Control Thymio Robot via Voice Commands

Using vosk Speech to Text Library

By Ahmad Agbaryah

Abstract

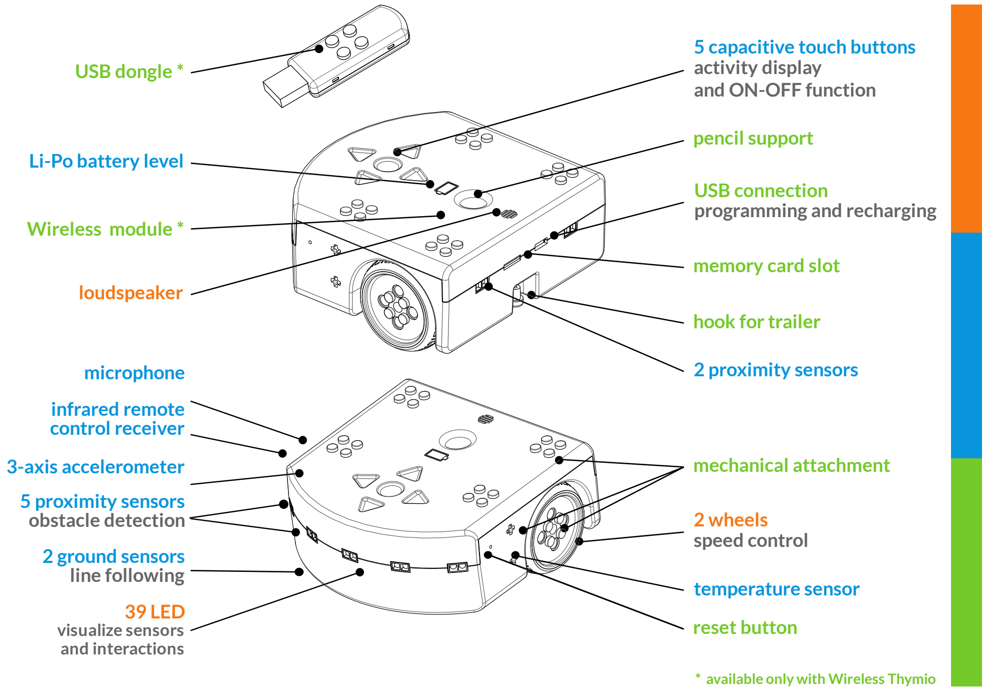
This proof of concept combining a wireless Thymio II with a python software on computer to control the movements of the robot. Speech to text is performed on the computer, using vosk library with an offline model. Once the python application recognizes new voice, it turns it to text and based on the result, the python script decides what command to send to the Thymio robot.

First, let’s have some base knowledge on the pieces of the proof of the concept.

What Thymio is?

Thymio is an open-source educational robot designed by researchers from the EPFL, in collaboration with ECAL, and produced by Mobsya, a nonprofit association whose mission is to offer comprehensive, engaging STEAM journeys to learners of all ages. For more information, visit the official web site at: https://www.thymio.org/

What is Thymio composed of?[1]


USB dongle *
Li-Po battery level
Wireless module *
loudspeaker
microphone
infrared remote control receiver
3-axis accelerometer
5 proximity sensors obstacle detection
2 ground sensors line following
39 LED visualize sensors and interactions
5 capacitive touch buttons activity display and ON-OFF function
pencil support
USB connection programming and recharging
memory card slot
hook for trailer
2 proximity sensors
mechanical attachment
2 wheels speed control
temperature sensor
reset button
* available only with Wireless Thymio

[1] Image from: http://wiki.thymio.org/en:thymiospecifications

What is VOSK?

Kaldi is a research speech recognition toolkit which implements many state of the art algorithms. Vosk is a practical speech recognition library which comes with a set of accurate models, scripts, practices and provides ready to use speech recognition for different platforms like mobile applications or Raspberry Pi. If you are doing research, Kaldi is probably your way. If you want to build practical applications with plug and play library, consider Vosk. (see https://alphacephei.com/vosk/faq).

The best things in Vosk are:

1) Supports 20+ languages and dialects - English, Indian English, German, French, Arabic and many more (for full list of supported languages, visit the official site: https://alphacephei.com/vosk/).
2) Works offline, even on lightweight devices - Raspberry Pi, Android, iOS
3) Installs with simple pip3 install vosk
4) Provides streaming API for the best user experience.

Demo

You may view our demo in the github page https://github.com/ahmad081177/control-thymio-via-voice or in the following YouTube link: https://youtu.be/3wKyVvp2RC0 .

What is Needed?

Hardware Pieces

To duplicate the demo on your end, you need the following:

1- Wireless Thymio II (with wireless dongle)

2- Laptop (With microphone)

Software Pieces

Following are the main pieces of the software to recreate the demo. In the next section, we are going to provide step by step instructions to setup the environment on your end.

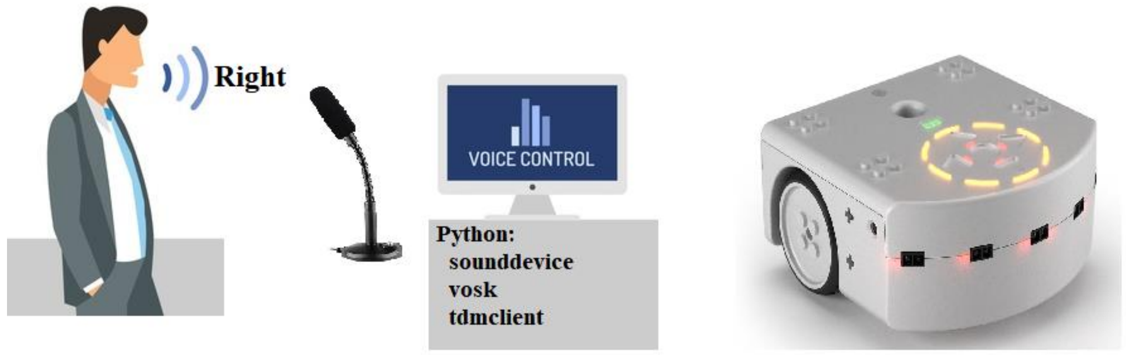
1- Python 3.x (Main development programming language)

2- Thymio Suite (Connect the computer with Thymio wirelessly)

3- VOSK model (offline speech recognition)

4- Python scripts to integrate all above together

Architecture of the Proof-Of-Concept



Person talks something, e.g., “right”, to control the robot to move to the right. The microphone receives the voice and translate the voice to audio-data thru “sounddevice” python library. VOSK library translates the audio data to text using the Speech-To-Recognition offline model (Speech to recognition is a natural language processing model that uses deep learning and other techniques to convert audio raw data to text). Python script receives the translated text and decide which command to send to the robot thru the “tdmclient” library.

Python

high-level programming language with dynamic semantics. Python supports modules and packages, which encourages program modularity and code reuse.

Pip

PIP is a package manager for Python packages, or modules if you like. A package contains all the files you need for a module. Modules are Python code libraries you can include in your project.

vosk

Vosk is an offline open-source speech recognition toolkit. It enables speech recognition for 20+ languages and dialects. Vosk supplies speech recognition for chatbots, smart home appliances, virtual assistants. It can also create subtitles for movies, transcription for lectures and interviews.

sounddevice

This Python module provides bindings for the PortAudio library and a few convenience functions to play and record NumPy arrays containing audio signals. (source https://pypi.org/project/sounddevice/)

tdmclient

Python package to connect to a Thymio II robot via the Thymio Device Manager (TDM), a component of the Thymio Suite. The connection between Python and the TDM is done over TCP to the port number advertised by zeroconf. Simple Python programs can run directly on the Thymio thanks to a transpiler. (source: https://pypi.org/project/tdmclient)

Software Setup

Follow the steps below to have the needed software pieces in your environment. In the next section, we are going to detail the python scripts of the software.

1) Install python v3 on your computer. You can find more details at the Python official site: https://www.python.org/downloads/. In this software, our main development programming language is python, it has plugins for the microphone, an offline speech to text and the TDM client library to control the Thymio robot.

2) Install the Thymio Suite on your computer. More details at the Thymio official site at: https://www.thymio.org/download-thymio-suite/.

3) Prepare Python environment:

- On your command line, type the following command (specify your own path) to create your own development workspace environment:

```
python3 -m venv c:\work\thymio
```

The above command allows the development environment to be isolated and not to clash with other environments. Installing addons will be for this environment only.

- Activate the environment by typing the following command on your command shell:

```
C:\work\thymio\Scripts\Activate.bat
```

The above command tells python that any addon installation will be for the current workspace only.

- Install required libraries by typing the following command:

```
(thymio) c:\work\thymio>pip install vosk sounddevice tdmclient
```

"pip install" tells python to install the following addons to the active environment.

The first library, vosk, is responsible for speech to text recognition (offline).

The second one, sounddevice, wraps the audio devices such as microphone and convert audio streamed from the microphone to raw data. Later, this raw data is fed into vosk model to convert it to text.

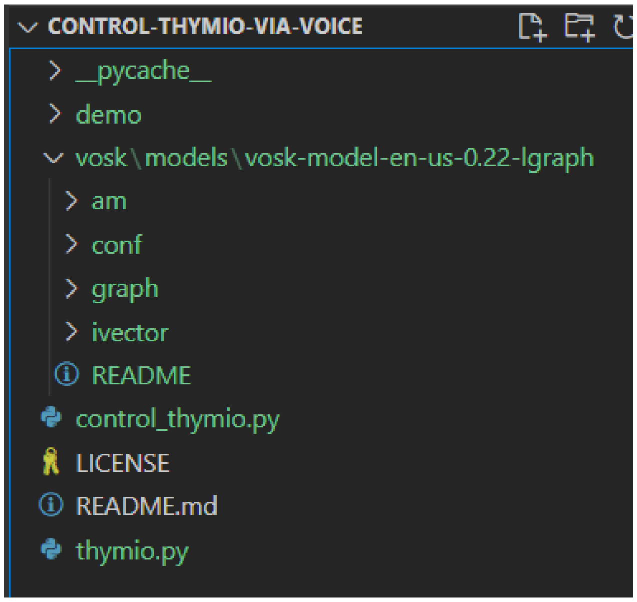
The last one is the Python Thymio wrapper to control the robot.

- Download proof of concept code from github to your local disk, e.g., c:\work\thymio\src\control-thymio-via-voice. Github code: https://github.com/ahmad081177/control-thymio-via-voice

- Download the VSOK model for offline recognition. Visit the page: https://alphacephei.com/vosk/models and pick any model you prefer. In my proof of concept, I've used both models: vosk-model-en-us-0.22 and vosk-model-en-us-0.22-lgraph.

  Place the model in your local disk, e.g., in c:\work\thymio\src\control-thymio-via-voice\vosk\models (If you place it otherwise, you may reflect the new path in control_thymio.py)

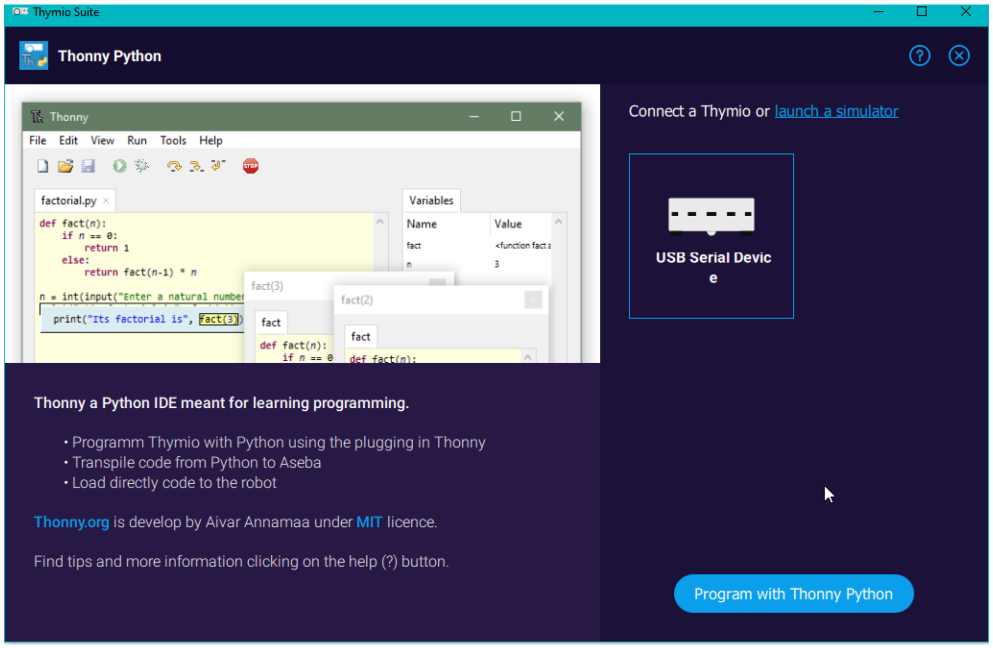
- Now you may have the following file structure:



```
CONTROL-THYMIO-VIA-VOICE
  > __pycache__
  > demo
  v vosk\models\vosk-model-en-us-0.22-lgraph
      > am
      > conf
      > graph
      > ivector
      README
    control_thymio.py
    LICENSE
    README.md
    thymio.py
```

Run the Software

Make sure you have Thymio II turned on and the wireless dongle is plugged in your computer. If you run Thymio Suite software, you may see something like:


Thymio Suite
Thonny Python
Thonny
File Edit View Run Tools Help
factorial.py
Variables
Name Value
Connect a Thymio or launch a simulator
USB Serial Device
Program with Thonny Python

```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(n-1) * n

n = int(input("Enter a natural number
print("Its factorial is", fact(3))
```

Thonny a Python IDE meant for learning programming.

- Programm Thymio with Python using the plugging in Thonny
- Transpile code from Python to Aseba
- Load directly code to the robot

Thonny.org is develop by Aivar Annamaa under MIT licence.

Find tips and more information clicking on the help (?) button.

In your command shell, type:

```
(thymio) c:\work\thymio\src\control-thymio-via-voice>python
control_thymio.py
```

The system will initialize the connection to the microphone, then initialize the connection to the Thymio II. VOSK library will transcript your voice. The python script will convert the text command to thymio's command and activate it.

Here is sample output for one of the demo's run:

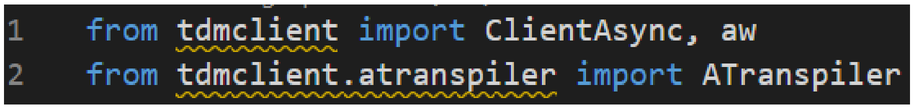
```
(thymio) c:\work\thymio\src\control-thymio-via-voice>python control_thymio.py
To create new Thymio interface
To start Thymio interface
Thymio interface has been started successfully
===> Initial Default Device Number:1 Description: {'name': 'Microphone Array (Realtek(R) Au', 'hostapi': 0,
'max_input_channels': 2, 'max_output_channels': 0, 'default_low_input_latency': 0.09,
'default_low_output_latency': 0.09, 'default_high_input_latency': 0.18, 'default_high_output_latency': 0.18,
'default_samplerate': 44100.0}
===> Build the model and recognizer objects.  This will take a few minutes.
LOG (VoskAPI:ReadDataFiles():model.cc:213) Decoding params beam=13 max-active=7000 lattice-beam=6
LOG (VoskAPI:ReadDataFiles():model.cc:216) Silence phones 1:2:3:4:5:11:12:13:14:15
LOG (VoskAPI:RemoveOrphanNodes():nnet-nnet.cc:948) Removed 0 orphan nodes.
LOG (VoskAPI:RemoveOrphanComponents():nnet-nnet.cc:847) Removing 0 orphan components.
LOG (VoskAPI:ReadDataFiles():model.cc:248) Loading i-vector extractor from
D:\workdir\units\weizmann\thymio\vosk\models\vosk-model-en-us-0.22/ivector/final.ie
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:183) Computing derived variables for iVector extractor
LOG (VoskAPI:ComputeDerivedVars():ivector-extractor.cc:204) Done.
LOG (VoskAPI:ReadDataFiles():model.cc:279) Loading HCLG from C:\work\thymio\src\control-thymio-via-
voice\vosk\models\vosk-model-en-us-0.22/graph/HCLG.fst
LOG (VoskAPI:ReadDataFiles():model.cc:294) Loading words from C:\work\thymio\src\control-thymio-via-
voice\vosk\models\vosk-model-en-us-0.22/graph/words.txt
LOG (VoskAPI:ReadDataFiles():model.cc:303) Loading winfo C:\work\thymio\src\control-thymio-via-voice\vosk\models\vosk-
model-en-us-0.22/graph/phones/word_boundary.int
LOG (VoskAPI:ReadDataFiles():model.cc:310) Loading subtract G.fst model from C:\work\thymio\src\control-thymio-via-
voice\vosk\models\vosk-model-en-us-0.22/rescore/G.fst
LOG (VoskAPI:ReadDataFiles():model.cc:312) Loading CARPA model from C:\work\thymio\src\control-thymio-via-
voice\vosk\models\vosk-model-en-us-0.22/rescore/G.carpaLOG (VoskAPI:ReadDataFiles():model.cc:318) Loading RNNLM
model from C:\work\thymio\src\control-thymio-via-voice\vosk\models\vosk-model-en-us-0.22/rnnlm/final.raw===> Begin
recording. Press Ctrl+C to stop the recording
{
  "text" : "move"
}
{
  "text" : "speed up"
}
{
  "text" : "speed"
}
{
  "text" : "left"
}
no input sound
{
  "text" : "stop"
}
{
  "text" : "start"
}
no input sound
{
  "text" : "left"
}
{
  "text" : "stop"
}
{
  "text" : "start"
}
{
  "text" : "right"
}
no input sound
{
  "text" : "stop"
}
{
  "text" : "back"
}
{
  "text" : "speed"
}
```

The scripts

As noticed, the software includes of some python's libraries, such as: tdmclient and vosk (which we already explained in the previous sections). The second part is the main two scripts that plugin all the puzzle pieces to create the working software.

Thymio.py

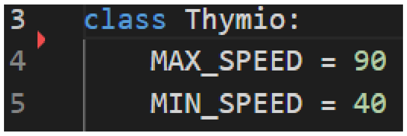
This script wraps the Thymio commands by calling APIs of the tdmclient library. Following is detail explanation about the commands of the script:



```
1   from tdmclient import ClientAsync, aw
2   from tdmclient.atranspiler import ATranspiler
```

Line#1: Importing ClientAsync and aw from the tdmclient. ClientAsync allow the python library to lock the Thymio robot at runtime. The aw library allows the python runtime to call the Thymio API on async manner

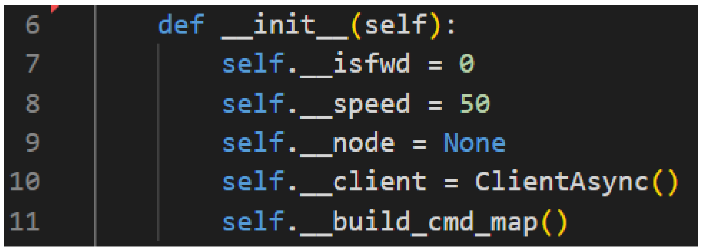
Line#2: importing the ATranspiler class. The transpiler receives a python code as string, compiles it and runs it in the Robot Thymio. A usage of this class to be presented later to play sound in the robot.



```
3   class Thymio:
4       MAX_SPEED = 90
5       MIN_SPEED = 40
```

Line#3: definition of new class called Thymio. This is the wrapper of the tdmclient API.

Line#4-5: definition of two constants, one for maximum speed and the other for minimum speed in order not to exceed those values while user asks to speed up or slow down the robot.



```
6       def __init__(self):
7           self.__isfwd = 0
8           self.__speed = 50
9           self.__node = None
10          self.__client = ClientAsync()
11          self.__build_cmd_map()
```

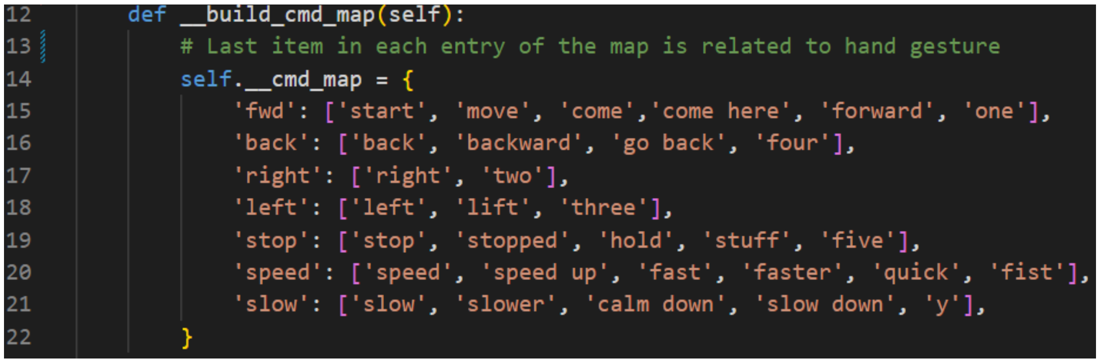
Line#6: definition of the constructor of the Thymio class.

Line#7: setting up the flag __isforward to 0 – flag indicates whether the robot is moving forward, then the value be 1 or the robot is moving backwards, then the value will be -1 or the robot is stopped, and the value will be 0.

Line#8: integer holds the current speed. Default is 50. The value will increase up to MAX_SPEED when user asks to speed up the robot. The value will decrease to MIN_SPEED when the user asks to slow down the robot.

Line#9-10: __node and __client, are objects parameter that holds the connectivity to the Thymio robot.

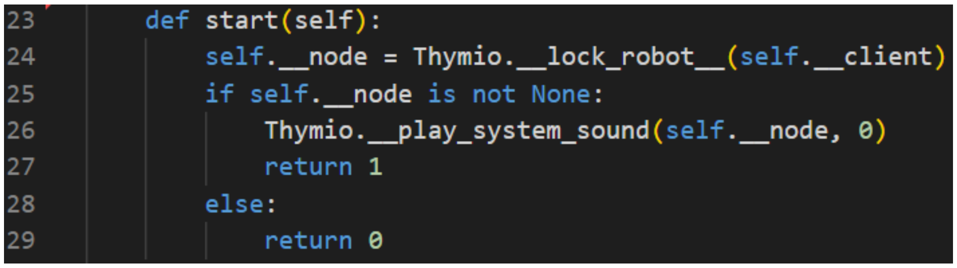
Line#11: __build_cmd_map() is a function call that build the __cmd_map attribute. (more information later)



```
12      def __build_cmd_map(self):
13          # Last item in each entry of the map is related to hand gesture
14          self.__cmd_map = {
15              'fwd': ['start', 'move', 'come','come here', 'forward', 'one'],
16              'back': ['back', 'backward', 'go back', 'four'],
17              'right': ['right', 'two'],
18              'left': ['left', 'lift', 'three'],
19              'stop': ['stop', 'stopped', 'hold', 'stuff', 'five'],
20              'speed': ['speed', 'speed up', 'fast', 'faster', 'quick', 'fist'],
21              'slow': ['slow', 'slower', 'calm down', 'slow down', 'y'],
22          }
```

Line#12: function definition to build the __cmd_map attribute.

Line#14-22: building the command map. Each entry in the map has key and value. The key is the command id and the value is an array of strings. Each value is used from the client(s) to control the robot thru on_command function. For instance, calling on_command("come here") will let the Thymio interface activate the "fwd" command.

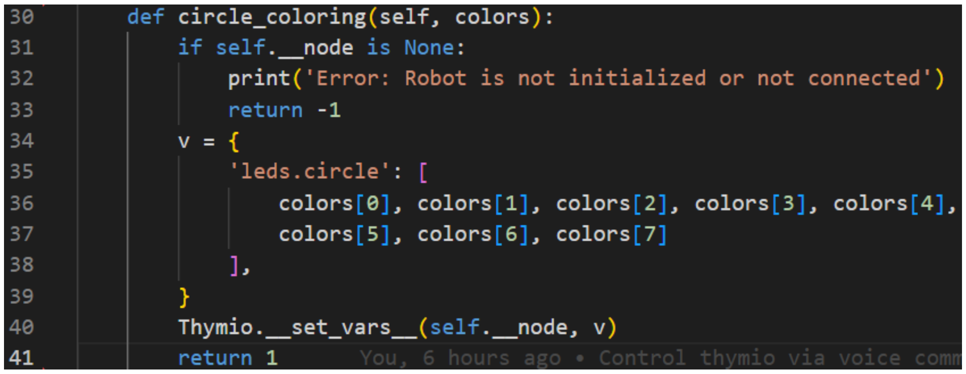


```
23      def start(self):
24          self.__node = Thymio.__lock_robot__(self.__client)
25          if self.__node is not None:
26              Thymio.__play_system_sound(self.__node, 0)
27              return 1
28          else:
29              return 0
```

Line#23: definition of start method. Method is responsible to search for free connected robot and lock it for this run.

Line#24: calls the static method __lock_robot__ that tries to lock a free robot and returns a node object of the Thymio interface.

Line#25-29: Checks if the __lock_robot__ was succeeded. If so, it plays the start sound and returns 1 indicating successful. Otherwise, it returns 0 in the Line#29



```
30      def circle_coloring(self, colors):
31          if self.__node is None:
32              print('Error: Robot is not initialized or not connected')
33              return -1
34          v = {
35              'leds.circle': [
36                  colors[0], colors[1], colors[2], colors[3], colors[4],
37                  colors[5], colors[6], colors[7]
38              ],
39          }
40          Thymio.__set_vars__(self.__node, v)
41          return 1        You, 6 hours ago • Control thymio via voice comm
```

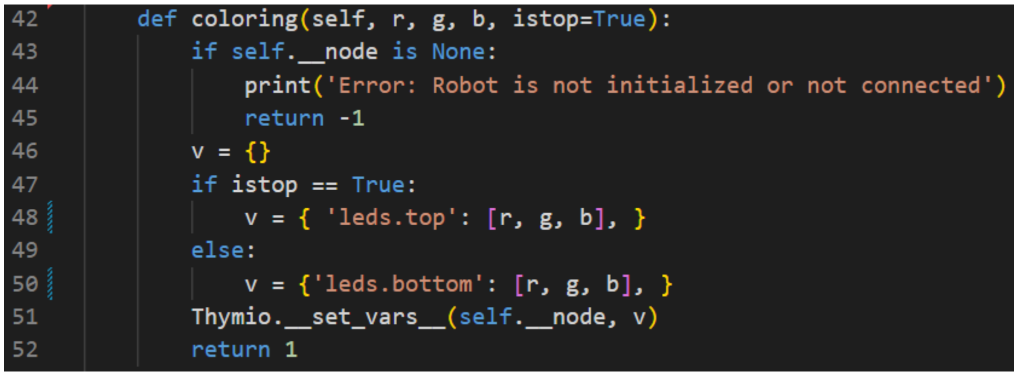
Line#30: definition of new method to control the robot's circle colors. The method accepts an array of integers of size 8. Each entry in the array indicates the color of each arc of the circle color.

Line#31-33: Checks if the __node attribute was initialized correctly, if not, prints an error message and returns -1 indicating failure.

Line#34-39: Preparing a variable called "v" that is a json key and value. The key is: "leds.circle" the name of the attribute that controls the leds of the robot. The value is an array of integers of size 8.

Line#40: Calling internal static method that calls Thymio interface to se the variable "leds.circle" in the connected robot.

Line#41: returns 1 indicating successful result.



```
42      def coloring(self, r, g, b, istop=True):
43          if self.__node is None:
44              print('Error: Robot is not initialized or not connected')
45              return -1
46          v = {}
47          if istop == True:
48              v = { 'leds.top': [r, g, b], }
49          else:
50              v = {'leds.bottom': [r, g, b], }
51          Thymio.__set_vars__(self.__node, v)
52          return 1
```

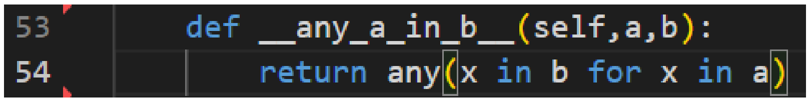
Line#42: definition of new method called: coloring that controls the top/down leds of the robot. First three parameter it receives indicating the color of the leds: red, green, blue values. The last optional parameter: istop indicating whether to color the top leds if false then the method will color the bottom leds. Default is true, i.e., coloring the top leds.

Line#43-45: Checks if the __node attribute was initialized correctly, if not, prints an error message and returns -1 indicating failure.

Line#46: prepare an empty json parameter called: v

Line#47-50: Checks if istop is set to True, then preparing "v" to hold the r,g,b values for the top leds. Otherwise, "v" holds the r,g,b values of the bottom leds.

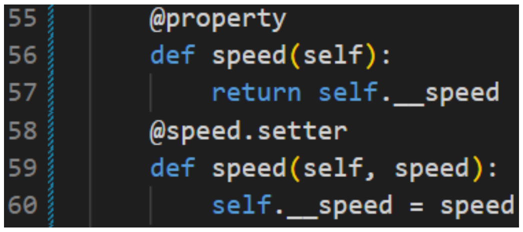
Line#51: calls an internal method to pass the “v” parameter (key and value) to the Thymio robot.



```
53      def __any_a_in_b__(self,a,b):
54          return any(x in b for x in a)
```

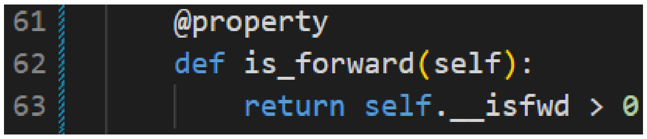
Line#53-54: definition of a utility method that checks if any value of the data structure “a” is presented in the data structure “b”.

For example: calling __any__a_in_b__([1,2,3], [4,5,6,7,1]) returns True since 1 is in both arrays. However, calling __any__a_in_b__([1,2,3,4] , [0,9]) returns False.



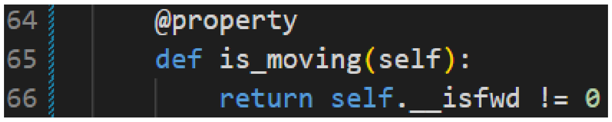
```
55      @property
56      def speed(self):
57          return self.__speed
58      @speed.setter
59      def speed(self, speed):
60          self.__speed = speed
```

The above code defines getter and setter properties for the __speed attribute.



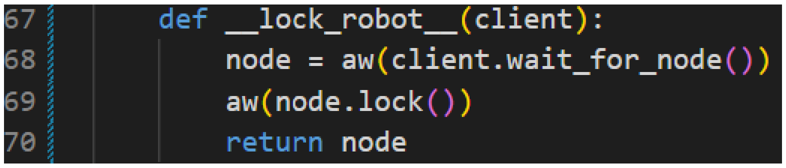
```
61      @property
62      def is_forward(self):
63          return self.__isfwd > 0
```

The above code defines getter property that checks whether the robot is moving forward. i.e., if the __isfwd property is greater than 0 (remember, __isfwd=0 when the robot is not moving, -1 when the robot is moving backwards and 1 when the robot is moving forwards)



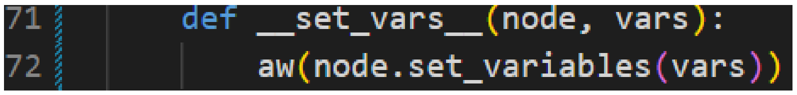
```
64      @property
65      def is_moving(self):
66          return self.__isfwd != 0
```

The above code defines getter property that checks whether the robot is moving (either forwards or backwards)



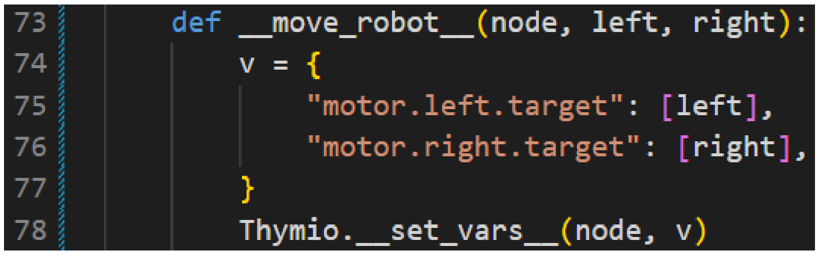
```
67      def __lock_robot__(client):
68          node = aw(client.wait_for_node())
69          aw(node.lock())
70          return node
```

The above code defines internal static method called: __lock_robot__ that waits for a connection to a free robot and then wait and lock it so no interruption will happen in the middle of voice commands.



```
71      def __set_vars__(node, vars):
72          aw(node.set_variables(vars))
```

The above code defines an internal static method that passes the vars variables to the Thymio robot thru node interface in an async manner. “vars” is a json variable with set of keys and values that are predefined in the tdmclient library. Usage of this method in the next piece of code:



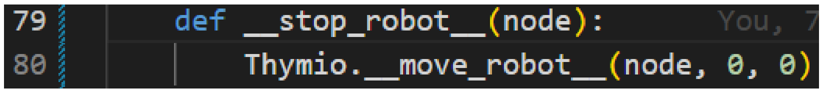
```
73        def __move_robot__(node, left, right):
74            v = {
75                "motor.left.target": [left],
76                "motor.right.target": [right],
77            }
78            Thymio.__set_vars__(node, v)
```

Line#73: The above code defines an internal static method called: __move_robot__ , receives an interface to the Thymio robot thru “node” variable and the values of the left and right motors.

Line#74-77: defines a json variable with the predefined keys to control the motor speed and the given values: left and right variables.

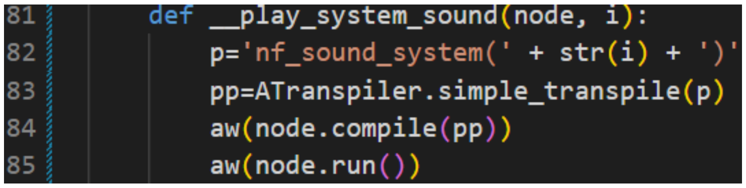
Line#78: calls the internal method to set the variable “v” to the robot runtime.

The above API will be called when the user asks to move the robot forwards or backwards.



```
79        def __stop_robot__(node):          You, 7
80            Thymio.__move_robot__(node, 0, 0)
```

The above code defines an internal static method called: __stop_robot__ which will stop the robot. In other words, it will set the left and right motors values to 0

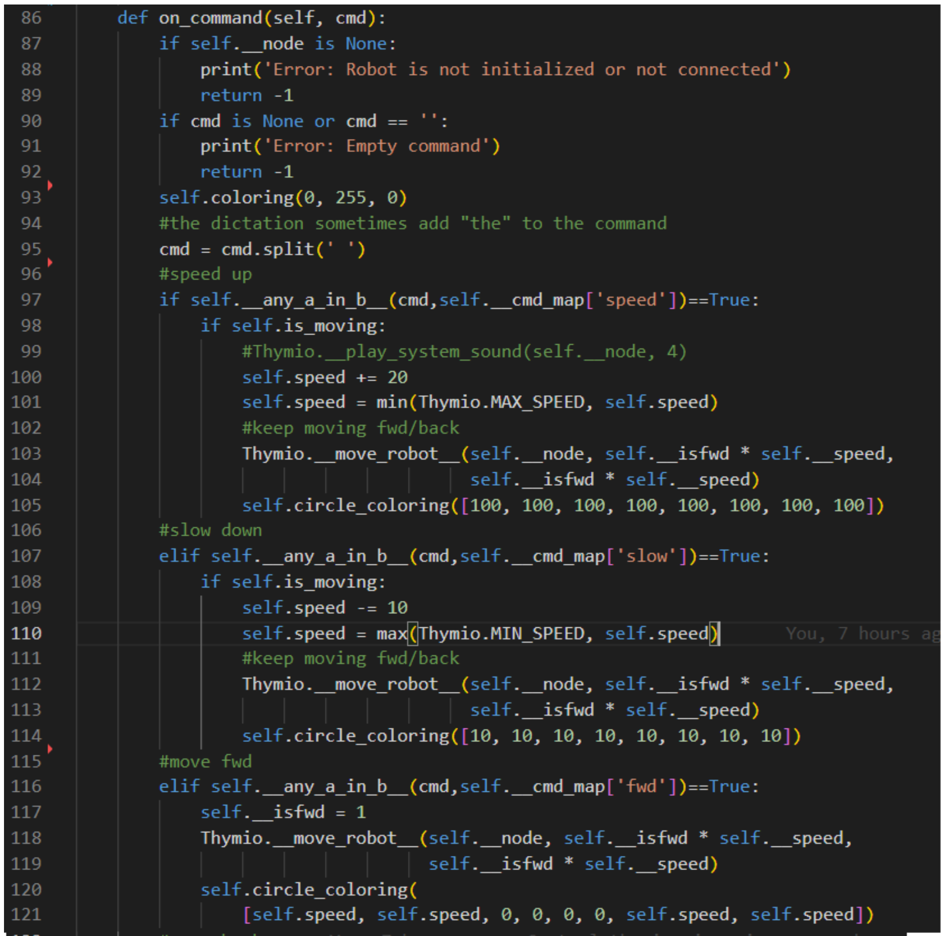


```
81        def __play_system_sound(node, i):
82            p='nf_sound_system(' + str(i) + ')'
83            pp=ATranspiler.simple_transpile(p)
84            aw(node.compile(pp))
85            aw(node.run())
```

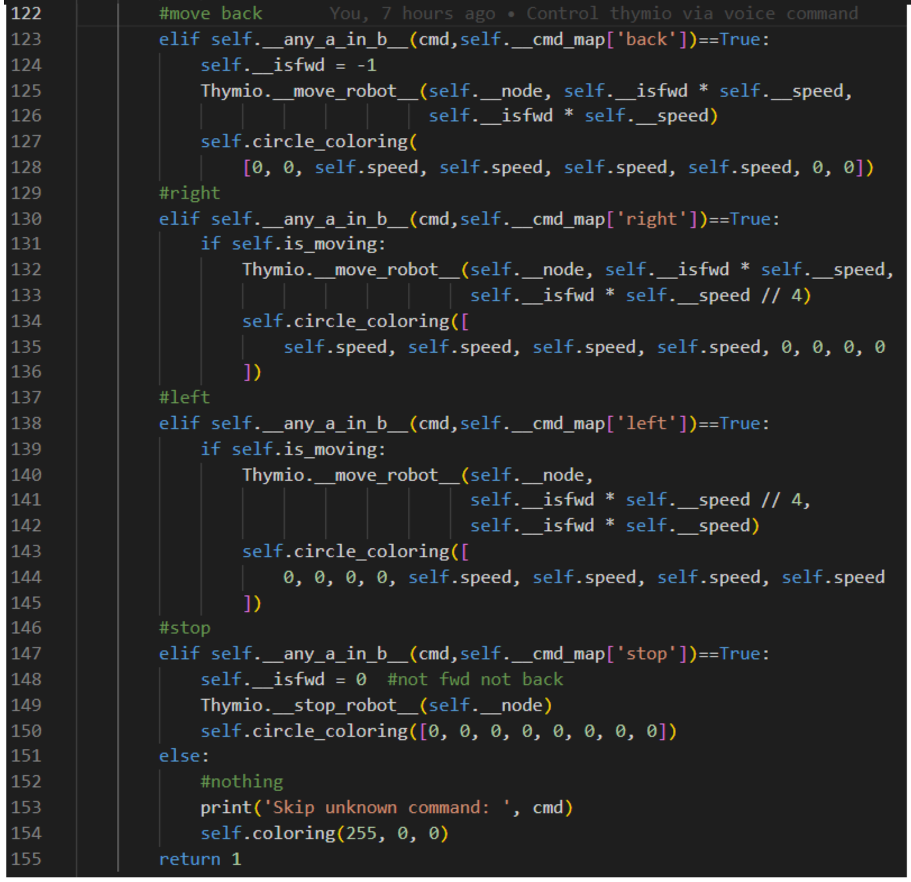
The above code defines an internal static method called: _play_system_sound, which will play a system sound on the robot. The parameter “i” is the index of the predefined system. For more information refer Seror the documentation of the tdmclient at https://pypi.org/project/tdmclient/

The above code prepares a piece of python code and store it in a variable called: p (Line#82). Then it asks to transpile it to Aseba code in Line#83. Then it compiles the Aseba code and run the code in the Thymio robot – Lines#84-85

And now to the main method that is called from the control_thymio script: on_command



```python
    def on_command(self, cmd):
        if self.__node is None:
            print('Error: Robot is not initialized or not connected')
            return -1
        if cmd is None or cmd == '':
            print('Error: Empty command')
            return -1
        self.coloring(0, 255, 0)
        #the dictation sometimes add "the" to the command
        cmd = cmd.split(' ')
        #speed up
        if self.__any_a_in_b__(cmd,self.__cmd_map['speed'])==True:
            if self.is_moving:
                #Thymio.__play_system_sound(self.__node, 4)
                self.speed += 20
                self.speed = min(Thymio.MAX_SPEED, self.speed)
                #keep moving fwd/back
                Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                      self.__isfwd * self.__speed)
                self.circle_coloring([100, 100, 100, 100, 100, 100, 100, 100])
        #slow down
        elif self.__any_a_in_b__(cmd,self.__cmd_map['slow'])==True:
            if self.is_moving:
                self.speed -= 10
                self.speed = max(Thymio.MIN_SPEED, self.speed)      You, 7 hours ag
                #keep moving fwd/back
                Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                      self.__isfwd * self.__speed)
                self.circle_coloring([10, 10, 10, 10, 10, 10, 10, 10])
        #move fwd
        elif self.__any_a_in_b__(cmd,self.__cmd_map['fwd'])==True:
            self.__isfwd = 1
            Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                  self.__isfwd * self.__speed)
            self.circle_coloring(
                [self.speed, self.speed, 0, 0, 0, 0, self.speed, self.speed])
```



```python
        #move back      You, 7 hours ago • Control thymio via voice command
        elif self.__any_a_in_b__(cmd,self.__cmd_map['back'])==True:
            self.__isfwd = -1
            Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                  self.__isfwd * self.__speed)
            self.circle_coloring(
                [0, 0, self.speed, self.speed, self.speed, self.speed, 0, 0])
        #right
        elif self.__any_a_in_b__(cmd,self.__cmd_map['right'])==True:
            if self.is_moving:
                Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                      self.__isfwd * self.__speed // 4)
                self.circle_coloring([
                    self.speed, self.speed, self.speed, self.speed, 0, 0, 0, 0
                ])
        #left
        elif self.__any_a_in_b__(cmd,self.__cmd_map['left'])==True:
            if self.is_moving:
                Thymio.__move_robot__(self.__node,
                                      self.__isfwd * self.__speed // 4,
                                      self.__isfwd * self.__speed)
                self.circle_coloring([
                    0, 0, 0, 0, self.speed, self.speed, self.speed, self.speed
                ])
        #stop
        elif self.__any_a_in_b__(cmd,self.__cmd_map['stop'])==True:
            self.__isfwd = 0  #not fwd not back
            Thymio.__stop_robot__(self.__node)
            self.circle_coloring([0, 0, 0, 0, 0, 0, 0, 0])
        else:
            #nothing
            print('Skip unknown command: ', cmd)
            self.coloring(255, 0, 0)
        return 1
```

The method is called: on_command and receives a string command variable called: cmd

Lines#87-89: Checks whether the robot is initialized correctly, if not, prints an error message and returns -1 indicating failure.

Lines#90-92: Checks if the command is valid (not empty). Otherwise, prints an error message and returns -1 indicating failure.

Line#93: color the top leds of the robot with green.

Line#95: If the client passes multiple words, such as: “move the robot”, then it replace the cmd to an array of strings. i.e., [“move”, “the”, “robot”]

Lines#97-105: First it checks whether the cmd (string or an array of strings) has any word corresponding to the “speed” entry in the commands map: __cmd_map. If so, then it checks if the robot is moving to speed it up – Line#98. Then it increases the speed by 20 – Line#100. Then, it checks if the speed passes the MAX_SPEED if so, then it set it to MAX_SPEED – Lines#100-101. Then, it asks the robot to keep moving with the new speed (Note: __isfwd=1 in case robot was moving forward and -1 in case the robot was moving backwards) – Lines#103-104. And finally, we color all the circle leds with strong light – indicating speed up.

Lines#107-114: in case user asks to slow down the robot. Same as before, this time we reduce the speed by 20 – Line#109, and make sure it is not passing the MIN_SPEED – Line#110. Then, asks the robot to keep moving with the new speed and color all the circle leds with light color – indicating slow down.

Lines#116-121: in case the user asks to move forward or start moving. The system sets the internal variable __isfwd to 1 (indicating moving forward) – Line#117. Then, asks the robot to move with the current speed forward (multiplying the speed by 1, will have positive value, means moving forward) – Lines#118-119. Finally, we color the side leds.

Lines#123-128: in case the user asks to move the robot backwards. The system sets the internal variable __isfwd to -1 (indicating moving backwards) – Line#124. Then, the robot moves with the current speed backwards (multiplying the speed by -1, will have negative value, meaning moving backwards) – Lines#12-126. Finally, we color the middle leds.

Lines#130-136: In case the user asks to turn the robot right. First, checks whether the robot is moving – Line#131, if so, it keeps the left motor’s speed as is and reduce the right motor’s speed by 1/4 – Lines#132-133. And finally, turns the four right leds on and turns off the others.

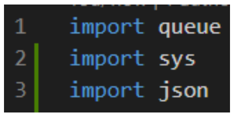
Lines#138-145: In case the user asks to turn the robot left. First, checks whether the robot is moving – Line#139, if so, it keeps the right motor’s speed as is and reduce the left motor’s speed by 1/4 – Lines#140-142. And finally, turns the four left leds on and turns off the others.

Lines#147-150: in case the use asks to stop the robot. First, set the __isfwd variable to 0 – meaning, robot is not moving – Line#148. Then, we call the internal method __stop_robot__ - Line#149 and finally turns off all the circle leds – Line#150

Last lines#151-154, in case unknown command we turn the top leds into red – indicating error.

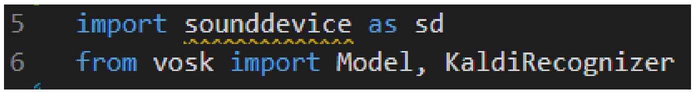
control_thymio.py

This script has the main entry to run the software. It starts the microphone connectivity and listens to audio from the user. Then, by calling to VOSK library, it translates the audio to text. And finally, calls the Thymio interface with the text command.



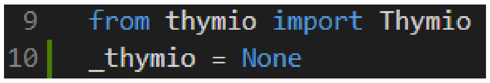
```
1   import queue
2   import sys
3   import json
```

The above code imports some standard python libraries. The first one manages queue data structure. The second one is the system library and the last one wraps json functions such as read, write, etc.



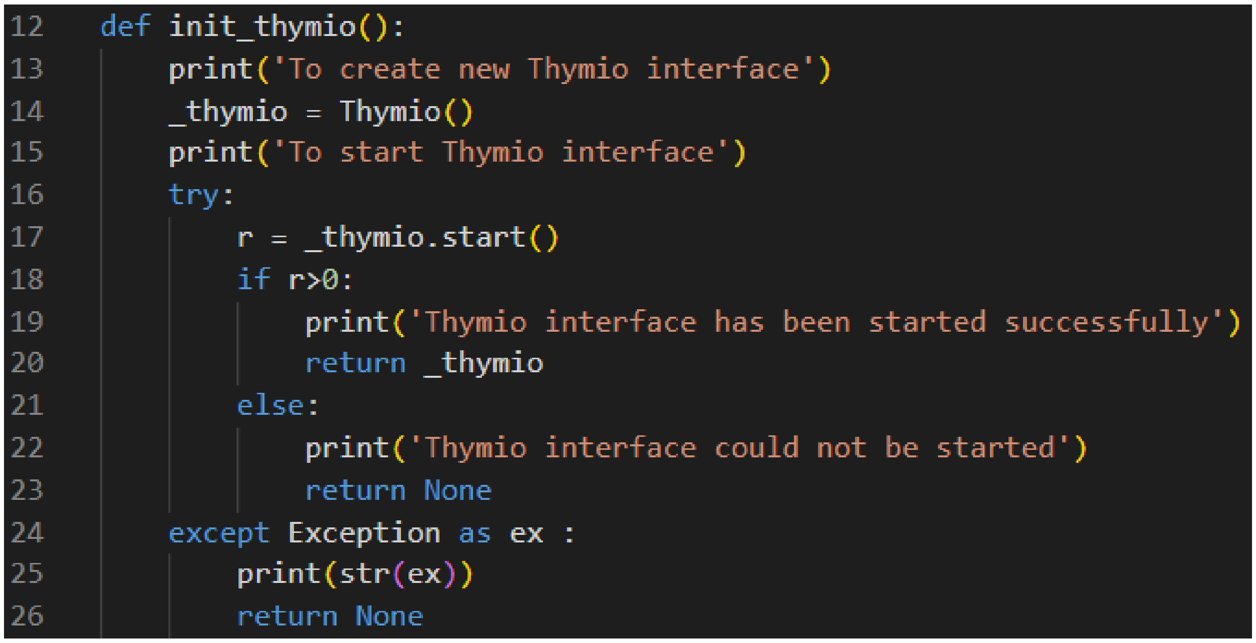
```
5   import sounddevice as sd
6   from vosk import Model, KaldiRecognizer
```

The first line above imports the sound device library for microphone connectivity. The second line loads the Speech to text models from VOSK library.



```
 9   from thymio import Thymio
10   _thymio = None
```

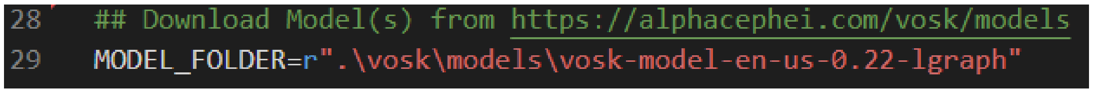
Line#9 imports the thymio class that wraps Thymio interface – previous section. Line#10 initializes _thymio parameter as none.



```
12   def init_thymio():
13       print('To create new Thymio interface')
14       _thymio = Thymio()
15       print('To start Thymio interface')
16       try:
17           r = _thymio.start()
18           if r>0:
19               print('Thymio interface has been started successfully')
20               return _thymio
21           else:
22               print('Thymio interface could not be started')
23               return None
24       except Exception as ex :
25           print(str(ex))
26           return None
```

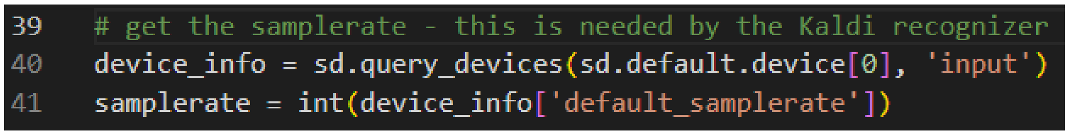
The above method is to initialize a thymio object – pointer to Thymio instance to send commands to.

Line#14 – create the Thymio object. Then, line#17 tries to start the connection to a free robot. If the connection was successful, a positive value is returned – line#18 and the instance to Thymio interface is returned. Otherwise, the function returns None – lines#21-23. In case of error, the function returns None as well – lines#24-26



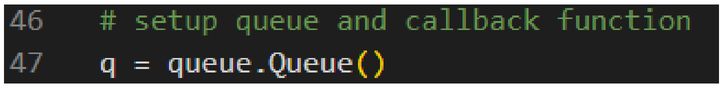
```
28    ## Download Model(s) from https://alphacephei.com/vosk/models
29    MODEL_FOLDER=r".\vosk\models\vosk-model-en-us-0.22-lgraph"
```

Defines new constant MODEL_FOLDER that holds the path to the local VOSK model. In case you want to use other model, you may modify this value.



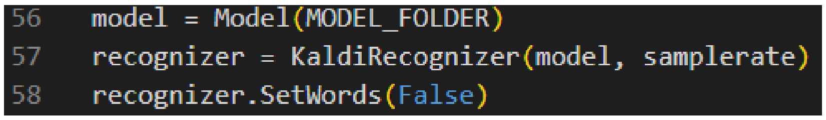
```
39    # get the samplerate - this is needed by the Kaldi recognizer
40    device_info = sd.query_devices(sd.default.device[0], 'input')
41    samplerate = int(device_info['default_samplerate'])
```

The above code gets the sample rate of the device, so we pass it to the speech to text recognizer to be able to convert the audio to text correctly.



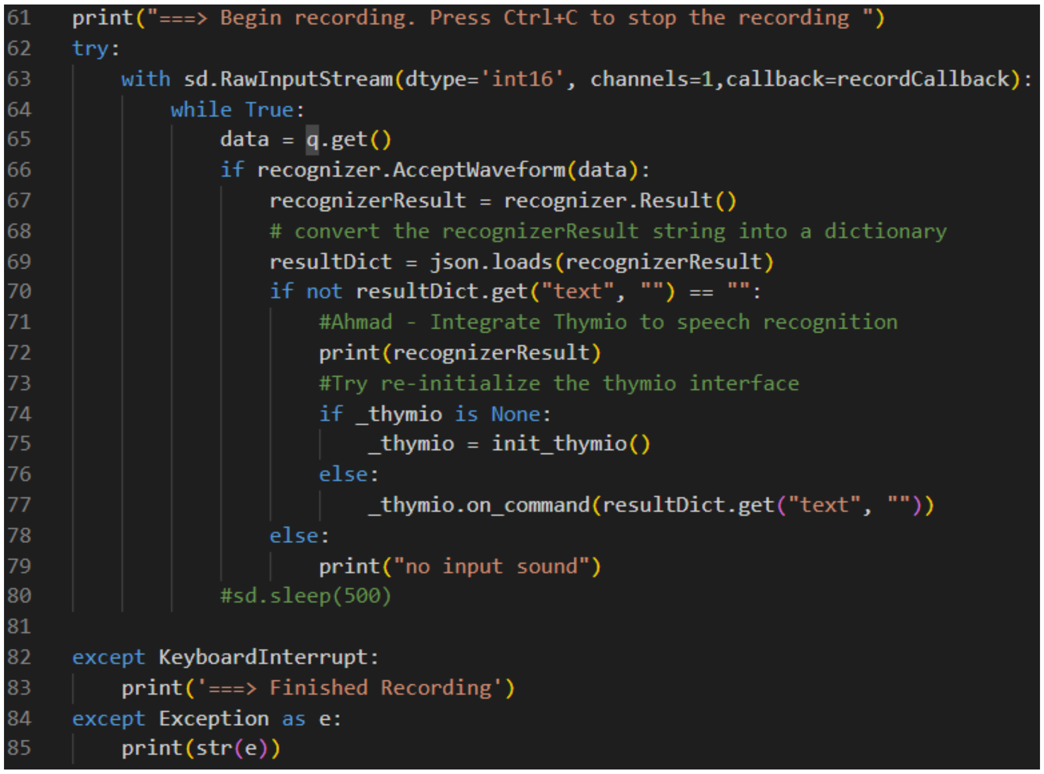
```
46    # setup queue and callback function
47    q = queue.Queue()
```

Initialize an empty queue to hold the audio callback's results. If the software was processing an audio and user keeps talking, the queue parameter will hold what the user has talked so far.



```
56    model = Model(MODEL_FOLDER)
57    recognizer = KaldiRecognizer(model, samplerate)
58    recognizer.SetWords(False)
```

Creates new offline speech to text model – based on the MODEL_FOLDER – line#56. Then initialize the recognizer of the model.



```
61  print("===> Begin recording. Press Ctrl+C to stop the recording ")
62  try:
63      with sd.RawInputStream(dtype='int16', channels=1,callback=recordCallback):
64          while True:
65              data = q.get()
66              if recognizer.AcceptWaveform(data):
67                  recognizerResult = recognizer.Result()
68                  # convert the recognizerResult string into a dictionary
69                  resultDict = json.loads(recognizerResult)
70                  if not resultDict.get("text", "") == "":
71                      #Ahmad - Integrate Thymio to speech recognition
72                      print(recognizerResult)
73                      #Try re-initialize the thymio interface
74                      if _thymio is None:
75                          _thymio = init_thymio()
76                      else:
77                          _thymio.on_command(resultDict.get("text", ""))
78                  else:
79                      print("no input sound")
80              #sd.sleep(500)
81
82  except KeyboardInterrupt:
83      print('===> Finished Recording')
84  except Exception as e:
85      print(str(e))
```

The above code is the infinite loop that keeps running until user press a key to stop the software.

Line#63 – the sound device keeps listening until user says something, then audio data is sent to the callback method: recordCallback which saves the audio data in the queue parameter: “q”

Line#65: checks whether the queue has any audio data, if so – line#66, it translates the audio data into text using the recognizer object – line#67.

Line#69 – converts the VOSK text object into json and asks for the “text” key – line#70. In case the recognizer was able to recognize the audio and translate it to some text, the lines#77 sends the text to the Thymio interface thru the on_command API. In case the Thymio interface is not initialized yet – line#74, we try to initialize it again.

Last lines are handling an exceptions/error.

Conclusion

It is not so hard to build the proof of concept in a school environment. You may try different VOSK models and different languages to control the Thymio robot (note, you need to modify the thymio.py file to react to the new commands). It is also recommended that students can be given an opportunity to experiment with the software themselves and try their own voice control.

The software can be easily extended to more commands, such as, control the colors and play sounds on the robot.

Summary

We have seen how easy it is to setup the environment. It is highly recommended to let the students try it. students may extend the commands they control the Thymio robot.

We recommend integrating such capability to the Thymio framework such as, Scratch.

Reference

Alphacephei. *Frequently Asked Questions.* Alpha Cephei Speech Recognition. https://alphacephei.com/en

Mobsya. *Thymio Home Page.* Thymio. https://www.thymio.org

Python Software Foundation. *tdmclient 0.1.18.* Python Package Index. https://pypi.org/project/tdmclient .