dear applicant

This document contains the work-level-agreement and the assignments to fulfill by the candidates of eteration r&d team (*see composiv.ai and eclipse-muto*).

To proceed further, please:

checkout the repository.
read and understand fully the content of this document.
reply the contact that has sent you this document with an estimated time of submission.
follow the tasks described in the further sections of this document.
submit your work as instructed in the relevant section.

In case of an emergency or any questions please contact your recruiter or the person in charge of this document:

name: deniz memis
email: deniz.memis@eteration.com

values

Writing well tested clean code that WORKS
Writing well understood documentation that JUSTIFIES the work done.
Being an ACTIVE contributor to feedback and/or review cycle.
Asking RELEVANT questions to eliminate uncertainties.
Learning in an enviroment with ESTABLISHED culture.
Nurturing the culture by COMMITTING time and effort.

checkboxes

The following content in the next two sub-sections below will serve you a great deal of benefit in your interview and work process. They are better to be embraced at the very beginning. Consider them as bonuses that will give you a head start. Please note down how many functional boxes are checked. Do not hesitate to reach out in case of any gray areas in any of them.

non-functionals:

Ability to read/write/think/understand in english (initially, the level of this is irrelevant. however the more is the better)

Ability to be physically present at the office when required (the consistency matters) Ability to get your way out in an Linux environment (ideally this was already your OS of choice to begin with).

Ability to understand a logical set of instructions (whether it is a tutorial or an article with academic foundations).

Ability to ask questions without hesitation or any long/overdue personal over-thinking (initially, the quality of the questions is irrelevant).

Ability to work in a transparent manner (audacity to say 'this is not done' or 'i did not do it' instead of hiding and ghosting in a dark room)

functionals:

Having a development environment on top of a linux distro, using an Ubuntu LTS: 18.04 or 20.04 (#general)

Beginner/intermediate skills of scripting in Python (preferable) or C++ (#ros, #general).

Ability to use command line interface (a terminal) in a *nix based environment (#general)

Ability to contribute and to collaborate with others via Git (#general)

Ability to read and debug a code segment (#general)

Basic understanding of web and web technologies (#general)

Basic understanding of robots and/or robotics (#ros)

Having a basic understanding of communication in a distributed architecture (#ros).

Having a basic understanding of how a pub-sub information flow works (#ros).

Having a basic understanding of OOP (#general, #ros).

*#ros: a label to indicate that it is a ROS related requirement *#general: a label to indicate that it is a general software related requirement*

preliminary

checkpoint ZERO

Please read and understand this document in detail, gave feedback to the person you are in contact with in the interview process.

checkpoint I

Initialize a new branch within this repository, entertaining the following format

feature/${your name}_${your surname}_${start date}

some examples, date is in DDMMYYYY format:

feature/john_doe_18102022

feature/jane_doe_01112022

ALL the work done will be submitted to this branch only.

checkpoint II

Read the official introduction to ROS.

Choose and install a ROS distro suitable for your system:

ROS Melodic Morenia (LTS) supported until May 2023, recommended for Ubuntu 18.04 ROS Noetic Ninjemys (latest LTS) supported until May 2025, recommended for Ubuntu 20.04

checkpoint III

Configure ROS environment (i.e. create a workspace, a catkin_ws if you like)

This is your ROS development workspace.

Notice that a ROS workspace has a /src folder where your task implementations will live as described in the next section.

Each and every task will be unoriginal, meaning that there are a lot samples and examplary implementations on the web.

Please take the maximum advantage of official ROS tutorials and your web search skills.

checkpoint IV

Write unit tests to your implementation.

tasks

checkpoint V

Within your workspace, create a ROS package with the following format:

composiv_tryouts

checkpoint VI

In your package, implement a publisher and subscriber:

composiv_talker

composiv_listener

checkpoint VII

In your workspace, create a launch folder and implement a launch file that will run the ROS executables you have implemented:

composiv_tryout.launch

report

checkpoint VIII

Document the work you have done by respecting the following attributes/qualities to your documentation:

It should have an executive summary for executives or any other non-technical stakeholders

It should be readable and understandable by anyone (up to a certain point at least). It should be a crystal clear description of technicalities, methods used or implemented for an onboarding developer who wants to see what is in the project so far.

It should be in a version controllable manner, meaning that the documentation could be improved or updated in iterations (choose your format wisely)

It should have references in case you make use of someone else's code base in any way. It should better have some illustrations (a diagram, a drawing or a sketch) of how things work. (a picture means a thousand words)

commit

checkpoint IX

Commit all your work to your branch and push it to remote server.

appendix

checkpoint ZERO: You have NOW started this assignment.
checkpoint I: You have NOW successfuly used Git to collaborate on a project (i.e. checking out an existing project, creating your own version where you will share your own features/work-done with collaborators)
checkpoint II: You have NOW successfuly read an online documentation and followed the installation instructions and now you have all the dependencies installed to start off with your own work.
checkpoint III: You have NOW successfuly created a ROS workspace (an overlay in ROS terms) and ready to develop your custom ROS nodes and services, kept following the set of instructions and worked your way out by googling/reading in case of any errors in the console.
checkpoint IV: You have NOW well-tested code base, where you could observe each unit works as expected or any breaking changes by a failing test before deployment.
checkpoint V: You have NOW a ros package ready to encapsulate your ROS nodes with different business logics, that can be executed by speciying the package name with the executable name.
checkpoint VI: You have NOW ROS nodes, one with a publisher (it sends its messages to a certaing topic) and another one with a subscriber (it receives messages from the previously mentioned topic)
checkpoint VII: You have NOW a launching mechanism that runs the ROS features you provide from a single point, instead of running each node individually.
checkpoint VIII: You have NOW a documentation ready to describe your work done. Congratz 😃
checkpoint IX: You have NOW shared your codebase belonging to your developed features in a collaboration environment.