

AI Large-Scale Online Voice Assistant

AI Large-Scale Online Voice Assistant

1. Online Voice Configuration
2. Create a Systemd Service
 - 2.1 Raspberry Pi 5
 - 2.1.1 Creating a Docker Container Startup Script
 - 2.1.2 If you are using the factory image of Yabo's large AI model, you do not need to recreate the script. Simply run the following command to restart the auto-start service:
 - 2.1.3 Related Commands:
 - 2.1.4 CSI Camera Auto-Start Service

If you are using the factory image of Yabo's large AI model, you do not need to recreate the script. Simply run the following command to restart the auto-start service:
 - 2.2 RDK X5
 - 2.2.1 Creating a Startup Script
 - 2.2.2 Create Systemd service file
 - 2.2.3 Managing and Debugging the Service
 - 2.3 ORIN
 - 2.3.1 Create a startup script
 - 2.3.2 Creating the Systemd Service File
 - 2.3.3 Managing and Debugging the Service

1. Online Voice Configuration

Before setting up auto-start, we must ensure that the program itself can work independently while online. This needs to be done by modifying the configuration file.

Raspberry Pi 5 requires entering a Docker container; RDK X5 and Orin controllers do not:

```
./ros2_docker.sh
```

If you need to enter other commands in the same Docker container later, simply enter `./ros2_docker.sh` again in the host terminal.

1. Open the configuration file:

```
vim ~/yahboom_ws/src/largemodel/config/yahboom.yaml
```

2. Modify Configuration Parameters:

Please check the following parameters in the file and ensure that their values match those shown below. If the parameters do not exist, add them.

```
asr:                                     #Voice node parameters
  ros__parameters:
    VAD_MODE: 2                          #vad sensitivity
    sample_rate: 16000                   #asr recording audio sampling rate
    frame_duration_ms: 30                 #vad frame size in ms
    use_online_asr: True                  #Whether to use online ASR
  recognition (True uses online, False uses offline)
  mic_serial_port: "/dev/ttyUSB0"        #Microphone serial port alias
  mic_index: 0                           #Microphone Index
```

```

language: 'zh'                                #asr language

model_service:                                #Model server node parameters
  ros__parameters:
    language: 'zh'                            #Large Model Interface Language
    useonline: True                          #Whether to use online speech
    synthesis (True uses online, False uses offline)
    llm_platform: 'tongyi'                   # Here we take Tongyi as an example

```

- `use_online_asr` and `useonline` must be set to `tongyi`.
- `llm_platform` must be set to `tongyi`.

With this setup, everything is online.

3. **Save the file** and **recompile** the project to apply the changes. This requires building within a Docker container:

```

cd ~/yahboom_ws
colcon build
source install/setup.bash

```

After completing this step, the program is now a fully online voice service.

2. Create a Systemd Service

Now, we will create a systemd service to automatically run `largemodel_control.launch.py` at system startup.

Note: If you are using the Yabo AI Large Model factory image, these scripts are already created; you only need to restart the auto-start container.

2.1 Raspberry Pi 5

2.1.1 Creating a Docker Container Startup Script

1. Creating a Script File:

In your home directory (`~`), create a file named `ros2-docker-autostart.sh`.

```
vim ~/ros2-docker-autostart.sh
```

2. Write the script content:

Copy and paste the following content into the script file.

```

#!/usr/bin/env bash

set -e

CN=${CONTAINER_NAME:-ros_humble_ai_service}
IMG=${IMAGE_NAME:-ros_humble_ai:v1.0}
WS=${HOST_WS_DIR:-"$HOME/yahboom_ws"}
ROS=${ROS_DISTRO:-humble}
CMD=${LAUNCH_CMD:-"ros2 launch largemodel largemodel_control.launch.py"}
AUDID=${AUDIO_UID:-$(id -u)}
XAUTH=${XAUTHORITY:-"$HOME/.Xauthority"}

```

```

mkdir -p "$WS"

x=()
if [ -d /tmp/.X11-unix ]; then
    command -v xhost >/dev/null 2>&1 && xhost +local:root || true
    x+=( -e DISPLAY=${DISPLAY:-:0} -e QT_X11_NO_MITSHM=1 -v /tmp/.X11-
unix:/tmp/.X11-unix )
    [ -f "$XAUTH" ] && x+=( -v "$XAUTH:/root/.Xauthority" -e
XAUTHORITY=/root/.Xauthority )
fi

a=()
if [ -S "/run/user/$AUID/pulse/native" ]; then
    a+=( -e PULSE_SERVER=unix:/run/user/$AUID/pulse/native -v
"/run/user/$AUID/pulse:/run/user/$AUID/pulse:ro" )
    [ -d "$HOME/.config/pulse" ] && a+=( -v
"$HOME/.config/pulse:/root/.config/pulse:ro" )
fi

d=()
[ -e /dev/video0 ] && d+=( --device=/dev/video0 )
[ -d /dev/bus/usb ] && d+=( --device=/dev/bus/usb )
[ -d /dev/snd ] && d+=( --device=/dev/snd )
[ -e /dev/mic ] && d+=( -v /dev/mic:/dev/mic )

if docker ps -a --format '{{.Names}}' | grep -qx "$CN"; then
    docker ps --format '{{.Names}}' | grep -qx "$CN" || docker start "$CN"
>/dev/null
else
    docker run -d \
        --name "$CN" --net=host --privileged --restart unless-stopped \
        --security-opt apparmor:unconfined \
        -v "$WS:/root/yahboom_ws:rw" \
        "${d[@]}" "${x[@]}" "${a[@]}" \
        "$IMG" \
        bash -lc "source /opt/ros/$ROS/setup.bash; [ -f
/root/yahboom_ws/install/setup.bash ] && source
/root/yahboom_ws/install/setup.bash; $CMD"
fi

echo "OK: $CN"

```

3. Save and exit

4. Confirm Docker starts on boot:

```
sudo systemctl enable --now docker
```

5. **Grant the script execute permissions**, then run it:

```
chmod +x ~/ros2_docker_autostart.sh
./ros2_docker_autostart.sh
```

6. verify:

```
docker inspect -f '{{.HostConfig.RestartPolicy.Name}}' ros_humble_ai_service
```

The output should be "unless-stopped". After that, Docker will automatically start the container every time the machine boots up, which can achieve the effect of automatically starting the voice assistant.

2.1.2 If you are using the factory image of Yabo's large AI model, you do not need to recreate the script. Simply run the following command to restart the auto-start service:

```
~/ros2_docker_autostart.sh
```

2.1.3 Related Commands:

Immediately stop the container, and also stop automatic startup:

```
docker stop ros_humble_ai_service
```

View the runtime logs of the Docker container:

```
docker logs -f ros_humble_ai_service
```

2.1.4 CSI Camera Auto-Start Service

To access CSI footage from within Docker, the host_stream.py file on the host machine needs to be started.

Therefore, if you need to access CSI camera footage during auto-start, you also need to add this file to the Systemd startup service.

Note: If you are using the Yabo AI large model factory image, these scripts have already been created; you only need to restart the auto-start service.

1. Enter command to write file

```
sudo vim /etc/systemd/system/host-stream.service
```

2. Copy the following content into the file.

```
[Unit]
Description=Host stream (runs ~/host_stream.py)
After=network.target

[Service]
Type=simple
User=pi
WorkingDirectory=/home/pi
ExecStart=/usr/bin/python3 /home/pi/host_stream.py
Restart=on-failure
RestartSec=5
StandardOutput=append:/home/pi/host_stream.log
StandardError=append:/home/pi/host_stream.log

[Install]
```

```
wantedBy=multi-user.target
```

3. Save and exit
4. Apply and start the service:

```
sudo systemctl daemon-reload
sudo systemctl enable host-stream.service
sudo systemctl start host-stream.service
```

5. Check service status:

```
sudo systemctl status host-stream.service
```

If you see ``Active: active (running)``, then the service has started successfully!

6. If you do not need the CSI camera auto-start service, run the following command to stop it:

```
sudo systemctl stop host-stream.service
sudo systemctl disable host-stream.service
```

If you are using the factory image of Yabo's large AI model, you do not need to recreate the script. Simply run the following command to restart the auto-start service:

```
sudo systemctl enable host-stream.service
sudo systemctl start host-stream.service
```

2.2 RDK X5

2.2.1 Creating a Startup Script

To ensure that `systemd` can correctly load the ROS2 environment, the best practice is to create a simple `bash` script to encapsulate our startup command.

1. **Create the script file:**

In the directory (`~/yahboom_ws/src/largemodel/`), create a file named `start_largemodel.sh`.

```
vim ~/yahboom_ws/src/largemodel/start_largemodel.sh
```

2. **Write the script content:**

Copy and paste the following content into the script file.

```
#!/bin/bash

# Source ROS2 Humble environment
source /opt/ros/humble/setup.bash

# Source Yahboom workspace environment
source /home/sunrise/yahboom_ws/install/setup.bash

# Start the largemodel control script
ros2 launch largemodel largemodel_control.launch.py
```

Important Note: Please ensure that `/home/sunrise/` in the script is replaced with your own user home directory path.

3. Save and exit

4. Grant script execution permissions:

```
chmod +x ~/yahboom_ws/src/largemodel/start_largemodel.sh
```

2.2.2 Create Systemd service file

1. Creating a Service File:

You need to use `sudo` privileges to create this file.

```
sudo vim /etc/systemd/system/largemodel.service
```

2. Write service configuration:

Copy and paste the following content into the service file.

```
[Unit]
Description=Robot Service
After=network.target sound.target graphical.target multi-user.target
Wants=network.target sound.target graphical.target multi-user.target

[Service]
Type=simple
User=sunrise
Group=sunrise
Environment=DISPLAY=:0
Environment=XDG_RUNTIME_DIR=/run/user/1000
Environment=PULSE_SERVER=unix:/run/user/1000/pulse/native
SupplementaryGroups=audio video
ExecStartPre=/bin/sleep 10
ExecStart=/home/sunrise/yahboom_ws/src/largemodel/start_largemodel.sh
Restart=on-failure
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

- `Environment=DISPLAY=:0` will output the GUI pop-up to the desktop. Users connecting via SSH can change this to `Environment=DISPLAY=:10`. You can find the specific parameters by typing `echo $DISPLAY` in the terminal.
- The path in `ExecStart` must be exactly the same as the path to the startup script to be executed.

3. Save and exit.

2.2.3 Managing and Debugging the Service

Now that your service has been created, we need to get `systemd` to load it and set it to start automatically on boot.

1. Reload the `systemd` daemon so it reads the service file we just created:

```
sudo systemctl daemon-reload
```

2. Set the service to start automatically on boot:

```
sudo systemctl enable largemodel.service
```

3. Start the service immediately:

```
sudo systemctl start largemodel.service
```

4. Check the service status:

This is the most important command to verify that the service is running successfully.

```
sudo systemctl status largemodel.service
```

* If you see `Active: active (running)`, congratulations, the `service` has started successfully!

* If the status is `failed` or something `else`, please continue to the next step for debugging.

5. View Service Logs (Essential for Debugging):

If the service fails to start, you can view all real-time logs generated by the `ros2 launch` command using the following command. This is crucial for troubleshooting.

```
journalctl -u largemodel.service -f
```

After completing all the above steps, the fully offline `largemodel` voice service will now start automatically every time you boot up.

To disable the automatic startup service, you can run the following command:

```
sudo systemctl start largemodel.service  
sudo systemctl disable largemodel.service
```

2.3 ORIN

2.3.1 Create a startup script

To ensure that `systemd` can correctly load the ROS2 environment, the best practice is to create a simple `bash` script to encapsulate our startup command.

1. Creating a script file:

In the directory (`~/yahboom_ws/src/largemodel/`), create a file named `start_largemodel.sh`.

```
vim ~/yahboom_ws/src/largemodel/start_largemodel.sh
```

2. Write script content:

Copy and paste the following content into the script file.

```
#!/bin/bash

# Source ROS2 Humble environment
source /opt/ros/humble/setup.bash

# Source Yahboom workspace environment
source /home/jetson/yahboom_ws/install/setup.bash

# Start the largemodel control script
ros2 launch largemodel largemodel_control.launch.py
```

Important Note: Please ensure that `/home/sunrise/` in the script is replaced with your own user home directory path.

3. Save and Exit

4. Grant Script Execution Permissions:

```
chmod +x ~/yahboom_ws/src/largemodel/start_largemodel.sh
```

2.3.2 Creating the Systemd Service File

This is the most crucial step. We will tell the system that we have a new service to manage.

1. Create the Service File:

You need to use `sudo` privileges to create this file.

```
sudo vim /etc/systemd/system/largemodel.service
```

2. Write the Service Configuration:

Copy and paste the following content completely into the service file.

```
[Unit]
Description=Robot Service
After=network.target sound.target graphical.target multi-user.target
Wants=network.target sound.target graphical.target multi-user.target

[Service]
Type=simple
User=sunrise
Group=sunrise
Environment=DISPLAY=:0
Environment=XDG_RUNTIME_DIR=/run/user/1000
Environment=PULSE_SERVER=unix:/run/user/1000/pulse/native
SupplementaryGroups=audio video
ExecStartPre=/bin/sleep 10
ExecStart=/home/sunrise/yahboom_ws/src/largemodel/start_largemodel.sh
Restart=on-failure
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

Extremely Important!!!

- Ensure that the paths in `WorkingDirectory` and `ExecStart` are exactly the same as your actual paths.

3. Save and Exit.

2.3.3 Managing and Debugging the Service

Now that your service has been created, we need to get `systemd` to load it and set it to start automatically on boot.

1. **Reload the `systemd` daemon** so that it reads the service file we just created:

```
sudo systemctl daemon-reload
```

2. **Set the service to start automatically on boot:**

```
sudo systemctl enable largemodel.service
```

3. **Start the service immediately:**

```
sudo systemctl start largemodel.service
```

4. **Check the service status:**

This is the most important command to verify that the service is running successfully.

```
sudo systemctl status largemodel.service
```

* If you see `Active: active (running)`, congratulations, the `service` has started successfully!
* If the status is `failed` or something `else`, please continue to the next step `for` debugging.

5. **View service logs (essential for debugging):**

If the service fails to start, you can view all real-time logs generated by the `ros2 launch` command using the following command. This is crucial for locating the problem.

```
journalctl -u largemodel.service -f
```

After completing all the above steps, the fully online `largemodel` voice service will now start automatically every time you boot up.