**COURSE: MOBILE ROBOTICS**

**ASSIGNMENT NO: 01**



**Submitted By**

**Muhammad Riaz**

**Regn Number**

**452759**

**Submitted To**

**Dr Yasar Ayaz**

**Department of Robotics and Artificial Intelligence**

**SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING**

**NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY**

**MOBILE ROBOT SELECTED FOR SIMULATION**

**ROBOT SELECTED :**

ROBOTONT- An Open-source and ROS-supported omnidirectional mobile robot for education and research .

**Harware Description and Working Principles:**

**Mechanical Assembly :**

The Mechanical Assembly of Robotont is shown in Figure1. The primary structure of the robot consists of a polycarbonate main chassis.

Inside the chassis are:

* Three wheel modules.’.
* Electronic circuit boards dedicated to various functions:
* Main microcontroller.
* Power management.
* Motor control.

Additionally, it houses the power supply an

d associated wiring.



**UGV Fleet Overview**

**A. Rover Hardware:**

Several criteria factored into the hardware selection for the individual rovers comprising the UGV fleet. First, the on-board computing resources needed to be easily programmable in order to deploy and run custom autonomous algorithms with varying sensor suites. Second, the vehicles were required to connect to a wireless network for communication with each other and a ground station operator. Finally, low level control algorithms for complex ground vehicles were not the intended use case for the fleet, so a simple and easy-to-control hardware solution was preferred. The GoPiGo3™ by Dexter Industries was chosen as the base platform for the UGV fleet. The GoPiGo3™ is a differential drive robot that uses a Raspberry Pi™ single board computer (SBC) to interface with its motors and other peripherals. The robot kit includes open source driver libraries in several programming languages allowing developers to create custom applications. Because the GoPiGo3™ kit includes motors with wheel encoders as well as the electronics needed to drive them, the only hardware changes to the nominal vehicle configuration were using a lithium polymer battery, attaching spherical Vicon™ motion tracking markers, and mounting a forward-facing Raspberry Pi™ camera. Twenty vehicles were constructed using this configuration shown in Fig. 1 with five reserved for spares and additional modifications to the sensor payload. To date, two alternate configurations have been created to satisfy significantly more demanding sensing requirements. Figure 2a shows a GoPiGo3™ outfitted with a Hokuyo™ lidar and VectorNav™ inertial measurement unit (IMU) to test sensor hardware interfaces and a state estimation pipeline before deploying them on a UAV. In Fig. 2b, an Intel Realsense™ depth camera and Phidgets™ IMU were mounted on the rover as a much less expensive sensor payload to explore visual-inertial localization and mapping. B. Command and Control Architecture The software for the rover fleet builds on the Autonomous Entity Operations Network (AEON) framework. As described in [3], the AEON framework facilitates software development for autonomous agents through a collection of utilities for writing concise, modular capabilities and connecting them through well-defined interfaces using the Data Distribution Service (DDS) [6] as the inter-process communication protocol. Figure 3 illustrates the core processes used to operate a single rover and how they instantiate the AEON design principles of modularity and layered functionality. Each rounded rectangle represents a standalone DDS-connected process (or node) and ovals represent hardware elements. The hardware interface layer exposes sensors and actuators to the DDS network, publishing data and receiving commands to and from higher layers.

**B. Command and Control Architecture**

The software for the rover fleet builds on the Autonomous Entity Operations Network (AEON) framework. As described in [3], the AEON framework facilitates software development for autonomous agents through a collection of utilities for writing concise, modular capabilities and connecting them through well-defined interfaces using the Data Distribution Service (DDS) [6] as the inter-process communication protocol. Figure 3 illustrates the core processes used to operate a single rover and how they instantiate the AEON design principles of modularity and layered functionality. Each rounded rectangle represents a standalone DDS-connected process (or node) and ovals represent hardware elements. The hardware interface layer exposes sensors and actuators to the DDS network, publishing data and receiving commands to and from higher layers.



**Software Utilities**

Prior to the UGV fleet, autonomous systems operations in the AISRB only utilized up to four simultaneously running agents. Simple tasks such as updating software or starting DDS nodes that used to be done manually quickly became tedious with twenty vehicles to manage. An existing utility script for deploying libraries and DDS nodes to a vehicle was upgraded to support multiple destinations. Another script allows users to quickly reboot or shut down many vehicles simultaneously. To alleviate the burden of typing dozens of commands to spin up demonstrations involving many agents, process management code from an existing project was broken out into a standalone node to start, stop, and query processes using messages sent over the DDS network. The process manager was installed on each vehicle and configured to start on boot, and a utility script was created to read extended markup language (XML) files containing process definitions and send them to the process manager to start or stop. In addition to the infrastructure automation, a graphical user interface (GUI) was created to test and debug the rover hardware interface and control layers shown in Fig. 3. The GUI, shown in Fig. 4, allows the user to select any vehicle connected to the DDS network, view and tweak controller parameters, send waypoint commands by clicking on the central plot, and view sensor data including vehicle pose (red chevron), lidar scans (black dots), and video.