THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

DEPARTMENT OF
COMPUTER SCIENCE
Au Reikura

2025

Contributors

J. Turner

J. Kasmara

MEET BUDDY 4
Raspberry Pi 6
Buddy the humanoid robot 6
Computer Vision 9
Pixels 10
Programming Buddy to detect objects 12
Summary 16
Useful Resources 17

MEET BUDDY

Welcome to the newest addition to the CSNeT programme, Artificial Intelligence Series. In today's session, we introduce you to our humanoid robot, Buddy.

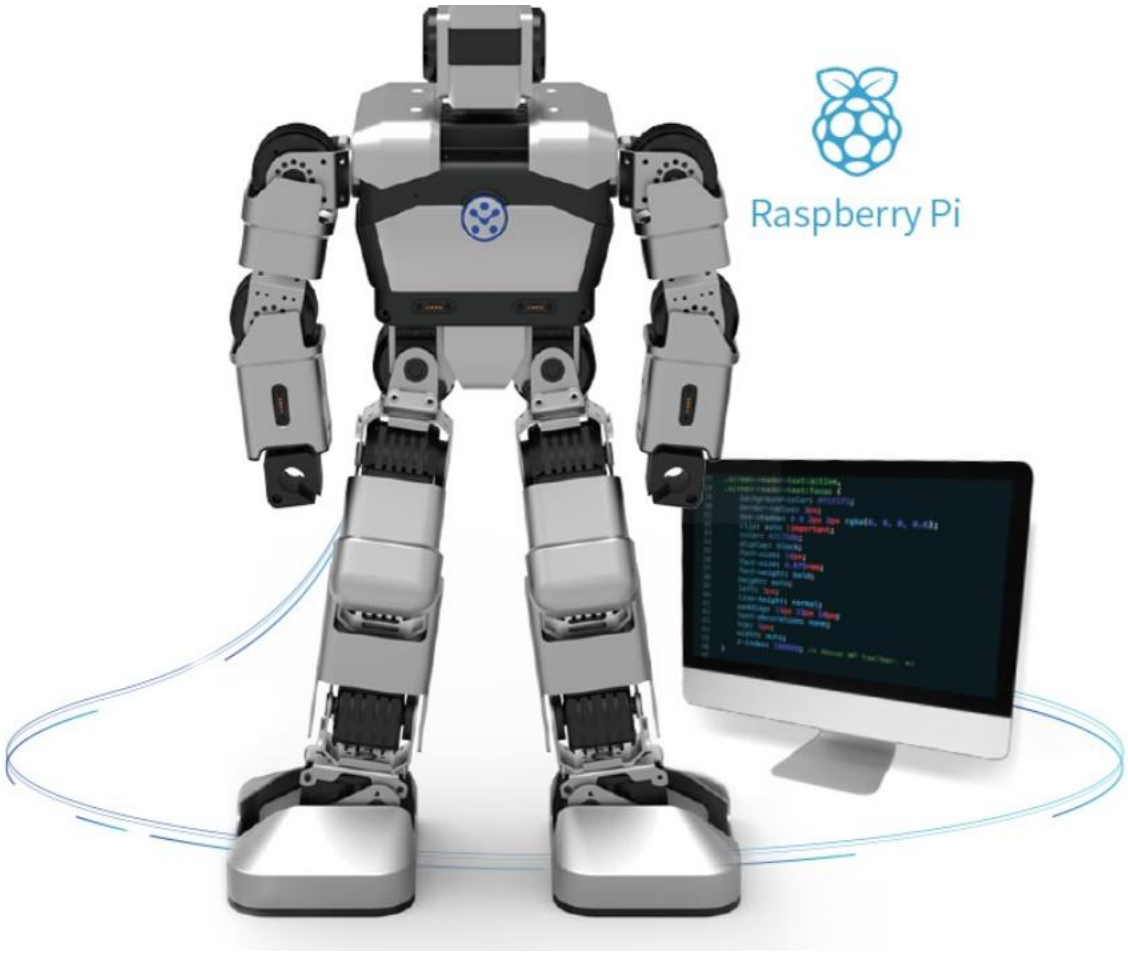


Figure 1: Our humanoid robot, Buddy

Buddy has a computer brain in a device called a Raspberry Pi. It is similar to your motherboard. The brain of a computer is the CPU (Central Processing Unit) chip. Buddy runs on a 1.2GHz 64-bit quad-core ARM Cortex-A53, whereas a typical laptop might use a 1.6GHz 64-bit dual-core Intel Core i5 or AMD Ryzen 5. Gaming computers often require at least double that speed, typically using an Intel Core i7 or AMD Ryzen 7. The ARM Cortex-A53 is chosen for Buddy's brain because it is well-suited for processing AI data.

Another feature of Buddy's brain is its memory. Buddy's memory is contained in a component installed on the motherboard and Raspberry Pi called RAM (Random Access Memory). Buddy has 1GB of RAM which means can it store up to 1GB of data temporarily at any given time, but this data gets wiped out as soon as quitting the program on Buddy. Additionally, Buddy has 16GB of built-in memory to store important data for long-term keeping. In comparison, a typical laptop may contain up to 8GB of RAM, while a gaming desktop may have between 16GB and 32GB of RAM.

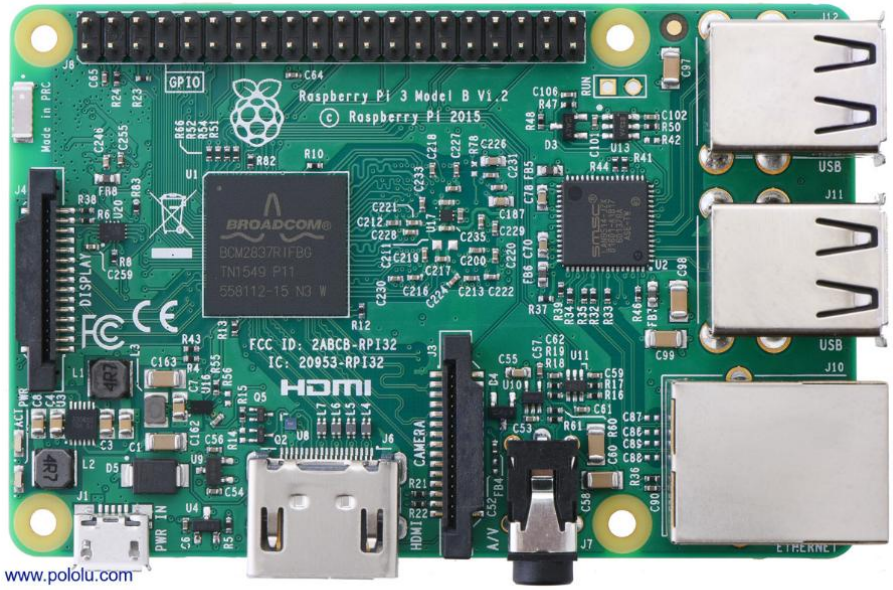
Before we load programs into him, we need a HDMI-HDMI cable and a monitor with an HDMI port.

Figure 2: HDMI-HDMI cable

Figure 3: HDMI-monitor

Raspberry Pi

Raspberry Pi, a single-board computer that runs on a Linux operating system. It is equipped with multi-core processors, providing sufficient processing power to handle complex tasks such as artificial intelligence and computer vision computations, multitasking, and running program applications. The Raspberry Pi supports a diverse range of I/O options, including HDMI, USB, Ethernet, and GPIO pins, making it ideal for projects requiring advanced computing capabilities, such as robotics and Internet of Things (IoT) applications that demand sophisticated software. Detailed specifications of the Raspberry Pi board, as shown in Figure 2, can be accessed from the manufacturer's catalog on the Pololu website.



Raspberry Pi 3 Model B, top view.

Figure 4: Raspberry Pi 3 Model B, top view.

Now, let's set up Buddy! Pause the video to connect the HDMI cable to Buddy's HDMI port and the other end to a monitor. Remember to connect the USB toggle of a mouse and the USB toggle of a keyboard to Buddy's USB ports. Feel free to ask staff for assistance. Now we are ready to program Buddy.

Buddy the humanoid robot

For the next 3 sessions of CSNeT Artificial Intelligence Series, you will get the opportunity to program Buddy, our humanoid robot (see Figure 1). However, before we begin the exercises, let's take a closer look at the robot.

Buddy is made by the robotics company UBTECH Robotics, which produce educational robots designed to promote learning in areas such as robotics, coding, and artificial intelligence.

Additionally, they make the UBTECH Education app, which provides a platform for coding and programming the robots (see Useful Resources).

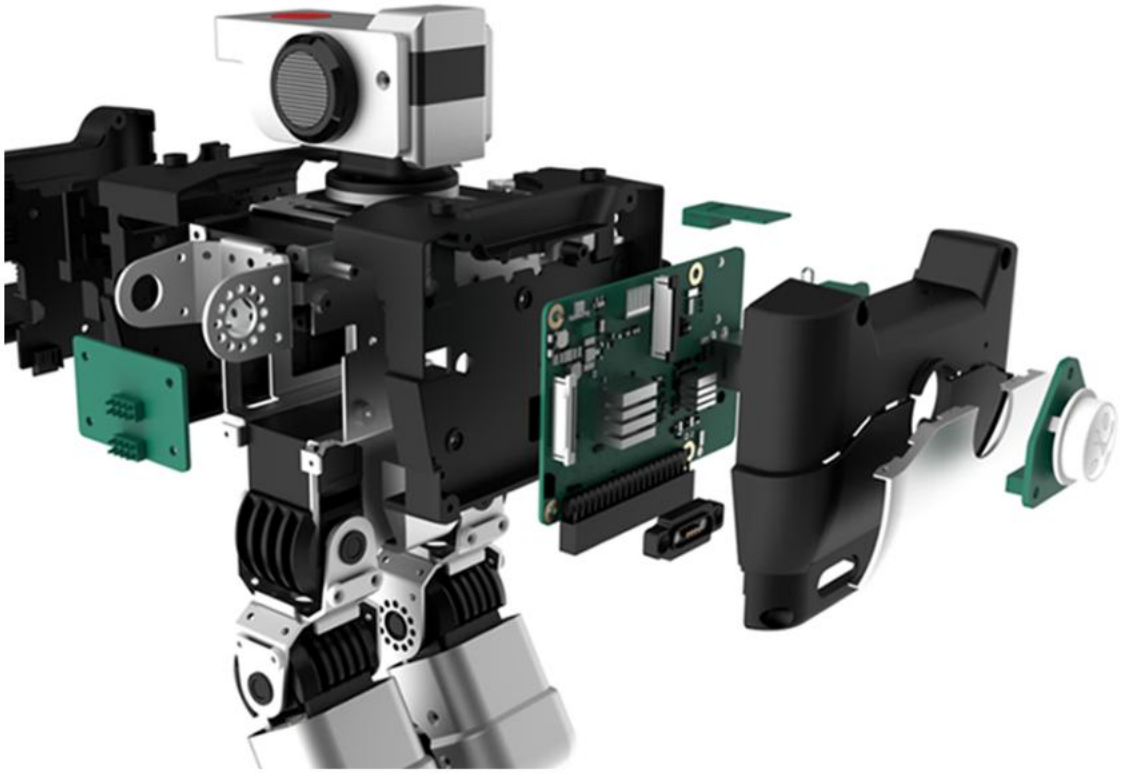


Figure 5: side section of Buddy

Buddy is made of a sturdy metallic build, featuring hinged joints, a rectangular head, and servo steering gears in its arms, legs, and torso as shown in Figure 3. The grey body is accented with black elements, giving it a sleek, industrial appearance suitable for futuristic robotics looks.

Buddy also has an on/off button on its chest (as shown in Figure 4). Pause the video. Press and hold the chest button for 2-3 seconds until it lights up blue, then release it. Once the robot is upright and its arms fall down, it will start successfully and announce, "Yanshee is ready for action”.

Figure 6: On/Off button located on Buddy's chest

If Buddy is running out of power, a red LED light on its back (as shown in Figure 5) will indicate this. To recharge its battery, plug the power cord into the charging port on the back of the robot. The power indicator will light up to indicate that it is charging, and will turn green once charging is complete.

Figure 7: Location of the battery charging indicator

Computer Vision

Let's dive into **Computer Vision**, a subfield of AI that focuses on enabling computers to interpret, analyse, and understand visual data from the world. According to IBM's website, computer vision is defined as:

"Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand."

Sources: *IBM, https://www.ibm.com/think/topics/computer-vision*

Figure 8: A computer vision-based traffic system in Bellevue, Washington State

By mimicking (and often surpassing) human visual perception, computer vision powers tasks like:

- **Object Detection:** Identifying objects in images.
- **Image Recognition:** Recognising and classifying images.
- **Facial Recognition:** Identifying people by their faces.
- **Motion Tracking:** Following movement in videos.

And it's not just for fun—computer vision has real-world uses, like detecting defects in fruits, for example, spotting imperfections in kiwifruits. Using advanced algorithms, machine learning

models, and deep learning techniques, computer vision processes vast amounts of visual information to extract meaningful insights from images or videos.

Pretty cool, right? 😊

Pixels

Now, coming back to our humanoid robot, Buddy. Who says Buddy doesn't have eyes? If you look at the front of its rectangular head, you'll see two eyes indicated by blue LEDs, one for the left eye and one for the right. Just check out the camera installed in Buddy's computer brain featuring 8 million pixels, staring right back at you!

Time to learn some cool terms:
By the way, 8 million pixels is the same as an 8-megapixel (MP) camera. But compare that to your phone, that is only 1/6 of an average phone camera resolution

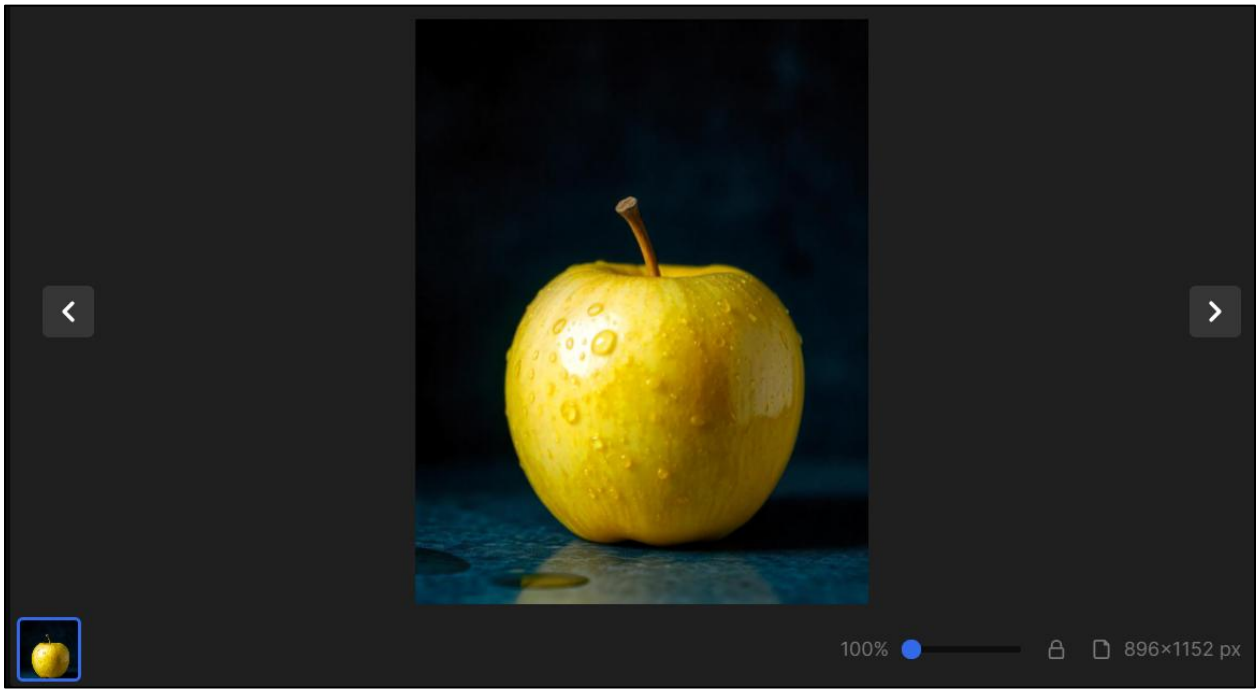
Pixel: The smallest unit of information in a photo.
Megapixel (MP): A unit of measurement that represents a million pixels.
More Megapixels = Higher Resolution: The more megapixels a camera has, the higher the resolution of its images.

More on pixels:

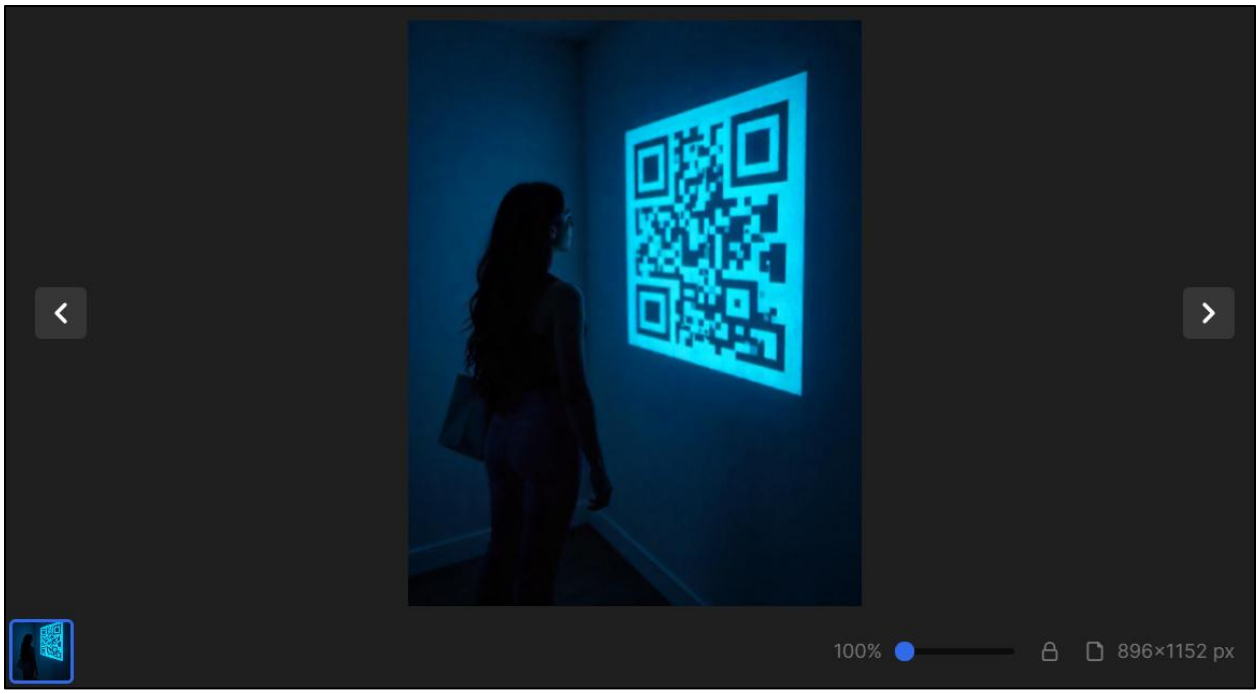
- The resolution of an image is determined by the number of pixels in each row and column.
- To calculate the number of megapixels in an image, multiply the number of pixels in a row by the number of pixels in a column.

Time for exercise:

- How many pixels in image 1? **ANS:** 1 million pixels



- How many pixels in image 2? **ANS:** 1 million pixels



Typically, a high-quality JPEG image might range from 3 million pixels to 7 million pixels, allowing for about 150 to 340 images to store in 1GB of RAM.

Programming Buddy to detect objects

Now that you understand computer vision and pixel fundamentals, let's introduce **Jupyter Lab**, which is a web-based interface to program Buddy for object detection. Before we make Buddy come to life, make sure you follow the “Programming Buddy Method 2” section in the Configuring Buddy.pdf guide**.**

On the Yanshee operating system, you should see the **Jupyter Lab** Desktop icon (Figure 9).

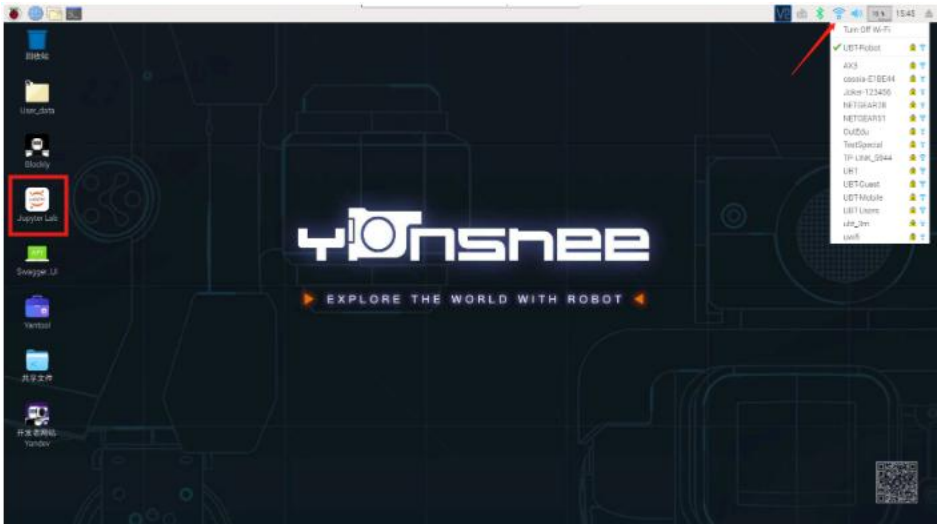


Figure 9: Yanshee operating system

Double-click the icon to open **Jupyter Lab** interface and once opened, your screen should see look like what is being shown in Figure 10**.**

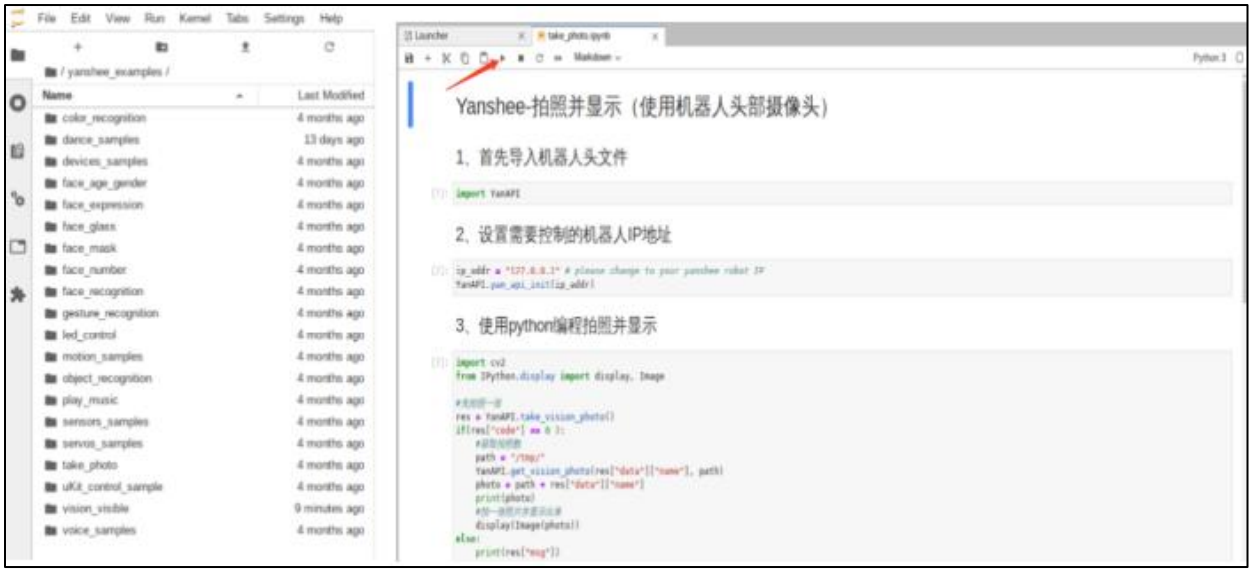


Figure 10: Jupyter Lab Interface

Create a new file called `object_detection.ipynb` and save it your folder. This is where we will write our code in Python for today's session. Feel free to ask staff for assistance if you get stuck.

Now we can start writing code to program Buddy in having the ability to detect objects. Buddy is ready to learn new things from us and to show off his new skills.

Firstly, we must include this code:

```
01    import YanAPI
02
03    ip_addr = "127.0.0.1"
04    YanAPI.yan_api_init(ip_addr)
05
06    res = YanAPI.sync_do_object_recognition()
07    print(res)
```

Line 1 tells the program to import a module, YanAPI. This allows the program to be able to use some of the object recognition functionality. The YanAPI documentation can be found on https://yandev.ubtrobot.com/#/en/api?api=YanAPI.

Line 3 defines a variable, `ip_addr`, to store the IP address of Buddy server. Buddy's server in this case is `"127.0.0.1"`. This will create connection between the Python program and Buddy, the humanoid robot. **Line 4** is initialising the `YanAPI` module with the provided IP address.

Line 6 calls a method from the `YanAPI` module. This `do_object_recognition()` will be used to analyse an image which will help in recognising object for Buddy. `sync_` is necessary to instruct the program to wait until there is a result before proceeding to the next line.

Line 7 calls the object recognition interface for identification.

Secondly, we must write a block of code that print the recognition results:

```
07    object_val = res ["data"]["recognition"]["name"]
08    if object_val != "none":
09         print("The result of recognition is:")
10         print(object_val)
11    else:
12         print("No object found")
```

Line 7 Accesses nested data from the result (`res`) returned by the object recognition function. To understand this better:

- `res:` this is a dictionary in Python. Dictionary is used when you need to store data that can be quickly retrieved.
- `["data"]["recognition"]["name"]:` these are the keys. This retrieves the value associated with `"name"`, which seems to store the recognized object's name.

Line 8 Checks if the value of `object_val` is not "none".

Line 9 prints the message `"The result of recognition is:"` to the console.

Line 10 prints the value stored in `["data"]["recognition"]["name"]`

Line 11 and 12 is where no object is found, it uses the else statement to print the message `"No object found"`

Let's have some fun by answering one short quiz before we get to **Today's Exercises**.

Quiz

1. Questions: What is a likely scenario where object_val = "none" ?

   Answer: where no object was recognised.

2. Question: Why is the IP address "127.0.0.1" used when programming Buddy?

   Answer: It represents the local server running on Buddy, allowing the Python script to communicate with Buddy.

3. Question: What happens when the object_val variable contains a recognised object's name?

   Answer: The program prints "The result of recognition is:" followed by the detected object's name.

Today's Exercises:

1. Begin with getting into group of 3.
2. Select the 3 objects from the pile to test Buddy's newly learnt skill.
3. Take turn raising the images to Buddy's eyes and make note how many out of the 3 objects raised, Buddy was able to recognise.
4. Show your findings in terms of probability to the student ambassadors.

Advanced Exercises:

See if you can use `start_voice_iat` method or `listen_res = YanAPI.sync_do_voice_asr()` from the API library in your code so that when you say "object recognition" to Buddy, then you will hear Buddy reply: "Object detecting, please wait", the robot's eye indicator will turn red; then use a photo of the object. When the robot successfully locates the object target, it will reply with the recognition result. For example: "I see a banana" or "This object is a rose". **Note:** When letting the robot recognise objects, it is best to ensure that the background is plain and simple, and that the natural light is not overexposed.

The main objects that can be recognised include: common flowers (sunflowers, carnations, egg plant, roses, lilies, lily of the valley, phalaenopsis, cosmos, daffodils, daisies, hyacinths, etc.), green plants (greenery, bamboo, fortune trees, Chinese orchids, tiger lilies, etc.), fruits (apples, oranges, bananas, dragon fruit, kiwi, strawberries, cherries, grapes, mangoes, peaches, etc.), and specific objects (AlphaMini robot, Yanshee robot, palm, cell phone, ordinary QR code, Rubik's cube, cell phone).

Once you think you are ready, ask a staff member to watch your code coming alive!

Summary

In this session, we explored the humanoid robot Buddy, delving into its hardware, software, and programming capabilities. Key concepts such as computer vision, object detection, and pixel fundamentals were introduced, emphasizing how Buddy processes and interprets visual data. We spent time in setting up Buddy properly using Raspberry Pi, programming it with Python via the Jupyter Lab environment, and utilising the YanAPI library for object recognition. Practical exercises encouraged hands-on application of these skills, including group activities to test Buddy's recognition accuracy and advanced challenges integrating voice commands. This session served as a comprehensive introduction to robotics and AI, laying the groundwork for further exploration in future sessions.

Useful Resources

- Pololu Robotics & Electronics. (2016). Raspberry Pi 3 Model B [Catalog]. https://www.pololu.com/product/2759
- Yanshee mobile app on App Store: https://apps.apple.com/us/app/yanshee/id1290088340
- Yanshee mobile app on Google Play: https://play.google.com/store/apps/details?id=com.ubtedu.alpha1x&hl=en_US
- Computer vision introduction: https://keremkargin.medium.com/computer-vision-fundamentals-and-opencv-overview-9a30fe94f0ce
- Yanshee manual: https://yandev.ubtrobot.com/#/en
- Yanshee API documentation: https://yandev.ubtrobot.com/#/en/api?api=YanAPI
- Object Recognition Guide: https://yandev.ubtrobot.com/#/en/guide