

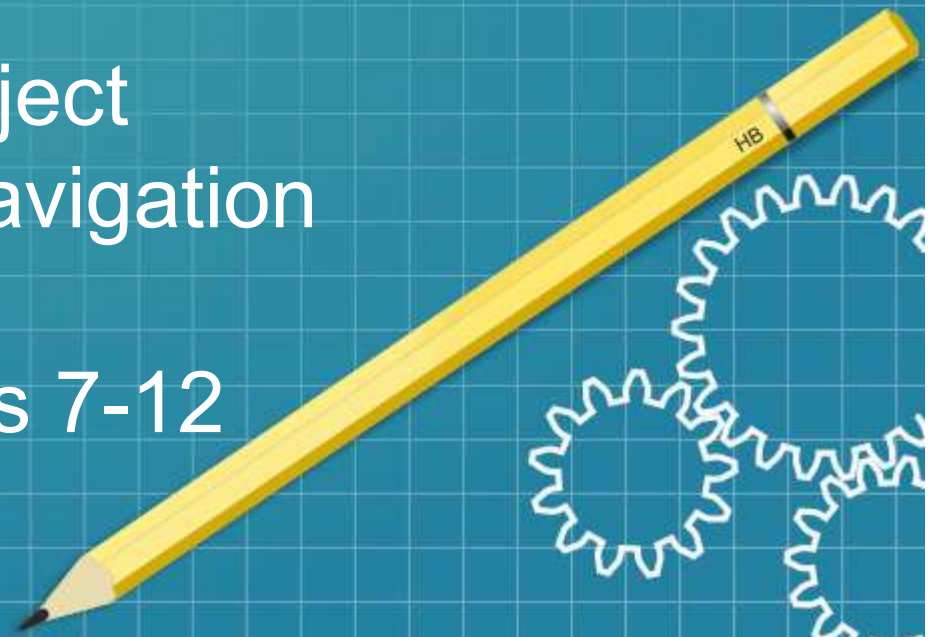


UWA
PERTH · AUSTRALIA

Mobile Robotics AUTO4508

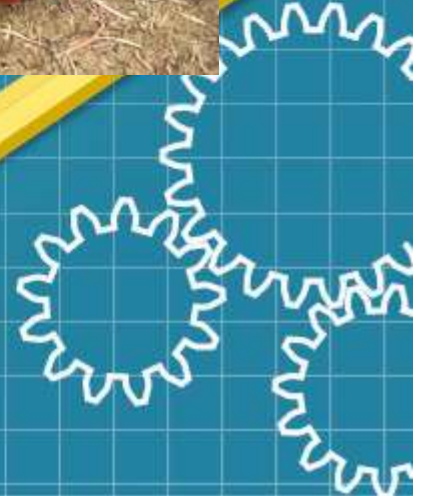
Group Project
Autonomous Navigation

2024 - Weeks 7-12



Overview

1. Equipment
2. Tasks to complete
3. General approach
4. Hardware
5. Operating system
6. Frameworks
7. Dev notes



Equipment



- Pioneer 3-AT Outdoor Mobile Robot Platform
- Industrial Linux PC with onboard screen
- GPS: not useful for this project
- IMU: Phidget Spatial 3/3/3
- Camera: Stereo Camera OAK-D V2
- Lidar: SICK TIMS7XXs
- Software: Ubuntu, ARIA or ROS

Tasks to complete 1/3



- Each team will be given a **number of GPS waypoints** that the robot has to visit, before **returning to its starting position**.
- Whenever a waypoint has been reached (within reasonable accuracy), the robot must **take a photo** of the marker object at the waypoint (e.g. an orange cone) and then head towards the next waypoint. Always leave this marker to **the robot's right side**.
- At each waypoint, an **additional object of interest**, e.g. a traffic sign or a large coloured bucket, will be displayed at a fixed height, but at an undetermined distance. Identify the object of interest and calculate its **distance from the waypoint marker**.
- Upon completion of the waypoint course, **print all marker photos and object distance** measurements on the screen.

Tasks to complete 2/3

- **Record the robot's driving path and display it** graphically on the robot's display with **all detected markers, objects of interest** and any obstacles.
- Implement a **user interface** with graphics and text on the robot's display that always displays the robot's **internal status and intended actions**.
- At all times, **avoid collisions** with markers, objects of interest and any other stationary or moving obstacles, such as walls, vehicles, people, bikes, etc.
- For safety reasons, implement a **Bluetooth link** between the robot's on-board PC and a gamepad controller:
 - Button 'A' enable automated mode. In automated mode, use the back pedals as a dead-man switch. If released, the robot has to stop.
 - Button 'B' enable manual mode (disable automated mode). In manual mode, the steering controls can be used to manually drive the robot forward/backward and left/right.

Tasks to complete 3/3



- **Project design report** (pdf), which includes
 - Report on which team member did what
 - Software design description/design choices
 - Diagrams, photos, screenshots, plots, etc.
 - Include page numbers
 - Max 10 pages plus 1 Title page
 - Do **NOT** include Program code, Table of contents, Half-empty pages, etc.
- **User Manual** (pdf)
 - Max 5 pages, no Title page
 - As if it was sold to a customer
- **Source code**, via email to lab demonstrator, clearly marking any imported code with referencing the source.

General approach 1/2



- This group project will require a **very substantial workload** within a short time-frame, through:
 - Catching up with required knowledge to program the robots (Ubuntu, Docker and ROS2).
 - Development of the code to perform the tasks.
 - Reporting and presentation.
- **It is not a competition**, team work within the group, and communication & collaboration on a general basis with other teams is encouraged and will prove necessary.
- **Do not reinvent the wheel !** Research the internet, as all the tasks are one way or another already documented.
- Ideal approach is to build out of open sources, and only code the last segments. You may not have the time to complete all tasks otherwise.
- **Stressed out work does not produce good and repeatable results.**

General approach 2/2



- Even the best plan never survives the first contact with the enemy... Be **adaptable**, prepare for contingencies !
- Make use of **all personnel resources** within your group. Communicate your availability and status to your team mates.
- Work at least in pairs for heavy/technical tasks, so the knowledge basis is not centralised.
- Do not let yourself or anyone slack down, and ensure the **workload is shared evenly**.
- Do not estimate that you are done when your personal task list is complete! Get involved in some other part of the team's project.
- **Keep time up your sleeve** for real life testing !

Hardware - Pioneer AT3 robot

- Pioneer 3-AT is a small four-wheel, four-motor skid-steer robot ideal for all-terrain operation or laboratory experimentation.
- The Pioneer 3-AT comes complete with one battery, emergency stop switch, wheel encoders and a micro-controller with ARCOS firmware.
- The robot is heavy (20kg), and needs to be lifted carefully (proper lifting techniques). Do not lift using the camera.
- Communications to the micro controller are done via serial port (RS232)



Hardware - PC



- The robot is mounted with an industrial grade PC outfitted with a 32GB of RAM.
- The hard drive is a Cactus 1 TB m.2 SSD.
- 2x USB3 and 4 usb 2.0, 2x serial ports, and other industrial connectors.
- WiFi adapter on board.

Hardware – OAK-D Camera

- 4 TOPS of processing power (1.4 TOPS for AI)
- Run any AI model, even custom architected/built ones
- Encoding: H.264, H.265, MJPEG - 4K/30FPS, 1080P/60FPS
- Computer vision: warp/dewarp, resize, crop, edge detection, feature tracking
- Stereo depth perception at 120 FPS with filtering
- Object tracking: 2D and 3D tracking



Hardware – Bluetooth controller

- PC has onboard Bluetooth card.
- The controller is DualShock IV clone.



Hardware – LIDAR



- Sick LIDAR TiM7xx, connected via ethernet cable with set ip address 192.168.0.1.



Operating systems

- The operating system **in use on the robots is Ubuntu 22.04 LTS**, although any other OS on the ground control station can be used.
- [Ubuntu command line tutorials](#)



Frameworks - Intro



- There are two software frameworks to use for the robot.
 - **Aria** – Library used to control the Pioneer AT3 robot (the basic drive commands are done for you).
 - **ROS2** – Industry standard robotics framework.

Any programming language can be used but you will get the most help in python or c++



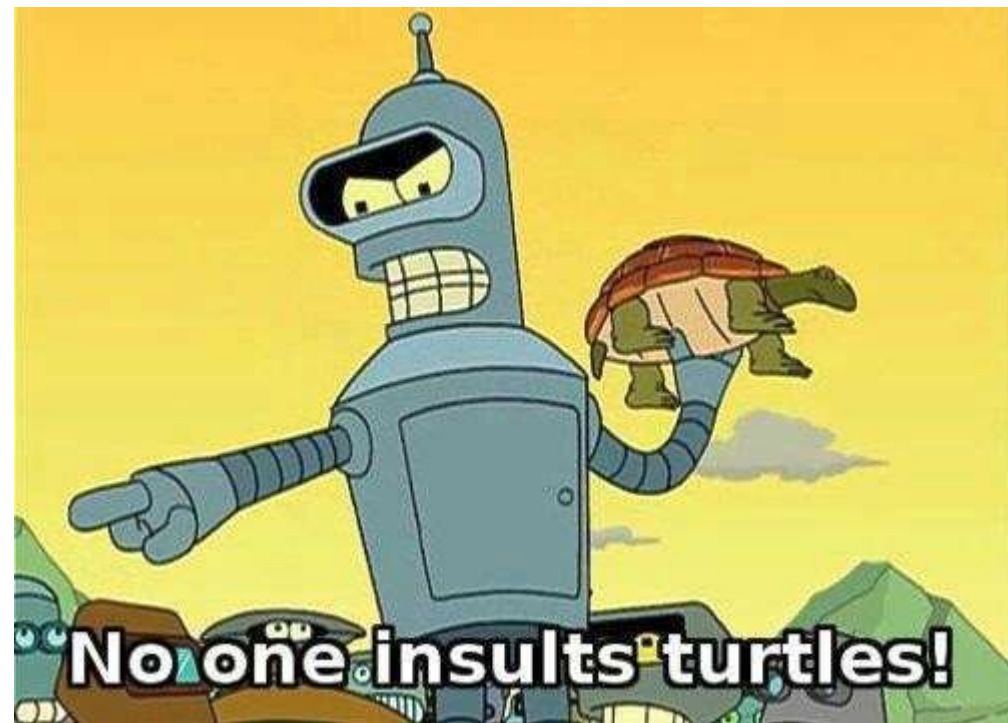
Frameworks - Aria



- **AriaCoda** is a C++ library used to **communicate with Pioneer** mobile robot controllers, many sensors and other accessory devices, and **includes various useful tools** for mobile robotics applications.
- Much of the library is mostly C++, but has C, Python and Java wrappers.
- AriaCoda is based on the deprecated ARIA library.
- Link: <https://github.com/reedhedges/AriaCoda>

Frameworks – ROS2

- **ROS (Robot Operating System)** provides libraries and tools to help software developers create robot applications.
- It provides hardware abstraction, device drivers, libraries, visualisers, message-passing, package management, and more.
- It will take a **steep learning curve**, but once familiar with the concepts, ROS may provide most of the required functionalities almost **out of the box**.



Dev notes - Ubuntu



- Ubuntu 22.04 on all bots, with all the build essential and compile support
- SSH server installed so possible to SSH, RDP or SFTP into the machine remotely.
- users have been already added to the dialout group, and does not require to sudo into the various ports.

Dev notes – Aria barebone



- The official Aria lib is deprecated and the compiling of ArNetworking was causing issues as properly obsolete.
- Using AriaCoda, which is a lightweight open source continuation of Aria, but fully functional.
- <https://github.com/reedhedges/AriaCoda.git>
- **Read the readme.md file !**
- All examples compiled directly with make and operational (teleop drive, battery monitor encoders, GPS, and co, etc).
- The examples compiled binaries are in the /Aria/examples folder, and the CLIs are listed in the devnotes.
- A couple of CLIs:
- `sudo ./teleopActionsExample -rp /dev/ttyUSB0`
- `sudo ./gpsExample -gpsPort /dev/ttyACM0 -rp /dev/ttys1`


Dev notes – DepthAi barebone

- **DepthAi** required to use the depth camera
- The install process mostly used the following tutorial:
- <https://github.com/luxonis/depthai>
- Python demo and examples in the subdirs. All working, again a couple of CLIs in the devnotes
- `python3 depthai_demo.py -gt cv`

- **WARNING !**
- As part of the install process, the procedure requires the following CLI:
- `python3 install_requirements.py`
- However it does install PyQt5 binding packages that clash with the ROS rqt visualiser.
- The following packages may require to be removed after the install of the prerequisites:
- `pip uninstall PyQt5`
- `pip uninstall PyQt5-sip`
- `pip uninstall PyQtWebEngine`



Dev notes – RosAria



- RosAria was developed for ROS1:
- http://wiki.ros.org/ROSARIA/Tutorials/How_to_use_ROSARIA
- If you wish to add more functionality or adapt the aria code you can but you will need to rewrite the code for ROS2.
- The node compiles without issue, providing Aria or AriaCoda is already available system wide.
- CLIs for starting rosaria and teleop node :
- `ros2 run ariaNode -rp /dev/ttyUSB0`
- There is a series of ROS parameters that will need to be passed to ensure that the odometry is configured properly.

Dev notes – depthai-ros

- depthai-ros compiles in a colcon workspace, following the tutorial:
- <https://github.com/luxonis/depthai-ros>
- Can use `-j2` flag to reduce the compile time load.
- `export ROS_PARALLEL_JOBS=-j2`
- CLIs for testing (ROS launch file) with rviz:
- `ros2 launch depthai_examples stereo_node.launch`
- Note the amount of RAM used (system monitor), close to 1GB for rviz alone, and only with a point cloud. Rviz would be better off running on the ground control station, to avoid depleting the robot's RAM and swap.
- Note that the demo nodes publishes already an attitude.

Dev notes – ROS GPS

- Navsat node installed directly from deb packages:
- `sudo apt-get install ros-noetic-nmea-navsat-driver`
- The node is started as follows:
- `roslaunch nmea_navsat_driver nmea_serial_driver _port:=/dev/ttyACM0 _baud:=9600`
-

This node is not the only option to publish the GPS data. If more in depth parametric and accessibility for other programs are required, the preferred method would then be to use a `gpsd` daemon and a corresponding `ros-gpsd` node.



RoadMap



Subjects to explore in priority over the next week:

- Bluetooth pairing of the game-pads
- joy node: <https://index.ros.org/p/joy/>
- teleop_twist_joy: https://index.ros.org/p/teleop_twist_joy/github-ros2-teleop_twist_joy/
- Phidgets IMU node: http://wiki.ros.org/phidgets_imu
- Transforms broadcasters for the sensors: <http://wiki.ros.org/tf>
- Nav stack setup: <http://wiki.ros.org/navigation/Tutorials/RobotSetup>
- ROS functionalities such as ros services

All the above can be supplemented by a good dose of tutorials on Youtube. Run your code at home using the simulation setup in gazebo, and get familiarised with the general navigation concepts !

Good Luck !



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. It makes use of the works of Mateus Machado Luna.

