

**CG1112 Engineering Principles and Practice**

Semester 2 2018/2019

**“Alex to the Rescue”**

**Design Report**

**Team: 02-01-01**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student # | Sub-Team | Role |
| HONG WEI YANG | A0182619X | Software | Lidar implementation |
| JIANG HONG QUAN | A0190306M | Hardware | Circuit design |
| KHAIRUL IMAN NASRI B AZHARIE | A0182359W | Software | Robot Movement |
| LEOW YONG KANG ANDREW | A0188376L | Hardware | Robot Calibration |
| NG WEE KIAT | A0182496R | Software | Color Detection |

**Main Aim & Design Ideas**

Our main goal is to use Alex to emulate the functionality of a Search and Rescue robot, where it can find its location in a foreign environment, map the area and subsequently be able to escape from the area. Alex would have to be remotely operated, hence in our design, we have implemented various mappings and communication paths so that the user would have enough information to control Alex based on just the mapping data provided back to the user. Our report below highlights our findings and possible implementations.

**Section 1 System Functionalities**

Alex is a remotely operated mobile mapping and detection robot. The operator utilizes a Secure Shell protocol (SSH) via laptop to access and control the Pi’s terminal over the network.

Alex contains a LIDAR unit which can perform 360 degrees environment scans within a 6 meters range. Information from the LIDAR is used to generate a 360 degrees 2D map which will be relayed to the operator throughout the operation.

The operator navigates Alex through the simulated environment manually using the generated map. Constant communication is engaged with the master control program (MCP) on Pi to translate commands into actual movement control signals for the connected Arduino board. These commands include speed control, distance of travel, turning angle and compass direction.

Alex will be able to detect objects scattered throughout the environment and determine their color (either red or green). Upon detection, information would be relayed to the operator. Subsequently, the operator could send an “identify object” command to Alex via MCP, for Alex to identify the color of the object (red or green).

The power consumption of the entire system including Arduino, Pi and Lidar is monitored by the operator. The operator should also able to selectively reduce the powe r consumption for each component to ensure that there is enough battery life to complete the tasks.

**Section 2 Review of State of the Art**

**ROBOCUE (Rescue Robot)**

Robocue is a search-and-rescue robot developed by Tokyo Fire Department in 2008, sized at 3.98×1.74×1.89m and weighs 3,860kg. It can be remote-controlled by the operator and used as far as 100m away. It is also tethered by a 328-foot cable and equipped with infrared cameras, a megaphone, and ultrasonic sensors that find victims in dangerous places where human rescuers cannot risk going, such as burning houses.

Upon locating the victims, it can pull victims inside its body via a conveyor belt. It has an onboard oxygen canister for those who might need it. After that, the person in question can be driven to a safe area.

**Main strengths**: Active scope camera coated with plastic cilia that can slither through crevices and find victims in disaster-stricken areas filled with rubble, the repulsion generated by the hair-like cilia structures and its surroundings allow the tube to move unobstructed through debris and rubble.

**Main weaknesses**: Due to its bulkiness, victims hidden within the rubble are unable to be saved immediately and could only be discovered using the scope camera, active personnel still have to tread the dangerous grounds to rescue the victims hidden deep within the rubbles personally. Additionally, if the victims are stuck underneath the rubble and cannot be extracted, the Robocue cannot pull the victims inside is body to keep them safe.



[1] Picture of the active scope camera [2] Structure of Robocue.

coated with plastic cilia enabling

ease of movement.

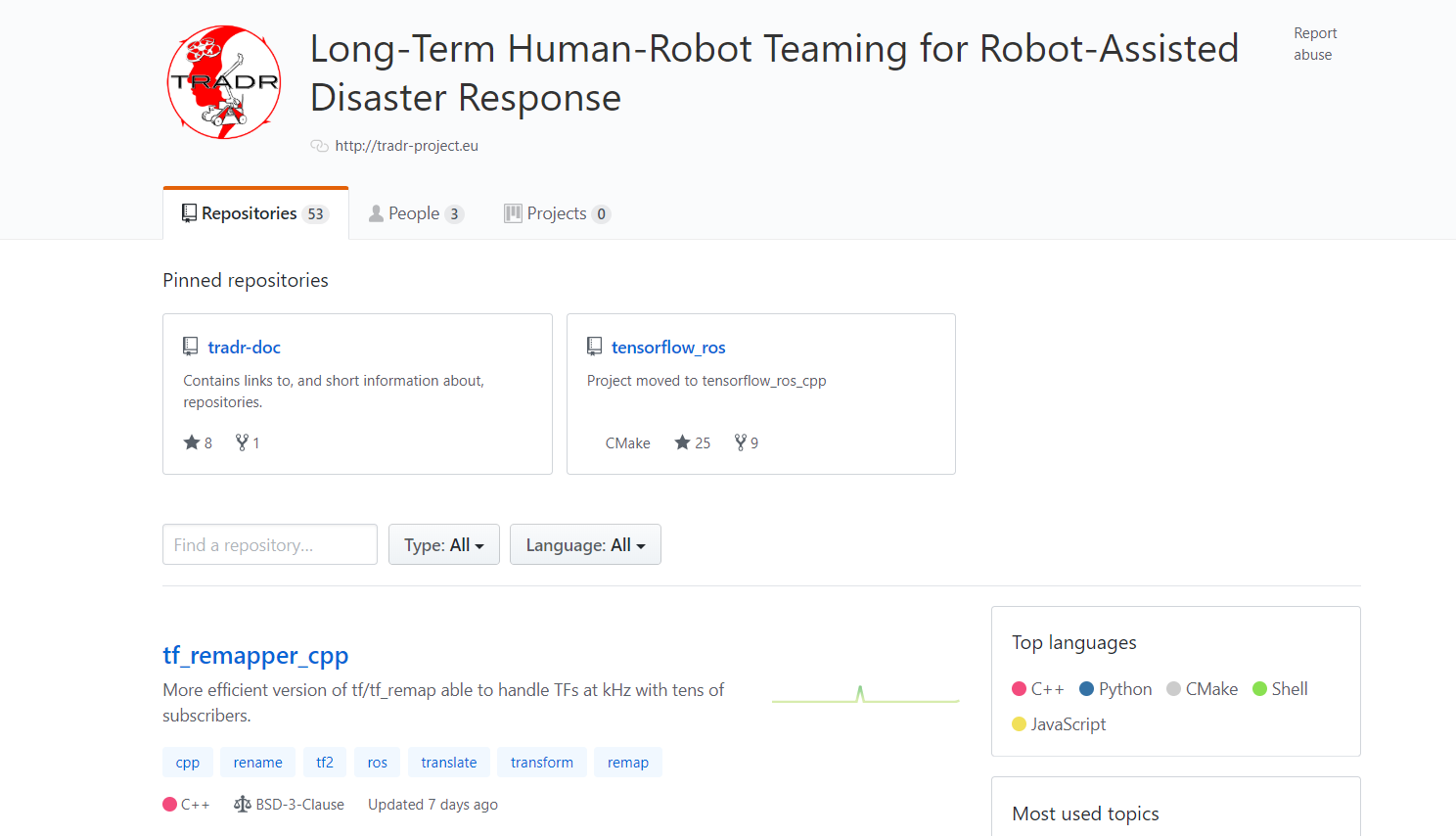
**TRADR Project**

The TRADR project develops cutting-edge software technology for human-robot teams to assist in urban search and rescue disaster response efforts, where duration of the mission would take several days or weeks. TRADR excels in human-robot teaming whereby the team

gradually develop its understanding of the disaster area over multiple possible asynchronous missions to improve the team’s understanding of the area via the robot’s learned model of the environment. The TRADR project releases several open source software libraries that were incorporated in their search and rescue robots/drones. Within their GitHub repository, we can find features for remapping environment, allowing for robot localization and semantics extraction called SegMap. Most notably, there’s a teleoperation package for Free Look Control that could be used with an Xbox controller for controlling the rescue robot over the traditional Tank Control.

**Main strengths:** Open-source libraries allow new features to be added by developers from all around the world and allow bugs to be fixed faster, which could be crucial in missions.

**Main weaknesses:** Emergency operations are usually not suited for TRADR since time is required to train the robot models to a state where it could be of use to the search and rescue team members.



[3] Picture of an autonomous rescue robot [4] Picture of the TRADR open-source

using TRADR software repository on GitHub

**Section 3 System Architecture**

The LIDAR would be connected directly to the Pi, as the sensor must create an accurate live plot of data of the for the Pi to process using SLAM algorithm.

The Motors would be connected to the Arduino. The Arduino would be in serial communications with the Pi. This movement would be transmitted after the SLAM algorithm determines the environment of the surroundings as well as the current location of Alex. Once the Arduino finishes executing the command packets sent by Pi, it would send an acknowledgement packet to allow for receival of the next command. The user will be the one sending command packets from the Pi through a remote connection from their laptops to the Arduino.

Color detection could be implemented using LDR sensors connected to Arduino. Arduino fetches RGB data constantly to Pi and upon detection of specific RGB values that signals a color has been found, an interrupt would be sent to Arduino and the information would be transferred to Pi via serial communication, allowing us to decide Alex’s next movement command based on the obstacle location.

Reduction of power consumption can be accomplished by reducing the components active on the Pi when they are not in use, as well as utilizing interrupts instead of polling to reduce the constant activity on the CPU.

The diagram below serves to illustrate the connections between the Pi, Arduino and sensors.

**A picture containing screenshot

Description generated with high confidence**

**Section 4 Component Design**

High Level Steps:

1. Initialization
2. User Movement Command
3. Movement Command Execution
4. Repeat parts 2 and 3 until user has detected escape route from mapped data

Part 1: Initialization

1. Instantiate Master program through selected robotics operating system (ROS).
2. Run LIDAR scan continuously, generating converted pixel plot data which updates every 3 seconds.
3. Using SLAM algorithm, identify fixed surroundings and current location of Alex relative to the surroundings.
4. Send mapped data of the environment to the user in the form of plot markings

Part 2: User Movement Command

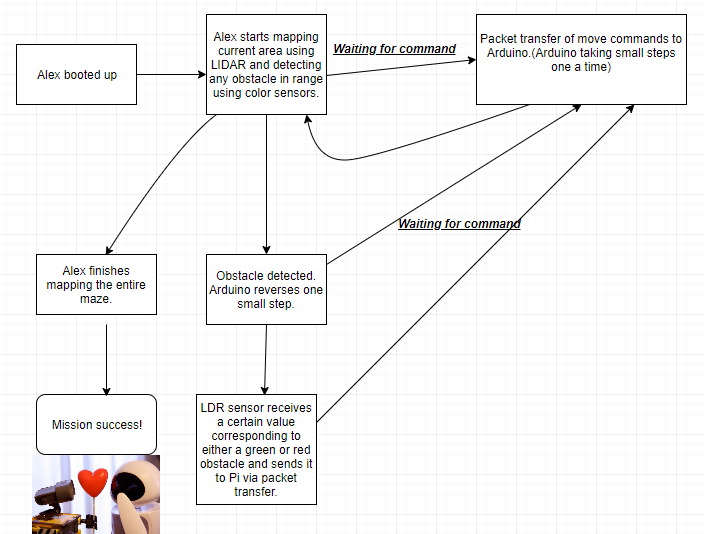
1. Based on the plot markings on returned environment map data, user would send the appropriate movement command to avoid the obstacles found in the surroundings and search for the exit.
2. Due to the sensitivity range of the LIDAR, it would be prudent to check for any close obstacles in front of Alex using either a proximity sensor (IR + IR Sensor circuit from EPP1) or a modified color detector sensor (Color sensor challenge from EPP1) attached to the front of Alex. The latter would have the added benefit of being able to simultaneously detect the additional feature of object color sensing for the additional features.
3. Relay the requested movement command from user to the Arduino through PI.
4. Arduino would execute said movement command. This movement commands would include going straight, going left, going right, making a U-turn and driving in reverse in the event Alex encounters a dead end.
5. (Optional feature) Warn the user if the potential command to be executed would result with collision with the obstacles (pre-calculating the distance moved by Alex if the command were to be made).

Part 3: Movement Command Execution

1. On the Arduino, the selected movement command would be executed, with the speed and angle of the turn being based on the direction of rotation of the wheels as well as the speed at which the wheels rotate.
2. For forward motion, wheels rotate at the same speed in the same direction.
3. For left/right turn, the respective turning side would have the wheel rotate in the reverse direction while the opposite side of the wheel would rotate in forward direction to spin left or right until a 90-degree turn is achieved in place before moving forward, allowing Alex to turn left or right.
4. For U-turning, it would be 2 cycles of left 90-degree spin to rotate Alex 180 degrees in place.
5. For reversing, wheels will rotate at the same speed in the reverse direction.
6. As the SLAM algorithm used would be the Hector SLAM, the speed of forward motion must be set to be below the max speed limits of the algorithm.

Part 4: Repeat parts 2 and 3 until user has detected escape route from mapped data

1. After every movement is executed, prompt the user and update current position of Alex via packet transfer (acknowledgement packets must be received by Arduino to continue execution of next instruction).
2. Prompt user to either run color check if it stops in front of an obstacle, and prompt for movement commands again after color detection. An alternative for prompting movement commands after color detection would be to reverse a fixed distance immediately to prevent collision when turning, since the object would be detected only if it is directly in front of Alex and blocking its way.
3. If color detected is either red or green, send a prompt to user to input the “identify object” command.
4. If no obstacles are found in front, immediately prompt via packet transfer for movement commands after the previous movement command completes.

****

**Section 5 Project Timeline**

|  |  |
| --- | --- |
| **Timeframe** | **Task** |
| **Week 8** | Finish assembly of Alex.  Identify additional components required to fulfill project requirements. |
| **Week 9** | Studio 1: Design SLAM algorithm  Studio 2: Design Robot turning algorithm |
| **Week 10** | * CELC workshops * Finalize code for Alex’s movement * Finalize SLAM and LIDAR integration * Integration of SLAM algorithm with the LIDAR to generate a reliable map of the environment. * Calibrate and draft movement codes required to move Alex around. (forward, reverse, turn left/right, spin on the spot for mapping) * Attach additional components required for the additional functionalities of Alex. * Test the remote control of Alex from command line using either SSH or VNC * Create detection of colored objects functionality. |
| **Week 11** | Studio 1: Cloud control of robot  Studio 2: Establish secure connection to robot |
| **Week 12** | Studio 1:   * Combine and integrate various segments of the software * Debug software-to-software communication * Debug software-to-hardware communication. * Test run of Alex in the lab   Studio 2: Mock Assessment  Presentation on project |
| **Week 13** | Studio 1:   * Final testing of robot functionalities * Final debugging of code that failed during mock assessment.   Studio 2: Final Evaluation |

**References**

*“*Rescue Robots Machines Play Vital Roles in Disaster Relief”

*Rescue Robots | Sci-tech | Trends in Japan | Web Japan*. [Online]. Available: https://web-japan.org/trends/09\_sci-tech/sci100909.html. [Accessed: 17-Mar-2019].

S. Toto, “Video: Rescue robot can pull victims into its body,” *TechCrunch*, 11-Aug-2009. [Online]. Available: https://techcrunch.com/2009/08/11/video-rescue-robot-can-pull-victims-into-its-body/. [Accessed: 17-Mar-2019].

“Mission - TRADR: Long-Term Human-Robot Teaming for Robot-Assisted Disaster Response,” *TRADR*, Nov-2013. [Online]. Available: http://www.tradr-project.eu/mission-2/. [Accessed: 17-Mar-2019].

“Long-Term Human-Robot Teaming for Robot-Assisted Disaster Response,” *GitHub*. [Online]. Available: https://github.com/tradr-project. [Accessed: 17-Mar-2019].