Module 1 - Introduction to Raspberry Pi and Jasper

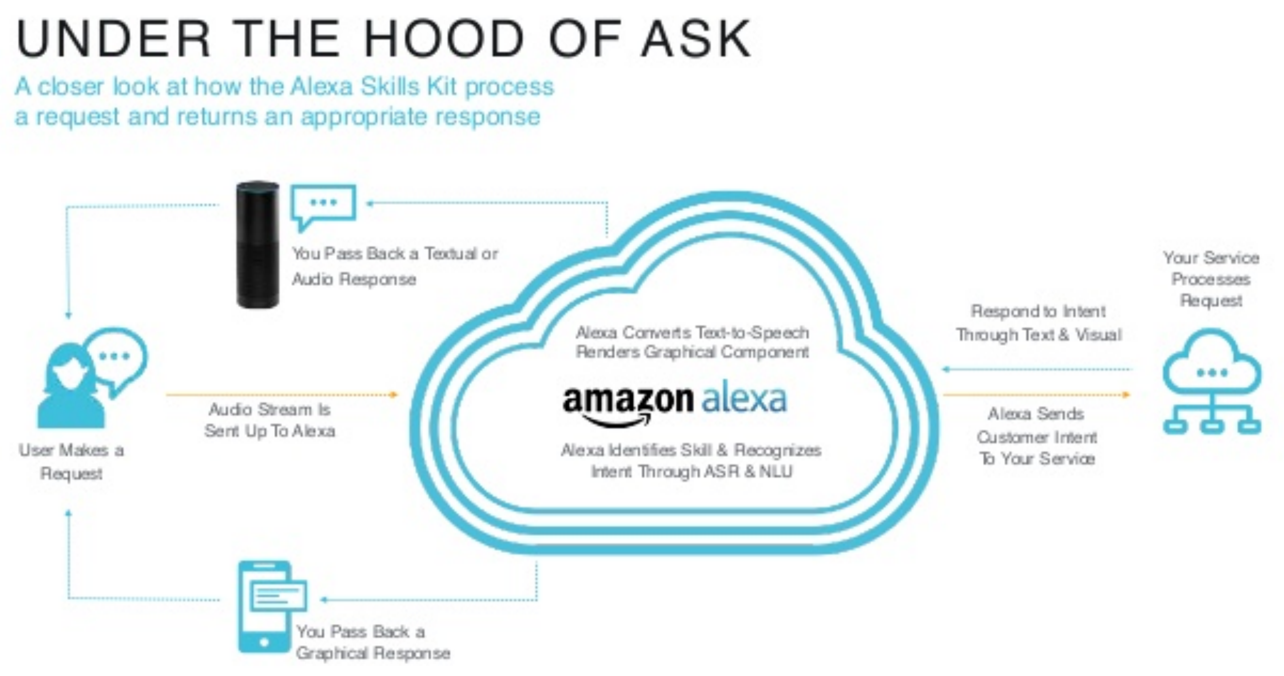
In this module we'll study personal assistant technologies in a hands-on manner. We'll run Jasper, a python version of a personal assistant, on self-contained Raspberry Pi hardware and then learn to write our own web services running on a laptop to mimic cloud infrastructure.

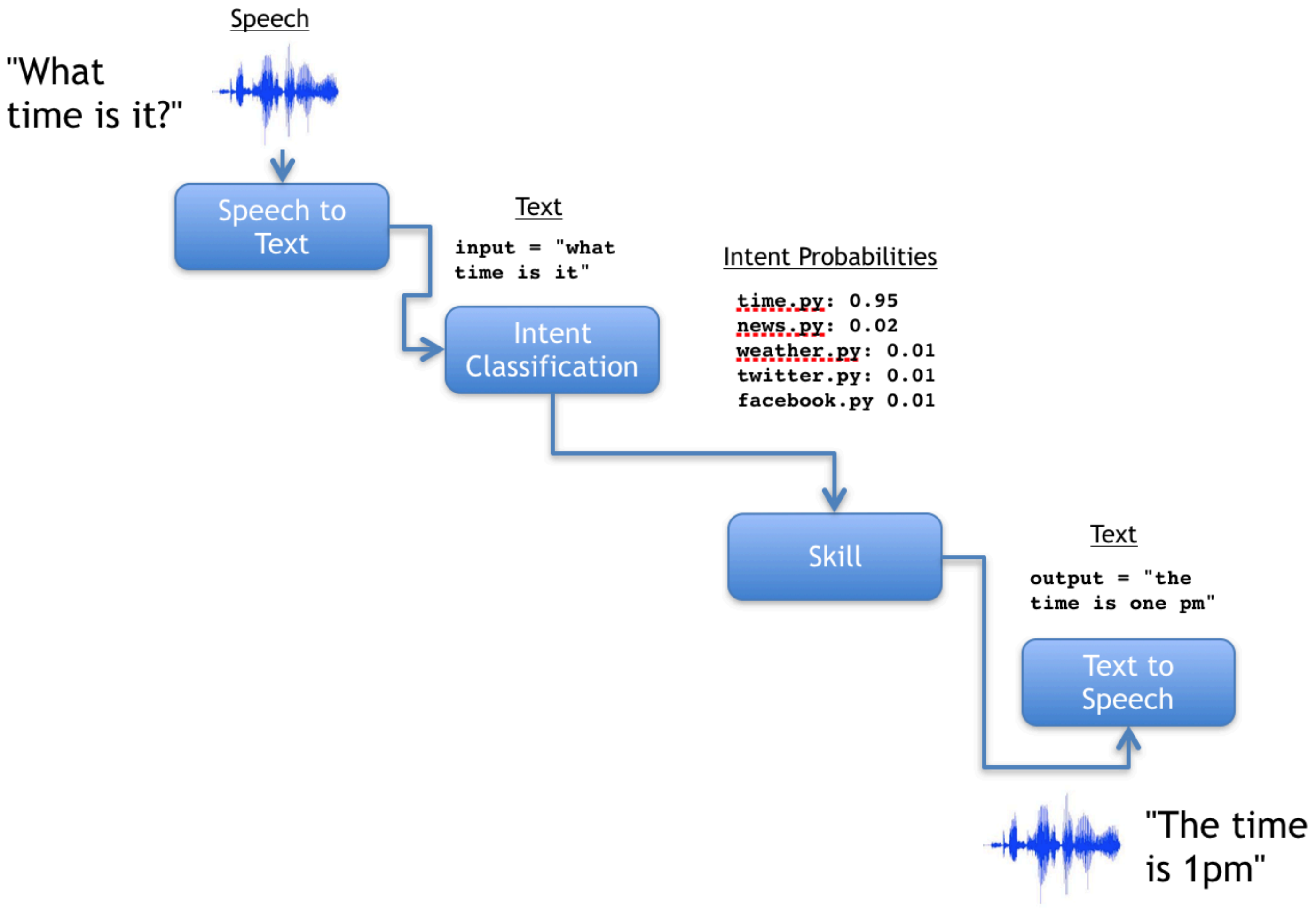
Skill Demonstration

Reddit news reader on Alexa.

Another demo:
https://www.youtube.com/watch?v=TKw3SeiIq4U

Personal assistant interaction

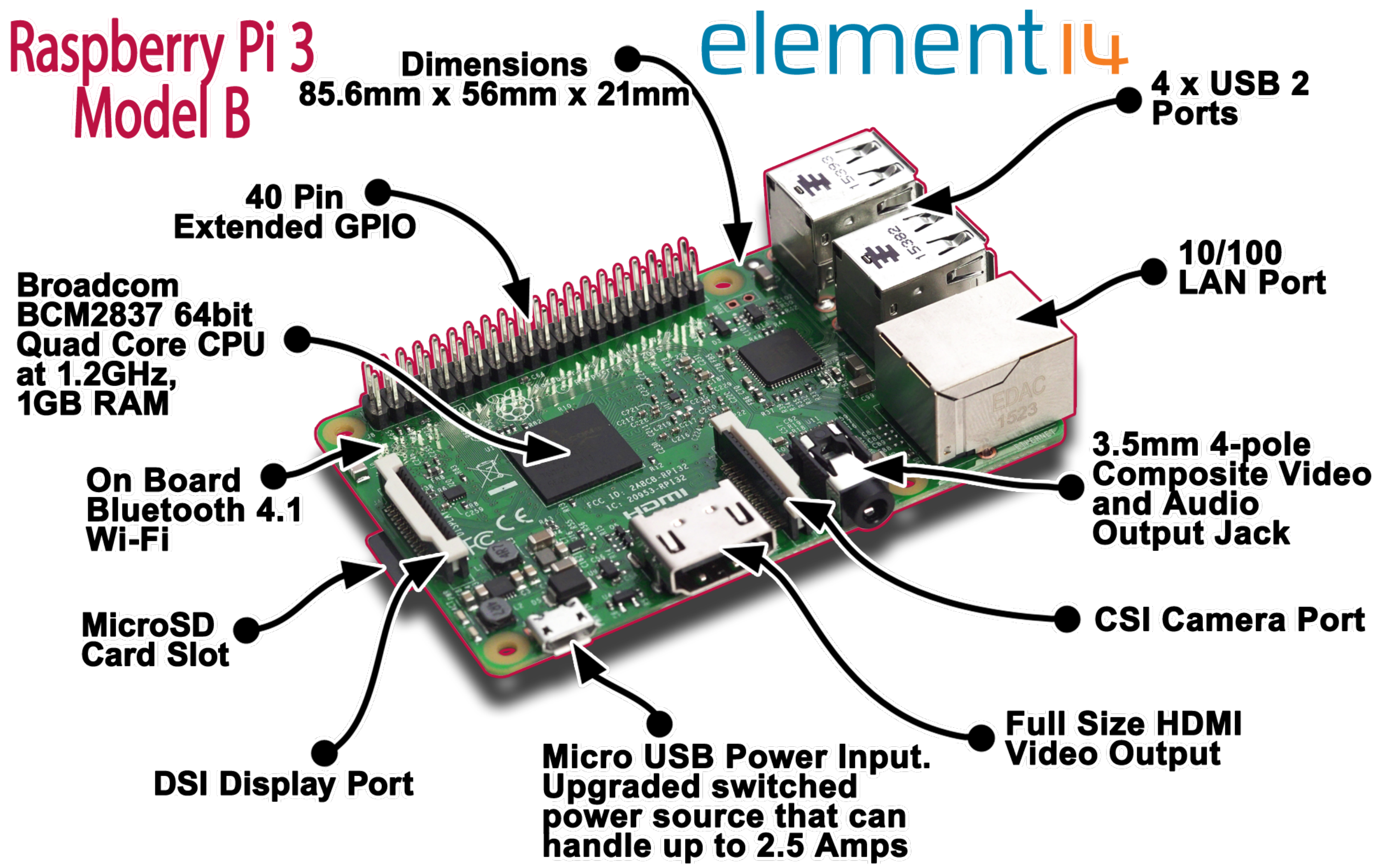

UNDER THE HOOD OF ASK
A closer look at how the Alexa Skills Kit process a request and returns an appropriate response
You Pass Back a Textual or Audio Response
User Makes a Request
Audio Stream Is Sent Up To Alexa
Alexa Converts Text-to-Speech Renders Graphical Component
amazon alexa
Alexa Identifies Skill & Recognizes Intent Through ASR & NLU
Respond to Intent Through Text & Visual
Your Service Processes Request
Alexa Sends Customer Intent To Your Service
You Pass Back a Graphical Response



We'll implement similar components with our own hardware and software:

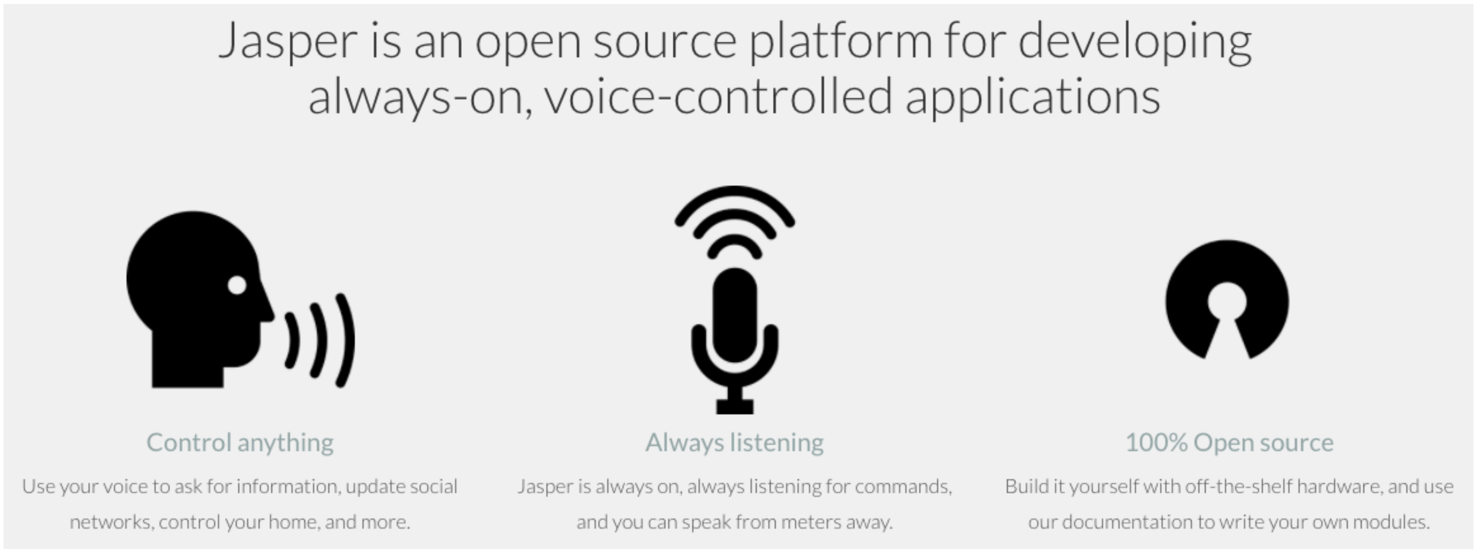
- Speech-to-text and text-to-speech
- Skills that run locally. Later, we'll build our own web services using Flask to run computationally heavy skills.
- How to replace simple keyword intent classification with machine learning and NLP methods for better accuracy and flexibility.

Raspberry Pi overview


Raspberry Pi 3 Model B
element14
Dimensions 85.6mm x 56mm x 21mm
4 x USB 2 Ports
40 Pin Extended GPIO
10/100 LAN Port
Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz, 1GB RAM
On Board Bluetooth 4.1 Wi-Fi
3.5mm 4-pole Composite Video and Audio Output Jack
MicroSD Card Slot
CSI Camera Port
DSI Display Port
Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps
Full Size HDMI Video Output

Jasper overview

Link: http://jasperproject.github.io/



Jasper is an open source platform for developing always-on, voice-controlled applications

Control anything

Use your voice to ask for information, update social networks, control your home, and more.

Always listening

Jasper is always on, always listening for commands, and you can speak from meters away.

100% Open source

Build it yourself with off-the-shelf hardware, and use our documentation to write your own modules.

Connect to Raspberry Pi through SSH

Make sure the Pi and your computer are connected to the same network.

Determine your IP address:

```
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:36:23:2b
          inet6 addr: fe80::3a44:883e:b333:5bf3/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:204 errors:0 dropped:0 overruns:0 frame:0
          TX packets:204 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:17184 (16.7 KiB)  TX bytes:17184 (16.7 KiB)

wlan0     Link encap:Ethernet  HWaddr b8:27:eb:63:76:7e
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::ee5b:10ea:84d4:a21b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:79 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9674 (9.4 KiB)  TX bytes:15230 (14.8 KiB)
```

The IP address that we want is under the `wlan0` interface and is labeled `inet addr:192.168.0.100`. Your output will vary from the example based on the IP addressed assigned by the wireless router.

SSH using the IP address using following command:

```
ssh pi@192.168.0.100
```

The username is `pi` and the default password on the provided disk image is `cogworks`.

The first time you log in with SSH, you will get this message, which you can accept:

```
Warning: Permanently added the RSA host key for IP address '<IP_ADDRESS>' to the l
```

Now we are remotely logged into the Pi and can control it without connecting to a monitor.

Note: if on a wireless router at home, you may be able to use the hostname `raspberrypi.local` like so:

```
ssh pi@raspberrypi.local
```

Run Jasper

- Go to the home directory (`cd ~`) and run the following: `jasper/jasper.py` .
  - Jasper will respond with a greeting with the name configured in your profile.
- Say “jasper” (or the wake word that you configured).
  - Jasper will beep and listen for a command.
- Say “time”.
  - This will invoke the time-telling skill and speak the current time.

Repeat the process and try other skills such as “weather tomorrow”, “news”.

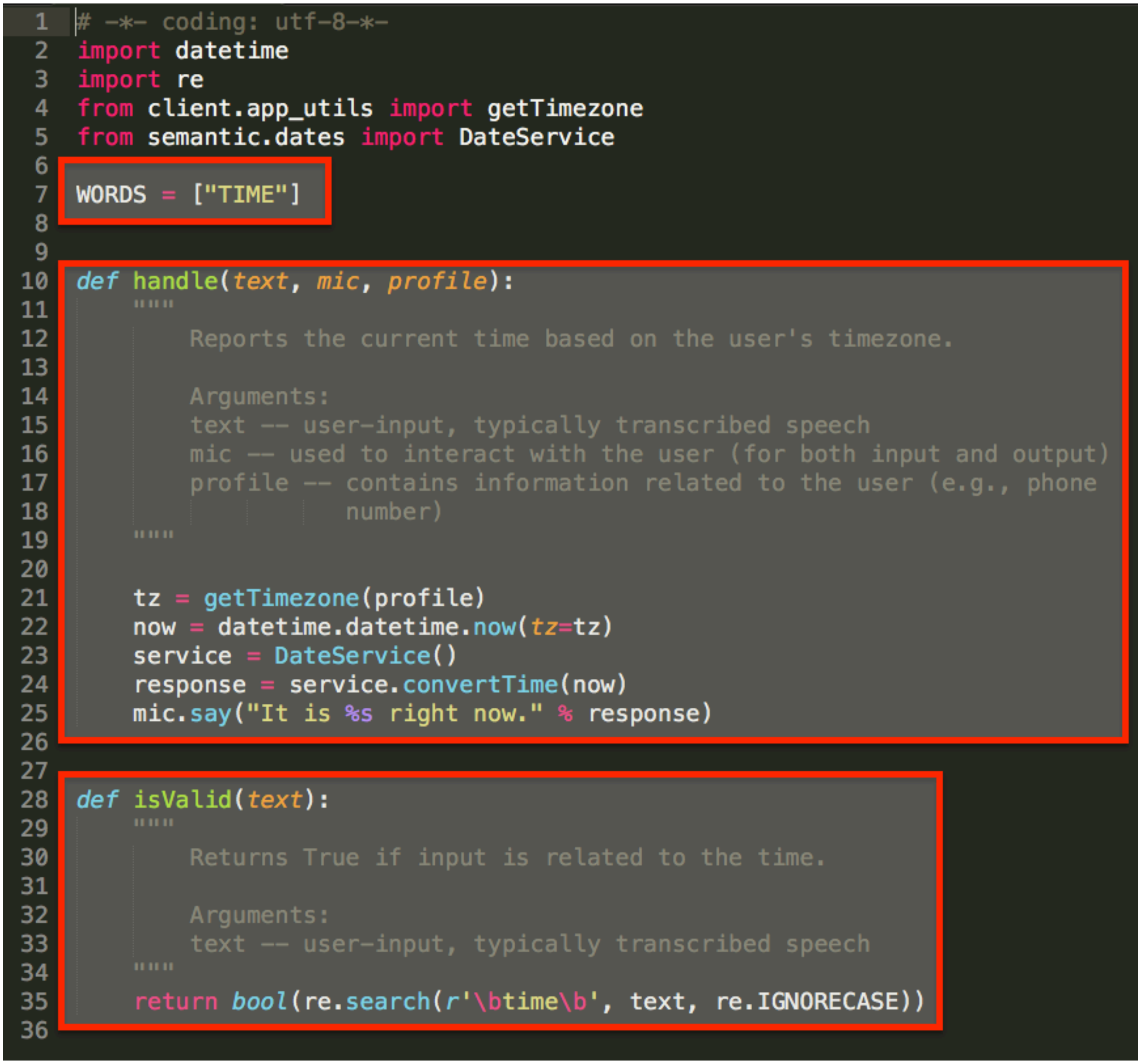
Jasper initially enters “passive listening” where it will wait until it hears the Wake word. After detecting the Wake word, Jasper enters “active listening” and waits for a command to invoke a skill.

Anatomy of a Skill

Stored in `jasper/client/modules` .

Common interface expected by `Brain.query()` :

- `WORDS` : list of strings that should trigger this skill
- `PRIORITY` : (optional) integer priority value for `Brain.get_modules()` to sort this skill in its checklist
- `isValid(text)` : returns True to `Brain` if input is related to this skill.
- `handle(text, mic, profile)` : All processing goes here. Can use information in `profile` and `text` and handles any audio input and output with `mic` .



```python
# -*- coding: utf-8-*-
import datetime
import re
from client.app_utils import getTimezone
from semantic.dates import DateService

WORDS = ["TIME"]


def handle(text, mic, profile):
    """
        Reports the current time based on the user's timezone.

        Arguments:
        text -- user-input, typically transcribed speech
        mic -- used to interact with the user (for both input and output)
        profile -- contains information related to the user (e.g., phone
                   number)
    """

    tz = getTimezone(profile)
    now = datetime.datetime.now(tz=tz)
    service = DateService()
    response = service.convertTime(now)
    mic.say("It is %s right now." % response)


def isValid(text):
    """
        Returns True if input is related to the time.

        Arguments:
        text -- user-input, typically transcribed speech
    """
    return bool(re.search(r'\btime\b', text, re.IGNORECASE))

```

Using TTS and STT modules

```python
from client import tts
from client import stt
from client import jasperpath
from client.mic import Mic

stt_engine_class = stt.get_engine_by_slug("sphinx")
stt_passive_engine_class = stt_engine_class
tts_engine_class = tts.get_engine_by_slug("google-tts")

mic = Mic(tts_engine_class.get_instance(),
          stt_passive_engine_class.get_passive_instance(),
          stt_engine_class.get_active_instance())

# Speak text
mic.say("I'm sorry, Hal, I can't do that")

# Transcribe audio (listens 10 seconds for keywords)
mic.activeListen()
```

Use Jasper in “local” mode

To bypass the microphone input and speaker output, you can run Jasper like so:

```
jasper/jasper.py --local
```

This will allow you to input words by typing and cause Jasper to print the spoken words.

Shutdown the Raspberry Pi

Use the following command to shutdown:

```
sudo shutdown now
```

Wait for the green light on the Raspberry Pi to stop blinking, and unplug the power. The green light indicates writing operations to the SD card.

Build your own Jasper skill

This minimally-complex skill causes Jasper to speak a greeting back to you.

```python

import random
import re  # Regular expressions

WORDS = ["HELLO"]


def handle(text, mic, profile):
    """
        Responds to user-input, typically speech text, by saying "hi"

        Arguments:
        text -- user-input, typically transcribed speech
        mic -- used to interact with the user (for both input and output)
        profile -- contains information related to the user (e.g., name)
    """
    first_name = profile["first_name"]

    messages = ["Hi",
                "Hello"

    message = random.choice(messages) + ", " + first_name

    mic.say(message)


def isValid(text):
    """
        Returns True if the input is "hi" or "hello"

        Arguments:
        text -- user-input, typically transcribed speech
    """
    return bool(re.search(r'\bhello\b', text, re.IGNORECASE))
```