

Jeg har valgt at opsætte min MQTT broker i sit helt eget docker miljø her på min raspberry.

Det har jeg gjort ved at lave et git clone på Mosquitto's officielle docker hub image.

Herefter kører jeg blot mit image, som starter MQTT containeren på port 1883.

Kommandoen for opstart af docker miljø:

```
docker run -it --name mosquitto -p 1883:1883 eclipse-mosquitto
```

Det hele tester jeg ved at lave en test publish af et topic, derefter subscribe til test topic i en ny SSH terminal hvor jeg kan bekræfte at beskederne bliver published.

Kommandoer til Publish og Subscribe tests af MQTT:

```
mosquitto_sub -h localhost -t "mqtt/test"  
mosquitto_pub -h localhost -t "mqtt/test" -m "Hello world"
```

Coordinator Xbee:

Xbee modulet jeg benytter i min arduino er opsat som en Coordinator med MAC adressen "0013A2004155B965", opsætningen er sket i programmet "XCTU".

Se bilag for [modtagelse af pakker som Coordinator](#).

Da vi kun bruger denne til at lytte på, og ikke til at sende fra, er opsætningen sådan set færdig, der er ikke behov for yderligere kode, og jeg er nu klar til at opsætte python scriptet som skal lytte på modulet.

Python script:

Nu er jeg parat til at modtage noget data, som jeg kan give min broker.

Scriptet er en enkelt python fil, hvor jeg lytter på min raspberry port "/dev/ttyUSB0" som er mit Xbee modul.

Koden for at kunne lytte på Xbee USB porten kan ses herunder:

```
xbee = XBeeDevice("/dev/ttyUSB0", 9600)  
xbee.open()
```

dataen jeg vil modtage definere jeg som bound rate 9600, da det er standarden for arduinos seriel kommunikation.

herefter opretter jeg forbindelse til min broker med "localhost" adressen og den førnævnte port 1883.

Koden for oprettelse af forbindelse til MQTT broker kan ses herunder:

```
broker = "localhost" # port  
port = 1883  
client = paho.Client(client_id="client", userdata=None,  
protocol=paho.MQTTv5)
```

For kontinuerligt at kunne modtage og publicere pakker, har jeg et while loop som kører så længe I er I, hvilket altid vil være korrekt.

I mit while loop læser jeg alt hvad der kommer fra min Xbee enhed, og hvis ikke der kommer noget, vil værdien være None, som også kaldes Null i andre programmeringssprog.

Når beskeden ikke er None, sætter jeg den nuværende tid som et timestamp med dato, samt en besked om at en person er gået ind i rummet, og publicere til mit i dette tilfælde test topic på brokeren.

Koden i mit While loop som publicere message kan ses herunder:

```
while 1 == 1:
    # Read data.
    xbee_message = xbee.read_data()
    if(xbee_message != None):
        # telemetry to send
        x = datetime.datetime.now()
        message = (x.strftime("%x") + x.strftime(" %X : Person entered the
room!"))
        # publish message
        ret = client.publish("mqtt/test", message)
```

Se bilag for [publicering af registreringer](#).

Fuld program kørsel:

Efter alt opsætning og kode nu er på plads, åbner jeg en SSH terminal igennem programmet Putty, og subscriber jeg til mit test topic på brokeren.

Derefter laver jeg en indgangs registrering og ser at beskeden kommer frem i min SSH terminal.

Se bilag om [modtagelse af indgangs registreringer](#).

Konklusion:

Tøjbutikken HuddiFrutti tøj kan nu, med denne løsning registrere ud og indgang af besøgende i butikken, samt at lyset på overetagen og eller andre rum automatisk tændes hvis der er kunder i butikken.

Ud fra min kravspecifikation kan jeg se at konkludere at jeg er kommet i mål med mine krav, samt tests af systemet. Og ved fysisk tests i klassen kan jeg se at både personlige og kollegiale tests går rent igennem, de er blandt andet dokumenteret i bilag og linket under mine accepttests.

Udfordringerne jeg igennem projektet har stødt på har hovedsageligt været kommunikationen af data mellem mine Xbee moduler, samt publicering af beskeder fra mit python script til MQTT brokeren, da localhost forbindelsen til en MQTT broker i et docker miljø havde bestemte krav ift. at man skulle specificere sig selv som værende Admin.

Udover dette har der været brugt en del tid på opsætning af docker miljøet, da min Raspberry Pi af ukendte årsager havde reserveret porten 1883, derfor definere jeg den udadtil som værende 11883 i stedet.

Dog er den stadig 1883 inde i containeren. Det kan ses på [bilaget her](#).

Diskussion:

Fordele / ulemper:

- *Ultralydssensor :*

Ulempen ved brug af ultralydssensor kan være præcisionen, da ultralyds præcision forværres drastisk jo længere afstand der er til nærmeste flade/væg eller lign. Her kunne i stedet være benyttet laser moduler til registrering af ind og udgang. Eller endnu bedre et kamera som ved hjælp af Machine Learning vil kunne registrere personer ved indgang samt udgang.

- *Xbee S2C :*

Fordelen ved brug af Xbee modulet er at pakker som sendes mellem Xbee modulerne let kan krypteres helt automatisk ved en enkelt indstilling under opsætning af modulerne i XCTU ved at sætte "EE (Encryption Enable)" til I.

- *MQTT Broker :*

Fordelen ved brug af en MQTT broker er at sensorer og andre IOT enheder let kan publicere data til brokieren, og at subscribers let får adgang til at subscribe til de data der har relevans for dem. Plus at enhver subscriber let kan publicere yderligere data til brokieren som evt. partnere kan subscribe til. På denne måde er der let tilgængelighed til data ved at subscribe til brokieren.

- *Sikkerhed :*

Fordelen ved dette setup er det sikkerhedsmæssige aspekt, da beskederne som overføres via Wifi er krypterede, samt der er password på netværket som brokieren kører på. Der kan opsættes brugeradgang med krypteres password for adgang til Brokieren, et link til denne er defineret under referencepunktet: [MQTT Authentication](#). Samt at forbindelsen til brokieren kan opsættes med TLS / SSL.

- Transport Layer Security og Secure Sockets layer er en handshake opsætning som sikrer og krypterer forbindelsen mellem enhederne. TLS / SSL

Reflektering:

Hvad har jeg lært?

Jeg føler at have lært opsætningen af arduino board med diverse moduler, samt hvordan pakker af data sendes mellem forskellige enheder ved hjælp af Xbee enhederne. Samt hvordan en MQTT broker fungerer ved hjælp af Topic, Publisher og Subscribers.

Det har været et godt projekt som giver en god forståelse for processen af at få sensor data sendt til en broker som gør dataen let tilgængelig via subscribing til topics.

Jeg har fået forståelse for brugen af de anvendte moduler samt enheder og hvordan kommunikationen på tværs af to sprog, som i dette tilfælde er C++ og Python faktisk ikke er besværligt, men til gengæld har lettet processen ved at anvende de korrekte biblioteker.

Ved brug af de forskellige teknologiers officielle dokumentation føler jeg at opgaven har været mere eller mindre lige til, der har været små udfordringer, dog har der været svar at hente da teknologierne er bredt anvendte i den virkelige verden.

Valgte jeg de rigtige teknologier?

Til opgavens formål vil jeg ud fra min nuværende viden mene at de korrekte teknologier til løsningen er valgt.

Ultralyds sensorerne registrere ud og indgang som de bør, og præcise nok til projektet målestok.

Xbee enhederne transportere let og sikkert dataen imellem hinanden og er let tilgængelig ved at lytte på Xbee USB porten.

MQTT brokern har været let at opsætte via docker, det er let tilgængelighed via SSH.

Testing har været lige til, ved hjælp af pub / sub kommandoerne som er tilgængelige i Mosquittos officielle dokumentation.

Opsætning samt forbindelse til Raspberry er gjort let ved brug af Raspberry image installeren og Putty. derefter har det ikke taget længe at lave det nødvendige miljø ved hjælp af ubuntu's apt-get package manager.

Referencer:

MQTT Authentication:

<https://www.hivemq.com/blog/mqtt-security-fundamentals-authentication-username-password/>

MQTT Sikkerhed:

<https://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals/>

Mosquitto dokumentation:

<https://mosquitto.org/documentation/>

Arduino dokumentation for kommunikation til MQTT broker:

<https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device>

Dokumentation for registrering af afstand med ultralyd HC-SR04:

<https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>

Feabhas guide til opsætning af MQTT Broker i docker:

<https://blog.feabhas.com/2020/02/running-the-eclipse-mosquitto-mqtt-broker-in-a-docker-container/>

Xbee Kryptering:

https://www.digi.com/resources/documentation/Digidocs/90001506/reference/r_encryption.htm?TocPath=Networking%7C_____4

MQTT Docker kommunikation med Python:

<https://thilake.medium.com/get-into-mqtt-in-2-minutes-python-docker-5d4e8b55cf1c>

Hvordan Paho biblioteket anvendes:

<http://www.steves-internet-guide.com/into-mqtt-python-client/>

<https://www.hivemq.com/blog/mqtt-client-library-paho-python/>

Skrivning af forord i rapporten:

<https://omatskrive.dk/rapporter-opgaver-praesentationer/rapportens-detajler/forord-og-indledning-hvad-er-hvad/>

Digi Xbee guide til data transfer:

<https://www.digi.com/support/forum/74379/xbee-api-mode-coordinator-router-end-device?show=74379#q74379>

https://xbplib.readthedocs.io/en/latest/user_doc/communicating_with_xbee_devices.html#communicatereceive_data

Installation af python packages med Pip:

<https://packaging.python.org/en/latest/tutorials/installing-packages/>

Test af MQTT Brokren:

<https://pimylifeup.com/raspberry-pi-mosquitto-mqtt-server/>

Bilag:

Arduino serial registrering af ind og udgang

```
Output  Serial Monitor X
Message (Ctrl + Enter to send message to 'Optibo

IN
Number of people in room: 1
IN
Number of people in room: 2
OUT
Number of people in room: 1
OUT
Number of people in room: 0
```

MQTT publish to topic test

```
RASP-MICHAEL@RASP-MICHAEL: ~
RASP-MICHAEL@RASP-MICHAEL:~ $ mosquitto_pub -h localhost -t "mqtt/pimylifeup" -m "Hello world"
RASP-MICHAEL@RASP-MICHAEL:~ $
```

MQTT Subscribe to topic test

```
RASP-MICHAEL@RASP-MICHAEL: ~
RASP-MICHAEL@RASP-MICHAEL:~ $ mosquitto_sub -h localhost -t "mqtt/pimylifeup"
Hello world

```

Modtagelse af indgangs registreringer

```
RASP-MICHAEL@RASP-MICHAEL: ~
RASP-MICHAEL@RASP-MICHAEL:~ $ mosquitto_sub -h localhost -t "mqtt/test"
06/23/22 13:28:27 : Person entered the room!
06/23/22 13:28:27 : Person entered the room!
06/23/22 13:28:28 : Person entered the room!
06/23/22 13:28:29 : Person entered the room!
```

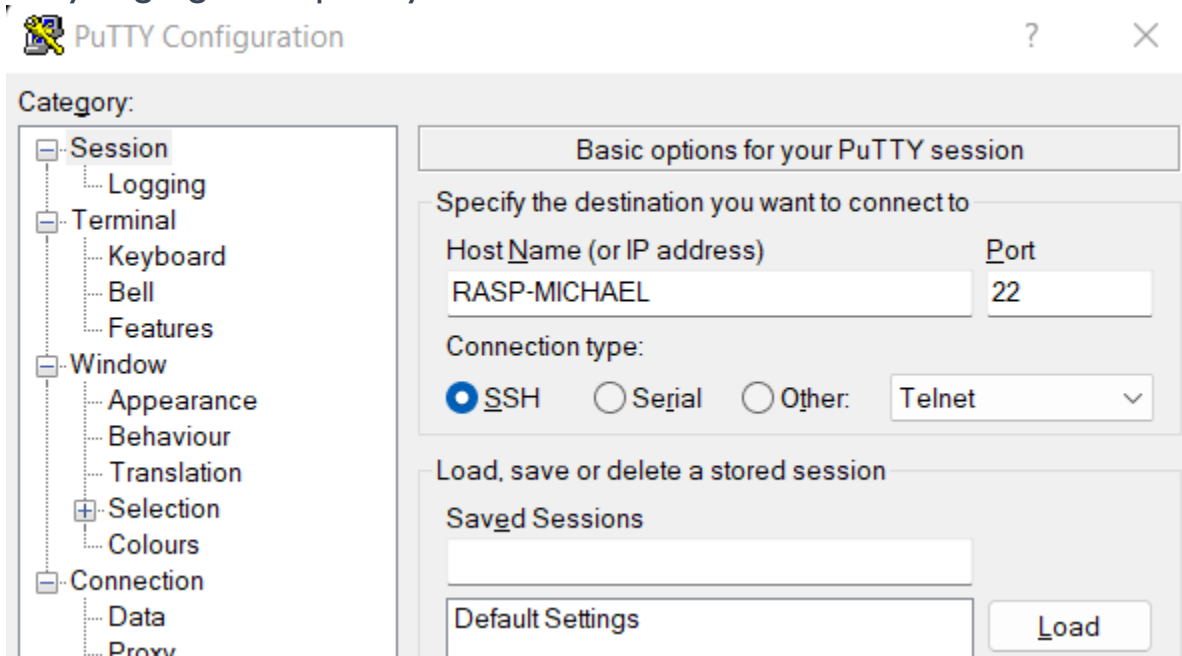
Ping Raspberry Pi

```
Windows PowerShell
PS C:\Users\92agg> ping RASP-MICHAEL

Pinging RASP-MICHAEL.local [fe80::b373:7ace:ceb3:9824%8] with 32 bytes of data:
Reply from fe80::b373:7ace:ceb3:9824%8: time=2ms
Reply from fe80::b373:7ace:ceb3:9824%8: time=3ms
Reply from fe80::b373:7ace:ceb3:9824%8: time=3ms
Reply from fe80::b373:7ace:ceb3:9824%8: time=4ms

Ping statistics for fe80::b373:7ace:ceb3:9824%8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 4ms, Average = 3ms
PS C:\Users\92agg>
```

Putty adgang til Raspberry Pi



SSH Coordinator modtager Zigbee data

```
RASP-MICHAEL@RASP-MICHAEL: ~  
RASP-MICHAEL@RASP-MICHAEL:~ $ nano script.py  
RASP-MICHAEL@RASP-MICHAEL:~ $ python script.py  
NO! Package were not sent!  
NO! Package were not sent!  
NO! Package were not sent!  
^CNO! Package were not sent!  
NO! Package were not sent!  
NO! Package were not sent!  
NO! Package were not sent!  
NO! Package were not sent!  
NO! Package were not sent!  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')  
bytearray(b'\x03\x04')
```

Tests af python script kaos!

```
RASP-MICHAEL@RASP-MICHAEL: ~  
RASP-MICHAEL@RASP-MICHAEL:~ $ ls  
opgave2  r2.py  r4.py  r6.py      script3.py  script5.py  t1.py  t3.py  t5.py  t7.py  t9.py  test3.py  test5.py  
r1.py    r3.py  r5.py  script2.py  script4.py  script.py   t2.py  t4.py  t6.py  t8.py  test2.py  test4.py  test.py  
RASP-MICHAEL@RASP-MICHAEL:~ $
```

SSH pubicer registrering

```
RASP-MICHAEL@RASP-MICHAEL: ~  
RASP-MICHAEL@RASP-MICHAEL:~ $ python r6.py  
Device 1 : Data published.  
Device 1 : Data published.  
Device 1 : Data published.  
Device 1 : Data published.  
█
```

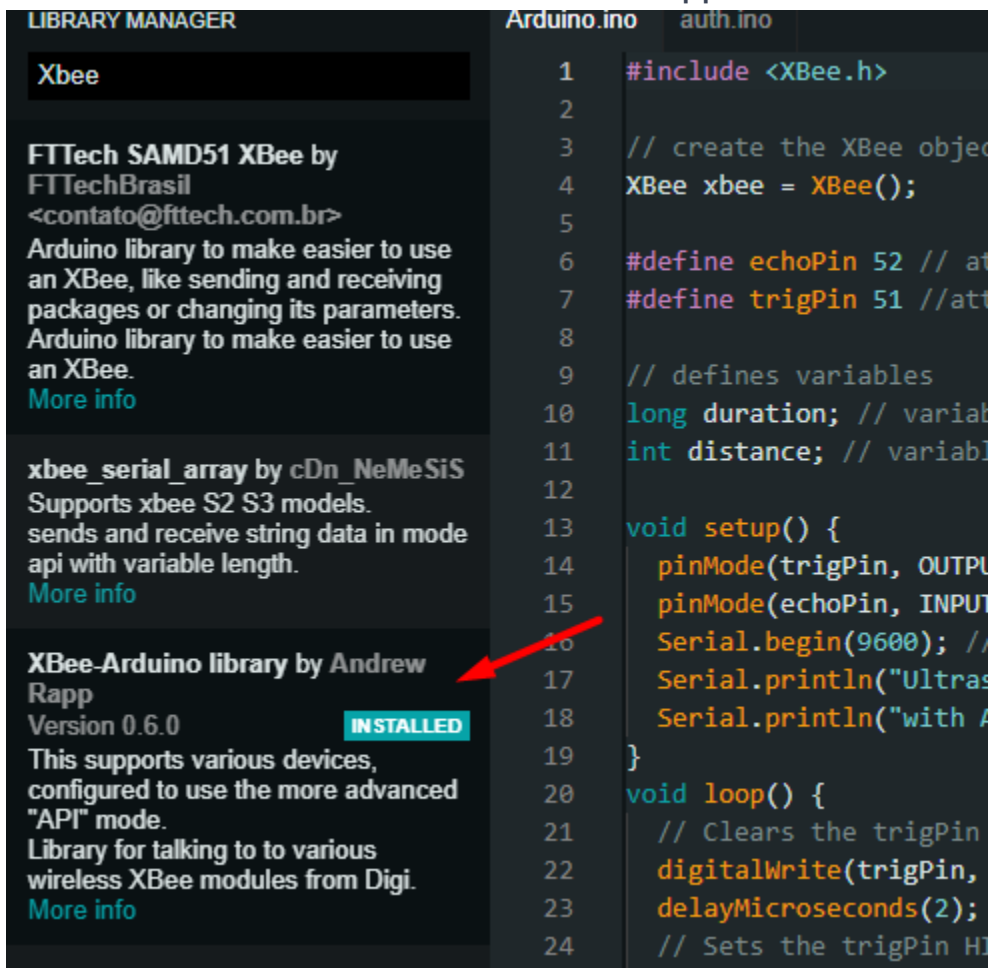
Forbundet til Raspberry via SSH

```
RASP-MICHAEL@RASP-MICHAEL: ~  
login as: RASP-MICHAEL  
RASP-MICHAEL@RASP-MICHAEL.local's password:  
Linux RASP-MICHAEL 5.15.32-v8+ #1538 SMP PREEMPT Thu Mar 31 19:40:39 BST 2022 aarch64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Tue Jun 21 09:43:36 2022  
RASP-MICHAEL@RASP-MICHAEL:~ $
```

SSH opstart af MQTT Docker miljø

```
RASP-MICHAEL@RASP-MICHAEL: ~  
RASP-MICHAEL@RASP-MICHAEL:~ $ docker run -it --name mosquitto -p 11883:1883 eclipse-mosquitto  
Unable to find image 'eclipse-mosquitto:latest' locally  
latest: Pulling from library/eclipse-mosquitto  
455c02918c45: Pull complete  
259f015a3f0d: Pull complete  
4c46f561843a: Pull complete  
Digest: sha256:43b90568c315eeae5cbdc75ef41aa109aef2170bc714443fe3586d565783d18  
Status: Downloaded newer image for eclipse-mosquitto:latest  
1655798182: mosquitto version 2.0.14 starting  
1655798182: Config loaded from /mosquitto/config/mosquitto.conf.  
1655798182: Starting in local only mode. Connections will only be possible from clients running on the  
1655798182: Create a configuration file which defines a listener to allow remote access.  
1655798182: For more details see https://mosquitto.org/documentation/authentication-methods/  
1655798182: Opening ipv4 listen socket on port 1883.  
1655798182: Opening ipv6 listen socket on port 1883.  
1655798182: Error: Address not available  
1655798182: mosquitto version 2.0.14 running  
█
```

Arduino IDE Xbee bibliotek af Andrew Rapp



The screenshot displays the Arduino IDE interface. On the left, the 'LIBRARY MANAGER' window is open, showing a list of libraries. The 'XBee-Arduino library by Andrew Rapp' is highlighted, with a red arrow pointing to it. This library is marked as 'INSTALLED'. The description for this library states: 'This supports various devices, configured to use the more advanced "API" mode. Library for talking to various wireless XBee modules from Digi. More info'.

On the right, the 'Arduino.ino' file is open in the code editor. The code is as follows:

```
1 #include <XBee.h>
2
3 // create the XBee object
4 XBee xbee = XBee();
5
6 #define echoPin 52 // at
7 #define trigPin 51 //att
8
9 // defines variables
10 long duration; // variab
11 int distance; // variabl
12
13 void setup() {
14   pinMode(trigPin, OUTPUT
15   pinMode(echoPin, INPUT
16   Serial.begin(9600); //
17   Serial.println("Ultras
18   Serial.println("with A
19 }
20 void loop() {
21   // Clears the trigPin
22   digitalWrite(trigPin,
23   delayMicroseconds(2);
24   // Sets the trigPin HI
```

XCTU Coordinator og Router opsat

The screenshot shows the XCTU software interface. On the left, the 'Radio Modules' list contains two entries:

- Router:** Function: ZIGBEE TH Reg, Port: COM8 - 9600/8/N/1/N - API 2, MAC: 0013A2004155B980
- Coordinator:** Function: ZIGBEE TH Reg, Port: COM7 - 9600/8/N/1/N - API 2, MAC: 0013A2004155B965

The main window displays the 'Radio Configuration [Router - 0013A2004155B980]' settings. The 'Product family' is XB24C, 'Function set' is ZIGBEE TH Reg, and 'Firmware version' is 4061. Under the 'Networking' section, the following settings are visible:

Parameter	Value	Unit/Type
ID PAN ID	1234	
SC Scan Channels	7FFF	Bitfield
SD Scan Duration	3	exponent
ZS ZigBee Stack Profile	0	
NJ Node Join Time	FF	x 1 sec

XCTU Coordinator modtager pakker

The screenshot shows the XCTU software interface with the 'Frames log' and 'Frame details' windows open. The 'Frames log' shows a single entry:

ID	Time	Length	Frame
0	10:57:58.187	24	Receive Packet

The 'Frame details' window shows the following information for the received packet:

- 90 (Receive Packet)**
- 64-bit source address:** 00 13 A2 00 41 55 B9 80
- 16-bit source address:** 4D A4
- Receive options:** 01
- RF data:** ASCII HEX, Test Package

XCTU Router sender pakker til Coordinator

XCTU Working Modes Tools Help

Radio Modules

Name: Router
Function: ZIGBEE TH Reg
Port: COM8 - 9600/8/N/1/N - API 2
MAC: 0013A2004155B980

Name: Coordinator
Function: ZIGBEE TH Reg
Port: COM7 - 9600/8/N/1/N - API 2
MAC: 0013A2004155B965

Router - 0013A2004155B980 Coordinator - 0013A2004155B965

Close Record Detach

Frames log

ID	Time	Length	Frame
0	10:57:58.083	26	Transmit Request
1	10:57:58.158	7	Transmit Status

Frame details

88 (Transmit Status)

Frame ID

01 (1)

16-bit dest. address

00 00

Tx. retry count

00 (0)

Delivery status

00 (Success)

Discovery status

00 (No discovery overhead)

Checksum

73

Copy packet information

Send frames

Name	Type
Send to Coordinator	Transmit Request

Send a single frame

Send selected frame

Send sequence

Transmit interval (ms): 500

Repeat times: 1

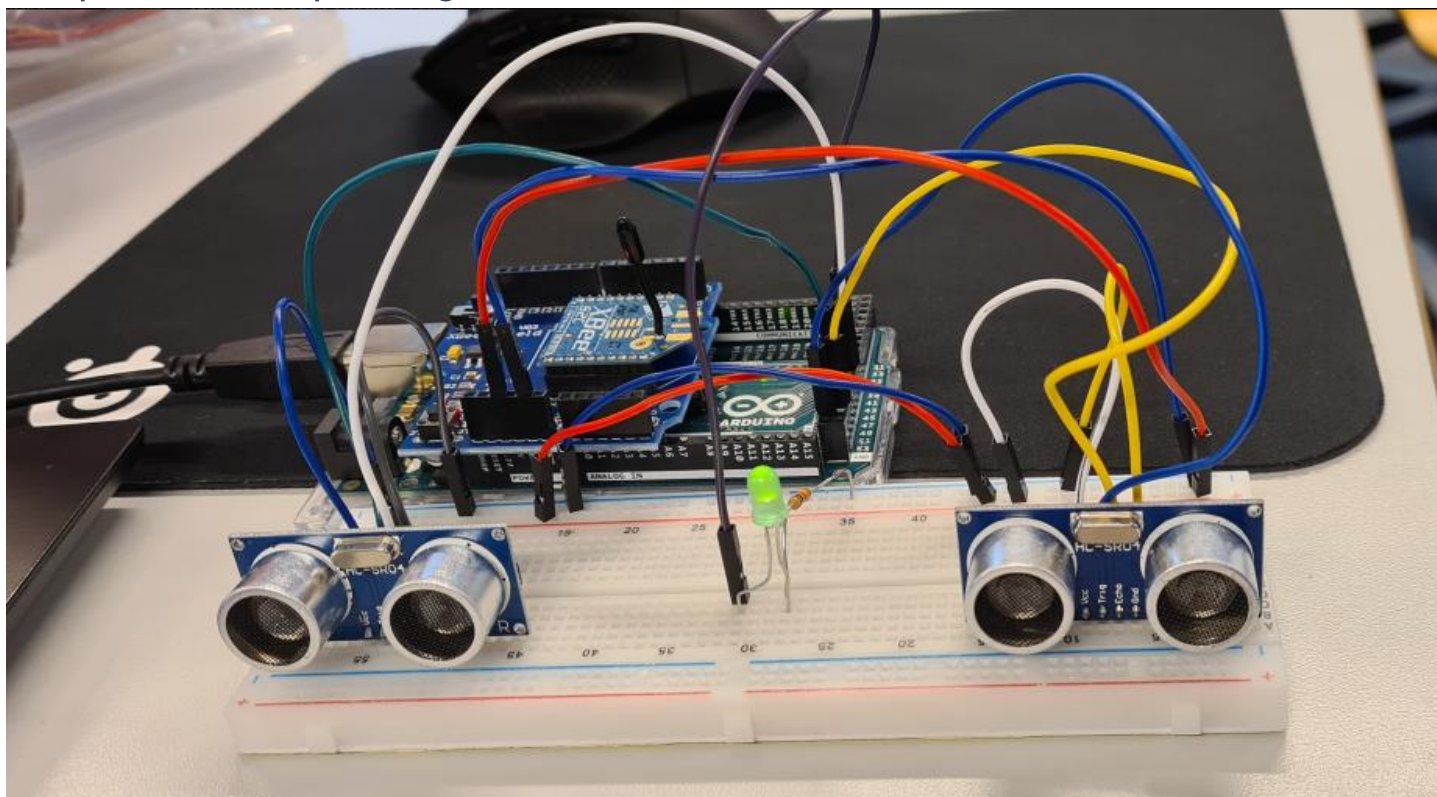
Loop infinitely

Start sequence

Raspberry Pi med Xbee enhed tilkoblet



Komplet Arduino opsætning



Docker container info

RASP-MICHAEL@RASP-MICHAEL: ~

```
RASP-MICHAEL@RASP-MICHAEL:~ $ docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0c56231f7761	eclipse-mosquitto	"/docker-entrypoint..."	3 days ago	Up 3 days	0.0.0.0:11883->1883/tcp	mosquitto

```
RASP-MICHAEL@RASP-MICHAEL:~ $
```