AI Elephant Project Setup Guide

This guide will help you set up the complete AI Elephant multi-agent system on your Raspberry Pi.

Prerequisites

Hardware Requirements

- Raspberry Pi 4 (4GB+ RAM recommended)
- MicroSD card (32GB+)
- Raspberry Pi Camera Module
- USB microphone or microphone array
- Multiple servo motors (SG90, MG90S, MG996R)
- Force-sensing resistors (FSRs)
- RFID reader module (RC522)
- RFID key fobs
- Proximity sensors (HC-SR04)
- Jumper wires, breadboards, power supplies
- 3D printer access for chassis parts

Software Requirements

- Ubuntu Server 20.04 or Raspberry Pi OS
- Python 3.8+
- ROS Noetic
- OpenCV
- Various Python libraries

Step-by-Step Setup

1. Operating System Setup

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install essential packages
sudo apt install -y python3-pip git curl wget build-essential
sudo apt install -y python3-opencv python3-numpy python3-scipy
sudo apt install -y portaudio19-dev python3-pyaudio
sudo apt install -y espeak espeak-data libespeak-dev
```

2. ROS Noetic Installation

```
# Add ROS repository
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main"
```

```
> /etc/apt/sources.list.d/ros-latest.list'
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -

# Install ROS
sudo apt update
sudo apt install -y ros-noetic-desktop-full

# Initialize rosdep
sudo rosdep init
rosdep update

# Setup environment
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc

# Install additional ROS packages
sudo apt install -y python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool
sudo apt install -y ros-noetic-cv-bridge ros-noetic-image-transport
```

3. Python Dependencies

```
# Create virtual environment (recommended)
python3 -m venv ~/elephant_env
source ~/elephant_env/bin/activate

# Install Python packages
pip install --upgrade pip
pip install rospy rospkg
pip install opencv-python
pip install numpy scipy
pip install SpeechRecognition pyttsx3
pip install transformers torch torchvision torchaudio
pip install requests
pip install RPi.GPIO
pip install mfrc522
pip install pyserial

# For local LLM support
pip install ollama-python
```

4. Ollama Setup (Local LLM)

```
# Install Ollama
curl -fsSL https://ollama.ai/install.sh | sh

# Start Ollama service
sudo systemctl start ollama
```

```
sudo systemctl enable ollama

# Pull a small model for the Raspberry Pi
ollama pull llama3.2:3b

# Test the installation
ollama run llama3.2:3b "Hello, I am an AI elephant!"
```

5. Hardware Configuration

GPIO Setup

```
# Enable GPIO, Camera, I2C, SPI
sudo raspi-config
# Navigate to Interfacing Options and enable:
# - Camera
# - I2C
# - SPI
# - GPIO

# Reboot after changes
sudo reboot
```

Camera Setup

```
# Test camera
raspistill -o test.jpg
# If successful, camera is working

# Install camera dependencies
sudo apt install -y python3-picamera
```

Audio Setup

```
# Test microphone
arecord -l  # List audio devices
arecord -d 5 -f cd test.wav  # Record 5-second test

# Test speakers
aplay test.wav

# Configure audio for best quality
sudo nano /boot/config.txt
# Add: dtparam=audio=on
```

6. Project Directory Setup

```
# Create workspace
mkdir -p ~/elephant_ws/src
cd ~/elephant_ws/src

# Clone or create your project
mkdir ai_elephant
cd ai_elephant

# Copy the agent code
# (Copy the provided Python code to ai_elephant_agents.py)

# Create launch files directory
mkdir launch config scripts
```

7. ROS Package Structure

Create the following files:

package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>ai_elephant</name>
  <version>0.1.0</version>
  <description>AI Elephant Multi-Agent System</description>

  <maintainer email="your.email@example.com">Your Name</maintainer>
  <license>MIT</license>

  <buildtool_depend>catkin</buildtool_depend>

  <depend>rospy</depend>
  <depend>std_msgs</depend>
  <depend>sensor_msgs</depend>
  <depend>geometry_msgs</depend>
  <depend>cv_bridge</depend>
  <depend>image_transport</depend>

  <export>
  </export>
</package>
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.2)
project(ai_elephant)

find_package(catkin REQUIRED COMPONENTS
  rospy
  std_msgs
  sensor_msgs
  geometry_msgs
)

catkin_package()

catkin_install_python(PROGRAMS
  scripts/ai_elephant_agents.py
  DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
)
```

8. Launch Configuration

Create `launch/elephant_system.launch`:

```
<launch>
  <!-- Core ROS -->
  <node name="roscore" pkg="roscore" type="roscore" />

  <!-- Supervising Agent (Main Controller) -->
  <node name="supervising_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
        args="supervising" output="screen" />

  <!-- Vision Agent -->
  <node name="vision_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
        args="vision" output="screen" />

  <!-- Audio Agent -->
  <node name="audio_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
        args="audio" output="screen" />

  <!-- Movement Agent -->
  <node name="movement_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
        args="movement" output="screen" />

  <!-- Gesture Agent -->
  <node name="gesture_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
        args="gesture" output="screen" />

  <!-- Touch Agent -->
  <node name="touch_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
        args="touch" output="screen" />

  <!-- RFID Agent -->
  <node name="rfid_agent" pkg="ai_elephant" type="ai_elephant_agents.py"
```

```
        args="rfid" output="screen" />
</launch>
```

9. Hardware Wiring Guide

Servo Connections

```
Movement Servos:
- Front Left Leg:  GPIO 18
- Front Right Leg: GPIO 19
- Rear Left Leg:   GPIO 12
- Rear Right Leg:  GPIO 13

Gesture Servos:
- Trunk Vertical:     GPIO 20
- Trunk Horizontal:   GPIO 21
- Left Ear:          GPIO 16
- Right Ear:         GPIO 26
- Left Eye H:        GPIO 6
- Left Eye V:        GPIO 5
- Right Eye H:       GPIO 22
- Right Eye V:       GPIO 27
- Tail:              GPIO 17
```

Sensor Connections

```
Touch Sensors (FSRs):
- Trunk Tip: GPIO 23
- Head:      GPIO 24
- Left Side: GPIO 25
- Right Side: GPIO 8
- Back:      GPIO 7

RFID Reader (RC522):
- SDA:  GPIO 24
- SCK:  GPIO 23
- MOSI: GPIO 19
- MISO: GPIO 21
- IRQ:  Not connected
- GND:  Ground
- RST:  GPIO 22
- 3.3V: 3.3V

Camera Module:
- Connect to CSI port

Microphone:
- USB connection
```

10. Build and Test

```
# Build the workspace
cd ~/elephant_ws
catkin_make

# Source the workspace
echo "source ~/elephant_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc

# Make scripts executable
chmod +x ~/elephant_ws/src/ai_elephant/scripts/ai_elephant_agents.py

# Test individual agents
rosrun ai_elephant ai_elephant_agents.py supervising

# Launch full system
roslaunch ai_elephant elephant_system.launch
```

11. Configuration and Calibration

Servo Calibration

```
# Create calibration script
nano ~/elephant_ws/src/ai_elephant/scripts/servo_calibration.py
```

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
servo_pin = 18  # Change for each servo

GPIO.setup(servo_pin, GPIO.OUT)
servo = GPIO.PWM(servo_pin, 50)
servo.start(7.5)  # Start at neutral

try:
    while True:
        angle = float(input("Enter angle (0-180): "))
        duty_cycle = 2.5 + (angle / 180.0) * 10.0
        servo.ChangeDutyCycle(duty_cycle)
        time.sleep(0.5)
        servo.ChangeDutyCycle(0)
except KeyboardInterrupt:
    servo.stop()
    GPIO.cleanup()
```

Audio Calibration

```
# Test speech recognition
python3 -c "
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as source:
    print('Say something!')
    audio = r.listen(source)
    print('You said:', r.recognize_google(audio))
"

# Test text-to-speech
python3 -c "
import pyttsx3
engine = pyttsx3.init()
engine.say('Hello, I am your AI elephant!')
engine.runAndWait()
"
```

12. Troubleshooting

Common Issues and Solutions

GPIO Permission Issues:

```
sudo usermod -a -G gpio $USER
# Logout and login again
```

Camera Not Working:

```
# Check camera connection
vcgencmd get_camera

# Enable legacy camera support if needed
sudo raspi-config
# Advanced Options > GL Driver > Legacy
```

Audio Issues:

```
# Check audio devices
cat /proc/asound/cards

# Set default audio device
```

```
sudo nano /etc/asound.conf
# Add:
# pcm.!default {
#   type hw
#   card 1
# }
```

ROS Communication Issues:

```
# Check ROS master
echo $ROS_MASTER_URI

# List active nodes
rosnode list

# Check topic communication
rostopic list
rostopic echo /elephant/status
```

Servo Issues:

```
# Check power supply (servos need sufficient current)
# Ensure separate 5V supply for servos
# Check wiring connections
# Verify GPIO pin assignments
```

13. System Monitoring

Create monitoring script

```
nano ~/elephant_ws/src/ai_elephant/scripts/system_monitor.py
```

```
#!/usr/bin/env python3
import rospy
import psutil
import json
from std_msgs.msg import String

def system_monitor():
    rospy.init_node('system_monitor')
    pub = rospy.Publisher('/elephant/system_status', String, queue_size=10)

    rate = rospy.Rate(0.5)  # 0.5 Hz (every 2 seconds)
```

```
    while not rospy.is_shutdown():
        status = {
            'cpu_percent': psutil.cpu_percent(),
            'memory_percent': psutil.virtual_memory().percent,
            'temperature': psutil.sensors_temperatures().get('cpu_thermal', [{}])
[0].get('current', 0),
            'disk_usage': psutil.disk_usage('/').percent
        }

        pub.publish(json.dumps(status))
        rospy.loginfo(f"System Status: CPU {status['cpu_percent']}%, RAM
{status['memory_percent']}%, Temp {status['temperature']}°C")

        rate.sleep()

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

14. Auto-Start Configuration

Create systemd service for auto-start:

```
sudo nano /etc/systemd/system/ai-elephant.service
```

```
[Unit]
Description=AI Elephant Multi-Agent System
After=network.target

[Service]
Type=simple
User=pi
WorkingDirectory=/home/pi/elephant_ws
Environment=ROS_MASTER_URI=http://localhost:11311
ExecStart=/bin/bash -c "source /opt/ros/noetic/setup.bash && source
/home/pi/elephant_ws/devel/setup.bash && roslaunch ai_elephant
elephant_system.launch"
Restart=always
RestartSec=5

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

```
# Enable the service
sudo systemctl enable ai-elephant.service
sudo systemctl start ai-elephant.service
```

```
# Check status
sudo systemctl status ai-elephant.service
```

15. Development and Testing

Testing Individual Components

```
# Test vision
rosrun ai_elephant ai_elephant_agents.py vision

# Test audio
rosrun ai_elephant ai_elephant_agents.py audio

# Test movement
rosrun ai_elephant ai_elephant_agents.py movement

# Monitor ROS topics
rostopic echo /elephant/command_received
rostopic echo /elephant/mahout_detected
rostopic echo /elephant/movement_status
```

Debugging Tools

```
# ROS graph visualization
sudo apt install ros-noetic-rqt-graph
rosrun rqt_graph rqt_graph

# Monitor system resources
htop

# Check GPIO status
gpio readall

# Monitor log files
tail -f ~/.ros/log/latest/supervising_agent-*.log
```

Next Steps

1. **Physical Assembly**: 3D print chassis parts and assemble hardware
2. **Calibration**: Fine-tune servo positions and sensor thresholds
3. **Training**: Train face recognition for mahout identification
4. **Testing**: Comprehensive system testing with all components
5. **Optimization**: Performance tuning and reliability improvements

Safety Considerations

- Always use appropriate power supplies for servos
- Implement emergency stop mechanisms
- Test all movements in safe, controlled environment
- Monitor system temperature and resource usage
- Have backup power solutions for critical operations

Support and Maintenance

- Regular software updates
- Hardware inspection and maintenance
- Log analysis for performance optimization
- Backup configurations and trained models
- Document any modifications or improvements

This setup provides a solid foundation for your AI Elephant project. The modular architecture allows for easy expansion and modification as you develop additional features.