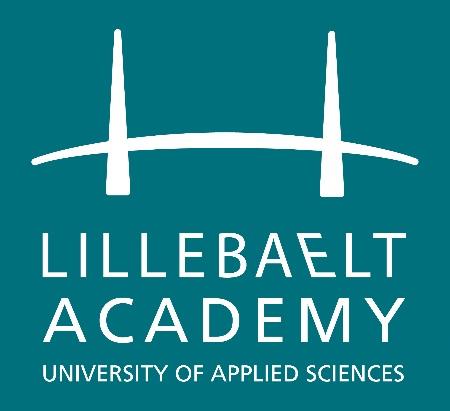
**IT Technology**

**Remote voice control of a system of robot vehicles**

**Appendix for project report**



LILLEBAELT ACADEMY

UNIVERSITY OF APPLIED SCIENCE

Authors

Dainius Čeliauskas

Juan Mohamad

[dain0084@edu.eal.dk](mailto:dain0084@edu.eal.dk)

[juan0314@edu.eal.dk](mailto:juan0314@edu.eal.dk)

Sunday, June 3, 2018

**Table of Contents**

[1. Simplified software installation instruction manual (Updated 18-05-30) 1](#_Toc515797489)

[2. PocketScriptHUB 5](#_Toc515797490)

[3. PocketScriptROBOT 9](#_Toc515797491)

[4. VoiceScript (old) 14](#_Toc515797492)

# Simplified software installation instruction manual (Updated 18-05-30)

This manual was created by and for the project group to quickly get Raspberries working with PocketSphinx, paho-mqtt and the main Python script. The keyphrase.dic, keyphrase.list, PocketScriptHUB.py and PocketScriptROBOT.py files can be found in the project’s GitHub page.

--------------------------------

Install Raspbian on Raspberry Pi

--------------------------------

Get Raspbian Stretch Lite iso file from internet

https://www.raspberrypi.org/downloads/raspbian/

Use Win32DiskImager to write iso file to sd card

https://sourceforge.net/projects/win32diskimager/

Create blank "ssh" file

(you can do it by creating a text file, then go to cmd, go to directory of your file and type:

rename yourfilename.txt ssh)

Create wpa\_supplicant.conf file -anywhere- with code like this:

ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev

update\_config=1

country=GB

network={

ssid="[Wifi network name]"

psk="[Wifi network password]"

priority=[number of priority]

}

network={

ssid="[PEAP protected Wifi network name]"

proto=RSN

key\_mgmt=WPA-EAP

pairwise=CCMP

auth\_alg=OPEN

eap=PEAP

identity="[your ID key]"

password=[your password in hash]

phase1="peaplabel=0"

phase2="auth=MSCHAPV2"

}

How to generate hash for PEAP protected network from your own password:

https://www.raspberrypi.org/forums/viewtopic.php?t=111100

(echo -n 'YOUR\_REAL\_PASSWORD' | iconv -t utf16le | openssl md4 > hash.txt)

Read more about PEAPv0/EAP-MSCHAPv2 here:

https://en.wikipedia.org/wiki/Protected\_Extensible\_Authentication\_Protocol#PEAPv0\_with\_EAP-MSCHAPv2

Insert ssh and wpa\_supplicant files into boot folder in sd card (visible in Windows)

Turn on raspberry. These files should be automatically moved from boot into where they need to be.

Find RPi IP with an IP scanner and connect to it with putty and WinSCP

https://www.advanced-ip-scanner.com/ (decent, portable IP scanner exe)

https://www.putty.org/ (for the terminal)

https://winscp.net/eng/download.php (GUI for easy exchange of files)

You can also use cmd for scanning IPs quickly using the 'arp -a' command.

Optional:

You can make the raspberry have a static IP to skip searching for it, but it's not recommended for large, constantly changing networks like Eal-Wireless. The RPi might not be able to connect to the network at all.

This article talks about it and how to do it:

https://caffinc.github.io/2016/12/raspberry-pi-3-headless/

Once connected, change the password, obviously:

passwd

Expand file system with sudo raspi-config

Update raspbian:

sudo apt-get update

sudo apt-get upgrade

sudo rpi-update

-----------------

Configure USB mic

-----------------

Insert mic into USB dongle

Run sudo nano /usr/share/alsa/alsa.conf and change in the file:

defaults.ctl.card 0

defaults.pcm.card 0

to

defaults.ctl.card 1

defaults.pcm.card 1

Create and edit .asoundrc file (sudo nano ~/.asoundrc), put the following:

pcm.!default {

type hw

card 1

}

ctl.!default {

type hw

card 1

}

Boost speaker output level with alsamixer to preferable levels

Test record audio from the mic:

arecord -D plughw:1,0 test.wav

-----------------------------------

Install Sphinxbase and Pocketsphinx

-----------------------------------

Type:

wget https://sourceforge.net/projects/cmusphinx/files/sphinxbase/5prealpha/sphinxbase-5prealpha.tar.gz/download -O sphinxbase.tar.gz

wget https://sourceforge.net/projects/cmusphinx/files/pocketsphinx/5prealpha/pocketsphinx-5prealpha.tar.gz/download -O pocketsphinx.tar.gz

Extract:

tar -xzvf sphinxbase.tar.gz

tar -xzvf pocketsphinx.tar.gz

Install bison, ALSA, swig, etc.

sudo apt-get install -y python python-dev python-pip build-essential bison libasound2-dev swig git

Compile Sphinxbase:

cd sphinxbase-5prealpha

./configure --enable-fixed

make

sudo make install

Compile Pocketsphinx:

cd ../pocketsphinx-5prealpha

./configure

make

sudo make install

Go for 2 coffee breaks each while they're installing, cause both of these take super long.

Test out the installation:

src/programs/pocketsphinx\_continuous -adcdev sysdefault -samprate 48000 -nfft 2048 -inmic yes

---------------------------

Install paho-mqtt

---------------------------

Type:

cd

sudo apt-get install paho-mqtt

---------------------------

Install pocketsphinx-python

---------------------------

Type:

cd

sudo apt-get install python-pyaudio

git clone --recursive https://github.com/cmusphinx/pocketsphinx-python/

cd pocketsphinx-python

sudo python setup.py install

The website for pocketsphinx-python shows the basic usage and how the code should look like.

-----------------------------------

Generate keyword list and test code

-----------------------------------

To create a new keyword list, type all the keywords in a txt file, new line for each keyword or combo

Go to http://www.speech.cs.cmu.edu/tools/lmtool-new.html, choose your file and click "Compile knowledge base"

Then download the .dic and .list files.

The .dic file tells the software how to pronounce words, while the .list file shows the thresholds for each keyword.

Put keyphrase.dic, keyphrase.list and PocketScriptHUB.py (or PocketScriptROBOT.py) files into pocketsphinx-python directory

Allow permission to run the script with "chmod 777 PocketScript\*.py"

Run PocketScript\*.py and see if it works correctly.

Try testing for accuracy, speed, response times, etc.

-----------------------------------------------------------------------

Make script start automatically at boot (and when opening SSH terminal)

-----------------------------------------------------------------------

Type:

sudo nano /home/pi/.bashrc

In the editor, add in the last line:

sudo python /home/pi/pocketsphinx-python/PocketScript\*.py

Then reboot the pi.

Now the script should work when the raspberry turns on. Also it can be disabled with Ctrl-C.

Note: by default the .bashrc file only loads upon successful login through SSH. In order for the .bashrc file to load on boot, you must have "Console Autologin" enabled in raspi-config under "Boot options".

# PocketScriptHUB

Script used for the hub device. Some lines of code may be moved to new lines because of formatting.

#!/usr/bin/python

import sys, os, pyaudio, time, socket, fcntl, struct

from pocketsphinx.pocketsphinx import \*

from sphinxbase.sphinxbase import \*

import RPi.GPIO as GPIO

from threading import Thread

import paho.mqtt.client as mqtt

import paho.mqtt.publish as publish

class Subscriber(Thread):

def run(self):

def on\_connect(client, userdata, flags, rc):

print("Connected with result code "+str(rc))

client.subscribe(MQTT\_PATH)

def on\_message(client, userdata, msg):

print(msg.topic+" "+str(msg.payload))

try:

MQTT\_SERVER = "iot.eclipse.org"

MQTT\_PATH = "voice\_mqtt/measurements/#"

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(MQTT\_SERVER, 1883, 60)

client.loop\_forever()

except KeyboardInterrupt:

GPIO.cleanup()

#set event that causes it to gracefully quit

#---------defs------------

def getHwAddr(ifname):

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

info = fcntl.ioctl(s.fileno(), 0x8927, struct.pack('256s', ifname[:15]))

return '-'.join(['%02x' % ord(char) for char in info[18:24]])

def blinkdiode():

GPIO.output(ledpin, GPIO.HIGH)

time.sleep(2)

GPIO.output(ledpin, GPIO.LOW)

time.sleep(2)

GPIO.output(ledpin, GPIO.HIGH)

time.sleep(2)

GPIO.output(ledpin, GPIO.LOW)

def invalidcommand():

print("Control word must be said before commands")

#-----LED setup (optional)----------

GPIO.setmode(GPIO.BCM)

ledpin = 4

GPIO.setup(ledpin, GPIO.OUT, initial=GPIO.LOW)

#------PocketSphinx setup----------

modeldir = "/home/pi/pocketsphinx-5prealpha/model/"

config = Decoder.default\_config()

config.set\_string('-hmm', os.path.join(modeldir, 'en-us/en-us'))

config.set\_string('-dict', '/home/pi/pocketsphinx-python/keyphrase.dic')

config.set\_string('-kws', '/home/pi/pocketsphinx-python/keyphrase.list')

config.set\_float('-samprate', 16000.0)

config.set\_int('-nfft', 512)

p = pyaudio.PyAudio()

stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, frames\_per\_buffer=1024)

stream.start\_stream()

decoder = Decoder(config)

decoder.start\_utt()

try:

Subscriber().start()

Subscriber.daemon = True

mac = raw\_input("Type MAC address of robot you want to control.\n")

print("Connection set to: " +str(mac))

MQTT\_SERVER = "iot.eclipse.org"

MQTT\_PATH = ("voice\_mqtt/commands/" +str(mac))

print("Ready to listen")

control = False

while True:

if (control == True):

GPIO.output(ledpin, GPIO.HIGH)

else:

GPIO.output(ledpin, GPIO.LOW)

buf = stream.read(1024, exception\_on\_overflow = False)

decoder.process\_raw(buf, False, False)

if decoder.hyp() != None:

print ([(seg.word) for seg in decoder.seg()])

#print ([(seg.word, seg.prob, seg.start\_frame, seg.end\_frame) for seg in decoder.seg()])

print ("Detected keyword, restarting search")

# command execution

if (seg.word == 'BLINK DIODE '):

blinkdiode()

print('Blinking diode')

elif (seg.word == 'BEGIN '):

control = True

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Control word accepted')

elif (seg.word == 'STOP '):

control = False

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Stopping')

elif (seg.word == 'DRIVE '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Going forward')

if (control == False):

invalidcommand()

elif (seg.word == 'BACK '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Going backward')

if (control == False):

invalidcommand()

elif (seg.word == 'LEFT '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Turning left')

if (control == False):

invalidcommand()

elif (seg.word == 'RIGHT '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Turning right')

if (control == False):

invalidcommand()

elif (seg.word == 'FIRST '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Changed duty cycle to 25')

if (control == False):

invalidcommand()

elif (seg.word == 'SECOND '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Changed duty cycle to 50')

if (control == False):

invalidcommand()

elif (seg.word == 'THIRD '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Changed duty cycle to 75')

if (control == False):

invalidcommand()

elif (seg.word == 'FOURTH '):

publish.single(MQTT\_PATH, seg.word, hostname=MQTT\_SERVER)

print('Changed duty cycle to 99')

if (control == False):

invalidcommand()

decoder.end\_utt()

time.sleep(0.02)

decoder.start\_utt()

except KeyboardInterrupt:

print("Exception: KeyboardInterrupt")

GPIO.cleanup()

# PocketScriptROBOT

Script used for all of the robot devices. Some lines of code may be moved to new lines because of formatting.

#!/usr/bin/python

import sys, os, pyaudio, time, socket, fcntl, struct

from pocketsphinx.pocketsphinx import \*

from sphinxbase.sphinxbase import \*

import RPi.GPIO as GPIO

from threading import Thread

import paho.mqtt.publish as publish

import paho.mqtt.client as mqtt

class Publisher(Thread):

def run(self):

def getHwAddr(ifname):

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

info = fcntl.ioctl(s.fileno(), 0x8927, struct.pack('256s', ifname[:15]))

return '-'.join(['%02x' % ord(char) for char in info[18:24]])

try:

MQTT\_SERVER = "iot.eclipse.org"

mac = getHwAddr('wlan0')

MQTT\_PATH = ("voice\_mqtt/measurements/" +str(mac))

global rpm

count = 0

while True:

count += 1

#print(count)

if (count >= 60):

print("RPM: ", rpm)

publish.single(MQTT\_PATH, "RPM of robot: " +str(rpm), hostname=MQTT\_SERVER)

count = 0

rpm = 0

time.sleep(1)

except KeyboardInterrupt:

GPIO.cleanup()

#set event that causes it to gracefully quit

class Subscriber(Thread):

def run(self):

def on\_connect(client, userdata, flags, rc):

print("Connected with result code "+str(rc))

client.subscribe(MQTT\_PATH)

def on\_message(client, userdata, msg):

global control

print(msg.topic+" "+str(msg.payload))

phrase = msg.payload

control = command\_execution(phrase, control)

try:

global mac

control = False

MQTT\_SERVER = "iot.eclipse.org"

MQTT\_PATH = ("voice\_mqtt/commands/"+str(mac))

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_message = on\_message

client.connect(MQTT\_SERVER, 1883, 60)

client.loop\_forever()

except KeyboardInterrupt:

GPIO.cleanup()

#set event that causes it to gracefully quit

#---------defs------------

def getHwAddr(ifname):

s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

info = fcntl.ioctl(s.fileno(), 0x8927, struct.pack('256s', ifname[:15]))

return '-'.join(['%02x' % ord(char) for char in info[18:24]])

def rpm\_increment(tachometerpin): # event detect callback, on separate thread

global rpm

rpm += 1

#print(rpm) # tachometer debugging

def blinkdiode():

GPIO.output(ledpin, GPIO.HIGH)

time.sleep(1)

GPIO.output(ledpin, GPIO.LOW)

time.sleep(1)

GPIO.output(ledpin, GPIO.HIGH)

time.sleep(1)

GPIO.output(ledpin, GPIO.LOW)

def forward():

GPIO.output(left1, GPIO.HIGH)

GPIO.output(left2, GPIO.LOW)

GPIO.output(right1, GPIO.HIGH)

GPIO.output(right2, GPIO.LOW)

def backward():

GPIO.output(left1, GPIO.LOW)

GPIO.output(left2, GPIO.HIGH)

GPIO.output(right1, GPIO.LOW)

GPIO.output(right2, GPIO.HIGH)

def stop():

GPIO.output(motorpins, GPIO.LOW)

def leftturn():

GPIO.output(right1, GPIO.LOW)

GPIO.output(right2, GPIO.LOW)

GPIO.output(left1, GPIO.HIGH)

GPIO.output(left2, GPIO.LOW)

def rightturn():

GPIO.output(right1, GPIO.HIGH)

GPIO.output(right2, GPIO.LOW)

GPIO.output(left1, GPIO.LOW)

GPIO.output(left2, GPIO.LOW)

def changespeed(dc):

motor1.ChangeDutyCycle(dc)

motor2.ChangeDutyCycle(dc)

def invalidcommand():

print("Control word must be said before commands")

def command\_execution(phrase, control):

print ("Detected keyword, restarting search")

# command execution

if (phrase == 'BLINK DIODE '):

blinkdiode()

print('Blinking diode')

elif (phrase == 'BEGIN '):

control = True

print('Control word accepted')

elif (phrase == 'STOP '):

stop()

control = False

print('Stopping')

elif (phrase == 'DRIVE '):

if (control == True):

forward()

print('Going forward')

else:

invalidcommand()

elif (phrase == 'BACK '):

if (control == True):

backward()

print('Going backward')

else:

invalidcommand()

elif (phrase == 'LEFT '):

if (control == True):

leftturn()

print('Turning left')

else:

invalidcommand()

elif (phrase == 'RIGHT '):

if (control == True):

rightturn()

print('Turning right')

else:

invalidcommand()

elif (phrase == 'FIRST '):

if (control == True):

changespeed(25)

print('Changed duty cycle to 25')

else:

invalidcommand()

elif (phrase == 'SECOND '):

if (control == True):

changespeed(50)

print('Changed duty cycle to 50')

else:

invalidcommand()

elif (phrase == 'THIRD '):

if (control == True):

changespeed(75)

print('Changed duty cycle to 75')

else:

invalidcommand()

elif (phrase == 'FOURTH '):

if (control == True):

changespeed(99)

print('Changed duty cycle to 99')

else:

invalidcommand()

return control

#-----motor setup----------

GPIO.setmode(GPIO.BCM)

ledpin = 4

enable1 = 17

enable2 = 23

left1 = 27

left2 = 22

right1 = 25

right2 = 24

tachometerpin = 26

enablerpins = [enable1,enable2]

motorpins = [left1,left2,right1,right2]

GPIO.setup(enablerpins, GPIO.OUT, initial=GPIO.HIGH)

GPIO.setup(motorpins, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(ledpin, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(tachometerpin, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)

motor1 = GPIO.PWM(enable1,25)

motor1.start(25)

motor2 = GPIO.PWM(enable2,25)

motor2.start(25)

#------PocketSphinx setup----------

modeldir = "/home/pi/pocketsphinx-5prealpha/model/"

config = Decoder.default\_config()

config.set\_string('-hmm', os.path.join(modeldir, 'en-us/en-us'))

config.set\_string('-dict', '/home/pi/pocketsphinx-python/keyphrase.dic')

config.set\_string('-kws', '/home/pi/pocketsphinx-python/keyphrase.list')

config.set\_float('-samprate', 16000.0)

config.set\_int('-nfft', 512)

p = pyaudio.PyAudio()

stream = p.open(format=pyaudio.paInt16, channels=1, rate=16000, input=True, frames\_per\_buffer=1024)

stream.start\_stream()

decoder = Decoder(config)

decoder.start\_utt()

try:

mac = getHwAddr('wlan0')

Publisher().start()

Publisher.daemon = True

Subscriber().start()

Subscriber.daemon = True

rpm = 0

GPIO.add\_event\_detect(tachometerpin, GPIO.FALLING, callback=rpm\_increment, bouncetime=200)

blinkdiode()

print("Ready to listen")

control = False

while True:

if (control == True):

GPIO.output(ledpin, GPIO.HIGH)

else:

GPIO.output(ledpin, GPIO.LOW)

buf = stream.read(1024, exception\_on\_overflow = False)

decoder.process\_raw(buf, False, False)

if decoder.hyp() != None:

print ([(seg.word) for seg in decoder.seg()])

#print ([(seg.word, seg.prob, seg.start\_frame, seg.end\_frame) for seg in decoder.seg()])

control = command\_execution(seg.word, control)

decoder.end\_utt()

time.sleep(0.02)

decoder.start\_utt()

except KeyboardInterrupt:

print("Exception: KeyboardInterrupt")

GPIO.cleanup()

# VoiceScript (old)

This is an earlier iteration of the speech recognition script, intended to be used with the VoiceMacro program. Some lines of code may be moved to new lines because of formatting.

#!/usr/bin/python

import sys, tty, termios, time

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

ledpin = 4

enable1 = 17

enable2 = 23

left1 = 27

left2 = 22

right1 = 25

right2 = 24

GPIO.setup(ledpin, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(enable1, GPIO.OUT, initial=GPIO.HIGH)

GPIO.setup(enable2, GPIO.OUT, initial=GPIO.HIGH)

GPIO.setup(left1, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(left2, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(right1, GPIO.OUT, initial=GPIO.LOW)

GPIO.setup(right2, GPIO.OUT, initial=GPIO.LOW)

motor1 = GPIO.PWM(17,25) # set motor1 pwm frequency here

motor1.start(25)

motor2 = GPIO.PWM(23,25) # set motor2 pwm frequency here

motor2.start(25)

def getch():

fd = sys.stdin.fileno()

old\_settings = termios.tcgetattr(fd)

try:

tty.setraw(sys.stdin.fileno())

ch = sys.stdin.read(1)

finally:

termios.tcsetattr(fd, termios.TCSADRAIN, old\_settings)

return ch

def blinkdiode():

GPIO.output(ledpin, GPIO.HIGH)

time.sleep(2)

GPIO.output(ledpin, GPIO.LOW)

time.sleep(2)

GPIO.output(ledpin, GPIO.HIGH)

time.sleep(2)

GPIO.output(ledpin, GPIO.LOW)

def forward():

GPIO.output(left1, GPIO.HIGH)

GPIO.output(left2, GPIO.LOW)

GPIO.output(right1, GPIO.HIGH)

GPIO.output(right2, GPIO.LOW)

def backward():

GPIO.output(left1, GPIO.LOW)

GPIO.output(left2, GPIO.HIGH)

GPIO.output(right1, GPIO.LOW)

GPIO.output(right2, GPIO.HIGH)

def stop():

GPIO.output(left1, GPIO.LOW)

GPIO.output(left2, GPIO.LOW)

GPIO.output(right1, GPIO.LOW)

GPIO.output(right2, GPIO.LOW)

def leftturn():

GPIO.output(left1, GPIO.LOW)

GPIO.output(left2, GPIO.LOW)

GPIO.output(right1, GPIO.HIGH)

GPIO.output(right2, GPIO.LOW)

def rightturn():

GPIO.output(left1, GPIO.HIGH)

GPIO.output(left2, GPIO.LOW)

GPIO.output(right1, GPIO.LOW)

GPIO.output(right2, GPIO.LOW)

def speed25():

motor1.ChangeDutyCycle(25)

motor2.ChangeDutyCycle(25)

def speed50():

motor1.ChangeDutyCycle(50)

motor2.ChangeDutyCycle(50)

def speed75():

motor1.ChangeDutyCycle(75)

motor2.ChangeDutyCycle(75)

def speed100():

motor1.ChangeDutyCycle(99)

motor2.ChangeDutyCycle(99)

try:

print ("Program started. Press X or say 'exit program' to exit")

while(True):

char = getch()

if(char == "b"):

blinkdiode()

if(char == "w"):

forward()

if(char == "s"):

backward()

if(char == "q"):

stop()

if(char == "a"):

leftturn()

if(char == "d"):

rightturn()

if(char == "1"):

print ("Changed duty cycle to 25")

speed25()

if(char == "2"):

print ("Changed duty cycle to 50")

speed50()

if(char == "3"):

print ("Changed duty cycle to 75")

speed75()

if(char == "4"):

print ("Changed duty cycle to 100")

speed100()

if(char == "x"):

raise KeyboardInterrupt

except KeyboardInterrupt:

GPIO.cleanup()