Automatic Dog Feeder : Software Task

# INTRODUCTION:

This document covers the deliverables for the software task for the Automatic dog feeder assessment task for Andromeda robotics. This document will cover the software system diagram, the software usage instructions, possible tests, considerations and assumptions and task requirements being met.

# SOFTWARE SYSTEM DIAGRAM:

ADF\_camera.py

(video recorder program)

ADF\_main.py

(central control program for the software system)

Sensor msg

Actuator msg

Recorded Videos

ADF\_sensor\_actuator.ino

RPi running Docker container

# RUN INSTRUCTIONS:

The first step is to setup Arduino Nano microcontroller by uploading the ADF\_sensor\_actuator.ino sketch using the Arduino IDE. After that connect the Arduino Nano and the Camera to the Raspberry Pi through its USB ports.

Then the docker image needs to be started up on the Raspberry Pi through its terminal with the proper command to provide the intended docker container the appropriate permissions to access the hardware I/Os and file access. For that, run

sudo docker run -v /dev:/dev /home/aditya/Videos:/root/recordings -it adf\_system

Once the docker container has started running, execute the following command to setup the overlay and underly environments for ROS2

source /opt/ros/iron/setup.bash

source ~/ros2\_ws/install/setup.bash

Once ROS2 environment is setup, execute the launch file to startup the ADF\_main and ADF\_camera nodes

ros2 launch adf\_py launch.py

This will startup the software system on the docker.

# TESTING:

In terms of testing, the following aspects of the system will need to be verified

* Individual components testing
* Reliability of ROS2 pub-sub communication system
* Reliability of Serial communication and packet reading on the Raspberry Pi side
* Effects of disconnection of peripherals at the Raspberry Pi
* Effects of disconnection between Arduino and Raspberry Pi
* Accuracy and minimum noise of sensors at the Arduino side
* Reliability of Servo actuators and their responsiveness
* Effects of running the complete software system for long periods of time

# ASSUMPTIONS AND CONSIDERATIONS:

In terms of assumptions, the majority assumptions were made on the electronics side of the complete ADF system. Due to the lack of the actual electronics diagram and microcontroller hardware allotted, an Arduino Nano was assumed to being utilized and the sensors and actuators were connected on assumed but appropriate pins on the Nano. Another assumption is the location of the Camera and Arduino ports in the /dev folder in the host Raspberry Pi. This can be resolved via static port directing for the specific peripheral hardware that will be used.

The main consideration being implemented here is that the main control program is fully in ROS2. The low level sensing and actuating task is offloaded to a companion microcontroller since it will have a quicker response to sensory changes. Another consideration is the presence of the necessary libraries – ROS2 core libraries, pyserial and opencv – in the host system(the Raspberry Pi). Another consideration is towards providing access to host file system to the docker container.

# CONCLUSION:

The developed software system is kept simple and yet meets the necessary requirement of the assessment task such as the main control logic being in ROS2, camera recording functionality in ROS2 and OpenCV and the ROS2 part of the software system being run in a docker. The Arduino sub-system also provides a low-latency embedded-level system that helps the overall system to react to real-world changes at high rate. Following these provided constraints has led to the development of a robust and highly mobile software that has potential for future feature expansion thanks to ROS2’s open source nature and wide developer community.