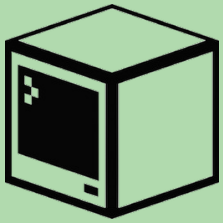


Fjernstyr en robot

Introduksjon

I denne oppgaven skal vi fjernstyre en robot fra en annen datamaskin brukeren, sende disse til en annen datamaskin, og tolke disse som en

Steg 1: Forberedelser

Denne oppgaven bygger videre på andre oppgaver:

- [ ] Send en beskjed over nettverk beskriver hvordan vi sender og r
- [ ] Hendelser -- Steg 1: Skattejakt viser hvordan vi kan gjenkjenne

- [ ] Bygg et Hus -- Steg 4: Funksjoner forklarer hvordan vi kan skrive

Gå tilbake og kikk på disse to oppgavene hvis denne blir vanskelig.

Redigere filer utenfor Computer

`edit`-programmet i ComputerCraft fungerer fint med små program
er for eksempel ikke lett å klippe og lime kode. Nå skal vi lære hvo
ComputerCraft.

Filene vi lager i ComputerCraft har forskjellig plassering på forskjel

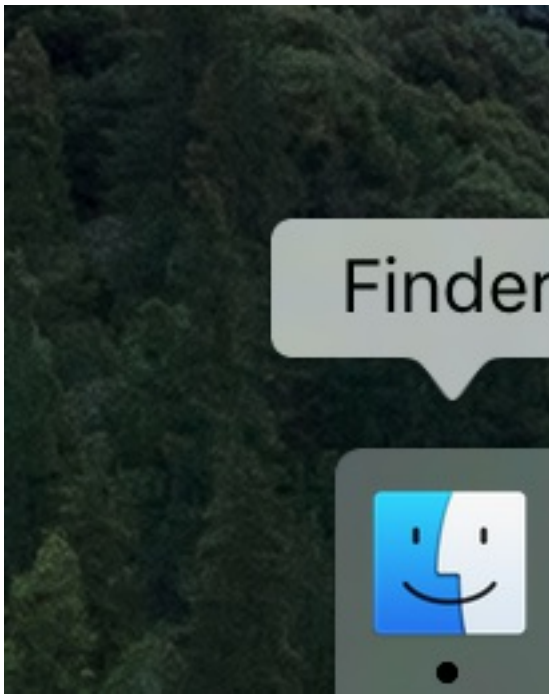
- [ ] Windows: `%AppData%\Roaming\.minecraft\saves\navn-på-di`
- [ ] Linux: `~/.minecraft/saves/navn-på-din-save/computer/id`
- [ ] Mac: `~/Library/Application Support/.minecraft/saves/na`

Vil du redigere filen `hello` fra spillet "Kodeklubben 2016" på datar
`AppData\Roaming\.minecraft\saves\Kodeklubben 2016\computer\`

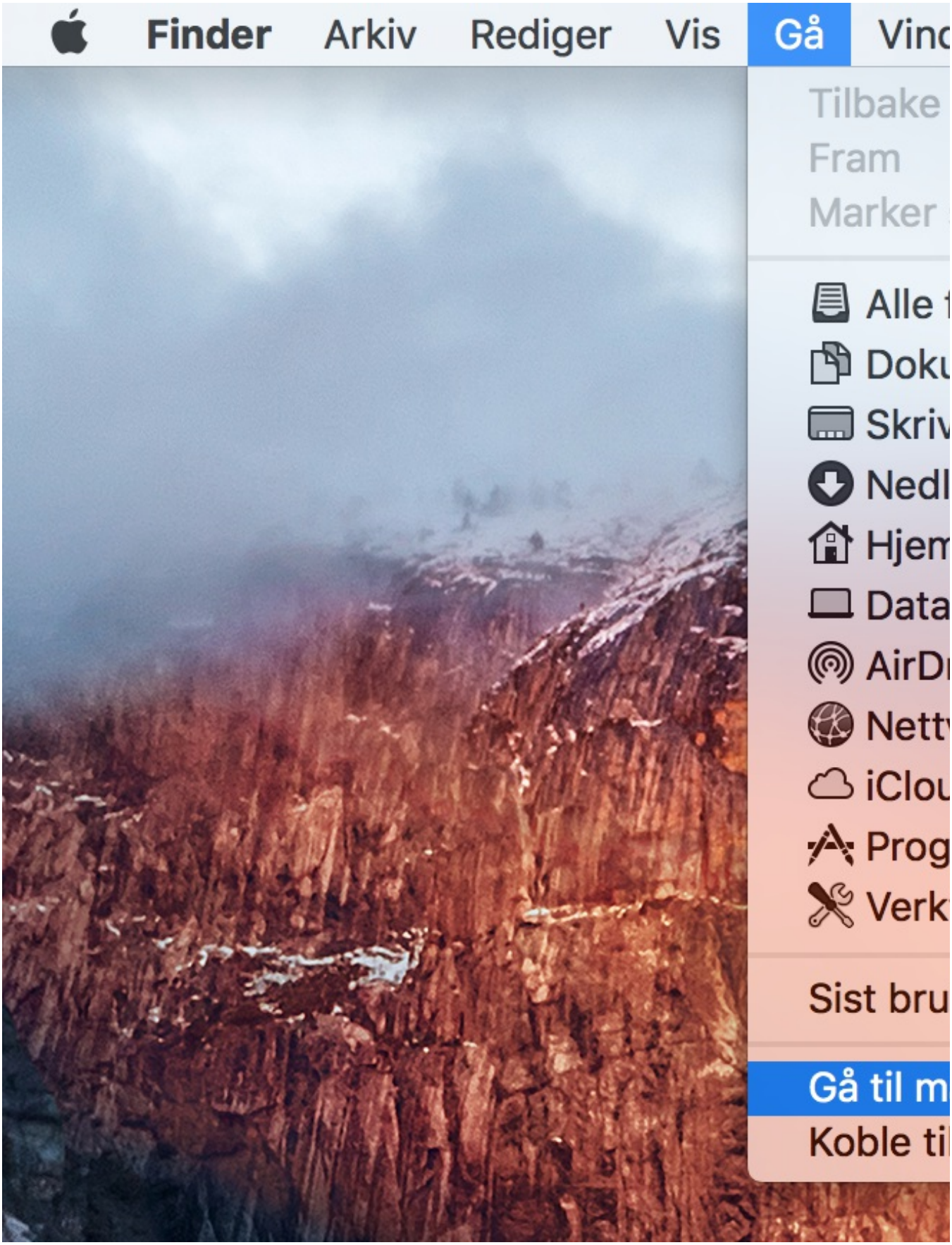
Finn filene på Mac

For å åpne filene på Mac kan du bruke Finder:

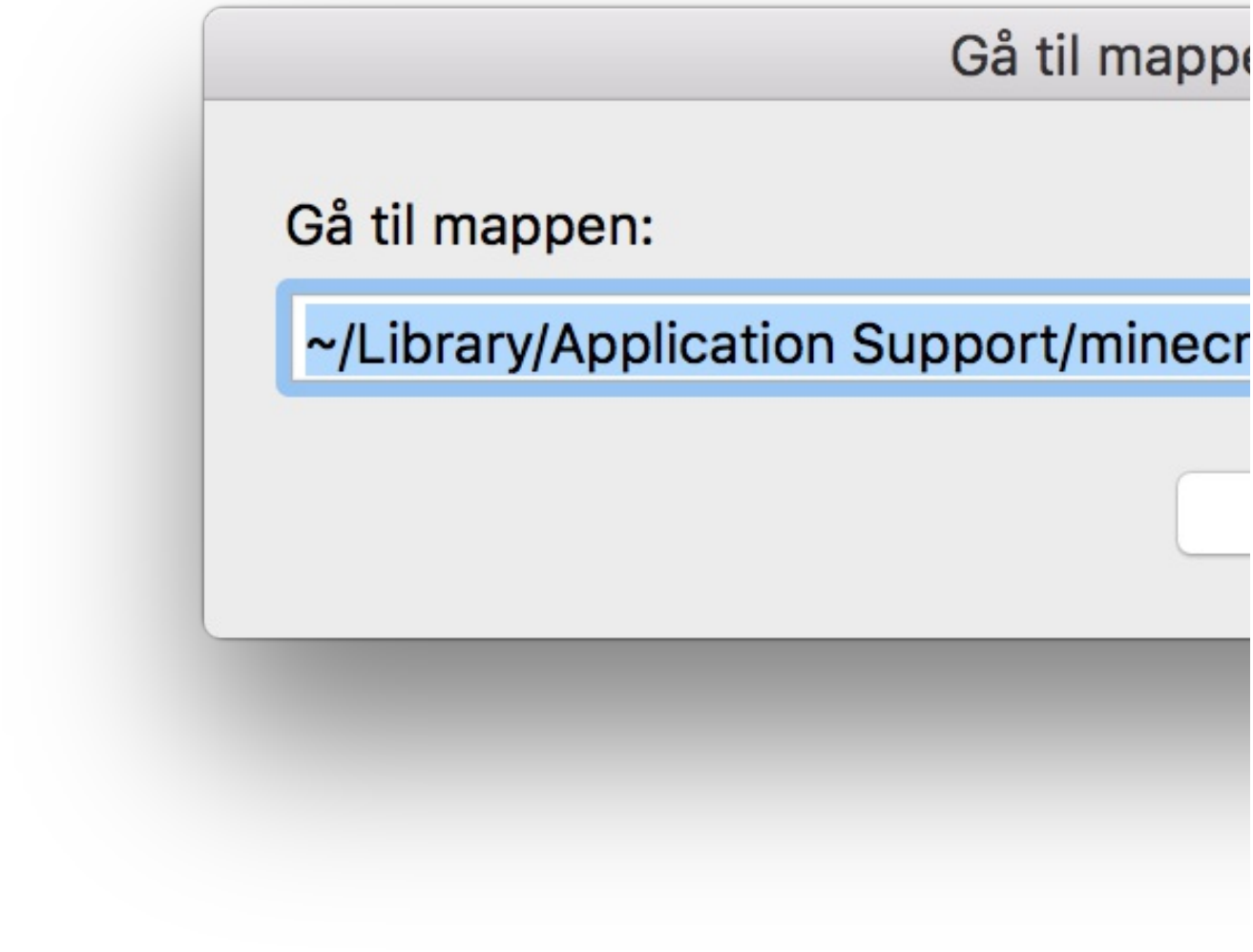
- [ ] Åpne Finder fra menyen



- [ ] Trykk på `Gå` og så `Gå til mappe`



- [ ] Skriv inn filplasseringen (`~/Library/Application Support/.min`



Her finnes programmene du har skrevet som egne filer. Åpne så progr

- [ ] Notepad++ er god og enkel. Støtter Windows.

- [ ] Github Atom er noe mer avansert. Støtter Windows, Mac og Linu

  Atom har en egen pakke for Lua-programmering. For å installere
  etter `lua` og installer pakken `language-lua`.

Gratulerer! Du kan nå klippe og lime kode mellom datamskiner. Eller v

Skriv koden selv!

Skriv koden du leser nå selv!

Du blir ikke vant med et programmeringsspråk uten å skrive kode
ikke klipp-og-lim. I stedet leser jeg et stykke kode, og skriver selv.
det skal være kommaer, `end`, `then` og andre småting som Lua ka

Og etter du har skrevet koden -- prøv litt fram og tilbake!

- [ ] Trenger jeg egentlig denne biten?
- [ ] Hva skjer om jeg fjerner denne setningen?
- [ ] Eller om jeg putter inn et annet tall her?

Programmering er lek! Ikke la noen fortelle deg noe annet!

Steg 2: Tastetrykk på fjern

Først programmerer vi kommandosentralen vår! Den må lytte etter ta
roboten.

Sjekkliste

- [ ] Vi bygger først kontroller. Craft en **Advanced Wireless Pocket**
er ikke bundet til én plass.

Filtyper

Filer har ofte etternavn som sier filtypen.

- [ ] Lua programmer heter ofte `program.lua`
- [ ] Nettsider heter ofte `nettside.html`: Nettsider skrives i et sp
- [ ] `dokument.docx` er et Word-dokument
- [ ] Kjenner du til andre filtyper?

Sjekkliste

- [ ] Vi kaller kontroller-programmet vårt `remote.lua`

```
edit remote.lua
```

- [ ] La oss starte med å lese inn hva brukeren trykker på:

```
function remoteMove()
  local action, keycode = os.pullEvent('key')
  if keycode == keys.w then
    return {"Go", "forward!"}
  elseif keycode == keys.a then
    return {"Denne", "må", "du", "fikse", "selv!"}
  else
    return {"Hjelp! Ukjent tast! Hva skal jeg gjøre???"}
  end
end

function main(  )
  print("Venter på kommando ...")
```

```
  rednet.open("back")  -- Hvor har du modemet ditt?
  while true do
    local command = remoteMove()
    print(unpack(command))
  end
end

main()
```

- [ ] Prøv! Hva skjer når du trykker på `w`? Hva skjer når du trykker p

- [ ] Roboten vår skal gjerne kunne svinge og gå bakover i tillegg. Le
eller `d`!

- [ ] Roboten kan i tillegg fly opp og ned. I Minecraft brukes `space` f
ComputerCraft Wiki for hvordan du bruker disse!

Nå kjenner vi igjen tastene for å bevege seg rundt. Neste steg er å se

Steg 3: Kommandotabell

Vi skal bruke en tabell til å lagre kommandoen vår. Lua kaller en tabel

La oss først leke oss litt med en robot.

Sjekkliste

- [ ] Pass på at roboten har nok kull!

- [ ] Åpne lua-tolkeren, og lag en tabell:

```
> lua
```

```
lua> tabell = {"hei", "på", "deg"}
lua> print(tabell)
lua> print(unpack(tabell))
lua> print(tabell[1])
lua> print(tabell[2])
lua> print(tabell[3])
```

Tabeller

- [ ] Prøver vi skrive ut en tabell alene, får vi *adressen* til tabellen
- [ ] Unpack lar oss bruke tabellen i en funksjon:

  `print(unpack({1, 2}))` er det samme som `print(1, 2)`.
- [ ] Vil vi ha ut et *element* fra tabellen, setter vi inn *indeksen* til e

Sjekkliste

Nå skal vi se på to måter å styre en robot på.

- [ ] Start `CraftOS` på en robot, og kjør først kommandoen på "vanli

```
> go forward 5
> turn left
```

... men vi vil da ikke løpe etter roboten og skrive inn nye kommandoe

- [ ] Åpne lua-tolkeren og lag tabeller for kommandoene over:

```
> lua
lua> command1 = {"go", "forward", "5"}
lua> command2 = {"turn", "left"}
lua> shell.run(unpack(command1))  -- Hva gjør unpack? Ka
lua> shell.run(unpack(command2))
```

- [ ] Hva gjør unpack? Hvordan kan vi bruke `shell.run()` til å gå fre

Nøtt

- [ ] Åpne et nytt program: `edit many.lua` (Du kan bruke en anne

```
cmd = {"go", "forward", "5"}
```

- [ ] Kan du lage en funksjon som kjører kommandoen `cmd` 2 gan

- [ ] Ekstra vanskelig: kan du lage en funksjon som kjører komma funksjonen?

  Du kan starte med dette:

```
function manyTimes( command, times )
  for i=1,times do
    -- Hva skal du gjøre mange ganger?
  end
end

command1 = {"go forward 5"}

manyTimes(command1, 7)
```

Sjekkliste

Nå skal vi endre på koden på fjernkontrollen vår så den faktisk sender over? Forklar til en annen programmerer eller voksen! Hva gjør progra

- [ ] Åpne igjen filen `remote.lua` på kontrolleren

- [ ] Endre koden til:

```
workerId = 5  -- Hvilket nummer er din robot? Hvordan fi

-- OBS: Ny fuksjon som inneholder deler av gamle remoteM
function toMoveCommand(keycode)
  if keycode == keys.w then
    return {"go", "forward"}  -- OBS! Nå må disse matche
  elseif keycode == keys.a then
    return {"Denne", "må", "du", "fikse", "selv!"}
  else
    return nil
  end
end

-- ... og nye remoteMove sender nå kommandoen dit den sk
function remoteMove()
  local action, keycode = os.pullEvent('key')
  command = toMoveCommand(keycode)
  if command == nil then
    print("Unknown key: " .. keycode)
  else
    print("Sending ".. unpack(command))
    rednet.send(workerId, command)
  end
end
```

```lua
-- Denne er endret litt
function main(  )
  print("Listening for move commands ...")
  rednet.open("back")
  while true do
  -- Her skal vi ikke lenger skrive ut, men flytte på ro
    remoteMove()
  end
end

main()
```

Steg 4: Motta kommando p

Gratulerer! Du har nå bygd en fjernkontroll!

Hva skal vi bruke denne til, mon tro?

Noe å styre?

Jepp, jeg er helt enig. La oss gjøre det!

Sjekkliste

- [ ] Åpne `react.lua` på roboten

Egne filer

- [ ] Husker du hvordan vi finner filene på vår egen datamaskin, s
- [ ] Hvis datamaskinen ikke har noen filer ennå, har den heller ik
  mappen til å dukke opp!

Sjekkliste

- [ ] Fyll så på med koden!

```
function main()
  -- Hva sier pheripherals? Hvilken side er modemet på?
  rednet.open("left")
  while true do
    sender, message = rednet.receive(99999)
    print("Message from " .. sender .. " received: ")
    print(message)
    shell.run(unpack(message))
  end
end

main()
```

`main()`

Metoden `main()` er ofte inngangsporten til et program. I for eksem
main-metode eller ikke. I andre språk må main være med. Eksemp
Java og Haskell.

Sjekkliste

- [ ] Hva brukes 99999 til?

- [ ] Hvorfor bruker vi en `while`-løkke? Hva skjer om vi ikke har en w

Steg 5: Kjør robot!

Gratulerer! Du har gjennomført en utfordrende programmeringsoppga
Vær stolt!

Nå skal vi prøve det selv!

Sjekkliste

- [ ] Start `react.lua` på din robot
- [ ] Start `remote.lua` på din Portable Computer
- [ ] Trykk `w` for å gå fremover!
- [ ] Legg til andre kommandoer du savner!

  Forslag:

  - [ ] Angrip
  - [ ] Bygg blokk foran
  - [ ] Kjør `excavate 4` her du er

Steg 6: Du vil ha mer?

Du har spilt ComputerCraft *lenge* og begynner å få et utall forskjellige andre kommandoer på roboten!

Programmet vi har skrevet for å motta og kjøre kommandoer, `react.` definert for å bevege seg rundt, for eksempel `excavate` og `dance`.

- [ ] Åpne kontrolleren

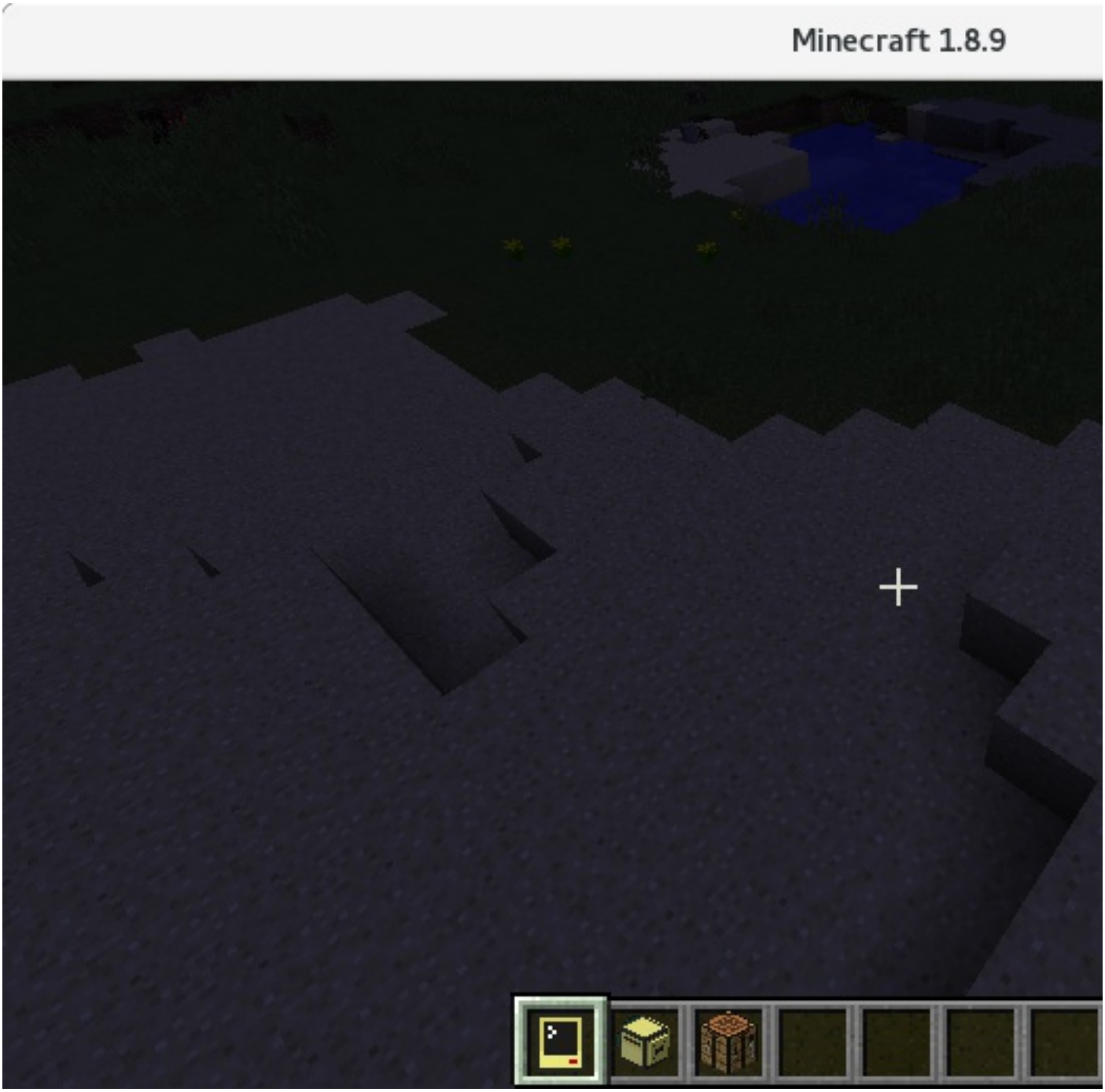
- [ ] Endre følgende i `ssh.lua`:

```
-- Starter jeg programmet sånn:
-- > ssh.lua 5
arguments = ...
-- ... blir arguments = "5"
remoteId = tonumber(arguments)
-- ... og remoteId = 5!

function pack(...)
  return arg
end

function remoteCommand()
  io.write("ssh@" .. remoteId .. "> ")
  local intputString = io.read()
  local commandTable = pack(intputString)
  print("Sending: ")
  print(unpack(commandTable))
  rednet.send(remoteId, commandTable)
end

function main(  )
  print("Listening for move commands ...")
  rednet.open("back")
  while true do
    remoteCommand()
  end
end
```

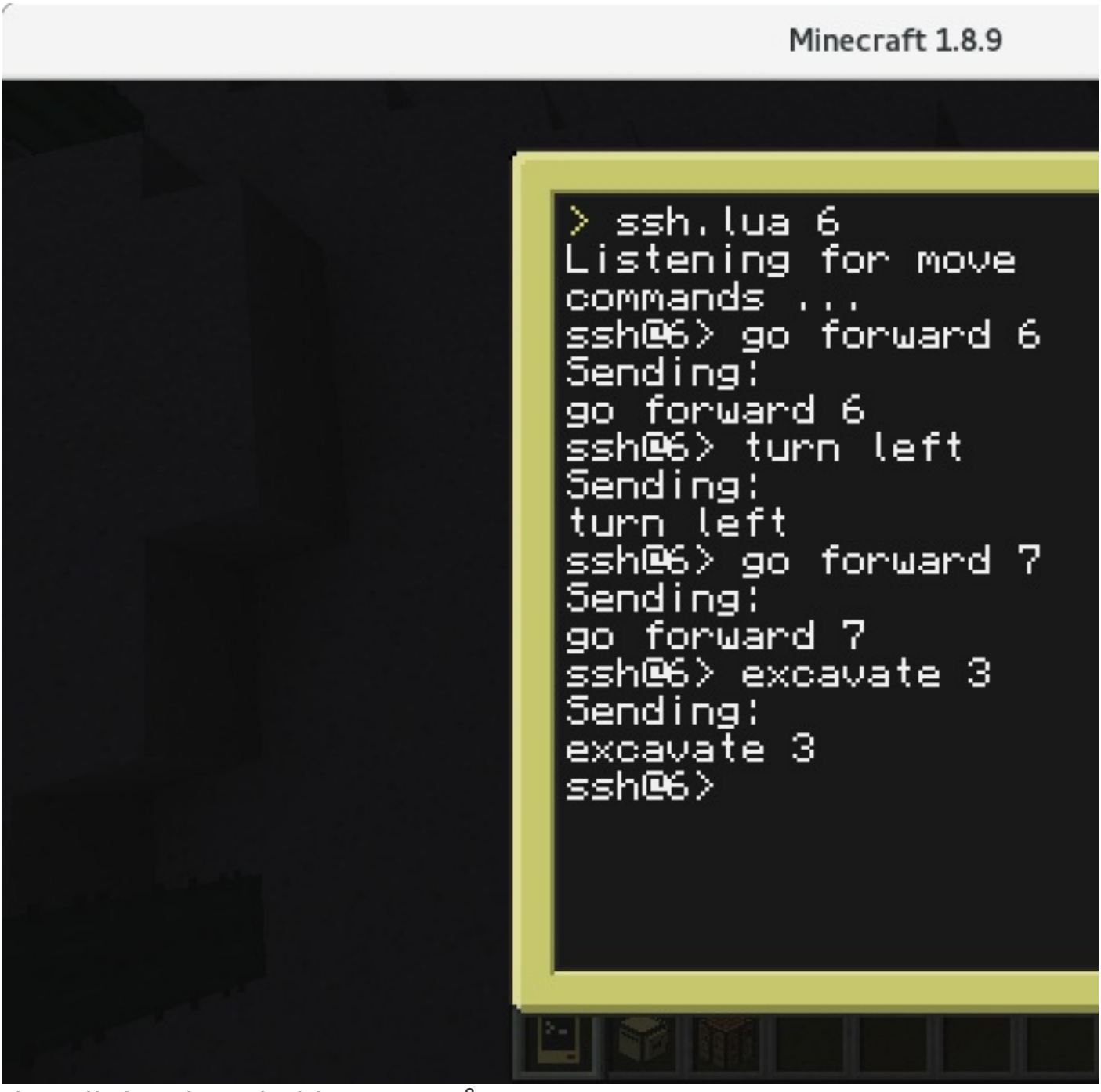
```
main()
```

- [ ] Start `react.lua` på roboten
- [ ] Start `ssh.lua` på kontrolleren

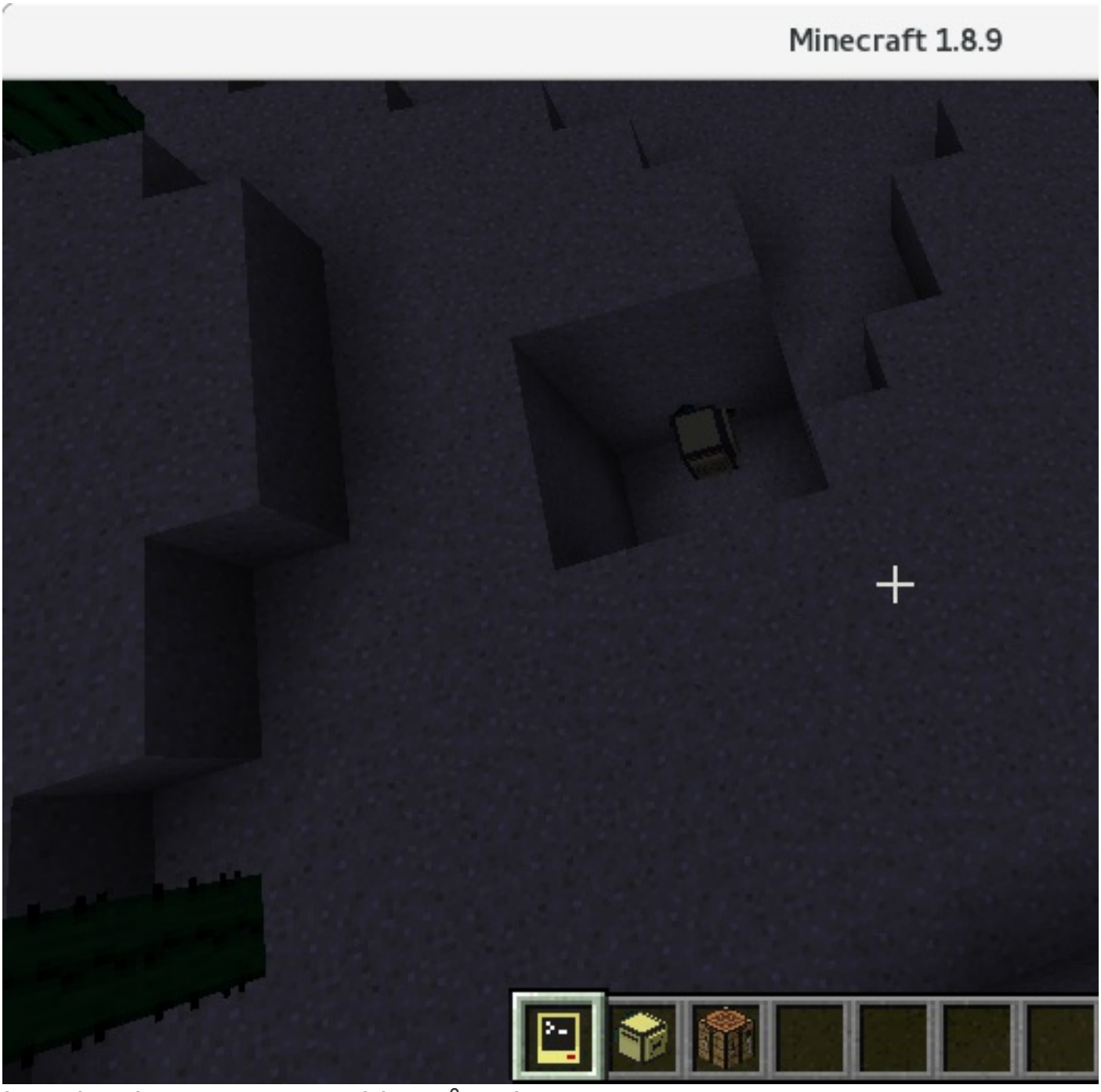
Fjernstyrt graverobot er nyttig! En graverobot kan craftes fra en anne gjorde:



Graveroboten er klar til høyre i bildet.



den til der den skal begynne å grave ...



har den kommet et stykke på vei!