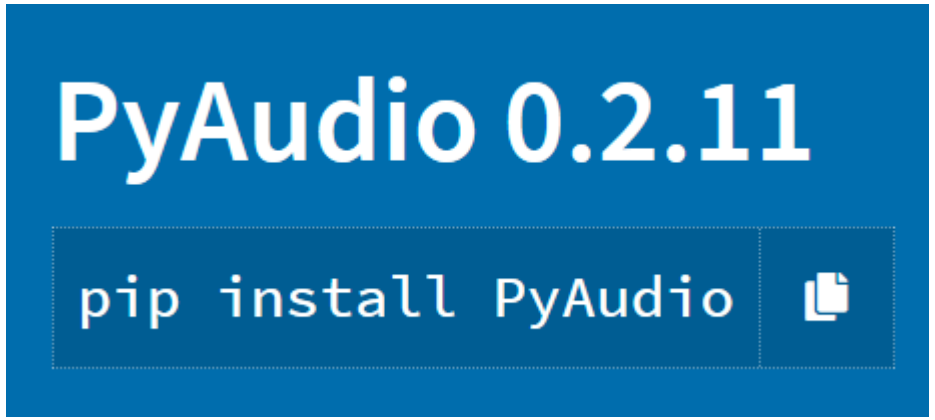
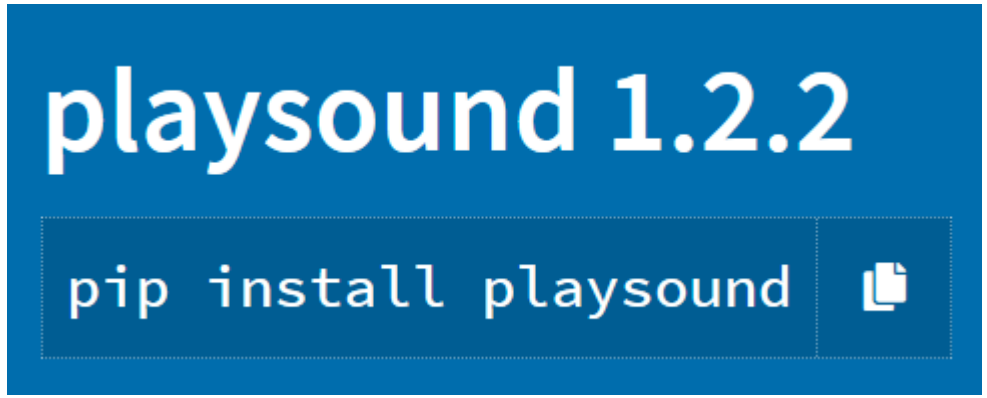
KASETSART
UNIVERSITY

We need to Speak

Speak



PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS.



The playsound module contains only one thing - the function (also named) playsound.
It requires one argument - the path to the file with the sound youd like to play. This may be a local file, or a URL.

Speak

Library	Platform	Playback	Record	Convert	Dependencies
`playsound`	Cross-platform	WAV, MP3	-	-	None

```
$sudo apt-get install mpg321 libmp3lame0 libmp3splt0-mp3
$pip install playsound
```

`winsound`	Windows	WAV	-	-	None
`sounddevice`	Cross-platform	NumPy array	NumPy array	-	`numpy`, `soundfile`

```
$sudo apt-get install ffmpeg libav-tools
$sudo apt-get install libasound-dev portaudio19-dev libportaudio2 libportaudiocpp0
$pip install pyAudio
```

`pyaudio`	Cross-platform	`bytes`	`bytes`	-	`wave`
`wavio`	Cross-platform	-	-	WAV, NumPy array	`numpy`, `wave`

Speak

Ispeak.py

```
from playsound import playsound
import pyaudio
import wave
print(' Speaking...')
playsound('./nictomeet.mp3')
playsound('./angus.mp3')
print(' Start to ask for speak')
playsound("./speak_after.mp3")
playsound("./signal.mp3")
```

Speak

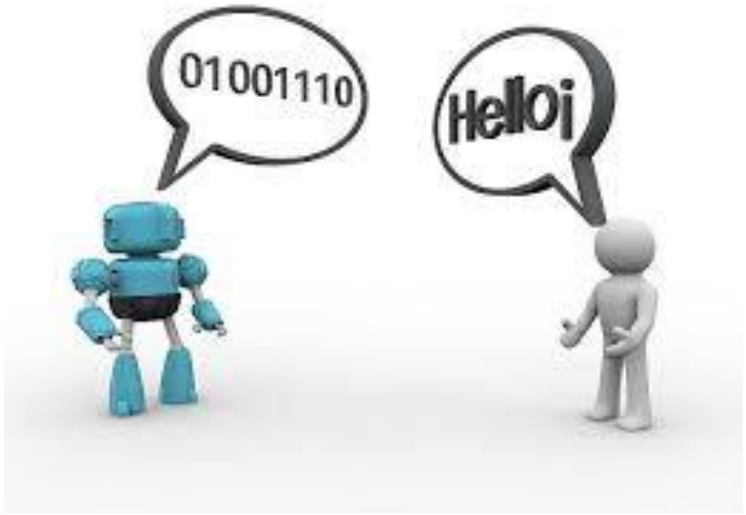
```
chunk = 1024        # Record in chunks of 1024 samples
sample_format = pyaudio.paInt16  # 16 bits per sample
channels = 2
fs = 44100          # Record at 44100 samples per second
seconds = 3
filename = "sound_out.wav“
print('Recording ……')

p = pyaudio.PyAudio()  # Create an interface to PortAudio
stream = p.open(format=sample_format, channels=channels, rate=fs,
frames_per_buffer=chunk, input=True)

frames = []  # Initialize array to store frames
```

Speak

```
for i in range(0, int(fs / chunk * seconds)):    # Store data in chunks for 3 secs
        data = stream.read(chunk)
        frames.append(data)
# Stop and close the stream
stream.stop_stream()
stream.close()
# Terminate the PortAudio interface
p.terminate()
print('Finished recording ....')       # Save the recorded data as a WAV
filewf = wave.open(filename,'wb')
wf.setnchannels(channels)
wf.setsampwidth(p.get_sample_size(sample_format))
wf.setframerate(fs)
wf.writeframes(b''.join(frames))
wf.close()
```

Speak

```
playsound("./signal.mp3")
playsound('./ihear.mp3')
playsound('./sound_out.wav')
playsound('./UV_Room.mp3')
playsound('./thanks.mp3')
```



Real Python

$python Ispeak.py

GOOGLE Speech to Text API

pypi v3.8.1 | status stable | python 2.7 | 3.3 | 3.4 | 3.5 | 3.6 | license BSD | build failing

Library for performing speech recognition, with support for several engines and APIs, online and offline.

Speech recognition engine/API support:

- CMU Sphinx (works offline)
- Google Speech Recognition
- Google Cloud Speech API
- Wit.ai
- Microsoft Bing Voice Recognition
- Houndify API
- IBM Speech to Text
- Snowboy Hotword Detection (works offline)

pip install SpeechRecognition

Quickstart: `pip install SpeechRecognition`. See the “Installing” section for more details.

To quickly try it out, run `python -m speech_recognition` after installing.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import speech_recognition as sr
from playsound import playsound

import sys
reload(sys)
sys.setdefaultencoding("utf-8")

# Record Audio
r = sr.Recognizer()
m = sr.Microphone()

#set threhold level
print(' Quiet Please ')
with m as source: r.adjust_for_ambient_noise(source)
print("Set minimum energy threshold to {}".format(r.energy_threshold))
```

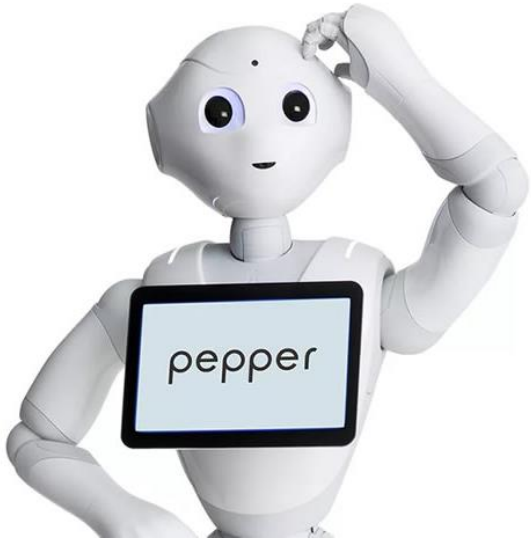
$ python talktome.py

```python
# Speech recognition using Google Speech Recognition
while (True):
    playsound("./speak_after.mp3")
    playsound("./signal.mp3")
    print("Say something!")
    with sr.Microphone() as source:
        audio = r.listen(source, phrase_time_limit=3)
        playsound("./signal.mp3")
    try:
        # for testing purposes, we're just using the default API key
        # to use another API key, use  key="GOOGLE_SPEECH_RECOGNITION_API_KEY"
        text_recognized = r.recognize_google(audio,  language='th-TH')
        print("Google Speech Recognition >> you said >>" + text_recognized)
        text_recognized.strip()
        if ( "สวัสดี" in text_recognized ):
            playsound("./nictomeet.mp3")
        elif ( 'ชื่อ' in text_recognized ):
            playsound("./angus.mp3")
        elif ( 'ข่าว' in text_recognized ):
            playsound("./covid.mp3")
        elif ( 'เพลง' in text_recognized ):
            playsound("./unknown.mp3")
        elif ( 'จบการทำงาน' in text_recognized ):
            break
        else:
            playsound("./dontknow.mp3")
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

ROS with Google Speech to Text API

$ python talktoROS.py → Std_msg String → $ python listenROS.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
reload(sys)
sys.setdefaultencoding("utf-8")

import speech_recognition as sr
from playsound import playsound
import rospy
from std_msgs.msg import String

# Record Audio
r = sr.Recognizer()
m = sr.Microphone()

pub = rospy.Publisher('TalktoROS', String, queue_size=10)
rospy.init_node('SpeechTalker', anonymous=True)
rate = rospy.Rate(1) # 1hz

#set threhold level
print(' Quiet Please ')
with m as source: r.adjust_for_ambient_noise(source)
print("Set minimum energy threshold to {}".format(r.energy_threshold))
```



ROS with Google Speech to Text API

$ python talktoROS.py → Std_msg String → $ python listenROS.py

```
while (True):
    playsound("./location.mp3")
    playsound("./signal.mp3")
    print("Say something!")
    with sr.Microphone() as source:
        audio = r.listen(source, phrase_time_limit=3)
        playsound("./signal.mp3")
    try:
        text_recognized = r.recognize_google(audio,  language='th-TH')
        print("Google Speech Recognition >> you said >>  " + text_recognized)
        if ( 'สวัสดี' in text_recognized ):
            playsound("./nictomeet.mp3")
        elif ( 'ชื่อ' in text_recognized ):
            playsound("./angus.mp3")
        elif ( 'ข่าว' in text_recognized ):
            playsound("./covid.mp3")
        elif ( 'ห้องรับแขก' in text_recognized ):
            playsound("./ok.mp3")
            playsound("./rooma.mp3")
            hello_str = 'ROOM A'
            break
```

ROS with Google Speech to Text API

$ python talktoROS.py → Std_msg / String → $ python listenROS.py


```
        elif ( 'ห้องครัว' in text_recognized ):
            playsound("./ok.mp3")
            playsound("./roomb.mp3")
            hello_str = 'ROOM B'
            break
        elif ( 'ห้องนอน' in text_recognized ):
            playsound("./ok.mp3")
            playsound("./roomc.mp3")
            hello_str = 'ROOM C'
            break
        else:
            playsound("./dontknow.mp3")
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition service; {0}".format(e))

print('Publish Location !')

while not rospy.is_shutdown():
    rospy.loginfo(hello_str)
    pub.publish(hello_str)
    rate.sleep()
```

ROS with Google Speech to Text API

$ python talktoROS.py → Std_msg String → $ python listenROS.py

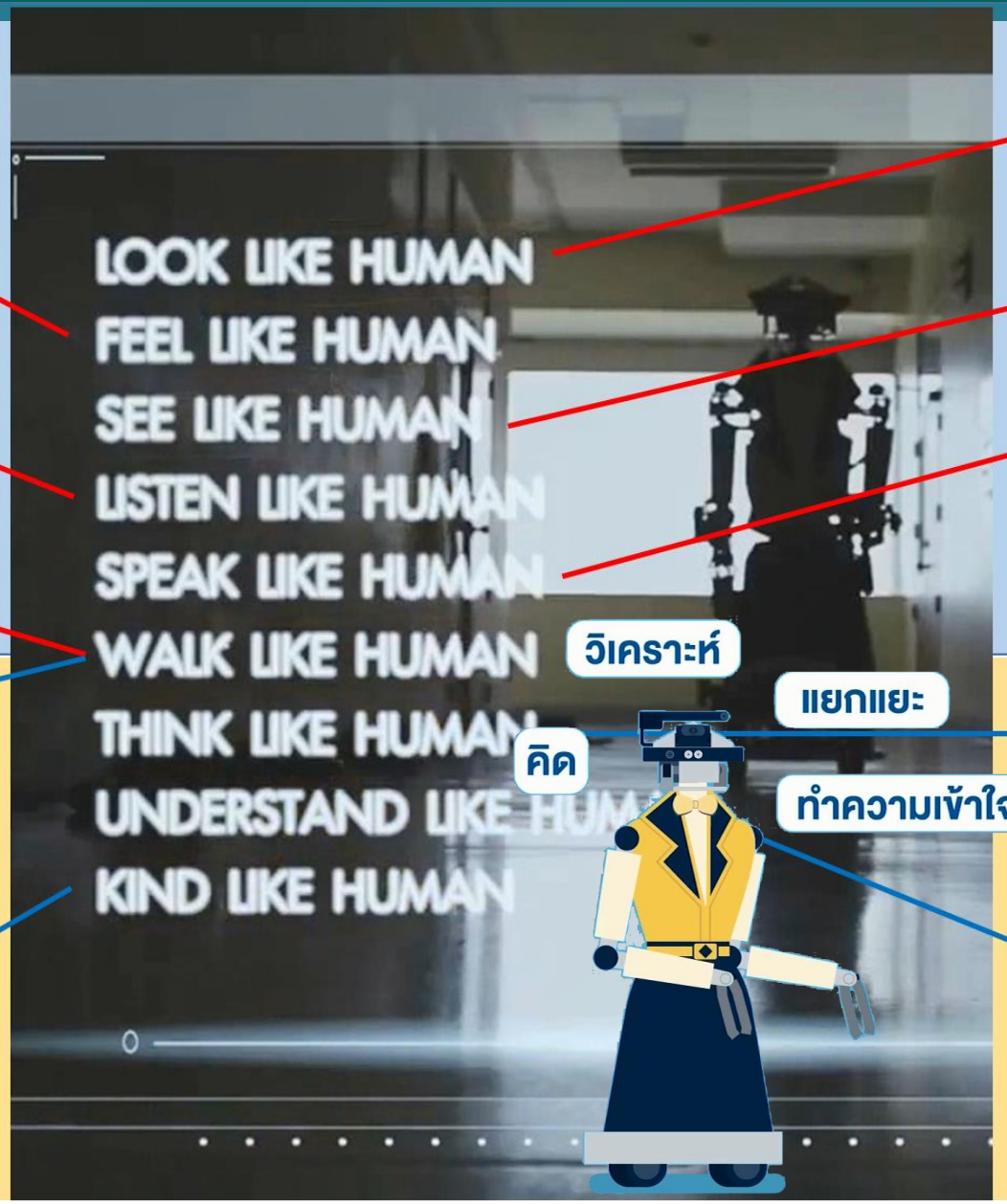
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
reload(sys)
sys.setdefaultencoding("utf-8")

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_caller_id() + " I will go to %s", data.data)

def listener():

    rospy.init_node('Speechlistener', anonymous=True)
    rospy.Subscriber("TalktoROS", String, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()

if __name__ == '__main__':
    listener()
```

@Home Robot

Force sensor

Microphone (array)

Wheels or Legs



Hardware/Arms

2D & 3D cameras

Speech to Text
Text to Voice

Sensor/Hardware

Navigation / Path Planning
Object avoidance
Sound direction

Human-Robot Interface
Emotion display
Chat Bot !

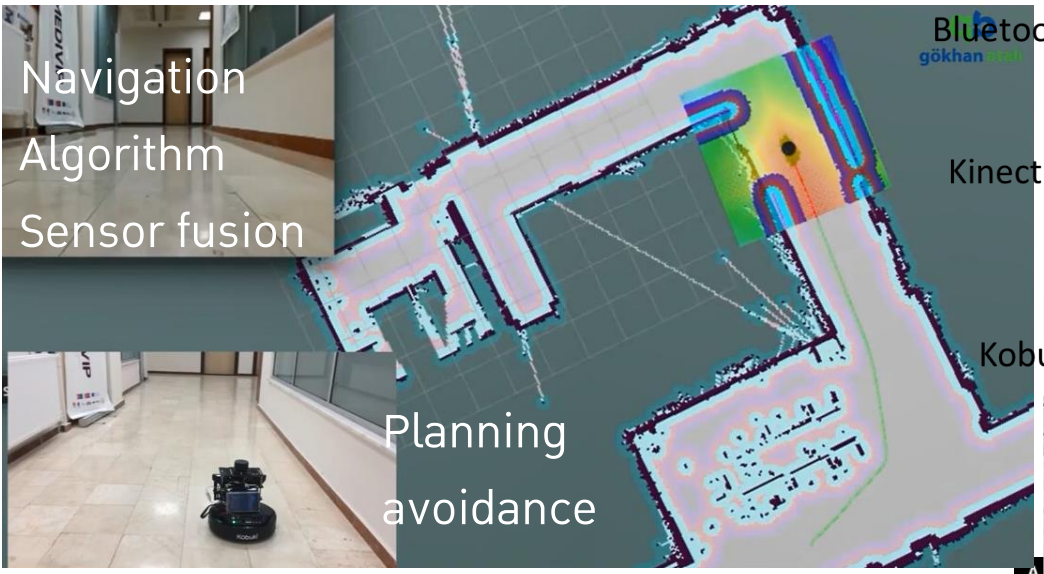
AI / NLP / Finite SM
Algorithms

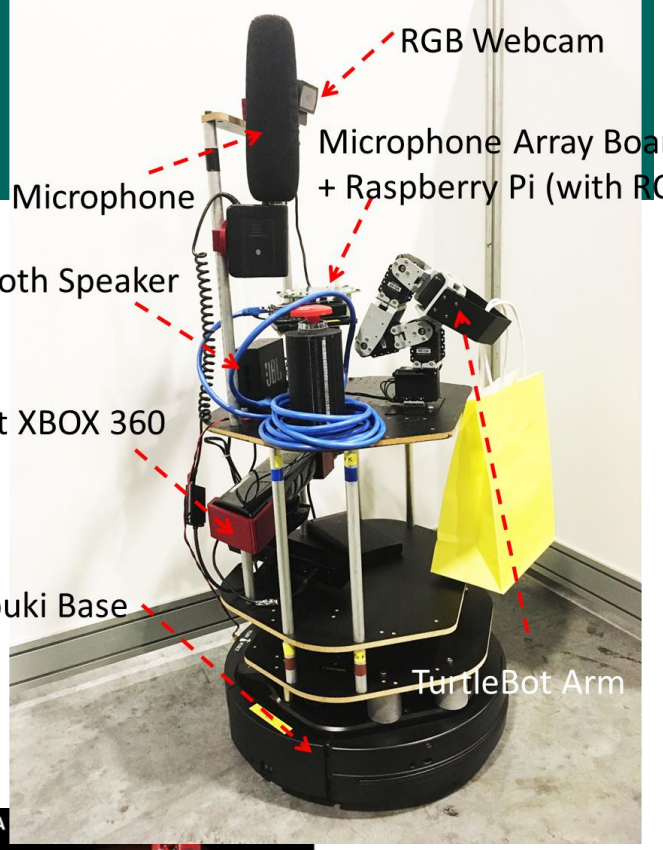
Recognition system
Machine Learning
Machine Vision

Software

@Home Robot

KASETSART UNIVERSITY





WiFi adapter
Standards: IEEE 802.11b/g/n

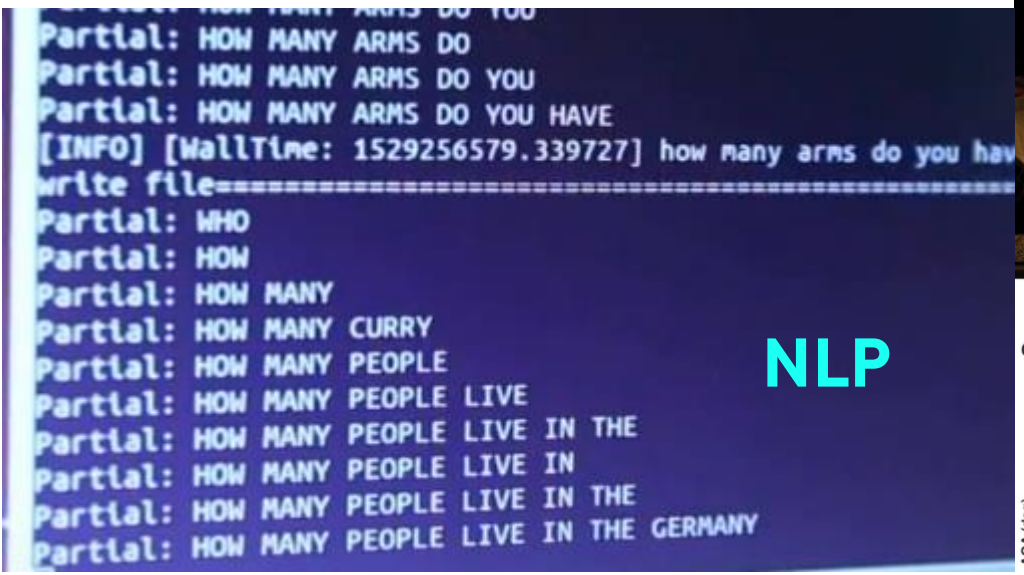
Wide-angle camera
Logitech C905

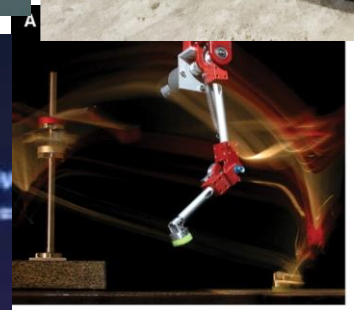
Speaker

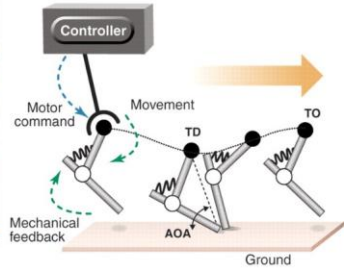
ZBOX nano XS
AMD E-450 processor 1.65 GHz
2 GB RAM, 64 GB SSD
USB 3.0, HDMI
Gigabit Ethernet
Memory card slot

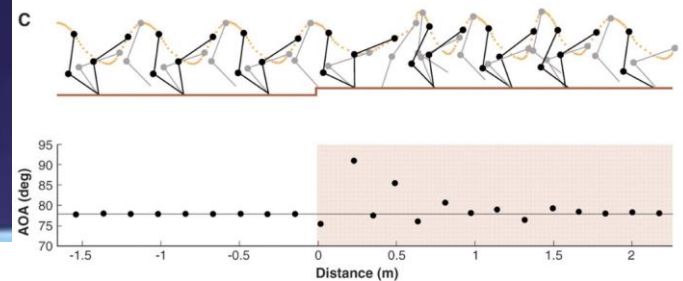
CM730 control board
Servo communication
3-axis accelerometer
3-axis gyroscope

LiPo battery
14.8 V, 3.6 Ah









20 actuators
Networked Dynamixel
MX actuators
6 per leg (MX-106)
3 per arm (MX-64)
2 in the neck (MX-64)

Lightweight materials
Carbon composite
Aluminum
ABS+

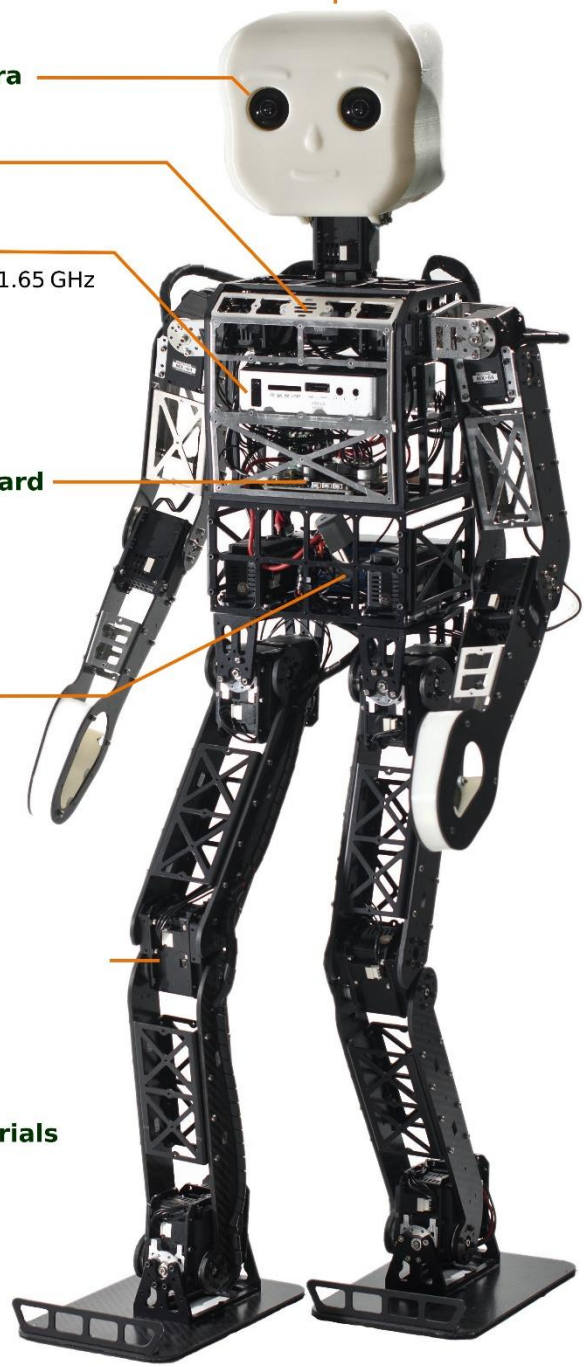


Photo: Felix Oprean

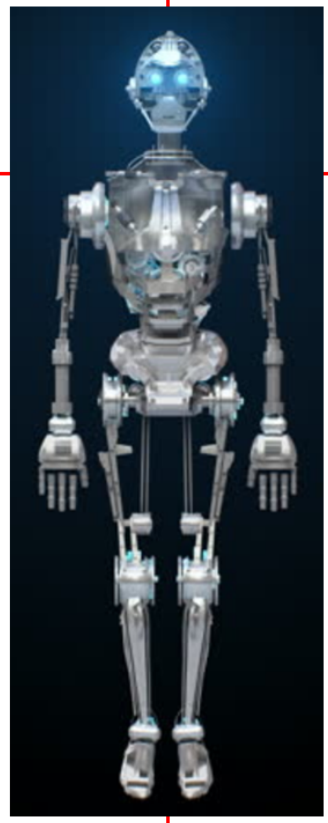
@Home Robot

Hardware

1. Mechanic Design
2. Embedded & Electronic Design
3. Mathematic Modelling
4. Low-Level Firmware
5. Low-level Control algorithm
6. Edge Processing Enhancement

System Integration (SI)

1. Combine Existing Technology
2. Create Application
3. Robustness & Optimization
4. Training
5. Education
6. Commercialization



software

1. Navigation & Avoidance & Path Planning
2. Robot Arm Motion for Specific Trajectory
3. High-Level Control algorithm
4. Machine Vision for 2D and 3D
5. Natural Language Processing
6. Recognition System
7. Decision System
8. Human-Robot Interfaces

THANK YOU