Design och tjänsteutveckling för
Internet of Things, vt23

Laboration 2

BL, JB

Syftet med denna laboration är att ge tillräckliga kunskaper för att kunna koppla ihop perifera agenter med en central kontrollenhet för att hämta, behandla och skicka data i ett trådlöst nätverk. Efter avslutad laboration har vi implemeterat ett system bestående av två agenter som interagerar med varandra via en central nod med hjälp av protokollet MQTT.

Uppgifterna III, V och VI ska redovisas/demonstreras för lärare i labbsalen. Förbered er på att diskutera lösningarna och se till att alla frågeställningar är behandlade, innan ni ber om att få redovisa.

Spetsuppgiften är en frivillig fördjupning för den som vill lära sig mer och/eller eftersträvar högre betyg. För betyg VG på laborationen ska samtliga grund- och spetsuppgifter vara redovisade och klara innan utsatt deadline.

I. Raspberry Pi, förberedelser

Sätt in uSD-minneskortet i RPi. Detta minneskort fungerar som ”hårddisk” för systemet, och är förberett med en *Raspberry Pi OS* image, ett komplett linux-system. Koppla därefter in skärm via HDMI-kabel samt tangentbord och mus via USB. När allt är på plats kan systemet spänningssättas med nätadaptern. RPi:n startar upp och en grafisk välkomstskärm bör efter en stund visas på skärmen.

Följ instruktionerna på skärmen, tips:

- Klicka gärna i rutan ”Use English language” (enklare att söka efter info på nätet om ev. felmeddelanden presenteras på engelska).
- När du ska skapa ett användarnamn: Inga mellanslag eller konstiga icke-basic-ASCII-tecken. Observera att inställningen för svenskt tangentbord eventuellt inte slår igenom direkt, så undvik för säkerhets skull specialtecken i lösenordet.
- Välj ”Skip” vid ”Select WiFi Network”
- Välj ”Skip” vid ”Update Software”

Efter omstart bör du bli automatiskt inloggad i ditt nya RPi-linux-system, gratulerar.

Tangentbordslayout

Starta ett terminalfönster med Terminal-ikonen i menyn uppe till vänster. Prova att skriva åäö. Om det inte fungerar: Gå till *Hallon-menyn -> Preferences -> Keyboard and Mouse -> Keyboard* och byt till svensk layout.

Internet-access och system-uppdatering

Anslut till WiFi-nätverket *UmU-wlan* via nätverksikonen uppe till höger. Detta näverk kräver inloggning via en browser för att ge internet-åtkomst. Starta webbläsaren Chrome, gå till www.umu.se och logga in med Umu-id (du kan behöva försöka ett par gånger innan inloggningsssidan dyker upp).

När du har verifierat att du har internet-access, kör följande kommando i ett terminalfönster:

```
$ sudo apt update
```

Vänta tills kommandot har kört klart och du har en blinkande cursor i terminalen, kör därefter följande kommando:

```
$ sudo apt upgrade
```

II. Filsystemet, terminalanvändning

Filsystemet med systemets alla kataloger och filer kan liknas vid ett uppochnedvänt träd med en rot vid stammens tjockaste del och sedan en uppdelning i något som kan liknas vid ett grenverk. Längst upp finns roten (/). Under roten finns ett antal kataloger med filer och eller ytterligare filer och kataloger. De flesta filerna och katalogerna i ett nytt system är systemfiler som en vanlig användare av systemet oftast inte behöver eller har rättigheter att modifiera. Användarens personliga filer finns typiskt i en katalog med namn */home/<anvnamn>/*
Denna katalog, hemkatalogen, betecknas ~

Öppna ett terminalprogram och prova följande kommandon:

pwd – print working directory
```
$ pwd
```

ls – lista innehåll i katalog
```
$ ls
$ ls /
$ ls /home
```

mkdir – skapa ny katalog
```
$ mkdir hejhej
$ ls
```

tree – semi-grafisk vy av filer och kataloger
```
$ tree
$ tree /home
$ tree hejhej
```

cd – change directory
touch – skapa ny (tom) fil
```
$ cd hejhej
$ touch huj.txt
$ cd ..          (två punkter är en genväg, ett alias, för katalogen ovanför nuvarande)
$ tree
```

mv – flytta eller byt namn på fil eller katalog
```
$ cd ~
$ touch apa.txt
$ mv apa.txt hejhej            (flytta filen till katalogen hejhej)
$ tree
$ mv hejhej/apa.txt hejhej/bepa.txt      (byt namn på filen)
$ tree
```

Övriga vanliga kommandon

cp – kopiera fil
rm – radera fil
rmdir – radera (tom) katalog

cat – lista innehåll i textfil
less – lista innehåll i textfil
grep – sök efter sträng i textfil

sudo – ge vanlig användare tillfälliga rättigheter att redigera systemfil eller köra program som vanligtvis bara system-administratören har åtkomst till. Behöver användas vid tex. system-uppdatering eller installation av nya program.

man – lista manual-sida för ett kommando
```
$ man ls
```

Prova följande kommandon, och studera manual-sidan för **ls** för att förstå vad de olika varianterna visar. Det går bra att ha två eller flera terminalfönster öppna samtidigt.

```
$ cd ~
$ ls
$ ls -l
$ ls -d
$ ls -a
```

Auto-complete med <TAB>

Tips för att undvika stavfel och misstag vid arbete i terminalen: <TAB>-tangenten kan användas för att automatiskt komplettera ett påbörjat kommando eller argument.

```
$ cd ~
$ cd he<TAB>
```

Om det påbörjade kommandot/argumentet inte kan auto-kompletteras unikt ger ett extra <TAB> en lista på möjliga alternativ. Fortsätt skriva fler tecken och prova <TAB> igen.

```
$ cd ~
$ cd Do<TAB><TAB>
$ cd Dow<TAB>
```

Avsluta applikationer med <CTRL>-c

Tangentkombinationen <CTRL>-c kan användas för att avsluta kommandon/applikationer (håll ned CTRL-tangenten, tryck därefter c).

```
$ tree /
```
zzz, blir aldrig klart…
```
<CTRL>-c
```
ha!

Terminalbaserade text-editorer

Det finns ett flertal text-editorer som inte har något grafiskt gränssnitt utöver det som kan visas i ett terminal-fönster. Exempel på kraftfulla sådana editorer är **vim** och **emacs** (för en historielektion – slå upp ”Editor war” på Wikipedia). Som standard finns den enklare editorn **nano** installerad på RPi. Ibland har man inte tillgång till ett grafiskt gränssnitt utan bara ett terminalfönster. Om man då behöver modifiera en konfigurationsfil eller ett script kan det vara bra att kunna hantera en terminalbaserad editor.

För att skapa en ny textfil i hemkatalogen:

```
$ nano ~/minfil.txt
```

Skriv en valfri text på några rader.

Spara ditt arbete med <CTRL>-O
Tryck <ENTER> för att acceptera det föreslagna alternativet eller välj ett annat namn.
Avsluta med <CTRL>-X

För att därefter lista innehållet i filen:

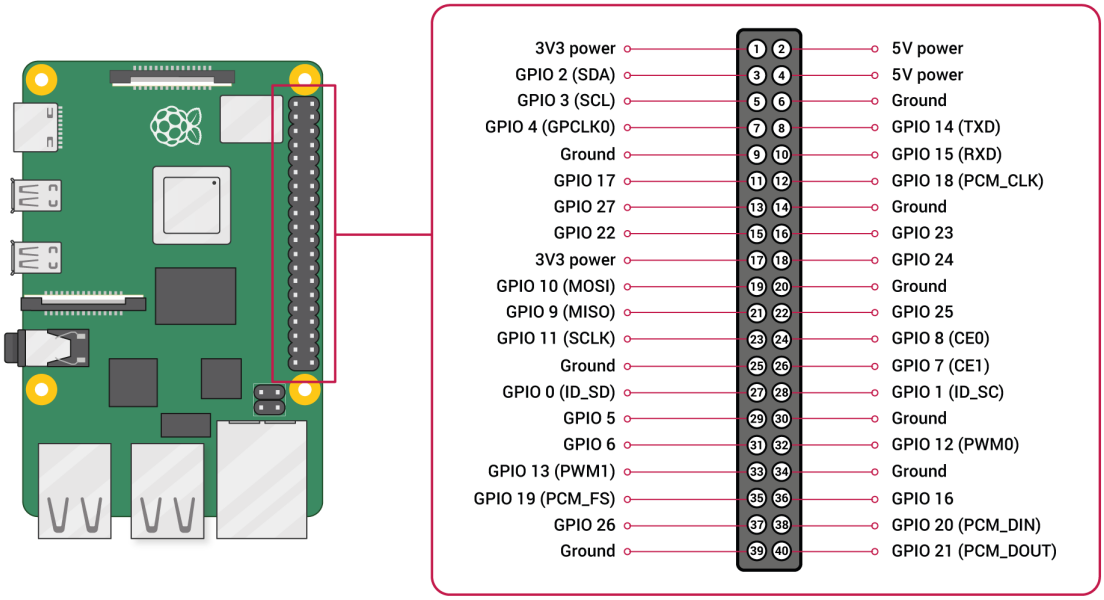
```
$ cat ~/minfil.txt
```

För att redigera den igen med nano:

```
$ nano ~/minfil.txt
```

III. RPi med extern hårdvara och Node.js

Node.js är en tolk för server-side JavaScript som kan användas för att sätta upp webbservrar och dynamiska webbsidor. Node.js ska här användas för att interagera med extern hårdvara kopplad till RPi:s expansionsportar.



Installera *Node.js* och den tillhörande pakethanteraren *npm* (Node Package Manager) genom att köra följande kommandon i ett terminalfönster.

```
$ sudo apt install nodejs
$ sudo apt install npm
```

För att Node.js ska kunna komma åt hårdvaran måste paketet/biblioteket *onoff* installeras. Gå till din hemkatalog och kör följande kommando. (Använd inte **sudo**, npm-paket installeras i hemkatalogen).

```
$ npm install onoff
```

Koppla in en lysdiod mellan GPIO4 och jord, via ett strömbegränsande motstånd. Koppla även en tryckknapp mellan GPIO14 och jord. Ett externt pull-up-motstånd (”några kΩ”) till 3.3V behövs till knappen. Se pinout-kartan ovan för att hitta rätt anslutningar, eller använd kommandot **pinout** i ett terminalfönster.

Skapa en ny textfil *blinka.js* med följande innehåll. Valfri editor. *Geany* finns tillgänglig via hallon-menyn, *nano* finns i terminalen, många andra kan installeras med *sudo apt install* ...

```
let Gpio = require('onoff').Gpio;
let LED = new Gpio(4, 'out');
let button = new Gpio(14, 'in', 'both');

function button_callback(err, value){

     console.log(”Button event, status: ", value);

}

button.watch(button_callback);

function timer_callback(){

     if (LED.readSync() === 0) {
          LED.writeSync(1);
     }
     else {
          LED.writeSync(0);
     }

}

my_interval = setInterval(timer_callback, 200);

function clean_exit() {

     console.log("Shutting down!");
     clearInterval(my_interval);
     LED.writeSync(0);
     LED.unexport();
     button.unexport();

}

process.on('SIGINT', clean_exit);
```

Kör scriptet och verifiera att det fungerar som det ska.

```
$ node blinka.js
```

Dokumentation

Utan dokumentation och referensmanualer kommer man inte långt här i livet ;)

Bekanta dig med dokumentationen för onoff-biblioteket. Se tex. beskrivningarna för konstruktorn *Gpio()*, och metoderna *writeSync()* och *watch()*.

https://www.npmjs.com/package/onoff

En generell guide för JavaScript-språket finns här:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

Dokumentation för Node.js-specifika funktioner finns via länken nedan. Se tex. avsnittet *Timers* som bla. beskriver *setInterval()*.

https://nodejs.org/docs/latest-v12.x/api/index.html

IV. RPi som MQTT-broker

MQTT-meddelanden hanteras av en server, en sk. *broker*. I denna laboration ska er egen RPi agera MQTT-broker.

Se till att er RPi har internet-access, och installera därefter brokern *mosquitto*, samt test-klienter, med följande kommandon.

```
$ sudo apt install mosquitto
$ sudo apt install mosquitto-clients
$ sudo systemctl enable mosquitto
```

Modifiera konfigurationsfilen för mosquitto genom att lägga till följande rader i */etc/mosquitto/mosquitto.conf*

```
listener 1883
allow_anonymous true
```

mosquitto.conf är en systemfil, så editorn måste köras med **sudo**, tex:

```
$ sudo nano /etc/mosquitto/mosquitto.conf
```

Starta därefter om mosquitto med:

```
$ sudo systemctl restart mosquitto
```

Använd klienterna *mosquitto_pub* och *mosquitto_sub* för att testa att brokern fungerar som den ska (kom ihåg kommandot **man** för att läsa manual-sidor). Ex:

```
$ mosquitto_pub –t hejhej –m hojhoj
```

Se till att ni får en känsla för hur pub/sub-upplägget i MQTT-kommunikationen fungerar innan ni går vidare.

V. RPi med Node.js-script som MQTT-klient

Gå till hemkatalogen och installera paketet *mqtt* för Node.js.

```
$ npm install mqtt
```

Dokumentation för mqtt-biblioteket finns här:

https://www.npmjs.com/package/mqtt

Skriv ett script som prenumererar på meddelanden med ett av gruppen valt topic. Meddelande *ON* ska tända lysdioden som är kopplad till GPIO4, meddelande *OFF* ska släcka lysdioden. (Använd mosquitto_pub för att publicera dessa meddelanden.)

Startup-hjälp(?)

```
const mqtt = require('mqtt');
const client = mqtt.connect('mqtt://localhost');

client.on('connect', function(){

     client.subscribe('groupX');

});

function message_callback(topic, message){

     // message is Buffer
     console.log("Msg received:", message.toString());

}

client.on('message', message_callback);
```

VI. ESP32 som MQTT-klient

I denna uppgift ska två ESP32-enheter, ESP1 och ESP2, kommunicera med varandra och med klient-script på RPi via MQTT-meddelanden. För att detta ska fungera måste ESP32-enheterna ha åtkomst till brokern (gruppens RPi) via WiFi. Detta hade varit enkelt att ordna med en vanlig hemma-WiFi-router, men universitetets regler kring WiFi/nätverk krånglar till det en aning. ESP32 kan inte ansluta till *eduroam* eller *UmU-wlan*. Vi får inte heller sätta upp en lokal WiFi-router med internet-access i labbet. Work-around: För följande uppgifter används ett lokalt WiFi-nät, som dock inte är anslutet till internet. Både ESP32-enheterna och broker-RPi:n måste vara anslutna till detta nätverk:

```
ssid: NETGEAR61
lösen: jaggedmountain461
```

Utrusta ESP1 med två tryckknappar.
Utrusta ESP2 med fyra lysdioder.
Utrusta RPi med en tryckknapp (LARM) och en lysdiod.

Implementera därefter följande funktionalitet:

Knapptryckningar på ESP1 ska styra lysdioderna på ESP2. Tänk att de två tryckknapparna representerar ett binärt tal där nedtryckt knapp representerar ’1’. 00 => ingen LED lyser, 01 => en LED lyser, 10 => två LEDs lyser, osv.

När LARM-knappen trycks ned ska samtliga lysdioder (på ESP2 och RPi) blinka. Under tiden LARM-knappen är nedtryckt ska knapparna på ESP1 inte påverka lysdioderna. När LARM-knappen släpps upp ska systemet återgå till normal drift.

Tips för RPi

Byt trådlöst nätverk med hjälp av nätverksmenyn uppe till höger.

Kommandot **ifconfig** kan användas för att ta reda på vilken IP-adress som tilldelats från WiFi-routern (wlan0 -> inet, IP-adress på formen 192.168.1.__)

Tips för ESP32

Dokumentation för MQTT i Mongoose OS:

https://mongoose-os.com/docs/mongoose-os/api/net/mqtt.md

Komplettera mos.yml

```
config_schema:

  - ["wifi.sta.enable", true]
  - ["wifi.ap.enable", false]
  - ["wifi.sta.ssid", "zzzzzz"]          # nätverksnamn
  - ["wifi.sta.pass", "zzzzzz"]          # lösenord

  - ["mqtt.server", "zzzzzz:1883"]       # IP-adress, RPi
  - ["mqtt.enable", true]


libs:

  - origin: https://github.com/mongoose-os-libs/mqtt
```

Spetsuppgift – Ljuv musik(?)

Spetsuppgiften utförs parvis (eller individuellt). All kommunikation mellan enheter sker via MQTT-meddelanden.

För par 1
Utrusta ESP1 med en avståndssensor SRF02 baserad på ultraljudsteknik. Utrusta ESP2 med en piezo-buzzer. Låt avståndssensorns avläsning styra frekvensen på tonen som piezo-buzzern genererar, i ett intervall som möjliggör musicerande. Vettiga frekvenser, valfri tonart.[1] För att aktivera funktionen i 10 sekunder ska en knapp tryckas ned.

För par 2
Skriv ett Node.js-script på RPi som till en textfil loggar musikstycket som skickas från ESP1. En knapptryckning ska läsa in loggfilen och spela upp det sparade musikstycket på ESP2.

Tips för SRF02

Dokumentation, I²C-läget ska användas:
https://www.robot-electronics.co.uk/htm/srf02tech.htm

Standard-I²C-adressen för sensorn anges som 0xE0. Detta är adressen på åtta-bitars-form. Mongoose-OS-funktionerna förväntar sig adressen på sju-bitars-form, 0x70 (=0xE0 bit-skiftat ett steg åt höger).

5V till sensorns matningspinne kan tas från ESP32-modulens anslutning märkt USB. Pull-up-motstånd (tex. 4.7kΩ) från SDA/SCL till 3.3V.

Till skillnad från temperatursensorn från laboration 1 kräver SRF02 ett kommando för att utföra en mätning. Sekvens:
1) Skicka kommando (tex. 0x51) till kommandoregistret (0).
2) Vänta 70ms så att mätningen hinner utföras.
3) Läs in resultatet från resultatregister (2/3).

Tips för filhantering på RPi

Node.js-dokumentationen, avsnittet *File system*.

Terminalkommandot **tail** med växeln **-f** kan vara användbart, se manual-sidan.

1 Se tex. https://en.wikipedia.org/wiki/Piano_key_frequencies