BSP-Documentation

This document will guide you on how to successfully launch the swarm, test it and to set all the necessary dependencies needed

---

1. Setting up the raspberry Pi 4/odroid XU4

1. **Download the ubuntu mate 20.04 LTS OS image**
   - For the **raspberry pi 4**:
     - Follow this link.
   - For the **odroid XU4**:
     - Follow this link.
2. **On a computer install the balena etcher**
   - once you have one of the above softwares, you will need to flash the ubuntu image (of step 1) to an sdcard (preferably 32Gb+).
3. After having installed the image on a sdcard you can insert it on the raspberry pi 4/odroid XU4 and boot it up. You should be now prompt to configure the OS. (Name suggestion: tello_controller_**n**, *where n is the number of the controller*).
4. **Enabling ssh** (insert the following commnads on the terminal:

```
$ sudo apt update
$ sudo apt install openssh-server
$ sudo ufw allow ssh
```

5. **Install ROS**. To do so follow the instruction on section 2
6. **Build the project workspace**. To do so follow the instructions on section 3
7. **Install the needed dependencies**. To do so follow the instructions on section 4
8. You can easily change the hostname of the machine by typing this on the terminal: `$ hostnamectl set-hostname new-name`

NOTE: you will now be able to connect from the `supervisor` to the raspberry pi/XU4 via **ssh** so that you don't need to have a monitor connected to each raspberry pi 4/XU4 to control it! Therefore on the supervisor you would type `ssh usename@ip_address`. Then you will be prompt to enter **yes** and most likely the **password** of your raspberry pi/XU4.

---

2. ROS installation

To run ROS you will need a Linux OS which can be either ubuntu 20.04 LTS or Debian. I do suggest that you go for the ubuntu as it was not tested on the Debian distribution.

Note that we are using the ROS Noetic. You can also go to this link to follow the instructions from the website (I advise you to try first on the website as it will contain the most recent updates).

Installation procedure

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
$ sudo apt update
$ sudo apt install ros-noetic-desktop-full
$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
$ sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

```
$ sudo apt install python3-rosdep
$ sudo rosdep init
$ rosdep update
```

NOTE: for the raspberry/XU4 instead of installing the `sudo apt install ros-noetic-desktop-full` only install the `sudo apt install ros-noetic-desktop` Also if you are using a zsh terminal instead of using `.bashrc` use `.zshrc` .

---

3. Building the project workspace

```
$ mkdir -p ~/catkin_ws/src/swarm
$ cd ~/catkin_ws/src/swarm
$ git clone https://github.com/PedroS235/TelloSwarm.git
$ git clone https://github.com/eric-wieser/ros_numpy.git
$ git clone https://github.com/joselusl/aruco_eye.git
$ git clone https://github.com/joselusl/robot_component_srvs.git
$ git clone https://github.com/joselusl/perception_msgs.git
$ git clone https://github.com/joselusl/pugixml.git
$ cd ~/catkin_ws
$ catkin_make
$ echo source ~/catkin_ws/devel/setup.bash >>~/.bashrc
$ source ~/.bashrc
$ roscd tello_ros/src
$ chmod +x ./*
$ roscd tello_formation/src
$ chmod +x ./*
$ roscd marker_localisation/src
$ chmod +x ./*
```

NOTE: if you are using a zsh terminal instead of using `.bashrc` use `.zshrc`. Also every time you build the project (do a catkin_make) you will need to source, which is done by typing `source ~/.bashrc` .

---

4. Installing the depedencies

In the terminal:

```
$ sudo apt install python3-pip
$ sudo apt install git
$ sudo apt-get install cmake libopencv-dev qtbase5-dev libqt5opengl5-dev libopenni2-dev
```

Installing Ucoslam:

1. First, download Ucoslam.
2. Unzip the folder, and rename it to ucoslam
3. Open a terminal window where the file is located and run the following commands:

```
$ cd ucoslam
$ mkdir build
$ cd build
$ cmake ../ -DBUILD_GUI=ON
$ make -j4
$ sudo make install
```

---

5. Steps to do before launching the swarm

Printing the aruco marker

First you will need to **print the aruco markers**. To do so you will go to the `aruco_eye/aruco_lib/resources/ArucoMarkers` folder and open the docArucosA4.docx and then print the number of aruco markers you will need, which is the first one, the `aruco_marker_0` .

Once printed you will need to **measure the size of the aruco marker on the paper**. Mine is of 0.178m (or 17,8cm). If your size is the same then you are set. If not, then you will need to go to the `aruco_eye/aruco_eye_ros/config` folder and open the `arUcoList.xml` and change the size number of the aruco marker that you printed.

Setting the environment

On the space/room where you will launch the swarm, you will need to **place the aruco marker in some wall**. You should place it between 30-50 cm above the ground. Once the aruco marker fixed, you should put some mark on the ground where the center of the room is, which will be the coordination (0,0) of the world. Then mesure the distance (x, y, z) between the (0,0, 0) of the world relative to the aruco marker. The other measurement are the goal positions. Again place some mark on the ground where you want the goals position to be. Then mesure the distance (x, y, 0) from the goal relative to the world. Once you have all the measurements, then you can go to the `marker_localisation/config` folder and open the `config.yaml` file. There you will find aruco_marker_0: [x, y, z, pitch, roll, yaw] and its where you should set the mesurements for the aruco marker in **meters. (**For the pitch roll and yam follow the instruction on the file)**.**

For the goals you will need to go to the `tello_formation/launch` folder and open the `static_goals.launch` file and change the distances accordingly, also in **meters** (same as for the aruco marker).

Setting the right name of the Tello wi-fi

In order to the controller connect to the Tello, you will need to specify the wi-fi name on the controllers. To do so go to `tello_ros/src/` and open the command_server.py script. On the start method you will find a line (should be line 27) commented `edit here the name of the Tello wifi name`. *You will need to change the name for every controller specifying which controller controls which drone.*

---

6. Setting the ROS network

Configuring the private network

On the `supervisor` and the `controllers` (supervisor a computer, and the controllers either the raspbery pi or the XU4 board) you will need to configure the network. To do so, go to the network settings and create a new network, and do the following steps:

```
1. go to the settings
2. go to network
3. click on the <+> symbol to add a new network
4. you can name it as you wish but I suggest Supervisor or controller0n (n being a number)
5. Select the MAC address with the only option you have
6. go to the IPv4 tab
7. select the manual setting
8. On the addresses click on the address field and type 10.0.0.n (choose n=1 for the supervisor and then others numbers for the controllers
9. on the netmask type: 255.255.255.0
10. you can now add this network by pressing the button <add> on the top right
```

NOTE: I chose the IP of the form 10.0.0.0, but you can choose some other IP, like the 192.168.0.1 for instance

Setting the ROS Master

For the controllers only

```
$ echo export ROS_MASTER_URI = http://10.0.0.1:11311 >> ~/.bashrc
$ source ~/.bashrc
```

NOTE: if you have changed the IP numbers, then instead of *10.0.0.1* you will need to insert the IP of your supervisor.

Setting the hosts

This step is to not have problems when pinging the machines. ***If you don't do this step you might have problems with the ROS connection.***

First type in the terminal `nano /etc/hosts`. Then you will need to add the right names and IP you have set accordingly. Here is an example how you should have if you have the same IPs and hostnames as me:

```
10.0.0.1    supervisor
10.0.0.2    controller01
10.0.0.3    controller02
10.0.0.4    controller03
...
```

Enabling to control the WIFI via ssh only on the controllers

This is to allow you to launch via ssh, otherwise you have not the permission to do it via ssh

```
$ touch /etc/polkit-1/localauthority/50-local.d/allow-ssh-networking.pkla
$ nano /etc/polkit-1/localauthority/50-local.d/allow-ssh-networking.pkla
```

Then paste the following inside the file:

```
[Let adm group modify system settings for network]
Identity=unix-group:adm
Action=org.freedesktop.NetworkManager.network-control
ResultAny=yes
```

Hardware side

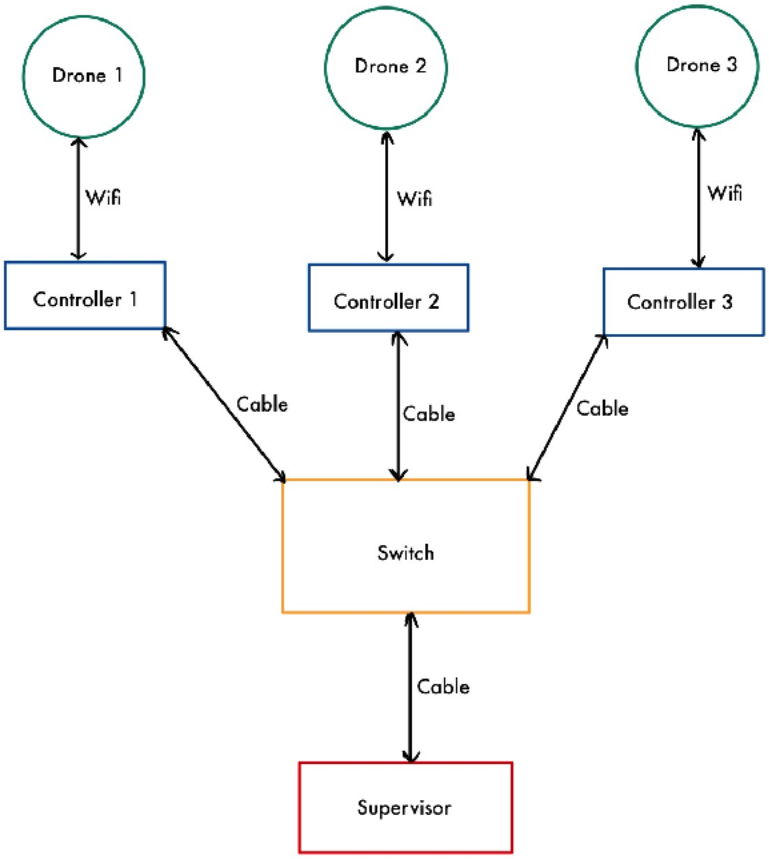
First you will need a switch that has the enough ports to connect all the controllers and the supervisor. then you will connect the both the controllers and the supervisor to the switch via an ethernet cable. If you have set all correctly the controllers and the supervisor should all be connected, and should be able to communicate with each other. To verify that everything works try pinging all the machines. Let's say you want to ping the controller01 from the supervisor. Then on the supervisor you type in the terminal `ping controller01`. You should see something like this:

```
64 bytes from 192.168.178.56: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 192.168.178.56: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 192.168.178.56: icmp_seq=3 ttl=64 time=0.050 ms
64 bytes from 192.168.178.56: icmp_seq=4 ttl=64 time=0.043 ms
64 bytes from 192.168.178.56: icmp_seq=5 ttl=64 time=0.026 ms
64 bytes from 192.168.178.56: icmp_seq=6 ttl=64 time=0.038 ms
64 bytes from 192.168.178.56: icmp_seq=7 ttl=64 time=0.039 ms
```

If for any reason you are getting a message error, like **host unreachable**, then make sure that the step **Setting hostsnames** is correct. If you don't know the hostnames you can check by typing `hostname`.

NOTE: you should ping every machine connected on the network in both directions. So above you pinged from the supervisor to the controllero1, now you should ping from the controller01 to the supervisor, and so on.

How the newtork should look like



---

7. Launching the swarm

On the supervisor

Launching the formation.launch

```
$ roslaunch tello_formation formation.launch grounded:=false
```

Open another terminal and type

Launching the aurco_detector.launch

```
$ roslaunch tello_formation sensors.launch id:=n
```

The n will be the number of the drone you have/will launch. For example, if you are launching 3 drones, then you will need to open 3 terminals and insert the command above and change the n to 0, 1 and 2, which correspond to the number of drones.

On the controllers

Launching the single_drone.launch

```
$ roslaunch tello_formation single_drone.launch id:=n
```

For the n, do the same as above for the **aruco_detector**

Open rviz

```
$ rviz
```

On the resources folder you will find rviz_config. Once you are in the rviz you can open it which is already configured, or you can configure it yourself by adding the topics needed.

---

8. Project files explanation

- The ROS packages wrote for these project are the following:
  - marker_localization
  - tello_formation
  - tello_ros
- The simulation folder contains a simulation of what the swarm of drones should be able to do.
  - Inside there lies 2 Python files, main.py and capt.py. capt.py is the algorithm that computes the trajectories that the drones should do. The main.py is the actual simulation, and you can run it by typing in the terminal the following:

    ```
    python3 main.py
    ```

tello_ros package

This package contains all the scripts that sends/receives commands from the drone and captures the video.

Source files

- **tello_client.py:** This script allows to send commands to the Tello drone and receive their answer.
- **video_stream.py:** This script will retrieve the video from the drone and publish to a ROS topic named `camera/image_raw`.
- **command_server.py:** This script allows to automatically connect to the drone's WiFi and will send the first commands which allows to communicate with the drone with commands.
- **camera_info_publisher.py**: This script will take a YML file which contains the different camera parameters and will publish these into a ROS topic named `camera/camera_info`
- **video_recorder.py**: This script allows to record the video being retrieved from the drone and save it into a .mp4 video file. (useful to create a map for Ucoslam)

Launch files

- **drone.launch**: This launch file initializes the scripts *command_server*, *video_stream* and *camera_info_publisher*.
- **prerecord.launch**: This launch file initializes the scripts *command_server* and *video_recorder*. It's purpose is only to record the video being captured from the drone.

tello_formation package

This package contains the scripts that allows to have the swarm of drones make some task. Currently, the task is to fly to a goal and land.

Source files

- **capt**: This script is responsible to compute the trajectory for each drone in order to land on a goal. The algorithm tries to choose the closes goal to the drone without any collisions with other drones.
- **supervisor**: This script is the responsible to send the commands to each controller, that commands a drone.
- **controller**: This script controls a unique drone and it receives the commands from the supervisor.

Launch files

- **static_goals.launch**: This launch file will publish the goals coordinates using ros tf2.
- **single_drone.launch**: This launch file will launch the `drone.launch` (from the tello_ros package). Note that an id (a number) is required. This id is to distiguish the different drones.
- **formation.launch**: This launch file will initialize all the necessary files to start the swarm.
- **sensors.launch**: This launch file initializes the tello_slam_ros launch files, also the aruco_eye launch files and the pose_estimation launch files.

tello_slam_ros package

This package is a ROS wrapper for Ucoslam.

Source files

- **tello_slam_ros_detector_node.cpp**: This node will retrieve the image from the drone and will pass it to ucoslam. After the image was processed, it publishes the pose of the camera to the TF tree.
- **tello_slam_ros_display_node.cpp**: This node will retrieve the image from the drone and will publish the image with the key points drawn in the image to a ROS topic named `tello_slam/tello_slam_observation_image/image_raw`

Launch files

- **tello_slam_ros_detector.launch**: This launch file initializes the tello_slam_ros_detector_node.
- **tello_slam_ros_display.launch**: This launch file initializes the tello_slam_ros_display_node.

pose_estimation package

This package creates the required TF frames to localize the drones in the world.

Source files

- **location_estimation.py**: This scripts creates new TF frames to locate the drones in the world, via the slam thechnology and the aruco_eye software.
- **static_transform.sh**: This script is just to facilitate the static transforms of the markers.

Launch files

- **static_markes.launch**: creates a TF frame for each visual marker placed in the world.

- **naive_estimation.launch**: runs the location_estimation node