Timebox4 – bt server

Oversigt

OpgaveNavn	Implementering af bt server		
Implementering af krav	Delvis implementering af EMB-3,4,2,5 Hel implementering af EMB-1		
Udført af	Jan og Marc	**Dato**	15-10-2021
Timebox	4	**Område**	Embed

Contents

Introduktion

Der skal på embedden opsættes en Bluetooth server, der kan modtage og sende Bluetooth beskeder, samt styre IO kaldende til låsen.

Analyse

For at kunne modtage og sende Bluetooth beskeder, vil der skulle sættes en Bluetooth server op, hvilket vil blive gjort med Python3 biblioteket pybluez.

For at få pybluez til at virke, vil der skulle hentes forskellige biblioteker, som kan variere alt efter opsætning, dog dem der skulle hentes til dette projekt er:

- bluetooth bluez
- bluez python-bluez

Der vil også skulle laves ændringer i Paspberry pi'ens Bluetooth settings filer for at der ikke bliver givet fejl ed pybluez:

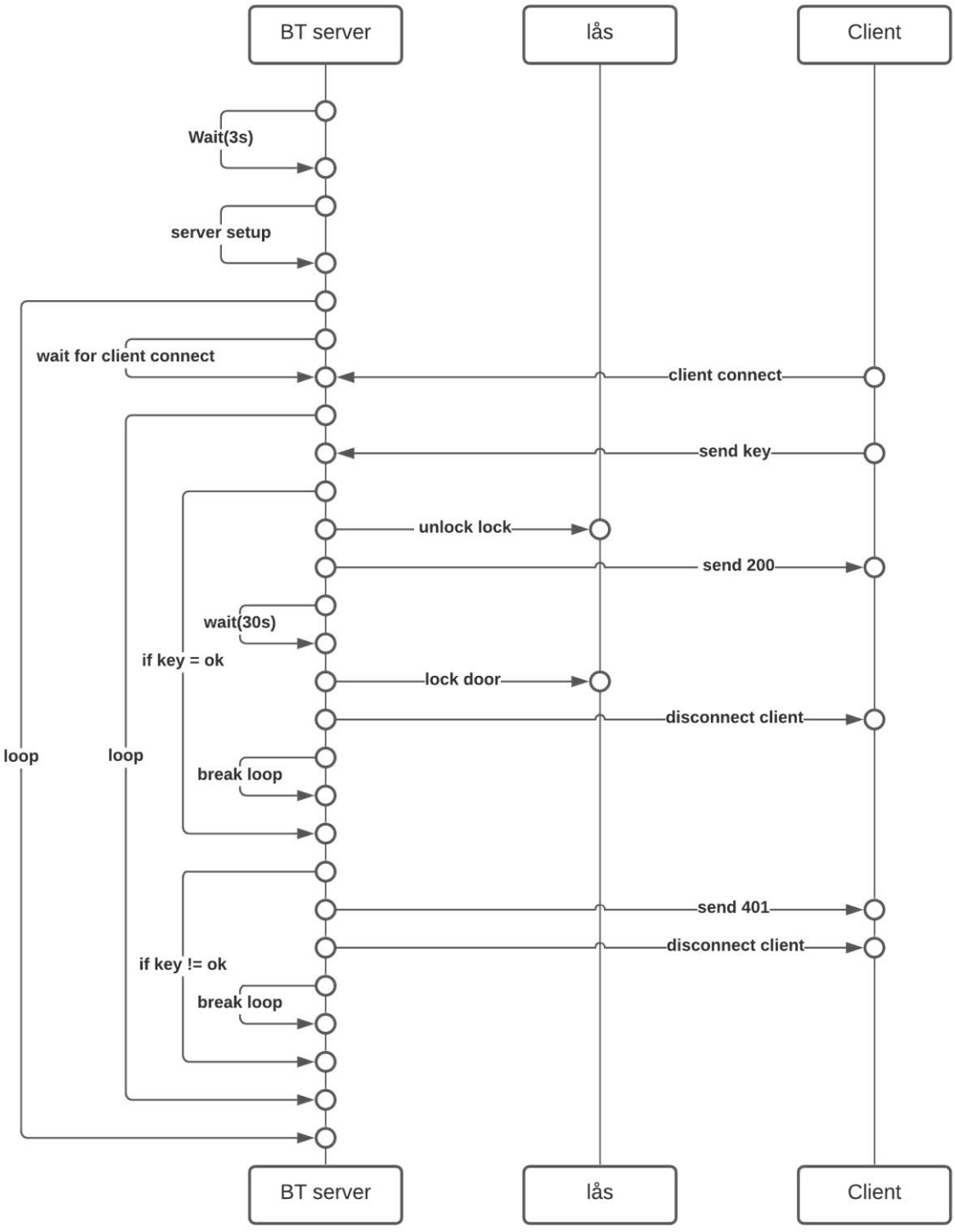
- Etc/bluetooth/main.conf => "privacy = off"
- /lib/systemd/system/bluetooth.service => "ExecStart=/usr/lib/bluetooth/bluetoothd -C –noplugin=sap -E"

Samt er nogle af de nødvendige kommandoer, efter der er blevet lavet ændringer til Bluetooth, som følgende:

- Genstart deamon => "sudo systemctl daemon-reload"
- Genstart bluetooth => "sudo systemctl restart bluetooth"
- Tjekke bluetooth status => "systemctl status bluetooth"
- For at kunne advertise eller blive discoverable => "sudo hciconfig hci0 piscan"
- Ændre device navn => "sudo hciconfig hci0 name "device name"

Design

Designet for funktionsflowet vil være som vist på swimlane diagrammet Figur 1.



Figur 1 swimlane for bt server

Implementering

Setup for serveren er som følgende:

1. sæt embedden til at kunne ses.
2. lav Bluetooth socket.

3. bind socket til en Bluetooth port.
4. begynd med at lytte på porten.
5. begynd med at advertise.

Dette kan også ses i Figur 2.

```
os.system('sudo hciconfig hci0 piscan')

lock_control = lock_control()
msg = 42
time.sleep(3)

server_sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
server_sock.bind(("", bluetooth.PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]

uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"

bluetooth.advertise_service(server_sock, "SampleServer", service_id=uuid,
                            service_classes=[uuid, bluetooth.SERIAL_PORT_CLASS],
                            profiles=[bluetooth.SERIAL_PORT_PROFILE],
                            # protocols=[bluetooth.OBEX_UUID]
                            )

print("Waiting for connection on RFCOMM channel", port)
```

Figur 2 bt setup

Herefter kan der laves et loop, som har et stop indtil der bliver accepteret en klient, som vist på Figur 3.

```
try:
    while True:
        client_sock, client_info = server_sock.accept()
        print("Accepted connection from", client_info)
        e = client_sock.send('hello')
```

Figur 3 client loop

Når der er modtaget en klient, kan der gåes ind I et nyt loop, hvor der bliver tjekket efter en "nøgle" besked fra klienten, som set på Figur 4.

```
try:
    while True:
        msg = client_sock.recv(1024)
        msg = str(msg)
        msg = msg.replace("'", "")
        msg = msg[1:]
        msg = msg[:2]
        print(msg)
        print(isinstance(msg, str))
        if lock_control.validate_key(msg):
            "Great open lock"
            print("Key is valid lock is now unlocked")
            if lock_control.unlock_lock():...
            else:
                "500 = internal error"
                client_sock.send("500")
                client_sock.close()
                "lock failed to unlock"
        else:
            "401 = unauthorized"
            client_sock.send("401")
            client_sock.close()
            print("Key is no good")
```

Figur 4 client handling

For hvert tilfælde vil der blive sendt en status kode til klienten, hvorefter klienten vil blive disconnected fra serveren.
De forskellige statuskoder brugt, er taget fra http status kode standarden, og er som følgende:

- 200 = succes
- 500 = intern fejl
- 401 = uautoriseret bruger

Selve serveren er sat til at køre som en baggrunds proces, ved opstart af Raspberry pi, ved at rette i ”/etc/rc.local”, som kan ses på Figur 5.

```
# Start bt server on upstart
sleep 10
python3 /home/pi/Desktop/embed-prog/bt_class.py &
```

Figur 5 rc.local

Verifikation

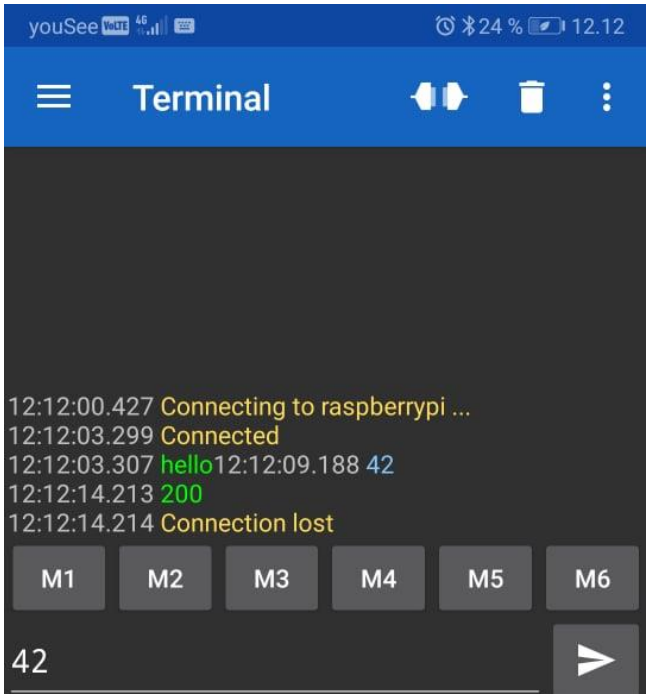
Antaget forberedelser:

- Bluetooth serveren er oppe at køre.
- Der er hentet en BT-terminal til ens telefon.

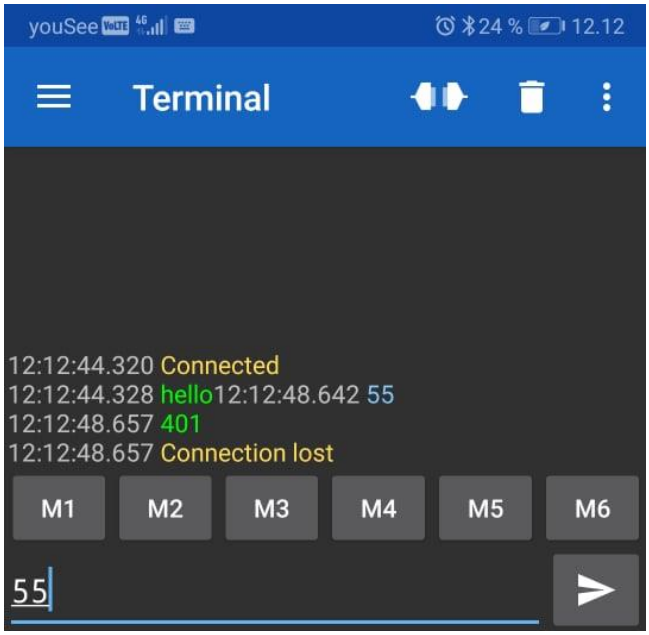
Tabel 1: Tests til verifikation af opgave

Test	Test Steps	Pre-requisites	Pass-betingelser	Resultat
Ok nøgle	1. opret BT-forbindelse til raspberry pi'en. 2. send "42".		1. der bliver modtaget en "200" retur kode. 2. relæ aktiverer 3. bruger bliver disconnected.	Bestået som kan ses på Figur 6
Ikke ok nøgle	1. opret BT- forbindelse til raspberry pi'en. 2. send "55".		1. der bliver modtaget en "401" retur kode. 2. bruger bliver disconnected	Bestået som kan ses på Figur 7

Testresultat



Figur 6 bt 200



Figur 7 bt 401

Konklusion

Der er blevet sat en Bluetooth server op, hvorfra man kan styre embeddens IO. Der kan oprettes forbindelse til embedden, samt kan der sendes beskeder frem og tilbage, mellem en telefon og embed.