SEMI-AUTONOMOUS RESCUE TEAM

Team Description Materials Q1 2019

Connor Kneebone, Matthew Williams, Jack Williams, Alexander Cavalli, Aaron Maggs, Ryan Ewyk, Riley Cockerill, Michael Cavalli, Charlotte Drury, Graham Stock, Ryan Millard-Cartwright

support@sfxrescue.com

Table of Contents

Semi-Autonomous Rescue Team

Team Description Materials Q1 2019

Logistical Information

Team Name	Semi-Autonomous Rescue Team (S.A.R.T.)
Organisation	St Francis Xavier College
Country	Australia
Mentor	Graham Stock
Contact Person	Graham Stock Phone: +61 417 439 452 Email: Graham.Stock@sfx.act.edu.au
Website	https://www.sfxrescue.com
Team Email	support@sfxrescue.com

1 Abstract

Figure 1 - The S.A.R.T. (from left to right) Graham Stock, Peter Crane, Connor Kneebone (back row), Ben Bartrim-Hallett, Kyle Ewyk (front row), Gerard Elias (far right) with fellow robotics team from SFX, ABCDEFC (middle) in Montréal, Canada 2018.

The Semi-Autonomous Rescue Team (herein known as the S.A.R.T.) is a group of STEM enthusiasts formed in late 2015 with the intent of developing and creating a robot capable of competing in the 2016 Rapidly Manufactured Rescue League (RMRL) at RoboCup in Leipzig, Germany.

The project started with the basic robot design specification provided by Curtin University in Western Australia. This was provided as a starting point onto which the teams could innovate and redesign into more advanced and capable robots. The original S.A.R.T. robot consisted of the basic chassis design from Curtin University, known as the "Emu Mini 2" paired with a Raspberry Pi B+ and Dynamixel AX-12A servo motors. Over the first eighteen months, the original team brought the robot from its initial design to a fully custom-designed, 3D-printed chassis with an Intel NUC as the main control board instead of the Raspberry Pi. In addition to this, the robot featured an oCam 5MP camera for streaming a high-quality video stream to the control panel over enterprise grade Wi-Fi and an Arduino Nano which collected data from a number of sensors in the robot.

Following the graduation of the first team of students, the new S.A.R.T. was formed in late 2017. Tasked with building a robot that improved upon the previous iteration, they redesigned much of the robot from scratch and made a number of major improvements for the competition in Montréal, Canada in mid-2018. This included a new main control board, the UDOO x86 Ultra, new custom moulded polyurethane wheels, a completely redesigned compact chassis, and a multitude of new sensors including laser distance sensors, a gas sensor, and a thermal camera.

With the graduation of the second S.A.R.T., the robot was again passed on to a new team. This time, the team consists of students from across Years 10 and 11. With the new team comes a huge range of opportunities for innovation and improvement for the robot.

Overall the S.A.R.T. have experienced great success in competition, having received awards and recognition for a number of achievements and advances, including:

- Tied 1st Place – Rapidly Manufactured Rescue League, RoboCup 2016, Leipzig, Germany
- 1st Place – Rapidly Manufactured Rescue League, RoboCup 2017, Nagoya, Japan
- Open Source Award – Rapidly Manufactured Rescue League, RoboCup 2017, Nagoya, Japan
- Open Source and Innovation Award – Rapidly Manufactured Rescue League, RoboCup 2018, Montréal, Canada

2 Introduction

After forming in late 2015, the Semi-Autonomous Rescue Team entered its first competitive event in 2016 at RoboCup in Leipzig, Germany. The robot's original design was based on the specifications and design supplied by Curtin University in Western Australia for the Emu Mini 2. This design was improved before the competition to include streaming camera footage from a Raspberry Pi Camera to a computer and controlling the robot itself over a short-range Bluetooth connection using a PlayStation 3 controller. The camera stream enabled the remote operation of the robot when the operator does not have a direct line of sight, or, in the case of RoboCup, when the operator cannot look at the robot while competing.

One of the major issues with the robot while competing in Germany was the latency of the video stream due to the massive congestion in the wireless radio environment – the delay made it difficult to navigate the robot through the courses accurately. The congested wireless environment inspired a greatly improved Wi-Fi network utilising an industrial grade access point (AP). Implementation of such a bulky device would require an enclosure, as well as a computer to control it, so a control panel with a mini PC, AP, Uninterruptible Power Supply (UPS) for the Power-over-Ethernet (PoE) injector and related peripherals was constructed. The UPS also allows the control panel to be portable, so the robot can be operated in remote conditions or without access to stable power. Observing the performance of other robots at the competition inspired changes to the robot itself. One such improvement was to modify the chassis for upgraded internal components arranged in such a way that the space inside the chassis was used more efficiently by the more powerful hardware.

After observations of the robot's performance and the performance of other teams in the RoboCup 2017 competition, it was concluded that a robot performs better with a longer wheelbase and larger wheels, up to a point. Having an excessively long wheelbase limits the robot's mobility in the courses, and the maximum size of the wheels directly correlates with the length of the wheelbase. Other

improvements to the software side of the robot's operation include real-time motion tracking and image recognition as well as more accurate temperature and distance measurement, as the current temperature sensor can only reliably indicate a dramatic change in temperature rather than the quantitative value. The infrared (IR) distance sensors cannot be configured to return an accurate measurement that makes sense. As for the robot's movement, the fine control over direction and speed afforded by the control panel was helpful in navigating through the courses. The improved controls allow the operator to drive the robot efficiently through a course or turn it around by flipping over against a wall to travel in the opposite direction without having to spin on its axis in a confined space.

In 2018, the new S.A.R.T. decided to redesign the robot from scratch. This included the chassis, the wheels and all the software written for the project.

The previous robot had issues in Japan where it would beach on parts of the course due to the depth of the chassis. One of the first changes that was made was to make the new chassis far shallower. The new board, the UDOO x86 Ultra allowed for this, as it wasn't as tall as the Intel NUC.

The decision was made to experiment with using four cameras instead of the one. This ended up being an excellent choice, providing the navigator with an almost 360-degree view, making the robot far easier to drive.

There were a number of new additions to the sensors too. Laser time-of-flight distance sensors replaced the previous infrared (IR) ones, three non-contact temperature sensors instead of one, an all-new thermal camera, and an air quality sensor that measures CO_2 and Total Volatile Organic Compound (TVOC) levels.

The interface was completely redesigned. The original modal-based interface worked well for the old robot, with minimal sensors and only one camera, but adapting that to the current robot with its four cameras was not going to be the most intuitive system. The new interface was built around the four cameras, which take up the majority of the screen. System information is now integrated into each side of the screen. The three additional cameras (left, right and back) can be swapped at the press of a button for the sensor display which contains the distance sensor graph, the temperature graph and the thermal camera display.

A major change in 2018 was the wheel design. The team researched, designed and made polyurethane tyres for the wheels. This replaced the original Plasti-Dip method which was cumbersome to do, and the material was no longer in production.

3 Team Member Roles

Each member of the S.A.R.T. has their own defined roles and skill set. The different strengths they possess (programming, front and backend web development, 3D modelling and rendering, network configuration and administration, building/construction, research, documentation) can be applied to the field. For example, programming in Python and web development for front and backend ultimately gets the robot moving and sending the data back for analysis via a web server hosted on the robot itself. 3D Computer Assisted Design (CAD) and rendering produce the images and simulations of the robot's use cases, as well as the physical design of the chassis and two-part wheel designs. On top of all that, a viable network must connect the robot to the control panel despite the difference in hardware and software on each device as created by individual team members.

These disciplines require teamwork and communication due to the different people working on the same project having different problem-solving methods and different ideas for implementing solutions to the same problem.

This reflects a real-world engineering process wherein the team who designs the product may not necessarily be the same team that manufactures the product, who in turn may not be the same team who programs the product or tests the product.

Detailed documentation is necessary in this case, as for the reasons outlined above. Every member of the team is experienced in documenting their ideas and processes through the website blog to communicate the design and updates through a medium that the entire team can consult readily with ease. It also allows for the easy sharing of multimedia content to explain concepts to not only the team but also the wider open source community

4 System Description

4.1 Hardware

4.1.1 Robot

4.1.1.1 Chassis Design

From the project's inception up until the competition in Japan 2017, the physical design for the robot went through nineteen different iterations of the chassis and wheels in order to make them suitable to be 3D printed on a large scale.

The original design idea for the robot was for it to be modular; different components could be swapped out if broken or if the current use case called for a more specialised part. Everything was to be self-contained and capable of being put together much like Lego, with modules attaching with a universal 'snap in' mechanism. After a few prototypes were designed and 3D printed, it was concluded that while the robot was modular, the physical dimensions were too large to fit in the maze. Additionally, the singular point of contact for the snap in connector was not robust enough to withstand what we would consider normal us. The additional parts to 3D print drove up the manufacturing time and cost. Overall, we determined that, as a proof of concept, modularity on this scale was impractical for real-world applications.

Starting from scratch, we looked at the original design of the Open Academic Robot Kit's Emu Mini 2. Considering its pros and cons, we realised that the form factor was ideal for navigating through the competition maze and the wheelbase allowed it to turn on the spot.

With all the new hardware we intended to use in the robot, a total redesign was required to ensure everything could fit in a similar footprint to the original Emu Mini 2. Simulated test fits were run on the nineteen different prototypes of the current chassis model, which involved using 3D models of the components (such as the Intel NUC, SSD, Camera, Arduino and battery) to verify that everything would fit inside the smallest package possible.

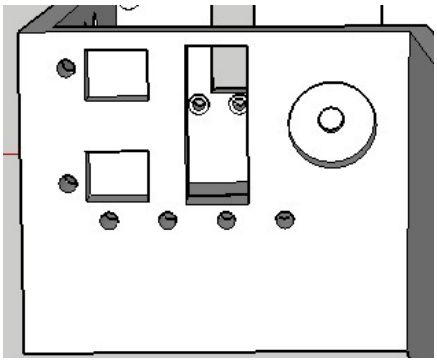

Figure 2 - The original servo mounts (from the Emu Mini 2) only allowed the servos to mount in a single orientation

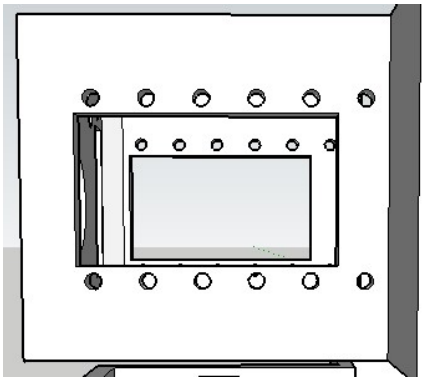

Figure 3 - The new design for a dual orientation servo mount. The only shortcoming of this design is the reduced strength of the chassis because of the rectangular holes in the sides where the greatest twisting forces are experienced.

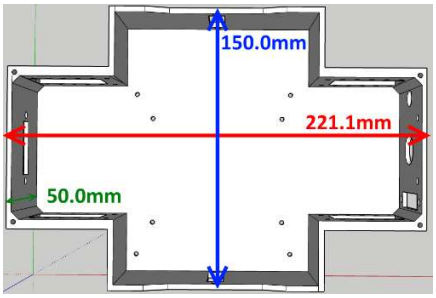

Figure 4 - (Top View) The dimensions of the final robot chassis.

While the basic rectangular design was inspired by the Emu Mini 2, some of the original features were omitted in favour of other features that we considered more important. One of the more notable modifications was the exclusion of a mechanical arm so that the robot was symmetrical on both top and bottom. The rationale behind this decision was that the robot could drive up against a wall and flip over. In our testing, we discovered that this was quicker than turning around, especially when the dimensions of the competition maze are too small for the robot to spin on its axis. The other notable change from the original Emu Mini 2 was the versatility of the servo mounts. Servos on the Emu Mini 2 could only be mounted in a single orientation due to the screw holes and power/data cable pass-through holes (Figure 2).

The Emu Mini 2 design differs to the design that we created, which had a single, larger cable pass-through hole and more screw holes where the servos mounted, giving us the flexibility to orient the servos however we wanted (Figure 3).

Our design gave us three options for wheelbase length, potentially having a short, medium or long wheelbase by simply changing the orientation of individual servos.

Unlike the Emu Mini 2, we did not add any extra holes or unnecessary embossed text as they provided an entry point for foreign objects, increased print time and made prints more likely to fail.

The final chassis consisted of a rectangular box of dimensions 221.10mm long, 150mm wide and 50mm high (Figure 4).

It had cut-outs for attaching the servos at two orientations, and a lid that prevented the internal components from falling out or foreign objects from entering the chassis. The physical size was the smallest that we could make it, given that we had to fit more powerful and larger hardware into it – mainly the small-form-factor computer, SSD, power delivery, battery, camera and the multitude of sensors, compass, accelerometer, gyroscope and Arduino.

Ultimately, this chassis design proved to be an effective design for navigating most of the courses. There were however a number of issues that become apparent whilst testing in real-life situations.

The robot went through some major redesigns of the chassis in 2018 in order to fix as many of the issues the team encountered in 2017 as possible.

Figure 5 - The Mark II getting beached on a course in Japan

One of the major issues in Japan was that the robot would get beached on some parts of the stepfield course (Figure 5). There were two obvious solutions to this, and we ended up implementing both. The first was bigger wheels and the second was to have a shallower chassis.

Since we were using the shallower UDOO x86 Ultra instead of the Intel NUC, we already had the ability to make the chassis somewhat shallower.

However, keeping the robot cool is important to maintaining reliable operation of the robot. We ran some CPU stress tests to see how hot the robot would get without and without the fan. While it never reached critical temperatures, it got close, so we decided the fan would be a worth keeping. However, both the fan and the heatsink ended up being too tall for a slimmer chassis. We decided we'd modify a custom low-profile heatsink instead.

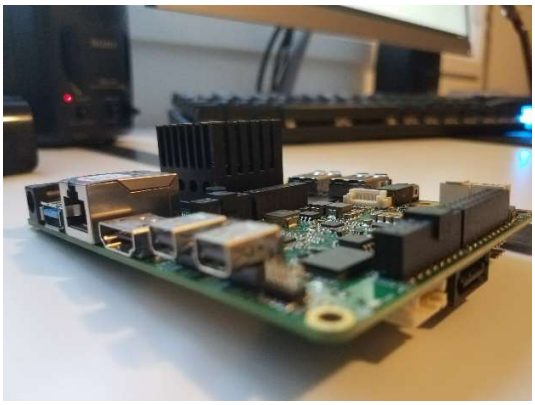

Figure 6 - Comparison of original vs custom heatsink

The heatsink we purchased was originally too large for the UDOO, but we cut it down to the correct size. A comparison of the original heatsink and the custom heatsink can be seen in Figure 6.

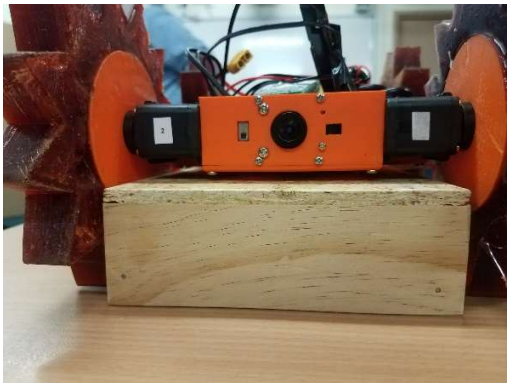

Figure 7 - The Mark III will hopefully no longer get beached!

We returned to the original design for the servo mounts. In Japan, we discovered that despite having the option to have a longer or shorter wheelbase, the longer wheelbase was far better for navigating the courses. Additionally, the bi-directional mounts suffered from reduced structural strength, and some ended up breaking during the competition.

We had a multitude of sensors we needed to include in the robot this time, as well as three additional cameras. The cameras were integrated easily into each side of the robot, however, some of the sensors proved to be difficult to fit on the chassis.

The thermal camera ended up overlapping with the camera on the front, just so both could fit. This was in addition to the distance sensor on the front. Likewise, on the back, the ethernet port, camera, distance sensor and temperature sensor were difficult to fit in the small area.

Another change was the lid. With the addition of the speaker and microphone, we added a speaker grill and a slot for the microphone. To help keep the robot operating at an acceptable temperature, we added a ventilation grille, with a filter to prevent foreign substances from entering the robot.

4.1.1.2 Wheels

The original design of the wheel was closely inspired by the wheels on the Emu Mini 2. To improve grip, the wheels were dipped in a rubber-based substance known as plasti-dip. This proved to be an effective method of improving grip on the wheels and was relatively easy to do.

However, it was a time-consuming process and needed to be reapplied in-between runs.

In 2018, the team created custom polyurethane resin wheels to solve this issue. They are durable, grippy and easy to rapidly mass manufacture. We also changed the wheel style to a hub and tyre design, to allow us to easily swap the types of wheels.

The team created two variations of these new wheels (Figure 8). The first of which featured larger teeth than the old wheels and was suited for traversing most terrain. In order to assist in navigating the 'hurdles' course, we experimented with a PVC-pipe oriented design. While these assisted in climbing the PVC pipes, they notably slowed down the robot and made it feel far more clunky to drive.

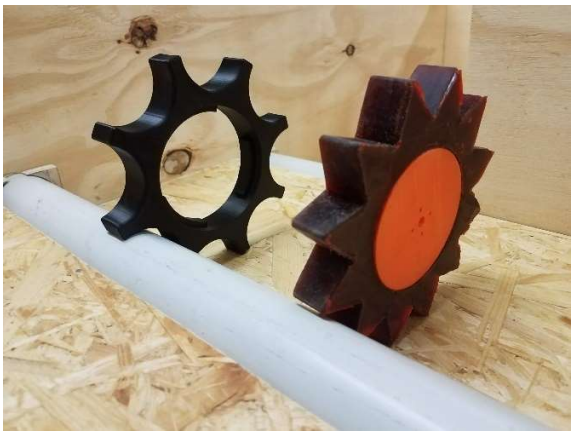

Figure 8 - Comparison of original tyre design and PVC-oriented design

4.1.1.3 Computational Unit

The UDOO x86 Ultra ("UDOO") is the main computational unit of the robot. It runs Ubuntu 16.04 LTS as its operating system.

The team used a Raspberry Pi in the earliest version of the robot which, while affordable and supported by a large community, lacks processing power and the ability to do any sort of high-speed network transfers.

The Raspberry Pi was replaced with an Intel NUC. This provided a huge number of advantages including large, high-speed solid-state storage, gigabit ethernet and high-speed (AC) Wi-Fi, and greatly increased RAM and processing power. The addition of improved network capabilities proved greatly beneficial and allowed us to stream high-resolution video streams in real time over long distances.

The UDOO additionally contains a built-in Arduino 101, eliminating the need to have a separate Arduino-based board in the robot as was done in previous versions. The additional processing power opened a realm of new opportunities for processing images and data on the UDOO itself.

4.1.1.4 Servos

The robot uses 4x Dynamixel AX-18A servos. Although we have also, at times, used the other models, mainly the AX-12A (lower torque, lower cost) and AX-12W (low torque, high speed). These can all be daisy-chained and connected to the UDOO via a USB2AX adapter.

4.1.1.5 Sensors

The robot has the following sensors:

- 4x Distance Sensors (VL53L0X) – One on each side, these are primarily used for autonomy. They provide a range of approximately 1200mm.
- 1x CO2 / TVOC Sensor (SGP30) – Measures carbon dioxide and total volatile organic compound levels in the atmosphere. This is used to assist in determining whether a victim is breathing and the dangers of sending in human rescue personnel.
- 1x Thermal IR Camera (AMG8833) – Displays a low-resolution thermal image, useful in identifying heat sources like a human body. It is situated on the front of the robot.
- 3x Non-Contact Temperature Sensors (MLX90614) – These are located on the back and sides (since we have the thermal camera on the front). These can measure the heat of an object up to approximately a metre away.

4.1.1.6 Power

The battery in use is a 0.8A 20-30C LiPo. During our testing this provided roughly 30 minutes of active operation. The battery outputs at approximately 11.1V but this can differ depending on the charge level so an UBEC voltage regulator is used to provide a consistent 12V to the UDOO and the SMPS2Dynamixel power board.

The four Dynamixel servos are daisy-chained, with one end of the chain connected to the USB2AX device and the other end to the SMPS2Dynamixel board which provides power to each of the servos. The order in which the chain is connected does not matter.

Each servo is assigned an ID, from 1 to 4. The IDs are assigned in a Z pattern from the front-left servo to the back-right servo.

4.1.1.7 Connectivity

An Ethernet port is available on the rear of the robot in case the wireless network cannot be used, such as in an environment with a large number of obstacles that would hinder wireless performance.

4.1.2 Control Panel

The idea behind the S.A.R.T. control panel is that everything needed to operate the robot is contained within a single portable package that is relatively lightweight and can be easily transported. (Figure 9).

Inside the control panel is an Uninterruptable Power Supply (UPS), Xirrus XR620 Wireless Access Point, and an AOC widescreen LED monitor all contained in a Pelican carrying case (Figure 10).

Figure 9 - A conceptual render of the early S.A.R.T. control panel, created to aid the communication of the idea before construction began.

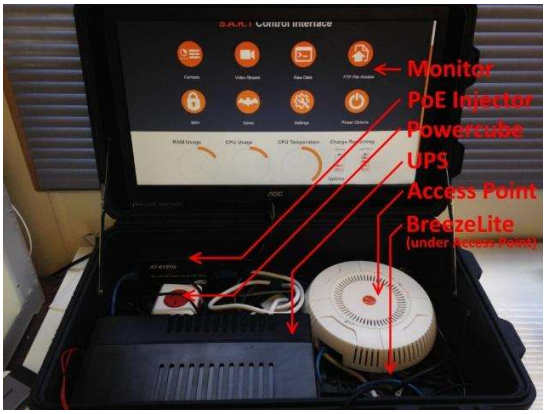

Figure 10 – The early S.A.R.T. control panel showing all the internal components.

The main PC inside the control panel has changed since the first version of the control panel.

The BreezeLite Mini PC was initially chosen based on its low power consumption, solid-state flash storage, small form factor and its passive cooling capability. The small form factor of the BreezeLite Mini PC meant that it could fit inside the case, leaving enough room for the Access Point, UPS and power delivery apparatus.

An Intel NUC was used in a more recent version of the control panel. The NUC offers similar features to the BreezeLite but is also able to run Linux-based operating systems such as Ubuntu, which aligns with our open-source philosophy and removes the need to purchase a license for Windows Server.

The wireless keyboard used to control the PC (and subsequently the robot) was a generic Microsoft device chosen mainly for its slim form factor (allowing the control panel lid to close without having to remove the keyboard) and the included touchpad, negating the necessity for a mouse. The operator can simply use a finger to point around the screen, whereas a mouse requires operators to potentially have to deal with uneven or poor reflective surfaces that can cause tracking problems with a traditional optical mouse.

In 2018, an experiment was done on a dual-screen control panel setup (Figure 11). Since the transition was made to a gamepad-based control scheme, the possibility of replacing the keyboard presented itself. While the screen in the top of the case remained the same, the wireless keyboard in the bottom panel was replaced with a Microsoft Surface Pro.

Figure 11 - The experimental dual-screen interface used in 2018

A dual-screen variant of the S.A.R.T. web-based control interface was used, showing the camera streams on the top screen, and sensor displays and interface controls on the bottom screen. This meant there was no interaction required with the top screen and all functionality could be controlled via the Xbox controller and touchscreen. The touchscreen controls included buttons to swap which camera is currently the primary one, an SSH terminal, and power controls.

This design however proved to be inconvenient when any manual work had to be done on the robot, due to the lack of keyboard. Additionally, a total of three cables had to be connected to the Surface Pro, which was inconvenient to detach and reattach. This design would be more appropriate in the future once the project has matured.

On all versions of the control panel, solid-state flash storage is used. This means that there are no moving parts that can be damaged if the control panel experiences a shock, unlike hard drives where a significant shock or drop can cause the head and spinning platter to misalign and as a result, not function correctly or at all.

The Xirrus XR620 Access Point was chosen for our applications based on its customisability and reliability, providing a fast, stable connection despite a considerable amount of external wireless interference. Even though it is a higher end Access Point and therefore rather expensive, it was given to us by our school so we would not have any issues with Wi-Fi unreliability, range or dropped connections.

The decision of the specific model of UPS we chose for the control panel came down to how long the battery would last based on our workload within the size constraints of the Pelican case. In Japan, we found having a UPS immensely helpful to convert 110 volts to the 240V Australian standard that we required for everything to work properly, with the aid of a transformer.

The screen we used was a generic 1600x900 resolution display that was chosen simply because it would fit inside the lid of the case and because it had a VGA connector which we used to connect it to the control panel PC. Using VGA rather than HDMI meant the cable would not come loose from the screen or PC during the competition.

Despite our use of a dedicated control panel, any device can be connected and used to operate the S.A.R.T. as long as the robot is configured to connect to the same network that the control panel is connected to. This is because the control interface is hosted on the robot itself via a web server, and all the user needs to do is enter the robot's IP address in the URL bar of their browser and have access to the control, streaming and data collection functionality.

4.2 Software

4.2.1 Robot

4.2.1.1 Operating System

Ubuntu 16.04 LTS was chosen as the robot's operating system. Its x86/64 architecture made it capable of running on the desktop-grade hardware in the UDOO x86 Ultra. Choosing a free operating system also allows others to replicate our project. All the software and code is distribution and platform agnostic and can run on any Linux distribution. The robot will likely be upgraded to Ubuntu 18.04 LTS. All the software should also run on popular ARM-based SoCs such as the Raspberry Pi, allowing for a low-cost version of the robot to be built.

4.2.1.2 Web servers

The robot runs the popular Apache2 webserver. This hosts the control panel interface on the robot which is accessible from any device on the S.A.R.T. network.

The robot's four camera streams are served over HTTP by a program called Motion. These are embedded into the control panel interface but can also be used directly by any image processing software that runs on the control panel (such as the QR code reader or hazmat detection).

4.2.1.3 Python Scripts

The scripts we run on top of Ubuntu are written in Python, chosen because of its versatility, ease of use and popularity. Some additional Python libraries were installed, such as *pyax12*, *asyncIO*, *WebSockets*

and *OpenCV* to manage the Dynamixels, asynchronous communication operation, control panel communication and vision respectively.

There are a small number of Python scripts that run on the robot. These handle the main functionality of the robot such as receiving and managing user input and streaming sensor data to the control panel.

servo_party.py

This script provides a reusable class to control the movement of the robot and is used by the gamepad script and autonomy scripts. It manages connecting to the servos via the *pyax12* library. It provides convenient functions for setting up and moving the servos.

control_gamepad.py

This script handles the movement of the robot via gamepad. It runs a WebSocket client, which receives a JSON formatted string from the control panel. The string contains all the required data from the controller, such as what buttons are pressed and what position the thumb sticks and triggers are at. It processes this and calculates the appropriate speed and power distribution of the servos based on the thumb-stick or trigger values.

sensor_stream.py

This script handles getting data from the onboard Arduino 101 and system data from the UDOO and then streaming it to the control panel via a WebSocket server. This includes all the data from the distance, temperature and gas sensors from the Arduino. It uses the Python module *psutil* to get the CPU usage, highest CPU core temperature, total RAM and current RAM usage from the UDOO itself.

stop_servos.py

In case of the all-to-common crash which leaves all the motors still spinning, this script will stop the motors from spinning. Although there is a failsafe in each of the main scripts that turns off the servos if the script stops, this provides a backup in case of an internal Python or system error.

auto.py

This script allows the robot to run in autonomous mode. It's in very early stages currently but in the future, it will allow the robot to navigate any of the main courses autonomously.

4.2.2 Control Panel

The control panel (in the context of software) can be defined as any device that connects to the S.A.R.T. robot and can load the interface in a web browser. Such a device can run any operating system (mobile devices may work theoretically, given a controller is paired with the device, although this is untested). The officially supported browser is Google Chrome, which allows the user to use all features of the robot. Mileage may vary when using other browsers.

For basic use, no other software is necessary on the control panel other than a web browser, as the interface itself is hosted on the robot's local web server.

However, the SART project includes a suite of control panel-side software, this includes hazmat sign detection, QR code reading and any future additions such as motion tracking. Most of these consist of Python-based programs that use the robot's camera streams that are available over network protocols.

4.3 The S.A.R.T. Network

The S.A.R.T. Network is a network to which both the robot and the control unit are connected to. All communication is done over the network using open-source web protocols. It uses the IEEE 802.11ac 5Ghz standard, as required by the competition. A Xirrus XR600 access point (AP) is used to allow the robot to connect completely wirelessly, however the option of running tethered via long-length ethernet cable is available, in cases of poor coverage, interference or where in need of additional reach.

The AP is the backbone of the network. Its industrial-grade wireless capabilities allow the robot to be operated from respectable distances with minimal latency and interference. Its interface provides access to many functions to edit and control the network as we please. The basic communication settings used during the 2018 competition was as follows:

Band: 5Ghz
Wi-Fi Mode: A
Channel: 140
Channel bonding: Disabled
Antenna count: Up to 4

The settings we used in the competition are by no means the limit of our access point's capability. It is capable of running 2.4GHz to 5GHz Wi-Fi bands in modes ranging from ac, a, b, g and n and delivering transfer speeds of up to 1.7Gbps to 240 individual clients. It is also capable of filtering out other Wi-Fi networks to ensure that it is the dominant network, which could be useful in a congested area. It can dynamically change channels depending on the congestion and traffic and also can bond those channels together for a more stable and reliable connection for greater range and signal strength.

The network is configured to use the 10.0.2.x subnet of addresses. The AP is statically assigned to 10.0.2.1 and addresses from 10.0.2.3 – 10.0.2.10 are reserved for each S.A.R.T. robot. The remaining addresses are part of a DHCP pool and are used for any other device on the network, such as the control panel. 10.0.2.2 is reserved in case the control panel needs a static IP in the future.

The advantage of using an industrial grade access point is that with all the bandwidth is concentrated on a single client, meaning that it is less likely to experience an outage. Because access points of this class are designed to be reliable in congested areas, the benefit to using this access point is because it helps guarantee a more stable, reliable and stronger signal, even in a congested environment similar to the one experienced at RoboCup 2017. The additional benefit this would provide in a rescue situation cannot be understated, being able to maintain a reliable signal is essential and the ability to channel bond and filter out other networks will greatly assist in this.

Communication between the robot and control panel is done mainly over WebSocket, a network-based communications protocol. The two scripts that primarily run on the robot, *control_gamepad.py* and *sensor_stream.py*, are a WebSocket client and server, respectively. *control_gamepad.py* acts as a client and receives the control data from the server on the control panel. *sensor_stream.py* serves all the

sensor data, which is received by the client, the control panel. All the data is sent as a human-readable JSON formatted string.

4.4 Control Interface

First responders in a rescue situation rate ease of use highly on what they want from a confined-space rescue robot. They need to be able to control the robot with minimal coding experience and prior training. This formed the basis of our design philosophy, right from day one the interface was built from ground up with ease of use and accessibility in mind.

Figure 12 - The original SART Interface, replaced in 2018

The culmination of this effort is a web-based control interface that can be used on any device with a compatible web browser. It is a complete front-end experience that removes the operator from the bare bones of the operating system via a sleek, modern and intuitive interface.

The original interface (Figure 12) was replaced in 2018 with a completely new design. This new design was necessary to accommodate the four cameras that are available on the new robot. Additionally, some features were removed, most notably the text-to-speech (TTS) and speech-to-text (STT) functionality as the Mark III uses voice-based communication. A few other rarely used functions were also removed such as FTP File Access and the ability to see the raw data.

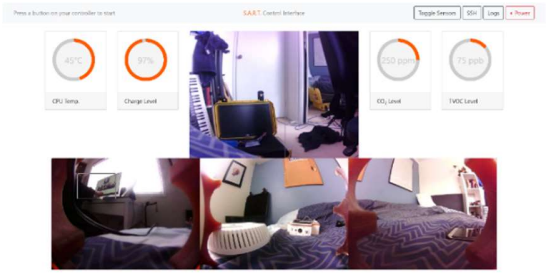

Figure 13 - The new SART Interface, redesigned to accommodate the three additional cameras

The main view of the interface consists of the robot's four cameras (Figure 13). To the sides of the main camera are a number of circular meters. On the left is the CPU temperature meter which reports the highest CPU core temperature of the UDOO, and the robot's current charge level percentage. On the right are the two gas sensors readings: CO_2 and TVOC levels.

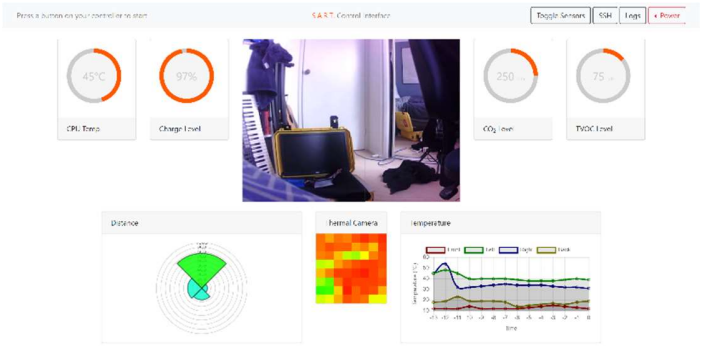

Figure 14 - The new interface's sensor view. With distance, thermal camera and temperature sensor graphs replacing the additional cameras

Sensor view (Figure 14) allows the user to see data from a number of the robot's sensors, all displayed on easy-to-understand and attractive graphs.

The *Toggle Sensors* button (mapped to the view button on an Xbox controller) in the menu bar toggles the display of the sensor view instead of the additional cameras view. The additional cameras (left, back, right) are replaced with three graphs for different sensors.

The **Distance graph** consists of four radial slices which expand based on the readings from the robot's distance sensors. These report up to the maximum range of the distance sensors which is about 1200mm.

The **Thermal Camera graph** is an 8 by 8 grid which shows the data from the robot's thermal camera.

The **Temperature graph** is a line graph showing the temperature readings from the different temperature sensors on the robot. It logs the most recent temperature readings and displays them as part of the graph.

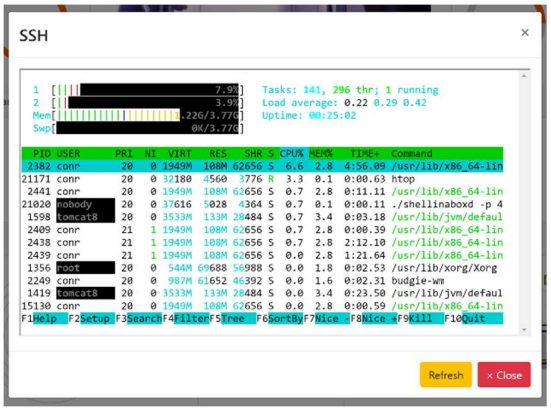

Figure 15 - The SSH terminal, allowing access to the underlying operating system

The **SSH terminal** (Figure 15) can be opened from the menu bar. The SSH terminal is powered by *ShellInABox* and it gives the operator complete access to the robot's underlying operating system. This can be useful for many different purposes, anything from killing unresponsive processes to monitoring activity on *htop* and eliminates the need to have an SSH client installed on the control panel.

The new interface features a completely new **control scheme**. The new interface was built around gamepad controller input, replacing the keyboard controls used by the previous interface. We opted for an Xbox One controller; however, any standard controller should work. This was done to make the interface more accessible and easier to learn, inspired by the United States Navy's decision to use Xbox controllers to control the periscopes of their most advanced submarines (Vergakis, B. 2017). The controller-based interface should be easier to learn and understand than a keyboard and mouse-based interface. It additionally allows the use of analog controls, notably the thumb-sticks on the controller. This allows the robot's speed to be controlled with far more precision than with the keyboard. The top left corner of the interface allows the user to see the current controller connection state and prompts the user to push a button on their controller to connect it if it's not connected.

The right-most button on the menu bar provides access to power options to let the operator quickly and safely reboot robot, or power it off completely, ready to be packed away.

Like the rest of the S.A.R.T. project, the entire interface is open source to allow other developers to contribute and to implement it in their own projects.

5 Operational Procedures

5.1 Setup

The setup process of the robot and control panel was designed to be as simple as possible. As mentioned previously, first responders in a rescue situation rate ease of use highly for robotic assistance. In a high-stakes environment, they need something that can go from packed to deployed in a matter of minutes.

One of our core design philosophies for user experience was simplicity, meaning the setup process we first demonstrated in Japan is remarkably straightforward and intuitive.

Firstly, all devices in the control panel are powered on simultaneously with a single, clearly visible power button. The operator can then power on the S.A.R.T. robot. After the boot sequence, it automatically connects to the waiting S.A.R.T. network.

In Japan and Canada, control scripts had to be started manually using the SSH console on the web interface. However, to simplify the process even more and to satisfy the overarching design criteria of an easy-to-use interface that requires minimal prior training to use, these scripts will run automatically on start up in the future and can be managed through the control interface which allows users to swap between manual and autonomous modes.

5.2 Pack Up

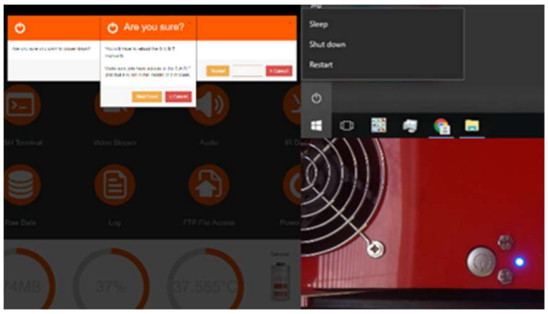

Figure 16 - The 3-step power-off process.

If the operator has already recovered the robot, the power-off process has three steps (Figure 16). The robot can be safely shut down using the power options in the interface. The control panel can then be turned off in the usual way in desktop operating systems (Start > Shutdown). Once the device has been safely powered off, power can be cut by depressing the UPS power button. Power can be cut to the robot using the switch.

Additional pack up steps may include placing the robot in its foam-lined hard carry case, as well as placing a protective foam shield between the monitor and keyboard of the control panel before closing the case. Optionally, the battery in the robot can be removed, although assuming it is not completely flat it is safe to leave it in indefinitely.

5.3 Mission Strategy

The Rapidly Manufactured Rescue League (RMRL) is a sub-division of RoboCup Rescue. It is a research competition that encourages students to develop innovative solutions to open response robotics problems in Search and Rescue, Hazardous Materials Response, Explosive Ordnance Disposal and Tactical Reconnaissance (oarkit.intelligentrobots.org). The RMRL brings this competition to high schools and undergraduate university classrooms, focussing on the challenges associated with operating small robots in confined spaces and low-cost prototyping, electronics and sensors to ultimately lower the barrier of entry into robotics research.

The specific strategy employed by the S.A.R.T. involves a careful analysis of each course to decide on the best route. For example, in the specific courses of stepfield and elevated ramps, the strategy for navigating was different from the dirt or gravel pits.

Our strategies changed over time. For example, in Japan, our strategy for clearing the stepfield and elevated ramps involved ‘hugging’ the left wall (when observed from behind), then executing a 90-degree turn and driving over the elevated section, as opposed to driving diagonally over the elevated section as this would often result in the robot becoming beached. If this was the resulting eventuality, our solution to becoming unstuck was to rapidly change the direction of the servos, effectively ‘rocking’

backwards and forwards to build up enough momentum to 'fall' off the perch where the robot was stuck. After the redesign of the chassis in 2018, getting beached was no longer an issue, which gave us the freedom to complete the course far more freely and without the fear of getting stuck. We could now complete the course by going the most direct route through the course.

A different strategy was employed for the dirt and gravel pits in Japan, because as the robot is relatively light, the smooth wheel design allowed for the robot to 'skim' along the top of the surface without sinking in. At the end of each course, the robot would drive up against the wall and execute a flip instead of turning around, which, in our experience was faster and more efficient, reducing the likelihood of becoming stuck perpendicular to the direction of the course. However, in Canada, the new design of the wheels created a new issue when navigating these courses. Now that the wheels have larger 'teeth', it was far easier for them to sink into the sand or gravel and had to be driven with far more care to ensure it did not slide down the slope.

For the hurdles course, which contains several PVC pipes that must be traversed, we created a completely new wheel design. While this design made it easier to climb this particular course, it limits the robot's speed and makes the robot feel far more clunky when driving.

These operation decisions were the result of careful experimentation with the courses and from previous experience controlling the robot.

6 Experiments & Testing

Much of the experimentation and testing phase occurred early in the development of the physical design and software development. Many of the experiments conducted exposed new problems that had to be solved for the changes made as a result of the experimental findings to be effective.

Unlike in previous years, in 2018, the team could use the real-world performance of the robot in Japan as a reference as to what worked well and what could be improved. This allowed for some major improvements such as fixing the beaching issue and improving the wheel design.

6.1 Experimental Design

It was through experimentation that it was concluded that a vertically symmetrical robot (the same on the top and bottom) would be useful in navigating through confined spaces, as a 180-degree turn was possible by driving up against a wall. Executing one of these turns resulted in the controls swapping and the stream upside down on the operator's end, so we had to patch the control interface to include a function that flips the controls and stream footage. With the inclusion of the additional cameras in the current robot, the ability to flip is not necessary, as the robot can be simply driven backwards, using the back camera instead of the front.

6.2 Materials Testing

A notable amount of research went into choosing and designing the materials used for the wheel. The first batch of wheels turned out well, with the only negative note being that the material looked like it set poorly, with a rough texture and uneven colouring. During the moulding procedure, we experimented with different quantities, notably not using dye, which provided a more appealing result. Careful consideration went into choosing the desired hardness of the wheels on the Shore A hardness scale. A hardness grade of 80A was chosen for the polyurethane resin tyres which would be roughly

similar to a leather belt. A grade of 25-40A was chosen for the silicone mould, which would be somewhere between the hardness of a rubber band and a pencil eraser (Mykin, 2018), hard enough to not stretch too much, but soft enough to allow the tyres to be removed from the mould easily.

6.3 Wireless Connection Range

Multiple experiments were run to verify the maximum range of the Xirrus WiFi Access Point. This was done by connecting to the S.A.R.T. network using a laptop (which had the same wireless network adapter as the robot) and walking in a straight line until the signal was completely lost. The average distance as calculated from the tests was around 90m in a crowded Wi-Fi environment which was thought to accurately simulate the conditions at the competition. Fortunately, this assumption turned out to be correct as we were able to operate wirelessly at ranges of over 50 metres during competition in Japan.

6.4 Streaming Reliability

During 2017, the reliability of streaming video footage was tested by loading the Mark II robot with numerous streaming clients to see at what point the stream cut out or the bandwidth of the access point was saturated. Having a low powered dual-core processor in the NUC, the stream began to drop frames, stutter and cut out after more than six clients were connected. The effects of this were exacerbated as the range increased but considering only one or two clients would need to be connected to the robot at any one time, the range and stream quality would remain normal.

In Canada 2018, the reliability of streaming four simultaneous video streams from the Mark III to a single client was tested. It was reliable to the point of around 30m through a single wall and other obstacles.

6.5 Battery Duration Testing

The usable time given by the UPS when the control panel is in the 'untethered' mode (i.e. not plugged in) was tested by building the power network required for the control panel to function. The worst-case scenario was simulated by initiating a synthetic load on the BreezeLite Mini PC and saturating the Access Point with a large file transfer over the Wi-Fi network using FTP. The results gave a minimum usable time of around 75 minutes.

The battery performance of the robot was also tested using a 1300 mAh LiPo battery. A series of tests determined a maximum run time of 90 minutes. We used the information gathered in these tests to ensure enough time was left between runs to swap batteries hourly during competition.

The battery performance of the Mark III robot was tested in 2018. The notably smaller 800 mAh LiPo battery provided approximately 30 minutes of continuous moderately-intensive operation. This proved to be more than sufficient for the competition's short runs, and batteries could be swapped between runs.

6.6 Experimenting at RoboCup

While we were at the competition in Canada, we made a few changes to the robot. The competition offered a far different experience to working on the robot back at school. Without some of the resources and most importantly, time, we were limited in what we could do and also how well we could make the changes we needed to do.

Taking the entire lid off the robot to change the battery before a run proved to be very time-consuming, and sometimes took long enough to use up some of our allocated slot time. Our solution was to cut the lid and use some duct tape to create a small flap that could be opened and closed with far more ease than removing the entire lid. It required unscrewing two screws rather than twelve.

RoboCup also allowed us to properly test the practicality of our new dual-screen control panel setup. Ultimately, it proved to be too difficult and tedious to work with and we chose not to even use the screen in the control panel, instead using the Surface, with its keyboard case by itself, situated in the control panel.

7 Current developments

The current members of S.A.R.T. are working on numerous improvements to the project for debut at the 2019 Sydney competition. This includes upgrades to the hardware and several software applications that either improve or add new features to the robot.

7.1 New chassis design

The prospective chassis design currently differs from the previous in only one way, the wheelbase has been enlarged by 50.9mm. It is proposed that a longer wheelbase will provide the robot with an enhanced ability to traverse terrain. A longer wheelbase also allows for the development of larger wheels, thus further increasing the robot's chances of success, and increases the volume available for internal hardware and sensors. A potential drawback to the longer wheelbase, the wheelbase of the previous chassis allowed the robot to turn on the spot within the maze, it is uncertain whether the new chassis retains this ability. It is suggested, that when the robot is high centred, a condition common in long wheelbase vehicles where an obstacle presses up into the centre, it could use the obstacle as a fulcrum and continue.

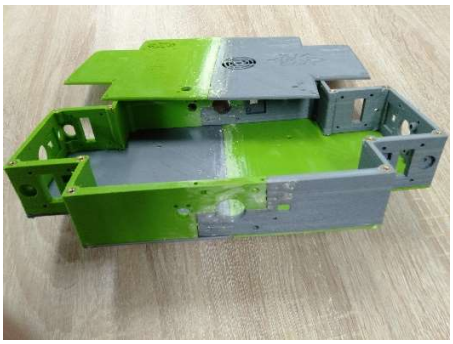

Figure 17 - The new chassis prototype

The chassis prototype (Figure 17) had to be printed in two pieces due to no printer bed of required size being available. To minimise the risk of fissure, the main body contains an overlapping design and is fixed with 2.5mm carbon fibre rods and acetone welding. The lid and base are also fixed with carbon fibre and acetone. In future, it would be preferable to mill the chassis as one solid body.

7.2 Autonomy

The current iteration of the autonomous functionality is a combination of a PID controller (currently primarily using the proportional component) and a finite state machine. The PID controller keeps the robot centred in the field, controlling the ratio of power to the left and right wheels while moving forward, and the state machine handles the intersections and ends of the fields. Currently, this only works on the most basic of fields, so the next iterations should work on some of the easier courses with obstacles, and eventually be stable enough to utilise SLAM.

7.3 SLAM

Through the use of a TurtleBot3 Burger that was sent to the team by our sponsor, Tribotix, the team started to look into simultaneous localisation and mapping (SLAM) (Open Source Robotics Foundation,

2018) (Robotis, 2018). S.A.R.T is in the process of integrating the robot operating system (ROS) project into the robot which provides SLAM-related software. ROS will allow S.A.R.T to access programs such as Rviz and gmapping which will help the team with integrating SLAM into the robot. SLAM will enable the team to make a 2D map of the area around it and allow the robot to create a path it can follow autonomously. The map will be seen in real time on the control panel which, in conjunction with the cameras, will allow rescuers to visualise the arena and terrain to better assess the situation before personnel must enter.

7.4 Image recognition

For the Mark IV robot, the team is working on using machine learning with OpenCV to detect hazmat signs. Building on the image manipulation employed in previous years, the new avenue for improvement is incorporating machine learning algorithms to classify the potential signs. The current algorithm being experimented with is the K-Nearest Neighbour algorithm to classify new images as it seemed quite simple to implement. The hardest part is finding a sufficiently large data-set to use as the training set, and we believe this may be a limiting factor for further advancements in this area.

8 Future Developments

The team has many great ideas for the improvement of the robot although some ideas are too risky, resource intensive, and/or time intensive to implement before the current competition. These ideas are listed below for posterity.

- Investigating avenues for incorporating an arm into our flippable design.
- Advanced machine learning capabilities.
- Investigating alternative materials and/or manufacturing methods for the chassis.
- Implementing LiDAR sensor array for increased accuracy and improved SLAM functionality.

9 Conclusion

9.1 What the team has learnt so far

So far through the process of designing and building the many prototypes and iterations of the robot, the members of the S.A.R.T. have all grown more skilled in their respective areas, leading to more sophisticated and streamlined programs and designs. This has been facilitated by previous members of S.A.R.T. who have returned to mentor the current generation to assist in the handover of the project and provide their first-hand experience of the competition.

As a result, we can anticipate potential problems before they occur and amend them before they become a major issue. Specific examples include wireless network optimisation, computer-to-computer communication via browser-based web applications on different operating systems, 3D design and rendering, 3D printing, image manipulation via a remote video stream, and controlling servos with various input devices over a wireless network.

9.2 New team goals

The new S.A.R.T. will continue the project, by improving the robot, writing blog posts, and preparing for Sydney 2019 alongside members of previous years, who will aid as mentors. The focus of the new generation of S.A.R.T. will be to:

- Improve the physical design of the robot to aid in the completion of course runs.
- Improve the sensory input of the robot to maximise our available multiplier.
- Implement intelligent algorithms, such as autonomous driving and SLAM, to mature the project, challenge the team and drive the competition forward.

9.3 What we hope to achieve

S.A.R.T provides the participants with the opportunity to acquire hands on experience in areas such as design and coding that they may not otherwise have access to. The necessity to learn new skills, such as new computing languages, and advanced concepts, such as SLAM, has led to the development of sophisticated skills beyond those that would have been achieved without access to this competition.

Participation in the competition will showcase and build upon team members' skills such as problem solving, creativity, initiative and teamwork. The need to analyse, evaluate and problem solve at competition level will give members real life experience in the world of engineering as well as insights into the worlds of innovation and business.

S.A.R.T also brings students of various ages and interests together that may not otherwise interact. The sharing of their unique abilities and perspectives benefits team members individually and the team collectively and the diverse skills provided by the unlikely gathering of students allows for the furthering of one's own skills. The competition will also provide opportunities for networking, exchanging ideas and learning from others. This exchanging of ideas and achievements creates a sense of community and builds cooperation and communication skills locally and internationally.

10 References

Intel 2016, *Intel® NUC Board NUC5CPYB and Intel® NUC Board NUC5PPYB Technical Product Specification*, viewed 2 March 2017, <http://www.intel.com/content/dam/support/us/en/documents/boardsandkits/NUC5CPYB_NUC5PPYB_TechProdSpec11.pdf>.

Intel 2016, *Intel® NUC Kit DN2820FYKH*, viewed 9 September 2016, <http://ark.intel.com/products/78953/Intel-NUC-Kit-DN2820FYKH>.

Lavrsen, K 2016, *Motion - Web Home*, viewed 20 May 2016, <http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome>.

Mykin Inc. 2019, *Rubber Hardness Chart,* viewed 11 March 2019, <https://mykin.com/rubber-hardness-chart>

Open Source Robotics Foundation, Inc 2018, *Turtlebot*, viewed 20 December 2018, <https://www.turtlebot.com/>

PC Case Gear 2016, *BreezeLite SN3-X5 Z8300 Windows 10 Mini PC*, viewed 13 September 2016, <https://www.pccasegear.com/products/36378/breezelite-sn3-x5-z8300-windows-10-mini-pc>.

RoboCup Federation 2016, *RoboCup 2017*, viewed 9 September 2016, <https://www.robocup2017.org/eng/index.html>.

ROBOTIS Inc 2016, *Dynamixel AX-18A*, viewed 13 September 2016, <http://support.robotis.com/en/product/dynamixel/ax_series/ax-18f.htm>.

ROBOTIS Inc 2018, *TurtleBot-3*, viewed 20 December 2018, <http://www.robotis.us/turtlebot-3/>.

S.B. 2017, *UDOO X86 User Manual*, viewed 2 Feb 2019, <http://download.udoo.org/files/UDOO_X86/Doc/UDOO_X86_MANUAL.pdf>

Sheh, R 2015, *The Open Academic Robot Kit*, viewed 4 May 2015, <http://oarkit.intelligentrobots.org/wiki/doku.php>.

Sheh, R 2017, *Controlling servos using the USB2AX V3.2*, viewed 3 March 2017, <http://oarkit.intelligentrobots.org/home/raspberry-pi/controlling-servos-using-the-usb2ax-v3-2/>.

Sidewinder n.d., *Enzotech CNB-S1L Low Profile Forged Copper Northbridge Heatsink,* viewed 8 March 2019, <http://www.sidewindercomputers.com/encnfoconohe.html>

Vergakis, B. 2017, *The U.S. Navy's most advanced submarines will soon be using Xbox controllers*, viewed 6 March 2019, <https://pilotonline.com/news/military/local/the-u-s-navy-s-most-advanced-submarines-will-soon/article_5c24eefc-8e70-5c4a-9d82-00d29e052b76.html>

Williams, J 2016, *S.A.R.T Interface*, viewed 20 May 2016, <https://drive.google.com/file/d/0B06CRDwuKGLLVVRrei15Wk9jLTg/view>.

Xevelabs 2017, *Product: USB2AX*, viewed 3 March 2017, <http://www.xevelabs.com/doku.php?id=product:usb2ax:usb2ax>.

11 Appendices

11.1 Appendix A - Components & Estimated Total Cost for Mark III

11.1.1 Robot

Component	**Price (USD)**	**Quantity**	**Total (USD)**
UDOO x86 Ultra	$267.00	1	$267.00
CPU fan for UDOO x86 heatsink	$7.50	1	$7.50
Dynamixel AX-18A	$92.83	4	$371.32
ELP 5MP HD USB Autofocus Camera	$35.28	4	$141.12
Adafruit VL53L0X Time-of-Flight Sensor	$14.95	4	$59.80
Adafruit AMG8833 IR Camera	$39.95	1	$39.95
Adafruit SGP30 CO2 / TVOC Sensor	$19.95	1	$19.95
Electret Microphone Amplifier - MAX9814 with AGC	$7.95	1	$7.95
Adafruit I2S 3W Class D Amplifier Breakout	$5.95	1	$5.95
Generic Stereo Enclosed Speaker - 3W 8Ω	$3.66	1	$3.66
Moulding materials for wheels (polyurethane, silicon, dye, etc.) *approx. cost.*	$10.58	4	$42.32
Custom Arduino Shield	$10.58	1	$10.58
JST connectors for sensors and shield	$3.53	1	$3.53
Ethernet Port	$3.33	1	$3.33
Generic 4 Port USB 2.0 Hub	$8.47	1	$8.47
Intel AC9260NGW Wi-Fi Card	$27.95	1	$27.95
MLX90614ESF-AAA Infrared Temperature Sensor	$23.42	3	$70.26
Ultimaker ABS 3D Printer Filament 1kg spool	$32.00	1	$32.00
Quanum 12V-5A (7.2 - 25.2V) Dual Output UBEC	$10.25	1	$10.25
SMPS2Dynamixel Power Board	$5.63	1	$5.63
USB2AX	$68.56	1	$68.56
Turnigy 800mAh 3S 30C Lipo Pack	$8.64	3	$25.92
Total			**$1233.00**

11.1.2 Control Panel

Component	Price (USD)	Quantity	Total (USD)
Pelican PROTECTOR CASE™ Camera Case (1514)	$229.66	1	$229.66
BreezeLite SN4-X5 Windows 10 Mini PC	$279.20	1	$279.20
CyberPower Value GP 1000VA/530W [VALUE1000EI] Line Interactive Ups	$138.40	1	$138.40
AOC E2070SWN 19.5in Widescreen LED Monitor	$87.20	1	$87.20
Xirrus XR620 WiFi Access Point (inc. PoE Injector & Ethernet Cables)	$460.00	1	$460.00
Microsoft All-In-One Media Keyboard	$45.60	1	$45.60
HDMI to VGA + Stereo Audio Converter	$22.36	1	$22.36
Allocacoc PowerCube 5 Power Outlets	$22.30	1	$22.30
0.5m VGA Monitor Connecting Cable	$5.56	1	$5.56
Comsol Male IEC-C14 to Female IEC-C13 Power Cable 2m	$11.90	1	$11.90
Comsol Male 3 Pin Plug to Female IEC-C13 Socket 2m	$11.90	1	$11.90
Total			**$1,314.09**

11.2 Appendix B – Robot Render and Components

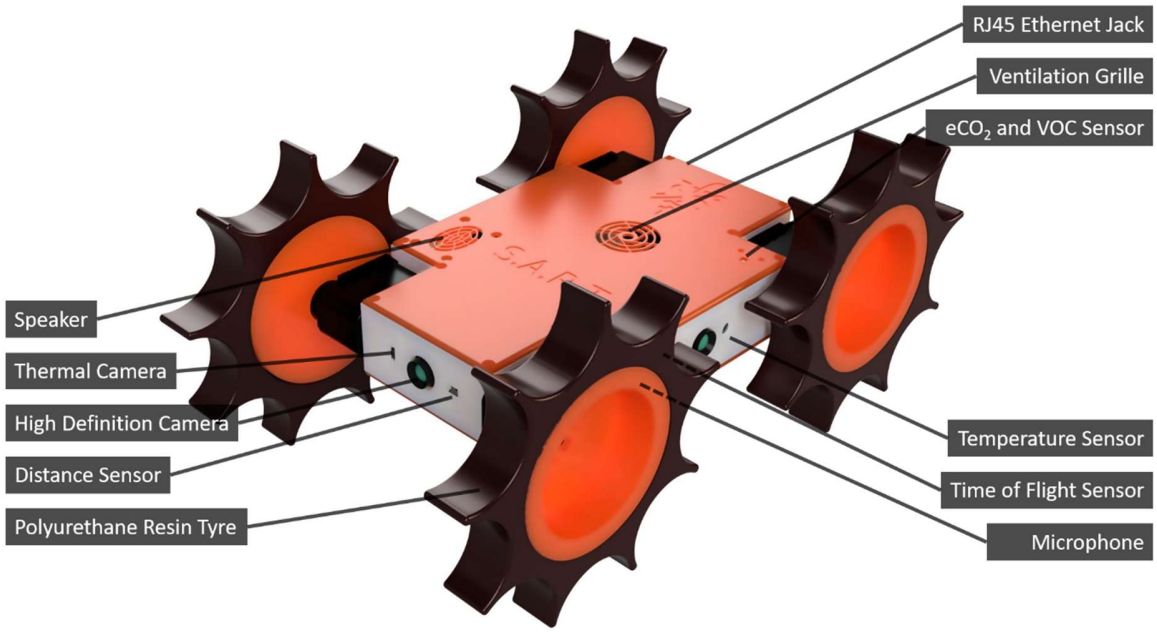

Figure 18 - Render of the Mark III robot, detailing external components.

11.3 Appendix C – List of Software Packages

Device	Software used
Control Panel (Microsoft Surface Pro *or* Intel NUC)	Microsoft Windows 10 *or* Ubuntu 16.04
	Mozilla Firefox
	Python 3.x
	Mumble Client
	CyberPower PowerPanel Personal Edition
Arduino 101	Adafruit MLX90614 library
	Adafruit SGP30 library
	Adafruit VL54L0X library
	Adafruit AMG8833 library
UDOO x86 Ultra	Ubuntu 16.04
	Motion
	Arduino IDE (CLI)
	Mumble Server
	Apache2
	Robot Operating System (ROS)

Process	Software packages used
S.A.R.T. Web-Based Control Interface	Web (JS / CSS): • jQuery • Bootstrap • WebSockets • Chart.js
Movement	Python: • pyax12 • asyncio • WebSockets
Image Recognition	Python: • OpenCV 3.1.2 • Numpy • Matplotlib
Motion Detection	Python: • OpenCV 3.1.2 • Numpy
QR Code Reading	Python: • OpenCV 3.1.2 • Pyzbar • Imutils • Numpy • Matplotlib
3D design of robot chassis and wheels	Google SketchUp
	Autodesk Inventor / Autodesk Fusion 360
Render of robot & control panel	IRender nXt SketchUp Plugin

11.4 Appendix D – List of Hardware

Component	Hardware
Robot Chassis	Heavily modified design inspired by the Emu Mini 2 from the Open Academic Robot Kit.
Robot Wheels	Original design inspired by the Emu Mini 2 from the Open Academic Robot Kit. Tyres moulded out of polyurethane resin in a silicon mould.
UDOO x86 Ultra	Central computational device on board the robot.
Dynamixel AX-18A	Servos, providing more power than the AX-12A.
ELP 5MP HD Camera	HD USB camera.
Adafruit AMG8833	Thermal camera. 8 by 8 grid.
Adafruit SGP30	CO_2 and TVOC air levels sensor.
Adafruit VL53L0X	Time of Flight laser distance sensor.
Electret Microphone Amplifier - MAX9814 with AGC	Microphone with automatic gain control.
Adafruit I2S 3W Class D Amplifier Breakout	Small speaker amplifier.
Generic Stereo Enclosed Speaker - 3W 8Ω	Small low-wattage speaker.
MLX90614ESF-AAA IR Temp. Sensor	Non-contact infrared temperature sensor.
Ultimaker ABS 3D Printer Filament	The 3D printer filament, a type of plastic extruded through a hot nozzle that builds a model by laying subsequent layers on top of each other.
Quanum 12V 5A Dual Output UBEC	Power delivery for the Intel NUC and servos.
SMPS2Dynamixel	Powers the servos via ~12V from the UBEC
USB2AX v3.2a	A small USB device that allows the servos to interface with the Intel NUC.
Turnigy 0.8Ah LiPo Battery	Powers the entire mobile apparatus.
Generic 4 Port USB 2.0 Hub	A small USB hub to allow us to connect all the cameras to the robot.
Intel AC9260NGW Wi-Fi Card	High speed (gigabit) Wi-Fi and Bluetooth M.2 network card.
Custom Arduino Shield	A custom Arduino Uno / 101 compatible shield to connect all the sensors to via JST connectors.

11.5 Appendix E – Web and Open Source Presence

Under our open source philosophy, everything we do is published online in the form of regular blogs, code repositories and 3D model downloads.

Website Blog

A regular blog detailing the design and implementation process over the course of the project.

https://www.sfxrescue.com

Code Repositories on GitHub

All our code is provided free to use and edit under the GNU General Public License on GitHub, where we encouraged other teams to contribute to or find inspiration in our solutions.

https://github.com/SFXRescue/

Editable 3D Models on Thingiverse

All our 3D models were made free to use and edit under the GNU GPL license on the 3D model sharing website Thingiverse.

https://www.thingiverse.com/SFXRescue/designs

YouTube Channel

The S.A.R.T YouTube channel has a series of videos including tutorials (Plasti-Dipping wheels to improve grip and daisy-chaining the Dynamixel servos), experiments (Wi-Fi range and stream tests) and sharing new features and developments.

https://www.youtube.com/channel/UCOM41hoo5jFGdlgnjjvApSQ/videos