Control Thymio Robot via Voice Commands

Using vosk Speech to Text Library

By Ahmad Agbaryah

Abstract

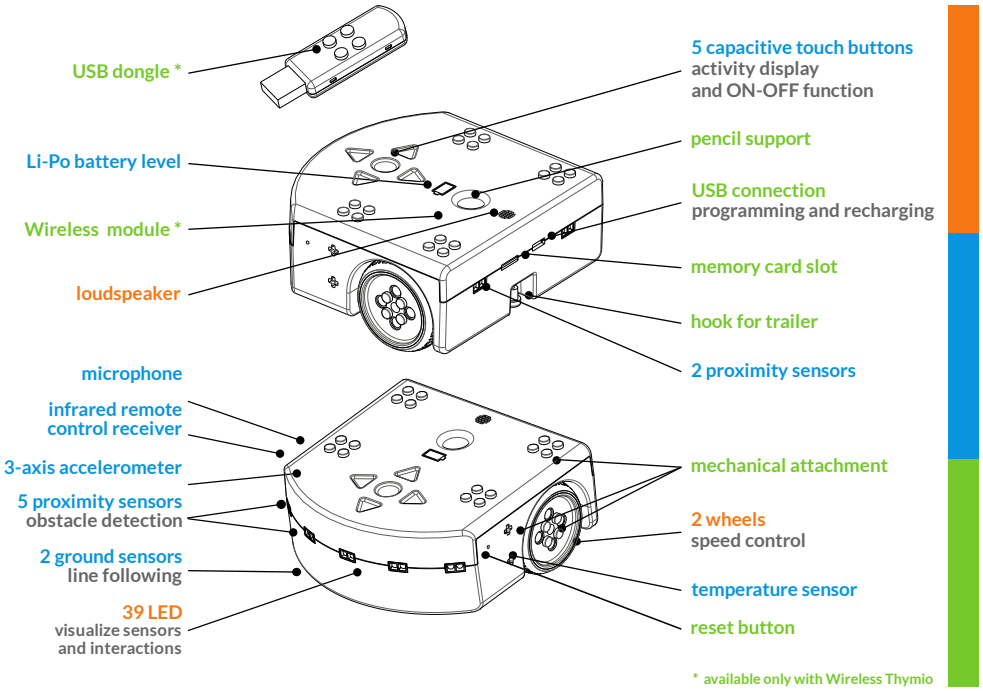
This proof of concept combining a wireless Thymio II with a python software on computer to control the movements of the robot. Speech to text is performed on the computer, using vosk library with an offline model. Once the python application recognizes new voice, it turns it to text and based on the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



[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/tree/main/demo 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.

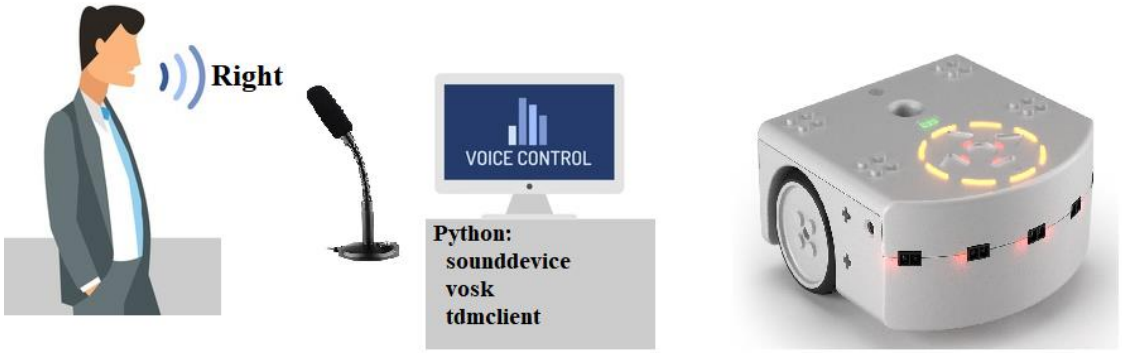
1- Python 3.x (Main development programming language)

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 (of the computer) receives the voice and translates the voice to an audio-data using the “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 in 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).

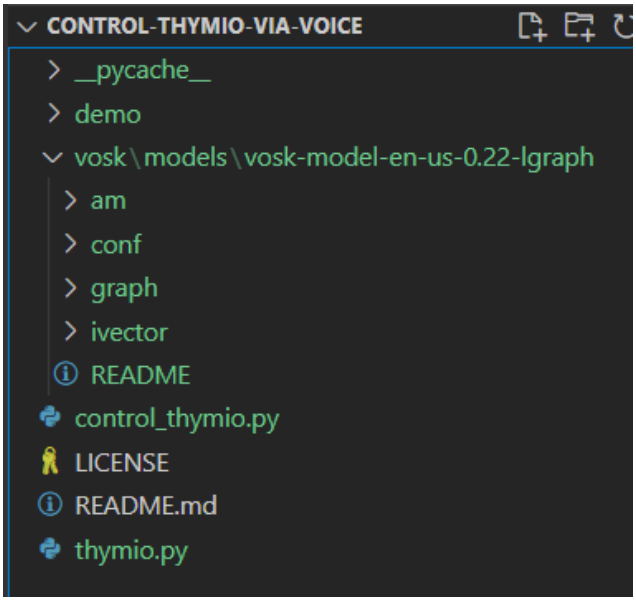
The second one, sounddevice, wraps the audio devices such as microphone and convert audio streamed from the microphone to raw data. Later, this raw data is fed into vosk model to convert it to text.

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
- If you want to use another VSOK model for offline recognition, visit the page: https://alphacephei.com/vosk/models and pick any model you prefer, then download it to your local disk at: *c:\work\thymio\src\control-thymio-via-voice\vosk\models*.
  In our proof of concept, we've used vosk-model-en-us-0.22-lgraph model.

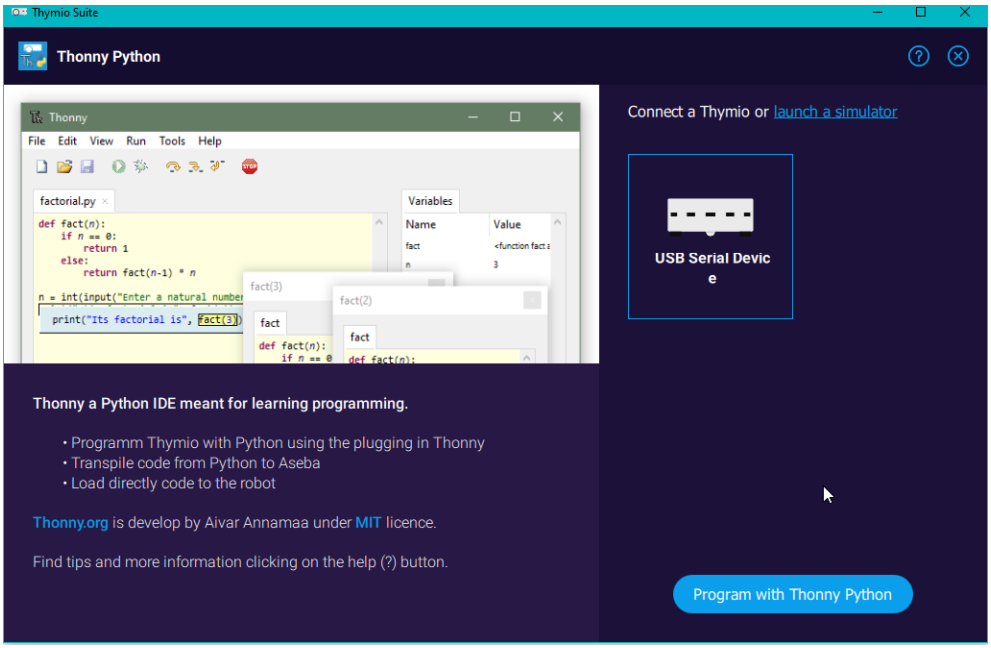
  If you decided to use other model, update the value of *MODEL_FOLDER* constant in *control_thymio.py* file.

- Now you may have the following file structure on your local disk:



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:



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....
===> Begin recording. Press Ctrl+C to stop the recording
{
  "text" : "move"
}
{
  "text" : "speed up"
}
```

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. It initializes a node object of the Thymio robot and lock the node to run commands on it without any interference. This is done by calling the *start()* method.

The main method of the *Thymio.py* is *on_command(cmd)* that receives the string command and tries to understand which command to run in the Thymio robot thru the *node* object. For instance, if the caller activates *on_command("forward"),* then it will ask the robot (again, thru the *node* object) to move forward. And this is done by changing the attribute of *"motor.left.target" and "motor.right.target"* to current speed value. Initial speed value is: 50, and it is increased/decreased by 20 if the user asks to *"speed up" /" slow-down"* respectively. More details are embedded in the code itself – see reference section below.

One piece of code that is worth to explain, since it is not standard call is to play the sound in the Thymio robot:

```
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

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.

This is done by, first, initializing several pieces. First, the Thymio robot and asks to lock a free robot thru: *Thymio.start()* API. Then, initializes the Recognition model of VOSK by creating new *Model* instance and passing it to the *KaldiRecognizer* object. Lastly, the script initializes a connection to the microphone and start listening to it thru: *sounddevice.RawInputStream(dtype='int16', channels=1,callback=recordCallback)* call.

The script then starts an endless loop, getting data from the audio's queue and trying to translate it to text calling: *recognizer.Result*() *API*. If the translation process was successful, it sends the data to the Thymio robot thru *thymio.on_command*. More details are embedded in the code.

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 in the Thymio framework such as, Scratch. Students can play around with the code to integrate other languages.

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 .

Appendix

Thymio.py

```
#TDM client library for Thymio.
from tdmclient import ClientAsync
# aw is to run async calls to the robot.
from tdmclient import aw
#Used to run a python script on the robot.
from tdmclient.atranspiler import ATranspiler
class Thymio:
    """Wrapper functions for the robot commands.
    Utilizes the tdmclient library to call the robot commands."""

    #constants for the Thymio's max and min speed
    MAX_SPEED = 90
    MIN_SPEED = 40
    #Constructor
    def __init__(self):
        #Used to determine if the robot is moving forward (1) or backward (-1) or
not (0)
        self.__isfwd = 0
        #The current speed of the robot
        self.__speed = 50
        #Reference to the node of the robot
        self.__node = None
        #Reference to the client
        self.__client = ClientAsync()
        #Build the commands map
        self.__build_cmd_map()


    def __build_cmd_map(self):
        """Build the commands map, the key is the command name and the value is
the values to be sent to the robot
        Last item in each entry of the map is related to hand gesture"""
        self.__cmd_map = {
            'fwd': ['start', 'move', 'come','come here', 'forward', 'one'],
            'back': ['back', 'backward', 'go back', 'four'],
            'right': ['right', 'two'],
            'left': ['left', 'lift', 'three'],
            'stop': ['stop', 'stopped', 'hold', 'stuff', 'five'],
            'speed': ['speed', 'speed up', 'fast', 'faster', 'quick', 'fist'],
            'slow': ['slow', 'slower', 'calm down', 'slow down', 'y'],
        }
```

```
    def start(self) -> bool:
        """Start the client and connect to the robot.
        Returns:True in case of success, False in case of failure"""
        self.__node = Thymio.__lock_robot__(self.__client)
        if self.__node is not None:
            #Play start sound to indicate successful connection
            Thymio.__play_system_sound(self.__node, 0)
            return True
        else:
            return False


    def circle_leds(self, leds) -> bool:
        """Turn on/off the leds in the Thymio leds-circle.
        leds: A list of integers of length 8.
        Returns:True in case of success, False in case of failure"""

        if self.__node is None:
            print('Error: Robot is not initialized or not connected')
            return -1
        #initialize the leds json with the correct format and values
        v = {
            'leds.circle': [
                leds[0], leds[1], leds[2], leds[3], leds[4],
                leds[5], leds[6], leds[7]
            ],
        }
        Thymio.__set_vars__(self.__node, v)
        return True


    def coloring(self, r, g, b, istop=True)->bool:
        """Set the color of the Thymio's leds.
        r: Red value (0-255)
        g: Green value (0-255)
        b: Blue value (0-255)
        istop: True to color the top leds, False to color the bottom leds.
        Returns:True in case of success, False in case of failure"""

        if self.__node is None:
            print('Error: Robot is not initialized or not connected')
            return -1
        v = {}
        if istop == True:
            v = { 'leds.top': [r, g, b], }
```

```
        else:
            v = {'leds.bottom': [r, g, b], }
        Thymio.__set_vars__(self.__node, v)
        return 1

    def __any_a_in_b__(self,a,b):
        """Check if any element of a is in b"""
        return any(x in b for x in a)

    def is_forward(self) -> bool:
        """Returns:True if the robot is moving forward, False otherwise"""
        return self.__isfwd > 0

    def is_moving(self) -> bool:
        """Returns:True if the robot is moving, False otherwise"""
        return self.__isfwd != 0

    def __lock_robot__(client):
        """Lock the robot and return a reference to the node"""
        node = aw(client.wait_for_node())
        aw(node.lock())
        return node

    def __set_vars__(node, vars):
        """Set the variables in the robot
        vars: A dictionary of variables to set"""
        aw(node.set_variables(vars))

    def __move_robot__(node, left, right):
        """Move the robot
        left: Left speed (MIN_SPEED-MAX_SPEED)
        right: Right speed (MIN_SPEED-MAX_SPEED)"""

        v = {
            "motor.left.target": [left],
            "motor.right.target": [right],
        }
        Thymio.__set_vars__(node, v)


    def __stop_robot__(node):
        """Stop the robot"""
        Thymio.__move_robot__(node, 0, 0)

    def __play_system_sound(node, i):
```

```
        """Play a system sound on the robot.
        i: Sound index (0-5)"""

        #Prepare the python script to play the sound
        p='nf_sound_system(' + str(i) + ')'
        #Transpile the script to Aseba
        pp=ATranspiler.simple_transpile(p)
        #Compile the script
        aw(node.compile(pp))
        #Run the script asynchronously
        aw(node.run())


    def on_command(self, cmd)->bool:
        """Execute a command.
        cmd: The command to execute.
        Returns:True in case of success, False in case of failure"""

        if self.__node is None:
            print('Error: Robot is not initialized or not connected')
            return False
        if cmd is None or cmd == '':
            print('Error: Empty command')
            return False

        #color the top leds with green - indicates that the robot is executing a
command
        self.coloring(0, 255, 0)
        #In case of multiple words in the command, split them
        cmd = cmd.split(' ')
        #speed up
        if self.__any_a_in_b__(cmd,self.__cmd_map['speed'])==True:
            #If robot is already moving, do speed up
            if self.is_moving:
                #speed up the robot by 20
                self.__speed += 20
                #Make sure the speed is not greater than MAX_SPEED
                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_leds([100, 100, 100, 100, 100, 100, 100, 100])
        #slow down
        elif self.__any_a_in_b__(cmd,self.__cmd_map['slow'])==True:
            #If robot is already moving, do slow down
```

```
            if self.is_moving:
                #slow down the robot by 20
                self.__speed -= 20
                #Make sure the speed is not less than MIN_SPEED
                self.__speed = max(Thymio.MIN_SPEED, self.__speed)
                #keep moving fwd/back
                Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                      self.__isfwd * self.__speed)
                self.circle_leds([10, 10, 10, 10, 10, 10, 10, 10])
        #move fwd
        elif self.__any_a_in_b__(cmd,self.__cmd_map['fwd'])==True:
            #Turn the robot forward flag on
            self.__isfwd = 1
            #Move the robot forward
            Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                  self.__isfwd * self.__speed)
            self.circle_leds(
                [self.__speed, self.__speed, 0, 0, 0, 0, self.__speed,
self.__speed])
        #move back
        elif self.__any_a_in_b__(cmd,self.__cmd_map['back'])==True:
            #Turn the robot forward flag off
            self.__isfwd = -1
            #Move the robot backward
            Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                  self.__isfwd * self.__speed)
            self.circle_leds(
                [0, 0, self.__speed, self.__speed, self.__speed, self.__speed, 0,
0])
        #Turn right
        elif self.__any_a_in_b__(cmd,self.__cmd_map['right'])==True:
            #If robot is moving
            if self.is_moving:
                #Turn the robot right -
                # the left motor is set to speed and the right motor is set to
1/4 speed
                Thymio.__move_robot__(self.__node, self.__isfwd * self.__speed,
                                      self.__isfwd * self.__speed // 4)
                self.circle_leds([
                    self.__speed, self.__speed, self.__speed, self.__speed, 0, 0,
0, 0
                ])
        #Turn left
        elif self.__any_a_in_b__(cmd,self.__cmd_map['left'])==True:
            #If robot is moving
```

```
            if self.is_moving:
                #Turn the robot left -
                # the left motor is set to 1/4 speed and the right motor is set
to speed
                Thymio.__move_robot__(self.__node,
                                      self.__isfwd * self.__speed // 4,
                                      self.__isfwd * self.__speed)
                self.circle_leds([
                    0, 0, 0, 0, self.__speed, self.__speed, self.__speed,
self.__speed
                ])
        #stop the robot
        elif self.__any_a_in_b__(cmd,self.__cmd_map['stop'])==True:
            #Set the moving flag to 0
            self.__isfwd = 0
            #Stop the robot
            Thymio.__stop_robot__(self.__node)
            self.circle_leds([0, 0, 0, 0, 0, 0, 0, 0])
        else:
            #nothing
            print('Skip unknown command: ', cmd)
            self.coloring(255, 0, 0)
        return 1

    # def __run_python_f(node, fname,v):
    #     p=fname+'(' + str(v) + ')'
    #     pp=ATranspiler.simple_transpile(p)
    #     aw(node.compile(pp))
    #     aw(node.run())
```

Control_thymio.py

```
#Managing queue of data
import queue
import sys
import json
#library for listening to the microphone
import sounddevice as sd
#Import the offline model and the offline recognizer from VOSK library
from vosk import Model, KaldiRecognizer


from thymio import Thymio
```

```
def init_thymio():
    """Initialize the Thymio connection"""

    print('To create new Thymio interface')
    _thymio = Thymio()
    print('To start Thymio interface')
    try:
        r = _thymio.start()
        if r==True:
            print('Thymio interface has been started successfully')
            return _thymio
        else:
            print('Thymio interface could not be started')
            return None
    except Exception as ex :
        print(str(ex))
        return None
#Initialize the Thymio interface
_thymio = init_thymio()


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

'''This script processes audio input from the microphone and displays the
transcribed text.'''

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

# display the default input device
print("===> Initial Default Device Number:{} Description:
{}".format(sd.default.device[0], device_info))

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

def recordCallback(indata, frames, time, status):
    """This is called (from a separate thread) for each audio block.
    Saves the audio block to a queue."""

    if status:
        print(status, file=sys.stderr)
    q.put(bytes(indata))
```

```
# build the model and recognizer objects.
print("===> Build the model and recognizer objects.  This will take a few
minutes.")
model = Model(MODEL_FOLDER)
recognizer = KaldiRecognizer(model, samplerate)
recognizer.SetWords(False)

print("===> Begin recording. Press Ctrl+C to stop the recording ")
try:
    # start the recording thread, and listens to the microphone
    with sd.RawInputStream(dtype='int16', channels=1,callback=recordCallback):
        while True:
            # get the next audio block from the queue
            data = q.get()
            # recognize the speech in the audio block
            if recognizer.AcceptWaveform(data):
                # returns a list of possible transcripts
                recognizerResult = recognizer.Result()
                # convert the recognizerResult string into a dictionary
                resultDict = json.loads(recognizerResult)
                if not resultDict.get("text", "") == "":
                    #Integrate Thymio to speech recognition
                    print(recognizerResult)
                    #Try re-initialize the thymio interface if it fails
                    if _thymio is None:
                        _thymio = init_thymio()
                    #If the thymio interface is initialized, send the command to
the thymio
                    else:
                        _thymio.on_command(resultDict.get("text", ""))
                else:
                    print("no input sound")
# if Ctrl+C is pressed, stop the recording thread and finish
except KeyboardInterrupt:
    print('===> Finished Recording')
except Exception as e:
    print(str(e))
```