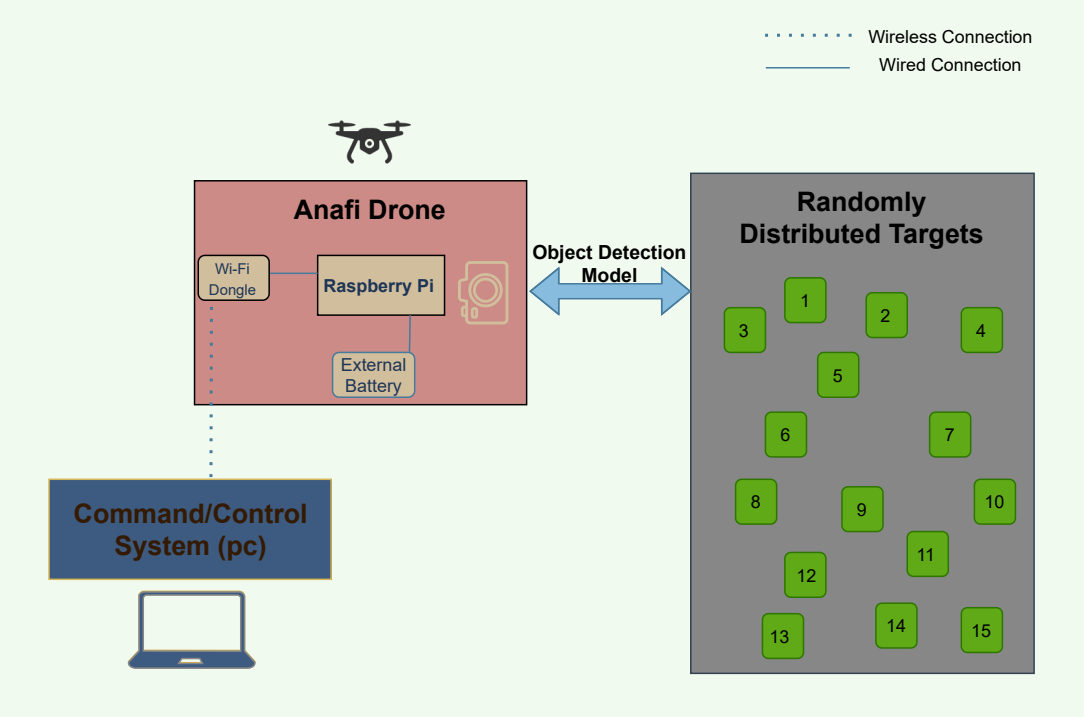


Fig. 1. A general overview of the solution.

The overview of the proposed solution is depicted in Fig. 1. The hardware components consist of an ANAFI drone, a Raspberry Pi with an external battery, and a Wi-Fi dongle. The built-in camera of the drone is used to take pictures of the targets and the Wi-Fi dongle is used to communicate with the command and control system, which sends high-level commands whenever the connection is present. An object detection model trained using RoboFlow detects and identifies the targets in the pictures. In addition, a deep reinforcement learning (DRL) agent is trained to visit virtual targets having the same mobility pattern as the real ones in a cyber-physical simulator called Sphinx. This finished model is loaded into the Raspberry Pi that acts as an onboard computer to the ANAFI drone and instructs it where to move to cover the targets in a minimum amount of time.

The following subsections will explain in more detail why this solution is chosen and what the integral components are.

0.1 Alternative solutions and tradeoffs

In Table 1, we can see some trade-offs between design A and design B, but in both designs, an onboard microcomputer must be present for flying control and the support of autopilot feature. There are three options for onboard microcomputers, which are shown in the following Table 2. Raspberry Pi 4 seems to be the winner since it has advantages in specifications, connection interfaces, and availability.

0.2 Selected solution overview

The proposed solution consists of two main parts: a commercial drone and a controlling system. We have chosen the commercial drones especially the Parrot ANAFI for several reasons. Firstly, it is supported by a continuously updated SDK and it can be controlled easily with a simple Python script which makes DRL development much easier and more stable.

Secondly, it has a good flight time as the ANAFI drone has a 2700 mAh battery. It can fly up to 25 minutes which is good enough for our application. Finally, its support of Wi-Fi 802.11 and GPS features is essential in our project for executing scripts and navigation.

For how the system will work, firstly, the user will import or choose the mobility pattern and set the constraints to the monitor device, which is a laptop. Then, the laptop will send basic high-level commands to the drone agent which is a Raspberry Pi that in turn will apply certain operations such as start and stop.

Once the user finishes importing the mobility pattern and starting the drone mission, the drone will take off and begin visiting areas and scanning for the most number of mobile targets based on the trained DRL model. The user will keep receiving live updates and the status of service on the control section using Wi-Fi. Most of the connections in the system are wireless, which will have benefits and drawbacks as discussed in the Section 0.4 below.

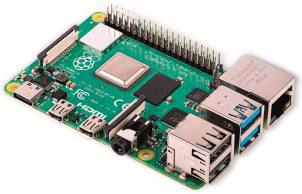
0.3 High level architecture

Fig. 2 shows a high-level architecture of the complete working system, in which a group of connected adapters and devices are combined into a single functional system. The architecture is composed of three sections namely interfacing, controlling, and targets. The interfacing section contains the drone that will handle the onboard computer, its power source, and the connection adapters. In the controlling part, a personal computer will be responsible for contacting the onboard computer to adjust settings, execute scripts, and get live updates and results. Finally, there will be multiple moving targets in the target section. For example, remotely controlled (R/C) cars are controlled manually and moving in a specific mobility pattern with varying directions and destinations. In the next section, hardware and software components will be presented in a more detailed manner.

Table 1. A comparison of the alternative solutions.

	Design A	*Design B*
Type	Custom made drone with onboard computer	Commercial drone with high performance and many features
Flexibility	Very flexible and customization is easy	Hard to customize or modify it since it flies under limited protocols and standards.
Price and Availability	Cheaper but the shipping and building processes must be considered	Expensive but in our case its available in our hands and ready to fly.
Onboard computer	Must have	Must have

Table 2. A comparison of the onboard computers.

	Raspberry Pi 4	NVIDIA *Jetson Nano*	DJI *Manifold*
Specifications	CPU: Cortex-A72 (ARM v8) 64-bit@ 1.5GHz | Ram: 4GB or 8GB LPDDR4-3200 SDRAM | GPU: Broadcom VideoCore VI	CPU: Quad-core ARM A57 @ 1.43 GHz | Ram: 4 GB 64-bit LPDDR4 | GPU: 128-core Maxwell	CPU: Quad-core, ARM | Ram: 2 GB DDR3L | GPU: Low-power GeForce graphics processor
Connection interfaces	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0	Gigabit Ethernet & M.2 Key E (for WiFi support).	10/100/1000 BASE-T Ethernet
Price & Availability	QAR 300 . Available and can be used on any drone	QAR 400 . Available but needs to be ordered and shipped	Very expensive and restricted to DJI drones and DJI company stopped selling it
Picture			

0.4 Hardware/software to be used

0.4.1 Software

There are three primary categories of software depending on the usage: simulation, training, and application, and they are listed in Table 3. The first part will focus on simulating the environment, testing the models, and flight control. Before discussing the software to be used, we have selected Ubuntu 18.04 (Bionic Beaver) as an operating system for several reasons. One key reason is that in addition to still being supported, it is compatible with the Parrot's Olympe and Sphinx programs, which are only supported on limited distributions and operating systems. Another reason is that it is a lite OS and can be installed on the onboard computer that will be attached to the drone. For the simulation part, using Sphinx and Gazebo software is very helpful to visualize the environment, control the drone, and apply the DRL model.

Sphinx is a simulation tool built on top of Gazebo to run the Parrot's drone firmware on personal computers, which comes with helpful features for simulation such as visualizing flight data at runtime, running the unmanned aerial vehicle (UAV) remotely, and executing scripts with the command line. Gazebo is a robot GUI simulation which simulates the visual and physical surrounding of drones and custom 3D objects. Fig. 3 shows how the Sphinx program looks like.

The training part is divided into reinforcement learning (RL), which is used to teach the ANAFI drone to complete the task, and object detection and identification, which is used to detect and count the targets required in determining the reward during the training. The RL agent is

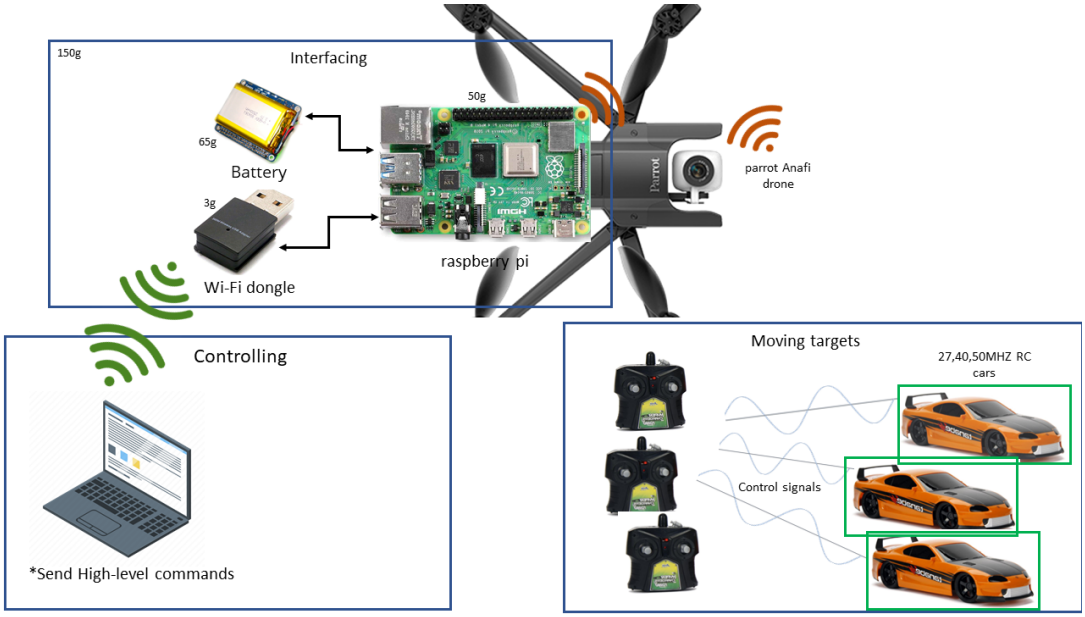


Fig. 2. The high-level architecture of the overall system.

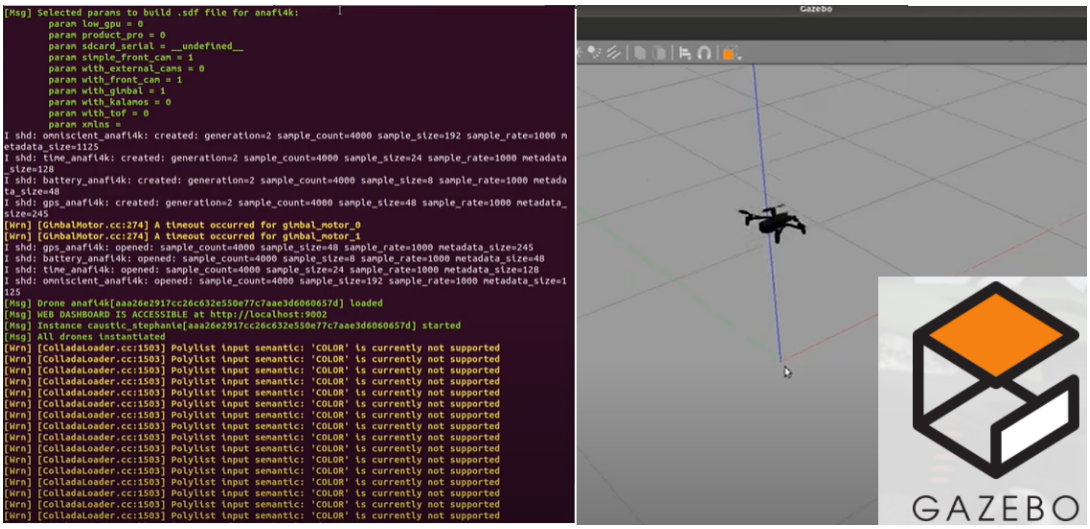


Fig. 3. The Sphinx program that runs on top of Gazebo.

Table 3. Software used in the project.

Software	*Logo*	*Justification*
Parrot Olympe	Parrot For Developers	A controller for the Parrot ANAFI drone. It makes controlling the drone possible using a Python script
Parrot Sphinx	Parrot For Developers	A simulator for the Parrot ANAFI drone. It loads the Parrot's drone firmware in the simulation environment. It is largely based on Gazebo.
Roboflow		A framework for Computer Vision development that can be used online. It facilitates labelling of images, splitting and merging datasets, applying image transformations and filters, and generating a link for the augmented datasets which will be used in the training notebooks.
Jupyter Notebook		A web application for writing, testing and sharing of code. It is free and does not require internet access like the Google Colab. It is used heavily in this project to experiment with new ideas in the Sphinx simulation.
Google Colab		A Jupyter notebook environment running in the cloud. It makes it easy to write, run, and share the code. Most importantly, it gives an option to use remote processing resources in addition to the local ones.

Fig. 4. An example of a user interface that will be built using sensor data coming from the ANAFI drone.

based on the DRL model developed by **Ged21**. The action space is composed of 9 movements in the forward, backward, left, right, forward-left, forward-right, backward-left and backward-right directions as well as a hover. The state space is given as

$$s = \{t, cell, [I_1, I_2, I_3, \ldots, I_m]\} \tag{1}$$

where t is the current time step, $cell$ is the ID of the cell above which the drone is, I_k is the binary variable that indicates if the target with an ID of k has been visited, m is the total number of targets, and the vector of length m is the container for the I indicators. For example, if only targets with ID's 3 and 7 have been visited in the current and previous time steps since the beginning of the episode, then the vector will have elements of one for I_3 and I_7 while the other elements are zeros. It follows that all the targets must be uniquely identifiable. In addition, the reward is how many new targets are captured in the current cell, and if the drone flies past the boundary, then a big negative reward will be incurred.

For the object detection, we used simulation tools to generate some training datasets. Firstly, we placed random objects and captured the images using the simulated drone camera. We have used a website called Roboflow which helped us labelling the objects and generate new datasets from the existing ones with different types of augmentation such as rotation and scaling. For the object detection model, Google Colab notebook was a sufficient tool to start training using convolutional neural network (CNN) YOLOv5 in addition to the Jupyter notebook which was very helpful in code experimentation.

For the application software, Parrot Olympe was used to send commands to the physical as well as the simulated drone and control the flight trip and how the drone moves. Parrot Olympe uses Python controller programming interface for Parrot drones which makes controlling simple and easy using a Python script. Moreover, Olympe allows us to read sensor data, such as the GPS fixes and camera feed, from the ANAFI drone. This will make it possible to build a user interface on the control and command station shown in Fig. 4 to monitor the progress of the drone when it is autonomously executing the task visitation mission.

0.4.2 Hardware

The main core of the hardware part is the drone, which will be the Parrot ANAFI. Table 4 lists the hardware components used in this project and their justifications. The second important

device is the Raspberry Pi. It acts as an onboard computer and is used in this project mainly to facilitate the low-level control of the drone by sending frequent control commands to the drone in a certain direction or keep hovering. The choice of action is determined by the DRL model that will be installed in it. Hence, the tasks of the Raspberry Pi are:

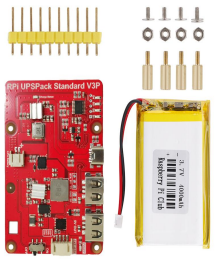
1. connecting to the drone's access point using a Wi-Fi interface,
2. controlling the drone by executing Olympe to send low-level control signals and to receive sensory data,
3. applying the DRL model supported by the command and control system, and
4. receiving high-level commands and sending data to the command and control system.

The Parrot ANAFI drone is connected to the Raspberry Pi, which is the onboard computer, using both devices' internal 2.4 GHz Wi-Fi interfaces. Firstly, we add the ANAFI drone's access point to the saved devices list in the Raspberry Pi. Once the Raspberry Pi boots up, it automatically keeps searching for the access point and connects to it once it is available.

For the connection between the command and control system and the Raspberry Pi, the Raspberry Pi will use a 300 MBps Wi-Fi adapter dongle connected to its USB port. This will allow it to create an access point to which the command and control device will connect and by which the Raspberry Pi can be controlled. This control of the onboard computer is done through the Secure Shell (SSH) protocol or the Virtual Network Computing (VNC) if the graphical interface is needed.

Regarding the power source for the Raspberry Pi, we thought of taking power directly from the drone's battery, but, after some research, we found that the ANAFI drone's socket is somehow different. It is also challenging and the drone battery is susceptible to shutting down immediately if the voltage reaches less than 3.0 Volt. So, we did not want to take the risk and used a lithium battery with a power board called UPSPack Standard Power Supply attached to the main Raspberry Pi board. It includes a 4000 mA h lithium battery, which provides enough power and time for our application.

Table 4. Hardware used in the project.

Hardware	*Picture*	*Justification*
Parrot ANAFI Drone		Available in the university, can be controlled easily with simple Python script, 4K-high resolution camera.
Raspberry Pi 4		Specifications are enough for our application, support Wi-Fi and its small size and weight is an advantage.
RPI UPSPack V3 with 4000 mA h lithium Battery		Support up to 4 hours which is more than enough , the board got an LED indicator for charging level also the weight and shape is an advantage.
Wireless N Nano USB Adapter		Cheap and do its job, good coverage range.
Laptop		Any laptop with good WiFi interface card will be enough for our case.