Project Name: German/English Translator

Student Name: Marie-Pierre Beaudoin

Student ID: 20086635

https://github.com/Mariep441/CompSystems

www.mpb-hdip-compsys.surge.sh

https://youtu.be/0yT_5Y4-Tvc

Grade Band	Combined knowledge	Networking Technologies	IoT Solution	Communication
Base	2 strands present in output. Basic knowledge of each exhibited. *comp sys - python, bash * database – SQL database	Physical/Data link layer solution. *Emails	Sensors: *SenseHat *AudioCard	Read me + video
Good	apply concepts from > 2 modules *comp sys - networks * database – SQL queries * ICT skills - Github knowledge	Wireless protocols. * WiFi *WLAN Interconnected devices	processing/ gateway function *Router / firewall	Portfolio/repository + documentation.
Excellent	apply concepts from > 2 modules *comp sys- mqtt, IoT * ICT skills – JSON credentials *Web development - website	Lightweight messaging. * MQTT URL * MQTT Broker	IoT Application of good prototypical standard. Used to evaluate overall suitability for a production system.	Additional communication resources *complete step by step instructions
Outstanding	Self-acquired knowledge * Google Cloud platform	*Excellent Use of 2 Google Cloud API	Novel solution of clear applicability to specific domain.	All the above, accessible project platform *web site

- **Short project description.**

My family and I moved to Luxembourg recently and my children are now learning Luxembourgish and German at school. I want to turn my Raspberry Pi into a voice recognition German to English translator. * As I do not speak German myself, for testing purpose, the project translates French (French-Canadian) to English.

After pressing the sensorHat, the recording will start for 5 seconds. An "R" is displayed on the Raspberry Pi right after the button is pressed and disappeared to let the user know the recording has started. This creates a .wav file with the required specifications to be sent to the Google Speech-to-text API. Once the audio file is converted to text, it is then sent to Google translate. eSpeak renders the English translation.

Using the Flow in Wia, an email is sent after the word reached the web application. Doing so, it is possible to see how the spelling of the word we are attempting to translate.

Finally, the words, translations and time stamp are published via a MQTT broker. The webserver acts as the subscriber receiving the data and sending them to the dbserver, where they are store into a SQL database.

Altogether, this offers a solution with clear IoT and domain application. It includes processing / gateway function and the IoT Application of good prototypical standard.

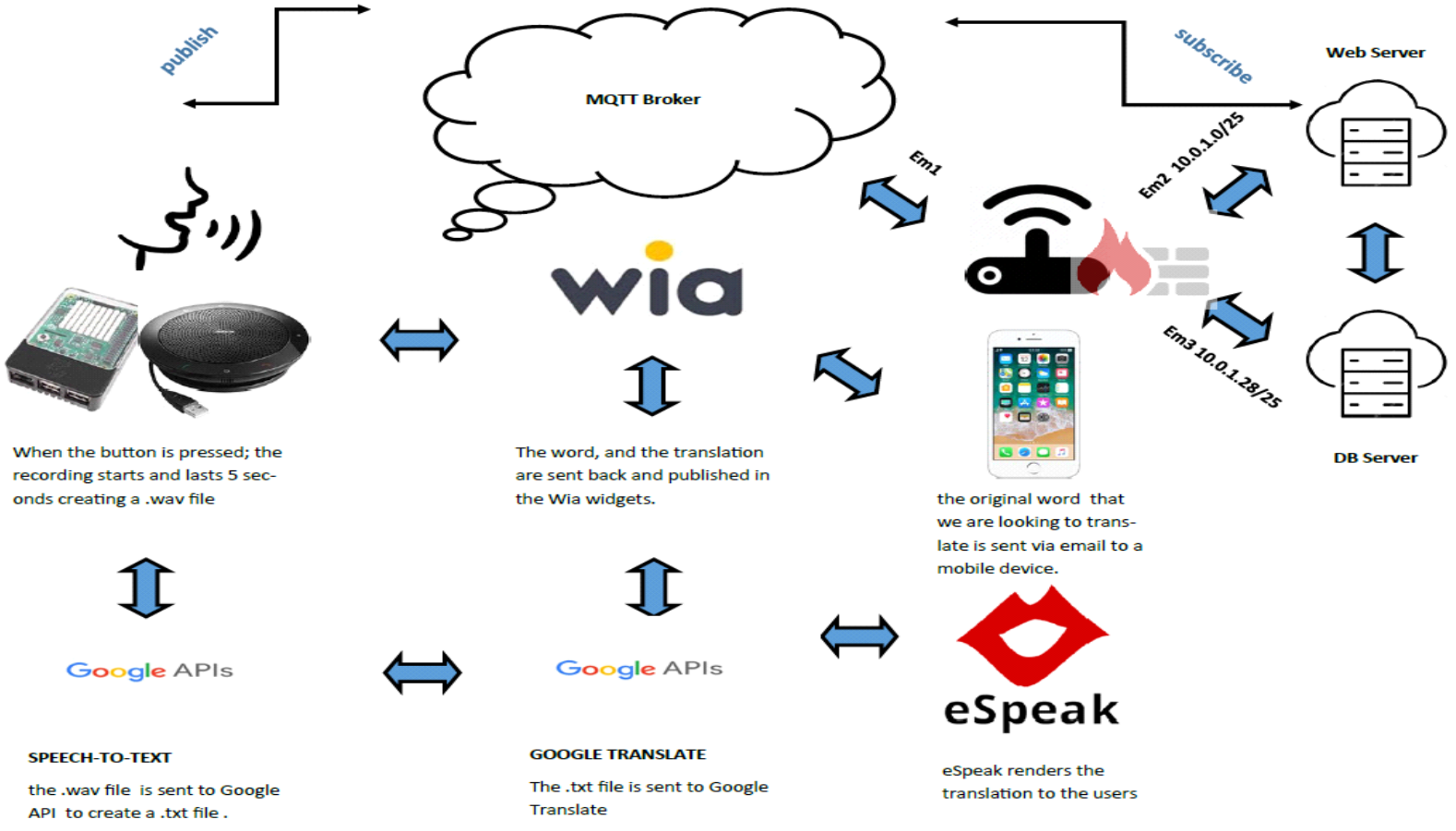
Tools, Technologies and Equipment

Equipment, Software:

- Raspberry Pi 3b+
- SenseHat,
- Jabra SPEAK 510 USB
- eSpeak
- Cloud MQTT Broker
- VirtualBox / Vagrant / webserver & dbserver (per lab week7)
- Google Cloud Platform (Google text-to-speech API & Google Translate)
- Wia

- **Workflow**

In the schematic, we can see a simple Physical/Data link layer solution, where emails are sent from Wia. Multiple wireless protocols have been utilized. The Raspberry Pi uses WIFI to connect to the network, the webserver is accessible via WLAN. Lightweight messaging architecture mediates between high and low level devices; the words, translations and time stamps are published via MQTT. The webserver acts as the subscriber receiving the data. Finally, the Google Cloud Platform is leveraged to provide the Speech-to-text and the Translation portions of the project.



- **Audio**

To verify the position of our sound card, we use the command lines below:

cat /proc/asound/cards
cat /proc/asound/modules

```
pi@raspberrypi:~ $ cat /proc/asound/cards
 0 [ALSA           ]: bcm2835_alsa - bcm2835 ALSA
                      bcm2835 ALSA
 1 [USB            ]: USB-Audio - Jabra SPEAK 510 USB
                      Jabra SPEAK 510 USB at usb-3f980000.usb-1.3, full speed
pi@raspberrypi:~ $ cat /proc/asound/modules
 0 snd_bcm2835
 1 snd_usb_audio
```

We must ensure that our USB Audio is used as the default setting of our raspberry pi. Replacing the default setting of 0 to 1

sudo nano /usr/share/alsa/alsa.conf

```
# show extended name hints
defaults.namehint.extended on
#
defaults.ctl.card 1
defaults.pcm.card 1
```

Finally, we create a config file with the following information:

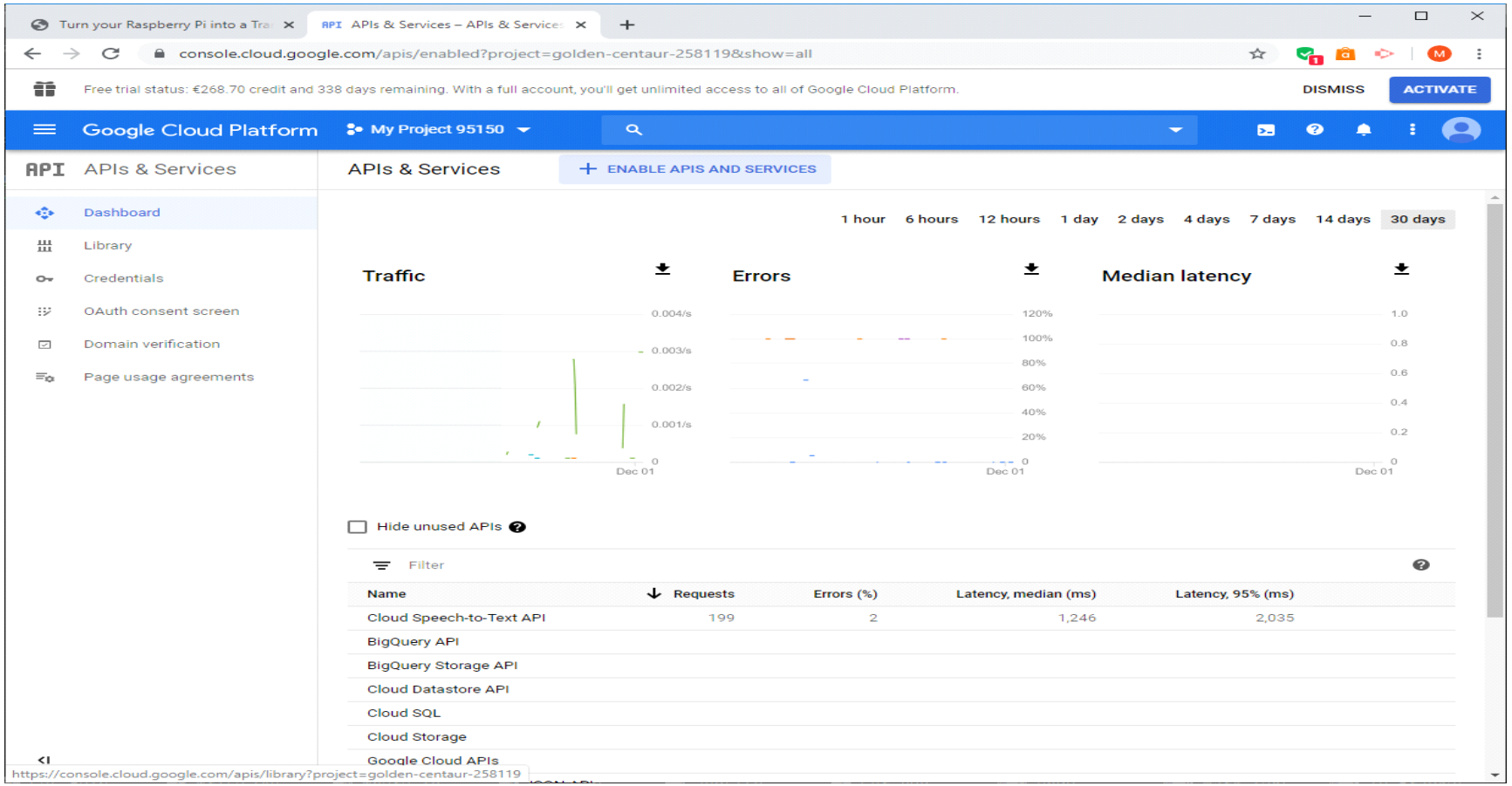
sudo nano /etc/asound.conf

```
pi@raspberrypi: ~
  GNU nano 3.2                                    /etc/asound.conf

pcm.!default {
  type plug
  slave {
    pcm "hw:1,0"
  }
}

ctl.!default {
    type hw
    card 1
}
```

This will make the default PCM (audio) output card #1 and the default control also card #1

- **APIs**

In order to use the Google API Speech-to-text, we need to create an account on the Google Cloud Platform. At Wit email address would give you access to the clouds for 365 days with a free credit of 268.70 euros. After gaining access to the platform, we need to download the credentials, in a Jason format, and store them within our project directory in the raspberry pi. The last step would simply be to enable the API that we would like to use for the project, in this case Google Speech-to-text API.



With regards to the Google Translate API, I have used "***pip install googletrans***" for a free and Unlimited Google translate API for Python

- **Codes**

The coding has been separated into three scripts. The Main script is in bash, in does call the other component and manage the entire process. The Button.py script control everything related to the SenseHat, it triggers the workflow on the button has been pressed. The last Script.py relates to the MQTT processing recalling the word and translation from previous steps and parsing them into MQTT URL and MQTT Broker

Main.sh

```
GNU nano 3.2                                                    main.sh

#!/bin/bash

  #run the python script to detect if the button have been pressed
  python button.py

  #use arecord to record the word to be translated in a format that can be used by google speech recognition
  arecord --device=hw:1,0 -d 2 --format S16_LE --rate 44100 -c1 word.wav

  echo "Converting Speech to Text..."

  #send the voice record to the Google cloud to have it a text file
  gcloud ml speech recognize  word.wav  --language-code='fr-CA'>file.txt

  #extract the word from the text file received from the Google cloud
  echo "You Said:"
  value=`cat file.txt`
  echo "$value"
  word=$(echo $value | sed -n -e 's/^.*"transcript": "//p'| sed 's/\" } ] } ] }//')

  echo "$word"

  #Export word to be ale to use it in the next python script
  export word

  #run the python script to send words to wia and to the dbserver
  python script.py

  echo "completed"
```

Button.py

```
  GNU nano 3.2                                                    button.py

#!/usr/bin/env python

import time

from wia import Wia
from sense_hat import SenseHat
sense = SenseHat()

wia = Wia()
wia.access_token ="d_sk_gmnO6Rpy0Pfnlpawq8NWWbfn"


while True:
  for event in sense.stick.get_events():
      if event.action == "pressed":
          sense.show_letter("R")
          wia.Event.publish(name="button",data="start recording")
          time.sleep(0.5)
          sense.clear()
          exit()
```

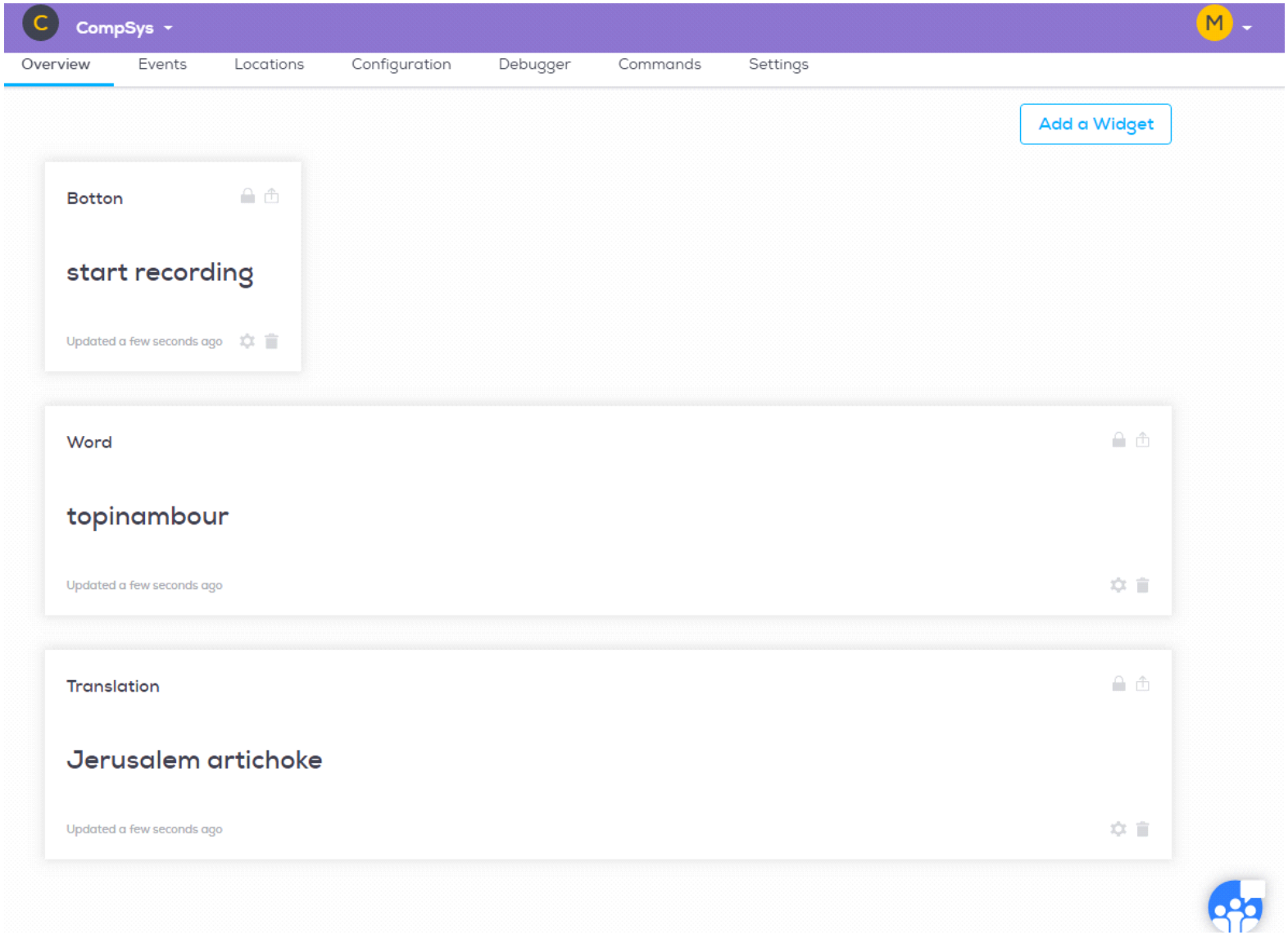
Script.py

```
 GNU nano 3.2                                script.py                                Modified

!/usr/bin/env python

mport paho.mqtt.client as mqtt
mport paho.mqtt.publish as publish
mport urlparse
mport sys
mport time
mport json
mport os
mport subprocess
rom translate import translation
rom wia import Wia
rom sense_hat import SenseHat

ense = SenseHat()

ia = Wia()
ia.access_token ="d_sk_gmnO6Rpy0Pfnlpawq8NWWbfn"

rd = os.environ['word']
rsl = translation

ia.Event.publish(name="word", data=wrd)
ia.Event.publish(name="translation", data=trsl)

 parse mqtt url for connection details
rl_str = "mqtt://test.mosquitto.org:1883/mpb/home"
rint(url_str)
rl = urlparse.urlparse(url_str)
ase_topic = url.path[1:]
uth=None
 Connect
f (url.username):
    auth = {'username':url.username, 'password':url.password}

 Publish a message

hile True:

    #Create JSON strings
    wrd_sensor=json.dumps({"word":wrd})
    trsl_sensor=json.dumps({"translation":trsl})

    #Create array of MQTT messages
    wrd_msg={'topic':base_topic, 'payload':wrd_sensor}
    trsl_msg={'topic':base_topic, 'payload':trsl_sensor}
    msgs=[wrd_msg,trsl_msg]

    #Publish array of messages
    publish.multiple(msgs, hostname=url.hostname, port=url.port, auth=auth)
    print("published")
    time.sleep(5)
    break
    exit()
```

Translate.py

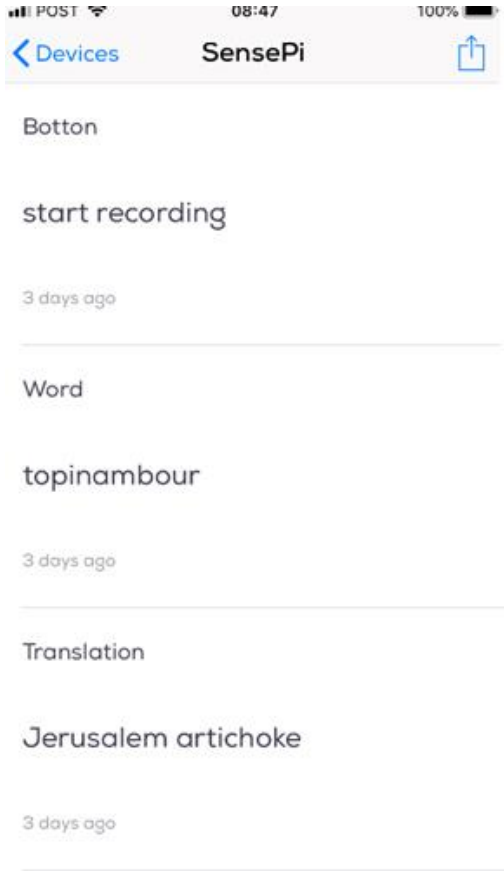
```
 GNU nano 3.2                                                translate.py

#!/usr/bin/python

import os
from subprocess import call

# Language Translator
from googletrans import Translator  # Import Translator module from googletrans package

translator = Translator() # Create object of Translator.

translated = translator.translate(os.environ['word'], src='fr', dest='en') # Pass both source and destination

translation = translated.text

# Source language auto detect by google trans
# By default destination language is English
print(" Source Language:" + translated.src)

print(" Translated string:" + translated.text)

call(["espeak","-s140 -ven+18 -z",translated.text])
```

- **Wia.io**

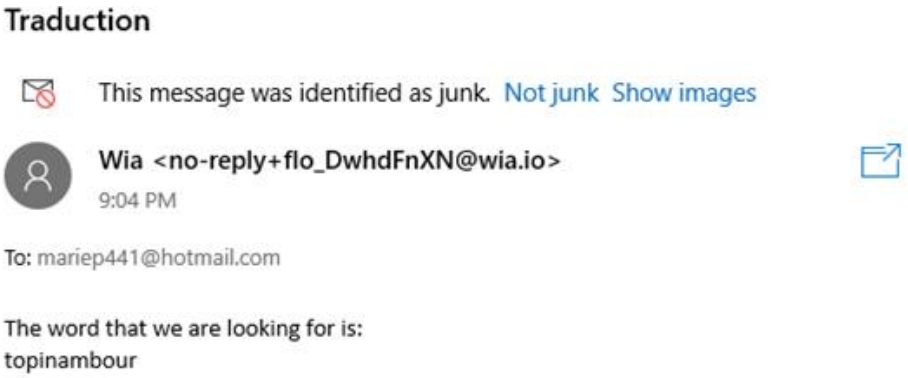

CompSys
M
Overview
Events
Locations
Configuration
Debugger
Commands
Settings
Add a Widget
Botton
start recording
Updated a few seconds ago
Word
topinambour
Updated a few seconds ago
Translation
Jerusalem artichoke
Updated a few seconds ago

From my iPhone

Wia application



Email sent by Wia



- **Data Persistence**

The Raspberry Pi is using lightweight messaging to publish the word, translation and time stamps to a Cloud MQTT broker. The webserver subscribes to the feed using the script below:

```
 GNU nano 2.2.6                                    File: client-sub.sh

!/bin/bash
cho "subscriber started"

osquitto_sub -h test.mosquitto.org -t 'mpb/home'  | while read RPI_DATA

o
cho  $RPI_DATA
D=$(jq '.word' <<< "$RPI_DATA")
RL=$(jq '.translation' <<< "$RPI_DATA")
S=$(jq '.timestamp' <<< "$RPI_DATA")


ysql -h 10.0.1.130  -uroot -e "insert into myProject.Traduction(timestamp,word,translation) VALUES(from_unixtime($TS), '$WD', '$TRL');"
```

The webserver is using its network connection with the dbserver to insert the data collected from the MQTT broker into a SQL database. Like we did in the lab week 7, I have created a database and table to store the data in MYSQL on the dbserver (ip address 10.0.1.130).

vagrant@webserver:~$ mysql -uroot -h10.0.1.130 -e "CREATE DATABASE myProject"

vagrant@webserver:~$ mysql -uroot -h10.0.1.130 -e "CREATE TABLE myProject.Traduction (word VARCHAR(255), translation VARCHAR(255), timestamp TIMESTAMP)"

To see the collection of data,

vagrant@webserver:~$ mysql -uroot -h10.0.1.130 -e "select * from myProject.Traduction"

```
vagrant@webserver:~/project$ mysql -uroot -h10.0.1.130 -e "select * from myProject.Traduction"
+---------------+------------------------+---------------------+
| word          | translation            | timestamp           |
+---------------+------------------------+---------------------+
| "topinambour" | null                   | 2019-12-01 20:04:47 |
| null          | "Jerusalem artichoke" | 2019-12-01 20:04:47 |
+---------------+------------------------+---------------------+
```