Justification document

Thomas Schenk

Contents

[Introduction 4](#_Toc136293040)

[1. Integration of a new camera 5](#_Toc136293041)

[Context 5](#_Toc136293042)

[Problem/opportunity 5](#_Toc136293043)

[Process 5](#_Toc136293044)

[Validation 5](#_Toc136293045)

[2. Project Plan Personal project 6](#_Toc136293046)

[Context 6](#_Toc136293047)

[Problem/Opportunity 6](#_Toc136293048)

[Goal 6](#_Toc136293049)

[Process 6](#_Toc136293050)

[Validation 6](#_Toc136293051)

[3. Integration process of new Lidar 7](#_Toc136293052)

[Foreword 7](#_Toc136293053)

[Context 7](#_Toc136293054)

[Problem 7](#_Toc136293055)

[Goal 7](#_Toc136293056)

[Process 7](#_Toc136293057)

[Final validation 7](#_Toc136293058)

[4. System Integration of Object Detection 8](#_Toc136293059)

[Context 8](#_Toc136293060)

[Problem 8](#_Toc136293061)

[Goal 8](#_Toc136293062)

[Process 8](#_Toc136293063)

[Validation 8](#_Toc136293064)

[5. Creating GUI 10](#_Toc136293065)

[Context 10](#_Toc136293066)

[Problem 10](#_Toc136293067)

[Goal 10](#_Toc136293068)

[Process 10](#_Toc136293069)

[Validation 11](#_Toc136293070)

[6. Wiki entry of GUI 12](#_Toc136293071)

[Context 12](#_Toc136293072)

[Problem 12](#_Toc136293073)

[Goal 12](#_Toc136293074)

[Process 12](#_Toc136293075)

[Validation 12](#_Toc136293076)

[7. Presentation Personal Project 13](#_Toc136293077)

[Context 13](#_Toc136293078)

[Problem 13](#_Toc136293079)

[Goal 13](#_Toc136293080)

[Process 13](#_Toc136293081)

[Validation 13](#_Toc136293082)

## Introduction

In this document you will find the justification of each piece of work done by me. Some parts are already done, while some problems will reiterate a few times throughout the sprint, depending on their size.

## Integration of a new camera

### Context

We would like to make the JACKAL more aware of its surroundings, while simultaneously make it easier for the end user to see what the robot sees. Some data is interpretable way quicker for a computer than a human, for example highly noisy point clouds. This is currently solved by a camera at the front of the robot, in its current configuration it is responsible for 3D scanning; as well as deliver a camera feed to the end user if desired.

### Problem/opportunity

The current problem is that the camera feed generated by the camera is very slow/choppy. This feed is not reliable at all for first person driving or accurate mapping. Let alone being able to monitor or recognize anything going on when far away from the robot. Furthermore the PO gave us a requirement which requires the robot to be able to see in the dark, the current camera did not support IR camera modes and therefore does not honour the requirements.

### Process

First, I made sure the camera worked according to the factory standards. I did this by installing the manufacturers software and checking all output feeds. After this, I researched how the current camera on the jackal actually talks to the rest of the system. This would be my beginning point of integration, since the system should start to listen to a new camera.

When I was done with this, I made sure the camera also worked in the ROS configuration on the robot; by running it locally on my pc with the same settings. The feed in this preliminary testing was very good and clear, so I was quite opportunistic to integrate it.

After this, I integrated the camera in the system. First of course all the required drivers, then I made the new camera talk to the existing system. Because I had first figured out how the previous camera worked, this was a very smooth job to do. I had to alter a few lines of code in setup files to accommodate for the new drivers.

### Validation

To validate this outcome, we can simply view the infrared camera stream and see if it outputs something which is not a black screen. This happened to be the case. The requirement was fulfilled.

We also wanted to fix the latency issues. To test the speed correctly, I needed to recreate the scenario like it will be when the end users use this product. Which is remotely via some sort of data tunnel. In my case, SSH is the option I went with. I tested the speed of the feed by having colleagues walk and perform quick jerky motions in front of the robot. If I could distinguish in a reasonable reaction time if there was a person or what they were doing; the speed would be adequate. This happened to NOT be the case. Sadly, the choppy feed was not due to the internal speed of the camera.

When testing the robot for its primary application, driving from first person view. The feed sadly wasn’t any better. Though it was actually doable to drive it, with little pauses in between it was no problem to navigate through the building.

## Project Plan Personal project

### Context

For my Personal Project I need to keep track of when I am going to do what and what my steps are. This way I can better plan the project, as well as project it to others a lot better.

### Problem/Opportunity

With this projectplan I could improve the way I manage and present my project better to others.

### Goal

Have a project plan which lists the scope, steps needed, the planning and a risk analysis.

### Process

I sat down and made sure to write down everything I knew about the project on a piece of paper. Then I slowly made it into separate boxes so I could plan and phase it better. Then it was a matter of making it into a project plan.

### Validation

I think the only possible validation for this is feedback from the teacher. Furthermore a good validation can be at the very end of the project. Did the project plan fit the phasing of the project? The better this fits, the better the project plan was.

## Integration process of new Lidar

### Foreword

This document is a collection of multiple smaller “products” I previously produced. However, after some time it became apparent that these products were all very similar to each other; the problem remained the exact same. Therefore I decided to group all of these into one document. Not only to increase the validity and testing capabilities of my work, but also to give the next group a more efficient carryover in the Lidar department.

### Context

The Jackal uses lidar data to map its surroundings. Together with RTAB-map, it can create a 2d map of the environment around this. The quality of this map is usually dependent on a few factors, most notably: the lidar used and its quality. The project group received a lidar which is far more accurate and powerful than its previous one.

### Problem

The current lidar is old and inferior to the newer lidar, it needs to be changed out. However, a lot of processes currently working perfectly fine are dependent on the lidar. So not only does the new lidar need to be integrated, changing the lidar to a new one means all these processes need to be reconfigured.

### Goal

Integrate the new lidar so that all processes work seamlessly with the new lidar, however dependant these processes may currently be of the old lidar.

### Process

Starting with basic installation of the required packages, I installed everything to work flawlessly in a test setup.

After that, porting it to the robot brought a few problems along the way. Firstly a problem regarding the launch files, which luckily was swiftly resolved.

After that, a much deeper problem concerning the local costmap came up; which after a lot of research I was finally able to solve.

## Final validation

Combining all sub validations and their problems into one validation, we let the jackal move around and go about its day with the new lidar. To our delight, no new hickups or problems occurred during this event. Effectively proving that the new integration was a succes.

## System Integration of Object Detection

### Context

To determine points of interests (POI) on the map, the JACKAL makes use of the installed camera to detect objects of interest. It does this using a recognition model using python. The operator can see these points of interest on the map and inform others around him.

### Problem

While the code for object detection is already present, the integration with the exploration package is very poor/nonexistant. This integration essentially contains a launch file which launches everything at once. The JACKAL currently does not have enough processing power to accommodate for both these high intensity programs at once. This causes the JACKAL to lag tremendously. Indirectly this causes both the exploration and the object detection processes to not perform, or perform very poorly.

### Goal

Create a different system architecture where computational loads are divided or made more efficient in order to not impede upon the quality of already well performing processes. After this has been established, make sure the object detection can be ran parallel to the exploration process in this new architecture.

### Process

To start this problem off, I first looked carefully at the problem; and what its final solution should contain. I considered ease of use, technical complexity but also ofcourse: what is this solution going to mean for the load on my robot?

After carefull consideration, I determined that the best way to split up computational loads; is to introduce a new computational device to the already setup ROS network. Which is setup by the JACKAL. This brought a new question to the table: How?

Again, after carefull consideration of how to best connect these now two devices; I settled on a physical link between the master (robot) and the slave (extra device). I made a new, more inherent to its name, script to quickly connect these two devices.

After this, I needed to run the object detection on the extra device; and the exploration package on the robot. Which proved to work near flawlessly; with the only exception to this being a bug in the part which marks the found object on the map.

### Validation

Validation for this problem can be split into two categories.

– New architecture  
To validate if the new architecture works, or in other words: are these two ROS systems connected?

We can simply run a ROS command which enables us to deduct this. This command is *rostopic list,* which lists all the topics a ROS machine can see. If we run this on the extra device and see topics such as: /scan, /move\_base or /velodyne; we know for sure these two systems are connected. Since these topics are ONLY published and made by the robot. Viewing them on our extra device, means we got a connection!

– Object Detection  
With the absence of the marker previously mentioned, how can we possibly determine that objects are detected?   
Luckily, this object detection does not only show its succes (or failure) by placing markers. Every object that is detected comes with a picture of that specific object. On this picture the model draws a border around the object it has detected. These are stored in a folder on the host machine, which we can check. If there are pictures in there; we know it works!

## Creating GUI

### Context

The Jackal is a product that will be used in high intensity situations, or very low intensity situations. Either way, it needs to have an intuitive and clear user interface. The operators must be able to react quickly on changing scenarios, without having to have technical knowledge of the product. Without such simplicity in its controls, its functionalities will be forever lost in the complexity of separate terminals and different visualization programs.

### Problem

The Jackal currently has no 1 place where the end user can both see and control the data on the Jackal. To be effectively rolled out as a product; it needs this. The operator in the field cannot be expected to carry a laptop and juggle around CLI (Command Line Interface, otherwise known as Terminal) windows or other programs currently needed for executing certain behavioural actions on the Jackal.

In short, the end user has too much responsibility when it comes to managing and performing the necessities for controlling and monitoring the robot.

### Goal

Something needs to be created to eliminate the need for multiple instances of programs, in whichever way shape or form, and which serves the end user with one concise GUI. The form of this product is very free, yet it must have one specific attribute: it must be able to do everything on one page/in one instance.

We need to take the responsibility of managing the system away from the user and expect our system to manage itself.

### Process

The first task was figuring out what type of program would be used to display everything. While an application on a desktop (like how you would launch Word) would ultimately be the most professional. We opted for a web based approach. Most notably for its already existing (albeit very vague) ability to interface with remote Ros environments. Furthermore, web applications are easy to port into standalone applications; making this work non-redundant if in the future a port like that is preferred.

After this, implementing Ros web connectivity was achieved by using ‘rosbridge’. A service which provides a WebSocket. Since our machine uses a VPN, this was a bit of a different setup than usual; but it proved to work.

After creating a websocket, specialized javascript libraries were used in order to create visualizations and user interactions.

### Validation

Validation for this product is very easy, it has a set amount of requirements; all of which can be individually met or not. In our case, the current requirements are met.

1, Live video feed  
2, Live Map feed  
3, Live Exploration control  
4, Live controls

A computer screen shot of a box

Description automatically generated with low confidence

As seen above, this is a screenshot during the development process where we can see both the live map, the live camera feed and the buttons to control some processes.

## Wiki entry of GUI

### Context

The GUI took a lot of work and googling to make any sense of the vague documentation in place for various javascript libraries used. Not only are some riddled with errors, but general knowledge about these topics also online seems to be few at best.

### Problem

The next group has absolutely no idea how the GUI works or how its structured currently. To avoid them having to research all the same (near useless) forums and websites I did; there needs to be a carryover to save them this hassle.

### Goal

Create a carryover which calls forth all the most important parts of the GUI and how that is structured.

### Process

Whilst already having the knowledge of the GUI, writing a wiki article about it was fairly straightforward. I first mapped what I wanted to tell on a piece of paper, then I wrote it out in one sitting.

### Validation

All my group members read the file I provided on the wiki, whilst having no knowledge of the GUI. After reading they were all quite up to speed with this GUI and could recite some basic workings of it needed to continue to build upon it.

## Presentation Personal Project

### Context

For the entirety of the project, we have been working on a personal project on the Mondays. A lot of work has been done for that personal project and in my case it is about an engine controller.

### Problem

Everyone around me is not up to speed with my work on the engine controller and does not know what I have done.

### Goal

Inform everyone around me about my personal project in a way that sticks to them and tells them methodically what I have done.

### Process

By condensing everything I know about my own project into a powerpoint, I perfected it over the course of a few days making the final version. Simultaneously, I prepared a physical demo of my product to showcase.

### Validation

To validate that my goal was met, I spoke to my teacher about the quality of my presentation as well as how he perceived it. Both answers were conform to the goal I had set.