UNIVERSITY OF KWA-ZULU NATAL

CONTRACTIBLE ARMS ELEVATING SEARCH AND RESCUE (CAESAR) ROBOT - IMPROVEMENTS AND MODIFICATIONS FOR URBAN SEARCH AND RESCUE (USAR) ROBOTS

Riaan Stopforth
(ZS5RSA)

Mechatronics and Robotics Research Group (MR^2G)
Search and Rescue Division
Durban

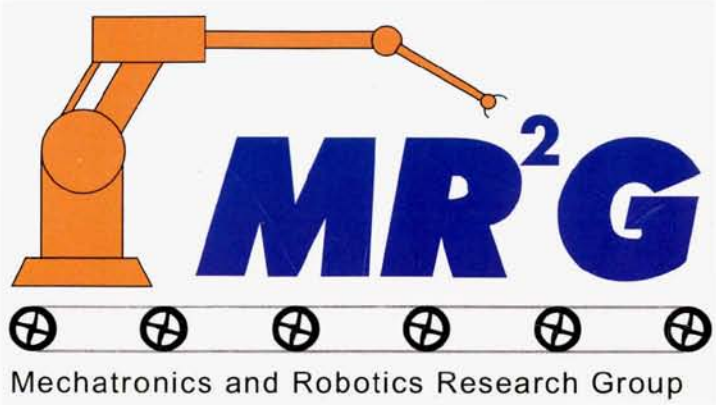
SUPERVISORS:

Prof. Glen Bright

Prof. R. Harley

2010

Submitted in fulfillment of the academic requirements for the degree of Doctoral of Philosophy in Engineering at the School of Mechanical Engineering, University of KwaZulu Natal.



Search And Rescue Division

UNIVERSITY OF KZN

PREFACE

The author hereby states that this entire thesis, unless specifically stated otherwise, is his own work, and has not been submitted in part or whole to any other university. This thesis records the work completed by the author at the School of Mechanical Engineering, University of KwaZulu-Natal from January 2008 to December 2009.

R. Stopforth

ABSTRACT

Rescuers have lost their lives in events requiring them to go into dangerous areas that have unstable structures and gases. Robots are necessary for search and rescue purposes, to access concealed places and environments to which fire fighters and rescue personnel cannot gain entry.

Robots that were previously used encountered problems with communication, chassis design, traction and sensory systems. Improvements are required for the successful localization of victims. Research on improvements in these areas were carried out for the use in the CAESAR (Contractible Arms Elevating Search And Rescue) robot.

Contributions were made in the area of Urban Search And Rescue (USAR) robots focusing on antenna design, communication protocols, chassis design, traction system and artificial intelligence on decisions relating to gas danger levels for humans and the robot.

The capabilities of CAESAR is audio, video and data communication irrespective of the orientation of the robot and the antennas. Penetration of radio frequencies through building material is possible. Reliable data communication is achieved with the designed Robotics Communication Protocol (RCP). The chassis is designed to have traction on unstable terrain and autonomously transform flipper arms for the best orientation. Materials for the body were selected and constructed to be able to withstand the unstable environments and high temperatures which they will encounter. The control station display gives the rescuers immediate indication of the gas concentrations detected by the on-board gas sensors. Developed analytical models determine the danger of the gas concentrations for victims, rescuers and the robots.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to the following:

- My supervisors, **Professor Glen Bright** and **Professor Harley** for their supervision and guidance.
- The members at the **University of KwaZulu-Natal**, who include the *staff* of the **School of Mechanical Engineering**, **IP Office** and **Innovation Company**, who have assisted and guided me through my research.
- The **Mechatronics and Robotics Research Group (MR^{2}G)**, for their openness and discussions of new ideas.
- **Mr. Peter Leonard**, who represented *Sentech* and *ICASA* for the communication accessibility arrangements and frequency allocation for testing the concepts to be proven.
- **Mr. Louis de Bruin (ZS5LP)** for assistance with communication testing
- **Mr. Dave Long (ZS5FR)** for his availability and arranging of facilities to test communication systems
- **Chief Mark TeWater**, **Mr. Alex Gloster**, **Mr. Kenny Naidoo** from *Ethekwini / Durban Metro Fire Department*, for their advice and information on the need and requirements for search and rescue purposes and arranging disaster incident testing. **Mr. Keith Lowes (ZS5WFD)** for arranging the use of emergency frequencies, supply of components and fire department resources.
- **Mr. Leo du Plessis** from *Durban Specialist Chemicals* for supplying us phenolic resin to perform the tests required for the research.
- **My family**, who have supported me during the research period, especially to my *mother* who has proofread my documentations, even though at times it must have been difficult trying to understand what I was attempting to convey.
- Finally, to **Jesus Christ** my Savior, who has helped me in all the research, problems, achievements and results. Without Him, I will not have been able to complete any of my life tasks.

LIST OF ACRONYMNS

ACK – Acknowledgment
AI – Artificial Intelligence
AP – Access Point
BLOS – Beyond Line Of Sight
COO – Cell Of Origin
CRC – Cyclic Redundancy Check
CSMA/CD – Carrier Sense Multiple Access Collision Detection
CTS – Clear To Send
DGPS – Differential Global Positioning System
EMF – Electro Motive Force
ESF – Emergency Support Function
FLIR – Forward Looking Infra-Red
GPS – Global Positioning System
GUI – Graphic User Interface
I^2C – Inter-Integrated Circuit
ICASA – Independent Communication Association of South Africa
ICS – Incident Command System
IDLH – Immediately Dangerous to Life or Health
ISM – Industrial, Scientific and Medical
IR – Infra-Red
LAN – Local Area Network
LOS – Line Of Sight
NAV – Network Allocation Vector
NIMS – National Incident Management System
NIST – National Institution of Standards and Technology
ppm – parts per million
RC – Robotic Confirmation
RCP – Robotic Communication Protocol
RO – Robotic One-way
RTS – Request To Send
SLAM – Simultaneous Localization And Mapping
SWR – Standing Wave Ratio
TLV – Threshold Level Value
UHF – Ultra High Frequency
USAR – Urban Search And Rescue

INTERNATIONAL ALPHABET PHONETICS

A	Alpha
B	Bravo
C	Charlie
D	Delta
E	Echo
F	Foxtrot
G	Golf
H	Hotel
I	India
J	Juliet
K	Kilo
L	Lima
M	Mike
N	November
O	Oscar
P	Papa
Q	Quebec
R	Romeo
S	Sierra
T	Tango
U	Uniform
V	Victor
W	Whiskey
X	X ray
Y	Yankee
Z	Zulu

CONTENTS

TABLE OF FIGURES

CHAPTER 1 - INTRODUCTION

1.1 Mechatronics

Mechatronics is the combination of Mechanical Engineering, Computer Engineering and Electronic Engineering. This collaboration is shown in figure 1-1 [1], which is one of many graphical expression of mechatronics.

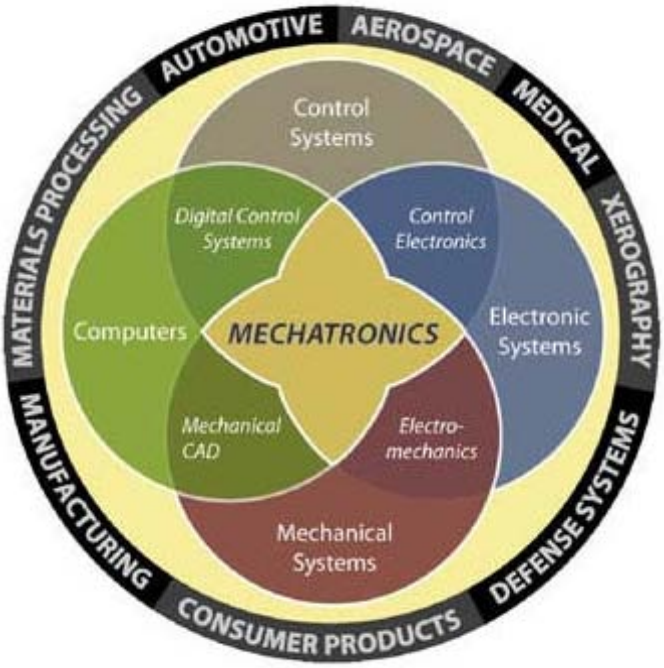


Figure 1-1: Graphical representation of Mechatronics

Mechatronics has developed from the technological development in the different fields to advance improvements in machinery, robotics and manufacturing designs.

1.2 Motivation for Research and Literature Survey

The literature survey that follows describes the use of Urban Search And Rescue (USAR) robots during previous disasters, highlighting the problems and constraints that were experienced. These problems will be considered to introduce improvements and possible solutions for the reviewed USAR robot.

Robots are required for search and rescue purposes. They should be able to access concealed places and environments to which fire fighters and rescue personnel cannot gain access. Three hundred and forty three firefighters died at the World Trade Center during the September 11 attacks in 2001 [2]. Rescuers often enter areas that have unstable structures, unaware that there may be no live victims to

rescue. Sixty five of these rescuers died as a consequence of searching in confined spaces that later flooded with water. [3] Robots could save the lives of victims and be first responders. Rescue workers typically have about 48 hours to retrieve victims due to survival constraints [4]. Several hours are lost when rescuers are unsure of a building's stability and have to wait until the rubble has stabilized. Routinely the rescuers have to evacuate when a body part of a possible survivor is found and the terrain becomes unstable [5]. Robots can stay in the unstable area and continue searching for survivors. In the future robots could possibly also be used to access mines, earthquakes, floods and landslips after an accident prior to rescue workers [6].

The development of search and rescue robots began with programs in the 1980s in Japan and the USA in particular. The Field Robotics group at Carnegie Mellon, under William Whittaker, had its origins in systems developed following the Three Mile Island disaster in 1979. Urban Search And Rescue (USAR) Robots were first extensively deployed at the collapsed World Trade Center site in 2001 [7]. The University of South Florida was involved in these rescue attempts. The robots that they used are shown in figure 1-2. The advantage of these robots over rescue members was their ability to immediately enter the disaster areas.

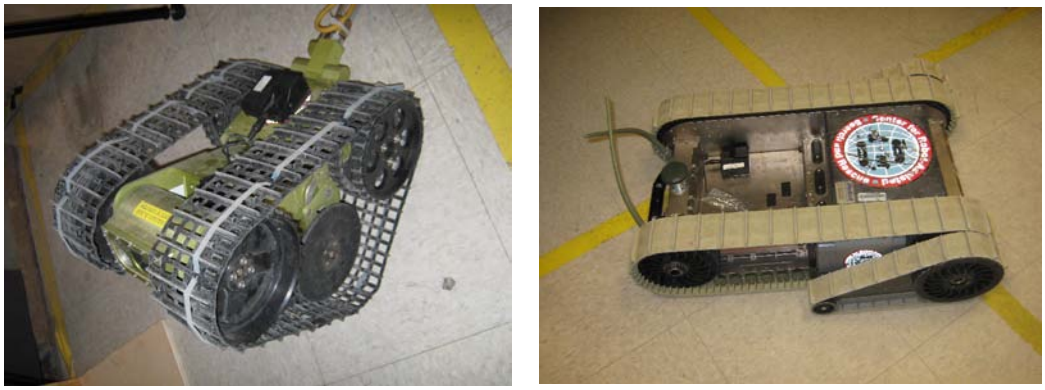


Figure 1-2: The Inuktun MicroVGTV and I-Robot Packbot were used in the rescue attempts at the World Trade Center in September 2001

The National Institution of Standards and Technology (NIST) have developed three grades of courses for test purposes. The Yellow course has a flat surface and is made from uniform material. Passageways are wide enough to permit large robots. The Orange course introduces a second level story, and the robots are required to climb stairs or up a ramp. Flooring material could consist of carpeting, tile or rubble and holes could exist in the floor. The Red course consists of piles of rubble with minimum or non-existing lighting. Passageways are narrow, extending under rubble or through pipes. [8]

Some of the performances required from USAR robots are victim location, hazard detection, quality of communication with humans and payload deployment. The payload consisting of a first aid kit, a radio, or food and water. USAR robots are able to maneuver over rubble, stairs and steep ramps. They are required to go through gaps and concealed areas. Acoustic, thermal and visual sensing are requirements

for successful search and rescue operations. Some form of autonomy is essential, in the instance that only limited bandwidth communications are available. [8]

Several problems were identified at the World Trade Center as well as at the testing grounds of the NIST. Firstly the robot's traction system malfunctioned [7]. More research was needed for the robots to withstand the harsh conditions of a fire [2]. Other issues observed were unstable control systems, chassis designed for narrow range of environmental conditions and limited wireless communication ranges in urban environment, as well as unreliable wireless video feedback [9]. Some robots were either too large or not easily maneuverable [4].

Other problems encountered were that the setup time of the robots was too extensive [7]. The human to robot ratio for transport and controlling was not consistently at 1:1 [7], which is ideal as the released human resources could be used to rescue victims. Setup times exceeding 5 minutes are considered to be vital time lost for the search [10]. Problems were identified regarding the communication with the robots [11]. Autonomous robots for USAR applications are not feasible as operators of robots are to work as a team, each contributing unique skills and capabilities [12]. Tasks of USAR robots include searching for survivors, inserting special sensors into the environment, collecting visual data of structural damage, carrying radio transmitters, carrying small amounts of food and medication to the victims and transporting rescue tools [13]. Certain objectives that USAR robots should meet are: to move in rough terrains, climb rubble and stairs, have high mobility and stability, consist of high integration modules, be radio-controlled, light-weighted, small and portable [14].

The initial aim in a disaster scenario rescue is the localization of victims and the detection of dangerous situations such as gas leaks, live wires and unsafe structures. The identification of such dangerous areas could save the lives of rescuers and victims. [7]

Communication protocol structures and suggestions for robot applications have been documented for the “further development of protocols for robot communication” [15]. These concepts only investigated the fields in the transport and content layer of typical communication protocols, but require “higher content layers to represent task dependent information” [15]. The IEEE 802.11b link used on the Packbot allowed for 200 m Line Of Sight (LOS) and about 2 rooms distance communication for Beyond Line Of Sight (BLOS) scenarios. Interference from other equipment using the 2.4 GHz frequencies decreased the quality of operation. Solutions for the handling of signal jamming and other radio traffic is desired. [16]

Two-way voice communication with a victim is important as this is one of the first steps for first aid assistance [7]. A thermal camera and two-way voice communication capabilities are both considered to be important and effective sensors for USAR attempts, as experienced at the 2005 La Conchita mudslides. [17]

Examples of other USAR robots are the ATR X-50 and the VGTV-Extreme, which each have different characteristics. The ATR X-50 [18], uses 2.4 GHz Wireless TCP/IP networks. It has a overall weight of 84 kg, with dimensions of 950 x 650 x 350 mm and can move on inclines with a 15° - 20° angle. Infrared range finders were used for obstacle detection and it has an on-board CCD camera. Suggestions for a more compact robot are made. [18]

The VGTV-Extreme has a tipping point of about 15° in the event that it should climb over obstacles [10]. Even though small robots are desired for USAR operations, larger stair-climbing robots are preferred [17]. The most common size voids that the robots would enter have a height between 0.33 m and 0.5 m.

The power to weight ratio of the robots needs to be examined by implementing a complex and efficient gearing system to maximize battery life. This will allow for maneuverability over uneven terrain. [7] Tests performed at the NIST indicated that the video feedback did not supply an indication on the steepness of inclines. The need for sensors to gauge the slopes is suggested as being beneficial [12].

Further studies in the failure of USAR robots were conducted and proved to be due to effectors and control systems that caused the most common of physical failures [9]. Slippage was the more common failure rather than errors. Robots in the disaster environments tested were unreliable. It appeared that the quality of each module within the robot influenced the quality of the robot's overall performance. The most common reason for failures is the complex control and effector systems. Mass produced sensors have proven to be the more reliable than custom-designed sensors. Research has also shown that the maintenance of robots must be accomplishable in the event of expected failures. Results of tests performed on 13 different robots confirmed, “limited mobility and unreliable wireless communication are problems which need to be addressed.” The recommendation resulting from these studies is that information needs to be filtered for failure of data. [9] The above mentioned problems could be evidence that designers and developers have not thought about the problems of operating in these environments.

The rubble pile at the World Trade Center was massive and there were confined areas which could only be accessed by small robots. Because of fires, robots responding to these scenarios would have to be highly heat-resistant. [19] Climatic conditions like flooding and slippage hazards also added to the risk for rescuers. [7] USAR robots are required to be water-resistance, as the electronic components can be damaged by the moistness of the environment or by sprinklers [17].

Control interfaces are required that will enable the operator to control the robot as desired. Future suggestions for research include the development of a better control interface for multiple robots. [20] Remote viewing of video feedback is essential and multiple observations stations are suggested, as this would enable other rescue members to also observe the video and have the advantage of their observing dangers or victims [17]. The operator should have higher authority than the robots.

Robots should be capable of accepting instructions to prevent the operator from reacting instead of pro-actively controlling of the robot. [21]

Field tests performed by the Center for Robotic Assisted Search and Rescue (CRASAR) at different disaster scenarios indicated the different advantages and disadvantages of robots described below [3]:

- Robots are able to enter places that humans cannot enter. These environments could be concealed, extremely hot and toxic.
- The safety and effectiveness of rescuers, are the most crucial in disaster scenarios.
- The rubble terrains were a challenge for the robots.
- Extreme heat within the pile of rubble caused the softening of the robot tracks and caused failure.
- The Foster-Miller Solen robot was abandoned as the density of the terrain interfered with the wireless network control.
- Robots are able to assist with five different tasks, namely “confined space search and inspection, semi-structured search, victim extrication, transport of medical payloads and monitoring.”
- The average time that a robot was utilized, was 6 minutes and 44 seconds.
- Robots such as the MicroTracs had to have a secondary operator next to the robot's point of entrance feeding the tether or rope. This resulted in this operator being in direct danger should there have been collapsing of the unstable environment. A tether has the disadvantage of being a hindrance for a successful operation, as it gets tangled and dragged [17].

The rescuers did not trust the robots that were used for the USAR scenarios. Forward Looking Infra-Red (FLIR) cameras are not portable on the VGTV, MircoTracs and Solem robots. Rescuers wanted air quality meters attached to the robots. These gas meters proved difficult to attach to the robots and readings were only possible once the robots returned from a search. Another suggestion that was made was that the robot and user interface needed to be usable with minimum training and that all researchers involved with rescue events have the sufficient training. [3]

As the research and development of USAR robots are of a newly developed field of study, the scientific publications are limited and for the most part refer to the tests performed at the World Trade Center disaster. The problems associated with the USAR robots at the World Trade Center have not been improved and are being used as the benchmark.

1.3 Thesis Contribution

The intention of this thesis was to produce a prototype of a USAR robot. The reviewed USAR robot has been improved and developed in different areas that have caused failure of performance in previous systems. These areas include communication, mechanical properties and construction and understanding the data received from the robot.

The specifications of the robot examined the requirements specified by rescuers. The requirements include voice, data and video communication, telemetry of gas concentrations, thermal video feedback, smallest possible size, easy control and durability in the environmental temperatures.

Three key areas were researched: vehicle traction and “transformality”, vehicle communication and an intelligent sensory system for harsh and environmentally unstable conditions. Testing and validation of the performance of the robot was documented and analyzed.

1.4 Research Objectives

The objective of this research was to investigate, design, assemble, test and determine the specifications of a robot that could be used for USAR scenarios. The research included the following topics:

- The traction system was researched, designed and thoroughly tested to allow the robot to maneuver through harsh and unstable environments. The robot was able to withstand the temperature of 200 degrees Celsius (too dangerous for humans) [22]. A thorough research of the material for manufacturing was conducted as this affected the concealing and protection of all the components. The traction system needed be more competent. Another feature that was regarded was the insulation and protection from environmental conditions, making the enclosure splash proof and impact resistant. This necessitated research into composite materials.
- Research was conducted to construct the transformable robot, compact in size, light weight and easily transportable. Investigation was extended to the ability of the robot to transform its shape and therefore enter into confined spaces.
- Studies on communication were conducted. Licensed and emergency frequencies were analyzed to prevent interference between the robots. Transmission protocol was considered, as well as the amount of power transmitted, considering that battery power was to be conserved. The robot required communication with the operator in line of sight as well as beyond line of sight. With the beyond line of site scenario, the physical environment has a great impact [11]. Further research is necessary to transmit video. This study should include ways to transmit and receive the signals and determine whether licensed frequencies are available.
- Research was conducted on the telemetry sensory system that is required in

the search for survivors. Research improvements pertaining to cameras will enable firefighters to determine the temperatures of the flames and surrounding conditions.

- Necessary investigation for autonomous transformation was conducted. This required a robot with omni-directional movement. A tracked system was utilized as it had a 30 % greater pull force compared to a wheel [23].
- The required robot needed to operate with a low setup time to prevent any delays. The operator had to be able to control the robot with a minimum amount of training. The human to robot ratio needed was 1:1, which is ideal as the released human resources could be used to rescue victims. This involved research and development of a control system that examined the situation the rescue members would encounter.

1.5 USAR Robot Specifications and Requirements

The World Trade Center disaster indicated many problems were associated with USAR robots. The problems associated with the USAR robots at the World Trade Center have not changed and are being used as the benchmark.

1.5.1 Mechanical Specifications and Requirements

Size: Height: 500 mm

Width: 1 m

Length: 1 m

Climb Gradient: 30° for long distances

45° for short distances

Size gap to enter: 1 m wide and 500 mm in height

Robot speed: Not applicable, as speed is dependent on the terrain, traction and stability.

Payload carrying ability: The robot had to be capable of carrying the weight of all the components needed for operation, which is a total of 25 kg.

Improvements and Contributions:

- The body had to be constructed to withstand temperatures of 200 °C for the time period of the search and rescue operation.
- Transformation was needed to allow the robot to maneuver over difficult terrain
- Traction system

1.5.2 Communication Specifications and Requirements

Frequency: UHF frequencies were needed for best possible signal penetration. The 453.500 MHz and 449.725 MHz simplex frequencies had been approved by the

Durban Metro Fire Department.

Data Communication: Improvements had be made in the data communication to prevent interference from other stations. The protocol size had be less than 50 % of common protocols.

Voice Communication: The control station operator needed to communicate with victims found.

Video Communication: A video feedback was required to examine the disaster area and to locate victims.

Antenna Design: The antennas necessitated investigated and design to improve the communication, so that a 360° omni-direction radiation is possible.

Transmission Power: At least 5 W of transmission power was needed where possible, as this is the maximum power approved from the Durban Metro Fire Department.

Improvements and Contributions:

- RF penetration through disaster environment
- Data communication protocol
- Antenna design for omni-directional communication

1.5.3 Electronic and Sensory / AI Specifications and Requirements

Power: The on board power had to be investigated to be sufficient for at least one hour of operation.

Control: The control station has full control over the robot, but the artificial intelligence has semi-control over the system.

Artificial Intelligence: The robot had to be able to detect objects that it needed to maneuver over as well as have the ability to transform when needed.

Gases: Dangerous gases in the environment needed to be detected, and a safety / danger analysis performed for a risk assessment for rescuers, victims and the robot. The gases of interest are methane, oxygen, carbon dioxide, carbon monoxide and oxygen.

Improvements and Contribution:

- A multi-agent robot control station.
- The artificial intelligence system for gas detection models and to warn the rescue workers of danger for humans and the robot.
- Develop equations for close estimations of gas concentrations for the sensors used.

The general system integration of the robot and control system is indicated in figure 1-3.

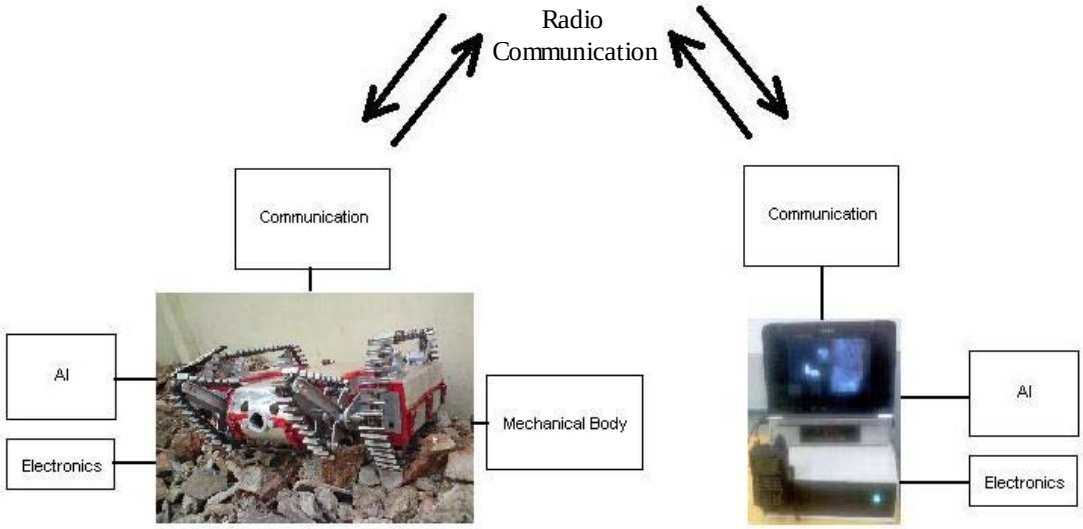


Figure 1-3: General system integration of the robot and control station

1.6 Training and Courses

Different courses and training were undertaken during the period of research, to achieve better understanding of the causes for disasters, the problems experienced, and different approaches to respective disasters. The following courses and training were completed:

- HAMNET Emergency Communication Training Certificate
- Emergency First Response Primary and Secondary Care
- FEMA Emergency Management Institute:
  - IS-100.a – Introduction to the Incident Command System (ICS 100)
  - IS-200.a – ICS for Single Resources and Initial Action Incidents
  - IS-271 – Anticipating Hazardous Weather & Community Risk
  - IS-700.a – National Incident Management System (NIMS), An Introduction
  - IS-804 – Emergency Support Function (ESF) #4 – Firefighting
  - IS-809 – Emergency Support Function (ESF) #9 – Search and Rescue
- Advanced Firefighting Certification
- Emergency Response Guide – HAZMAT Awareness Competency

1.7 Peer-Review Research Publications, Conferences and Presentations

Peer-review conferences and presentations were performed to assess the validity of the research and to receive feedback on the contribution performed. A peer-reviewed chapter in a book and journal papers have been submitted for publication, to inform other researchers of the possible ways to solve the problems experienced in USAR robot operations. Many of these publications have been referenced

throughout the thesis. Patents have been filed for the mechanical and communication systems implemented in the research. Various interviews and media publications on the research were done, which occurred locally, nationally and internationally.

1.8 Summary

A mechatronics concept is shown which indicates the integration of the mechanical, electronic and computer engineering systems. The history of previous problems and failures of USAR robots and the reasons for improvement and development were discussed. These problems have been experienced during tests performed at disaster sites around the world. The contribution and objectives were explained with the specifications in the areas of mechanical, communication, electronic and sensory / AI. Training and courses attended for research preparation and testing are given.

CHAPTER 2 – COMMUNICATION

The improvements made on the communication aspect of the research are described in this chapter. The context and reasoning of information is given in the following order:

- Introduction and reasoning for the use of UHF frequencies as implemented with the reviewed USAR robot.
- Introduction of the radio modules that were used and the alterations that were made to operate on the desired frequencies and transmission power.
- Explanation of the IEEE 802.11 Standard protocol and the reasoning why it was not used.
- Introduction and reasoning for the design and development of the implemented Robotics Communication Protocol (RCP) packet formation and communication procedure. Integration of the RCP with micro-controllers and other modules within the reviewed USAR robot.
- Video and voice communication performed with the modifications of purchased radios, thermal cameras and amplifiers.
- Discussion of the characteristics of the standard ½ and ¼ wavelength antenna designs. Explanation of the development of the ¼ wavelength antenna used in the reviewed USAR robot.
- Explanation of the characteristics of the loop shaped folded dipole antenna that were initially developed and implemented. The eggbeater antenna is introduced with the alterations required for the reviewed USAR robot.

2.1 Implemented Communication

The interference from other equipment that was experienced at the World Trade Center was mainly due to the robots using Industrial, Scientific and Medical (ISM) bands. Examples of equipment that caused interference, due to using the same frequencies, were other USAR robots and devices using bluetooth, wifi and similar technologies. Many electronic communication units use the ISM bands which are unlicensed frequencies that have certain constraints. Examples of these constraints are transmission power and signal transmission across territorial boundaries. As USAR robots are used to save lives, it is suggested that licensed frequencies are utilized. This will significantly prevent interferences. The output power between the control unit and the robot can be constrained to prevent a signal from one unit overwhelming the signals from other units.

Another reason for failed robot communication is the loss of signals between the robot and its control unit. This is caused by either the high frequency used as the signals cannot penetrate the building rubble or the regulations that restrict transmission power within these bands. As wavelength is inversely proportional to the frequency and the antenna size is proportional to the wavelength therefore the higher the frequency, the smaller the antenna will be. Higher frequencies are

capable of penetrating more dense materials than lower frequencies. Transmission efficiency decreases at higher frequencies as a result of factors such as the effect of dust particles, which resonate at these frequencies absorbing energy from the signals. Therefore it is best to use a frequency in the center of the two extremes that will allow optimization for radio communication. The comparison of the different factors that are considered are shown in figure 2-1.

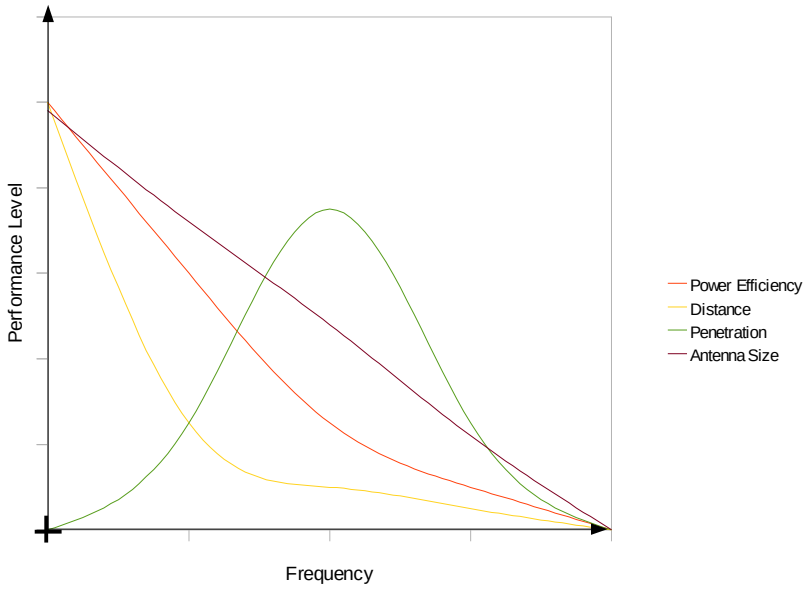


Figure 2-1: Comparison of factors considered as frequency increases

Figure 2-1 expressed a frequency range 300 kHz to 3 GHz. The power efficiency graph shows efficiency (%) vs frequency (Hz). As the frequency increases, the efficiency of the transmitters decreases. The distance of transmission decreases as the frequency increases. Low frequencies can be transmitted over distances of thousands of kilometers, as the waves reflect off the different layers in the atmosphere. Higher frequencies are only possible to transmit to a receiver in line of sight.

Low frequencies are not able to transmit through dense materials, while this is possible at the higher frequencies, up to a level of the UHF range. After this range, the penetration through dense materials decreases as the energy is absorbed by the resonating particles. The penetration distance is determined by the transmission power being used at the transmitter.

The antenna size is inversely proportional to the frequency. The robot requires antennas that are mounted within the chassis, therefore higher frequencies are desirable as the antenna size decreases to fit within the vehicle, but the other properties of the frequencies also need to be considered. Subsequently the decision

is to use UHF frequencies as these are able to penetrate with a relatively low power output and have a relatively good signal penetration property.

2.1.1 Radio Modules

The Radiometrix narrow band FM multi-channel UHF TR2M-433-5 transceiver modules were used for the data communication, as they are programmable for UHF frequencies. A photograph of one of these modules is shown in figure 2-2 [24].

Figure 2-2: Radiometrix TR2M radio module

The features of the TR2M modules are:

- Can be programmed to operate on any 5 MHz band from 420 MHz to 480 MHz.
- Fully screened
- 1200 baud dumb modem
- User configurable via RS232 interface (2400 baud)
- Low power requirements
- 4.5 V – 16 V power supply
- Current consumption: 110 mA transmit, 27 mA receive
- Operating temperatures: -10 to 60 °C

These data modules will be valuable for the USAR robot, enabling the programming of the modules to operate on the frequencies supplied by the fire department. The power consumption is low which is vital for power saving. As the thermal camera can be operated in negative temperatures, it allows for search operations in cold conditions. Insulation from the outside will allow for operation in further extreme conditions.

The only problem that occurs regarding these modules are their inability to transmit more than 10 mW. An output power of 5 W is required for efficient communication with the restrictions of buildings and other power absorbers. A RF amplifier is needed to solve this problem.

RF amplifiers that amplify 10 mW output power from the transmitter to 5 W are either not readily available or are expensive. The modules that are usually available have a minimum input power of at least 100 mW. In order to solve this problem, the final stages of Motorola MCX100 radios were used. The need arose for two of the three RF amplification stages as the amount of power that these final stages produce was sufficient, whereas the three final stages produce more than 5 W output power. Refer to Figure 2-3 for the interconnection of these stages. The disassembly and reconstruction of these stages require the addition of discrete components as not all the modules in the radio were used and the impedances of the missing modules are to be replaced.

The circuit of the RF amplifier was traced with a probe to determine the amplification of each stage. A discontinuation for a closed loop circuit was found by tracing the power point that was not powering the circuit of the first stage of the RF amplifier. This closed loop circuit was terminated to another module not used. By modifying the impedance on this point, a different output power was produced from the RF amplifier. It was determined that a resistance of 680 Ω limited power to the RF amplifier, which allowed it to produce 5 W output.

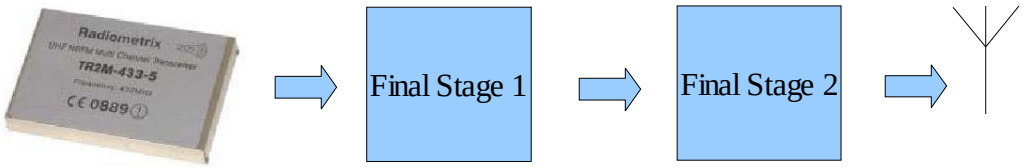


Figure 2-3: Transmission process block diagram

A problem occurred in the reception, as the signal was not able to reach the TR2M module from the antenna due to the RF amplifier not being bi-directional. This could possibly be solved by connecting the antenna directly to the TR2M module and then reception would be possible, but the high output power from the RF amplifiers would terminate the operation of the TR2M module, as there is high power penetrating the sensitive module.

This problem was solved by the implementation of a switching circuit on the output to the antenna. Figure 2-4 illustrates the concept of this circuitry. While the two relays are in position 1, the TR2M module can receive data. Should the TR2M module need to transmit, then the relays are switched over to position 2, which will then connect the TR2M module to the RF amplifier and in turn with the antenna. This prevents the need for two antennas and allows for only one radio module for data communication at each station. The switching is activated by the micro-controller that controls the serial communication.

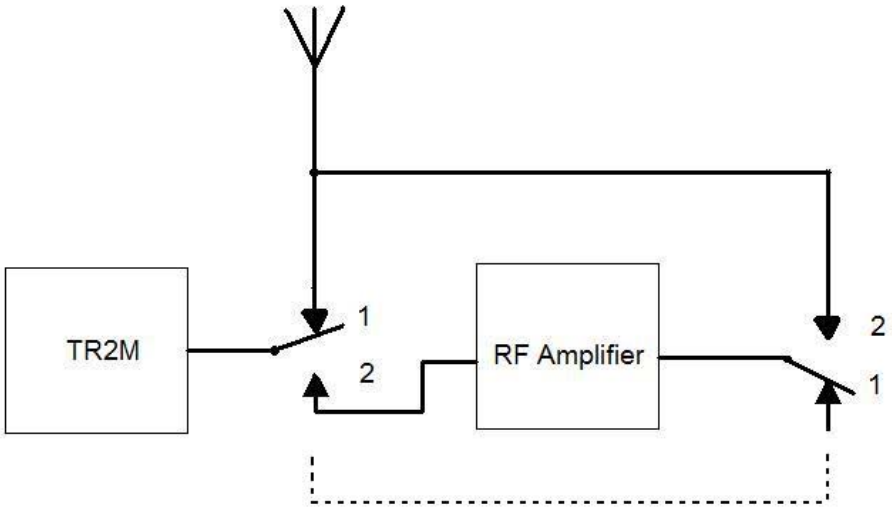


Figure 2-4: TR2M and RF Amplifier with the appropriate switching

The modules are connected to a computer via a RS-232 adapter. The configuration to connect these modules to be programmed by a computer is shown in figure 2-5 [24].

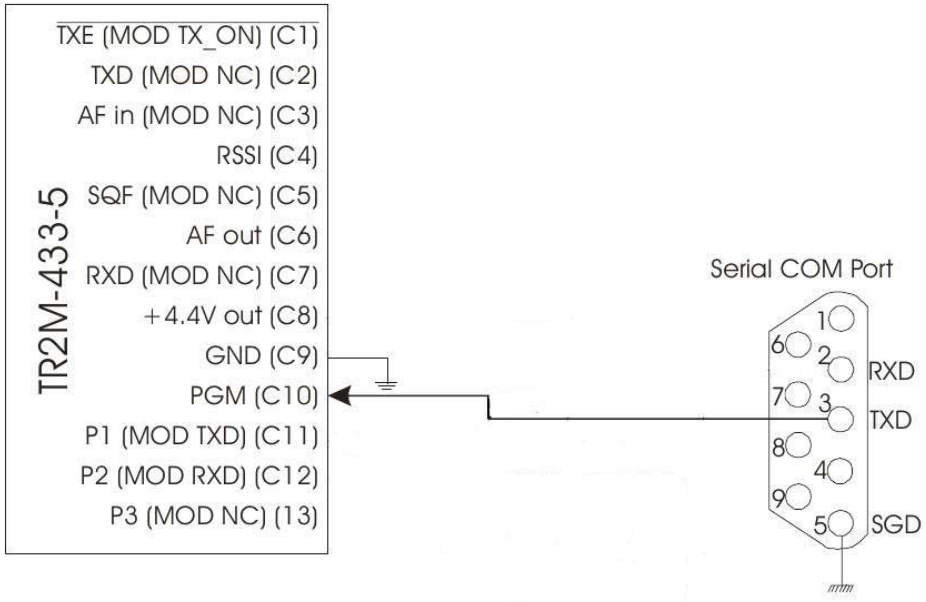


Figure 2-5: Configuration for RS-232 terminal programming

The modules are programmed with the function:

LOAD aa nnnnn

GOCHAN aa

where aa is the channel spacing, which is set to 00 as only a single frequency is being used

nnnnn is the synthesizer N register value which is calculated as:

$$N = \frac{f_{RF}}{25\text{kHz}} = \frac{449.725}{25kHz} = 17989 \qquad \{1\}$$

Both transceivers must be set to use 1200 bps, 8 Data bits, no parity, ½ stop bits and no flow control.

Regarding the above system, UHF and dedicated frequencies are used with 5 W transmission power from the antenna, allowing for penetration through building materials. A 10 km LOS communication is possible with this frequency and power transmission selection. This has not been used in previous USAR robots, which used 2.4 GHz frequencies and have 200 m LOS and about 2 rooms distance BLOS scenarios [16].

2.1.2 Protocols

The use of protocols is important for data to be successfully transmitted. Using available protocols was an option, but the performance and efficiency must be considered. Most existing protocols have been developed over many years and by various people. These protocols are optimized for best performance for a specific task. The IEEE 802.11 standard for Wireless communication (2.4 GHz) was investigated to determine its capability for USAR robot communication. After the problem of signal penetration was identified with Wireless communication, a decision was made to use the UHF frequencies 453.500 MHz and 449.725 MHz.

2.1.2.1 IEEE 802.11 Standard

The collision detection and IEEE 802.11 Standard is described by Brenner [25].

A wired Local Area Network (LAN) uses a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) mechanism. Should a system want to send a packet, it will first monitor to determine if there is traffic being transmitted on a network segment. If no traffic is detected the system will then transmit a packet. In the event of two systems transmitting at the same time, the collision detection protocol will identify that the packet was not transmitted successfully and will wait for a period of time before retransmission. The period of time that the system has to wait is determined by the exponential random backoff algorithm.

The exponential random backoff algorithm used is calculated by each individual system. The system must choose a random number (n) between 0 and a given number and wait for this number of slots before checking the medium again for transmission. The algorithm is executed in the following situations:

- The system senses that the medium is busy before transmission
- After each retransmission
- After a successful transmission

The collision detection works well on a wired LAN, but not on a wireless LAN. This is firstly because the duplex radios are needed to allow both transmission and reception simultaneously. This results in an increase of price for wireless equipment and two frequencies that are needed. Another reason is that assumptions can't be made that all stations can hear each other while this is definite with a wired LAN.

The IEEE 802.11 protocol can be described by means of two systems. Systems A is allocated in the wireless area and system B is the Access Point (AP). System A senses the medium, and if it detects that it is not active, then it will transmit a Request To Send (RTS) packet. Should system B receive the packet, it will verify the Cyclic Redundancy Check (CRC) and return a Clear To Send (CTS) packet. Once system A receives the CTS packet, it knows that no collision has occurred, or subsequently it will retry.

The RTS and CTS packets contain the duration of the required transaction. All stations in the wireless area will receive these packets, and will set their Virtual Carrier Sense indicator called Network Allocation Vector (NAV) for the given duration. This allows other stations to transmit after this time period while decreasing the possibility of a collision occurring.

The transmission of smaller packets is better considering that there is a higher bit error rate on a radio link, and the probability of a packet being corrupt increases with size. An additional reason is the smaller the packet, the lower the overhead will be in a retransmission should a packet corruption have occurred.

The IEEE 802.11 protocol could be used for communication between the robots, but there is not always an Access Point available for the wireless communication. The communication between the robots will be an Ad-Hoc style. Since UHF frequencies are being used, the data rate will be less in comparison to that used by wireless communication, as they use frequencies in the 2.4 GHz band and the quality factor bandwidth decreases as frequency decreases. Due to the bandwidth being decreased, additional collisions might occur and therefore smaller packet sizes are needed. More data transmission from other stations could occur when the packet sizes are smaller.

2.1.2.2 Robot Communication Protocol

The Robot Communication Protocol (RCP) [26] uses different protocol fields from the wired and wireless LAN protocols. The problem when using wireless communication technology is that it utilizes the 2.4 GHz band which causes the dust particles of buildings to resonate at this frequency and to absorb energy which can prevent penetration through buildings. A further problem with the use of the IEEE 802.11 protocol is that its packets contain header details that will not be utilized for the USAR robots. This results in unnecessary data being transmitted and will therefore occupy the use of the medium. In view of the fact that the baud rate of the data communication modules are 1200 bps, unnecessary data transmittance must be prevented as this can saturate the medium.

Another problem pertaining to the existing protocols is that they may possibly contain non-printable characters that cannot be processed by certain computers and micro-controllers. The printable characters are those that have an ASCII value between 31 and 127.

Utilization of a new wireless communication protocol is therefore required for USAR robots. A decision was made to use callsigns to identify the robots and control units to prevent communication interference. A six character callsign that consist of letters of the alphabet and numbers was assigned to each robot and control unit. This gives a combination of $36^6 = 2.17 \times 10^9$ different callsigns available.

There are two types of protocols that need to be transmitted namely: a “one way packet” that is sent from one station to the other and that needs no confirmation (referred to from now on as a Robotic One-way (RO) packet) and a packet which is sent from one station to the other and which replies with an acknowledgment of reception packet (referred to from now on as a Robotic Confirmation (RC) packet).

There are four packets for the robotic network namely, Request-To-Send (RTS), Clear-To-Send (CTS), Acknowledgment (ACK) and Data packet. The different packets with their fields are explained below.

RTS / CTS / ACK Packet

The packet format for the RTS, CTS and ACK packets are shown in figure 2-6.

Size	1 byte	1 byte	2 bytes	6 bytes	6 bytes	1 byte	1 byte
Field	Start	Type	Duration	RA	TA	Checksum	End

Figure 2-6: RTS / CTS / ACK Packet

Start: The start character is for stations to identify the commencement of the packet. This is indicated with the hash (#) character. Should a station only start receiving in

the middle of a transmission it will then recognize this and discard the packet. The purpose for the necessity of a start byte is that the transmission is asynchronous on a single channel.

Type: This field indicates the type of packet that is being sent. The indication for the RTS, CTS and ACK packets are the characters 0, 1 and 2 respectively.

Duration: The duration for the complete data transmission process from one station to another is specified in this field. This provides the other stations with the time period to delay before attempting to transmit. The duration is specified by the number of characters. Time periods are calculated from the sum of the two bytes multiplied by x, where x is the time period for each character to transmit.

$$x = \frac{8\,bits}{baud\ rate} \qquad \{2\}$$

Should the ASCII values of these fields be a “#” or “!”, then the most significant byte must be incremented and the least significant byte must be decremented.

RA: This is the address of the receiving station. This field presents the opportunity for other stations to identify whether that the packet is destined for them or not. Should the packet not be intended for the station, the rest of the incoming packet can be disregarded and the station can start processing other incoming packets after the delay duration.

TA: This is the address of the transmitting station and is used by the receiving station to identify whether the packet is from its approved station.

Checksum: This verifies the integrity of the packet. The field value consist of the sum of all ASCII values of all characters in packet modular 94 and the addition of 32. Should the receiving station receive a packet that is not approved then it is subsequently dropped. If the value of this field should be equal to ASCII “#” or “!” then the duration field is incremented and the checksum is recalculated. This field must be a printable character and not a control character (I.e. the character must have an ASCII value between 31 and 127)

End: This indicates the end of the packet with an exclamation mark (!) character.

Data Packet

The format of the Data packet is shown in figure 2-7.

Size	1 byte	1 byte	2 bytes	6 bytes	6 bytes	0–255 bytes	1 byte	1 byte
Field	Start	Type	Duration	RA	TA	Data	Checksum	End

Figure 2-7: Data Packet

Start: The start character is for stations to identify the commencement of the packet. This is indicated with the hash (#) character. In the event that a station only starts receiving in the middle of a transmission, this will be identified and the packet will be discarded. The motivation for a start byte is that the transmission is asynchronous on a single channel.

Type: This field indicates the type of packet that is being sent. The identification of a RO Data packet is the character 3 while for a RC Data packet it is the character 4. The other possible values (except for the character values for # and !) for this field are reserved for future use.

Duration: The duration of the transmission is given here. This provides the other stations with the time period that they have to delay with before attempting to transmit. The duration is given by the number of characters. Time periods are calculated from the sum of the two bytes multiplied by x, where x is the time period for each character to transmit.

$$x = \frac{8\,bits}{baud\ rate} \qquad \{3\}$$

Should the ASCII values of these fields be equal to ASCII “#” or “!”, then the most significant byte must be incremented and the least significant byte must be decremented.

RA: This field has the same purpose as in the RTS / CTS / ACK packet.

TA: This field has the same purpose as in the RTS / CTS / ACK packet.

Data: The data for instructions or information between the stations is stored in this field. An instruction could be a task that a specific robot must perform, while information could be what was requested by the control station. The only characters that are not allowed in this field are the hash (#) and the exclamation mark (!) seeing that these are the start and end characters respectively. Control characters are also not allowed in this field.

Checksum: This field has the same purpose as in the RTS / CTS / ACK packet.

End: This indicates the end of the packet with an exclamation mark (!) character.

2.1.2.3 Communication Procedure

The description of the communication procedure is described by means of two stations; station A and station B. Should station A want to transmit, it would observe whether no transmissions are occurring. If none are detected, then station A starts transmitting a RTS packet. All the stations in the vicinity of station A will delay transmission for the period of the duration field in the RTS packet. The delay duration period consists of the sum of the following:

- the time period needed to transmit the RTS packet
- the time period needed to transmit a CTS packet
- the time period for the Data packet
- the time period to transmit an ACK packet (if this is needed), which is equal to 18 (as the ACK packet has 18 characters).
- the sum of the processing time at each station, which is dependent of the processing speed at each station.

Station B receives the RTS packet and replies with a CTS packet which contains a delay duration period which is:

- the sum of the time period for the CTS packet
- the time period to transmit the Data packet
- the time period to transmit an ACK packet (if this is needed)
- the sum of the processing time at each station.

Station A responds with the Data packet that contains a delay duration period which is the sum of the time period for:

- the time period to transmit the Data packet,
- the time period to transmit an ACK packet if this is needed
- the sum of the processing time at each station.

Station B will reply with an ACK packet should the last received packet have a type value of 100. This packet will contain a delay duration period which is the sum of the time period to transmit the ACK packet as well as the processing time at each

station.

Given that there is no Access Point that is stationary in a disaster scenario, there is no single station that controls communication within the network as with a wifi network. In figure 2-8 four stations are shown with their respective radio coverage. C1 and R1 are control unit 1 and robot 1 respectively and C2 and R2 are control unit 2 and robot 2 respectively. Each station has an equal radius of communication distance. For communication between stations to be possible, the range circle needs to overlap with the transceiver of another range circle.

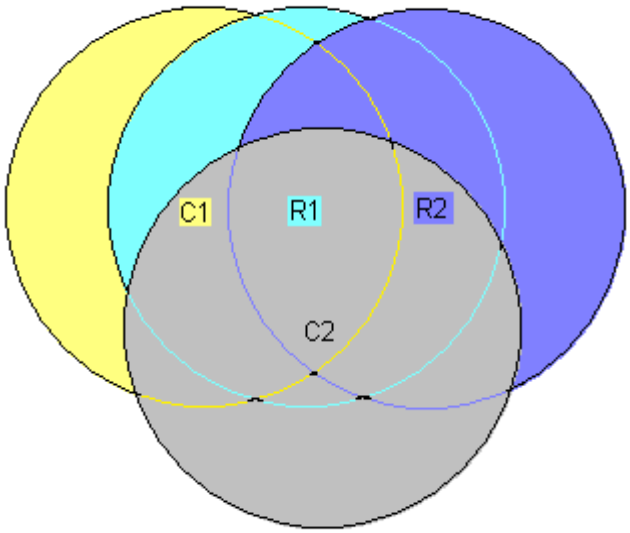


Figure 2-8: Radio Coverage of two control units and two robots

As noted in figure 2-8, C1 is in radio coverage with R1 and C2; R1 is in radio coverage with R2 and C2; R2 is in radio coverage with C2. Since C1 and R2 are not in radio coverage, packets requesting transmission will not be received between these two stations. This is not a great disadvantage as the different stations operate in an ad-hoc system. Of importance is the aspect that each robot is able to communicate with its own control unit.

Should a RTS packet be transmitted by C1 then R1 will receive the request and reply with a CTS packet. This CTS packet, which will contain a duration field, will be received by R2 as well. In view of the fact that R2 has received this packet, it will delay any transmission for this time period before trying to transmit again.

In the instance that both C1 and C2 transmit an RTS at the same time, R1 will then receive data that will be a combination of data from the two control units. R1 will reject this data, as it will not recognize it or, because it is not an acceptable packet. After a time-out period C1 will realize that R1 has not responded and will transmit the RTS again if required.

As the RTS packets are relatively small, the overhead of retransmission would be small if two stations should transmit at the same time. The sum of data being sent in the Data packet is limited to 128 characters and it need not necessarily be sent in a specific format, providing the format is understandable between the respective control units and the robots.

The advantage of the RCP is that a computer system could be connected to a modem that uses the same protocol and this modem could then transmit and receive instructions and data to a large network of robots. In this situation the computer would be the control unit and would not be dedicated to only a single robot. This network of robots could then be controlled to perform a task that could have a greater efficiency than that of a single robot, as implemented with swarm activity.

The RCP packets that are used to control a robot have smaller packet sizes of at least 38 % compared to those used by hard-wired computer network protocols and 33 % compared to that used by IEEE 802.11 protocol. Communication between the robots and their control units is more reliable when used in a network scenario. The use of a computer network protocol could be valuable when the robots have to transmit data and information that involves more than just the basic instructions. The RCP solves the data corruption problems experienced before with USAR robots that used the same frequencies for data communication.

2.1.3 Modular Approach for Layered Model

A layered model similar to the OSI model is needed for data communication. Each layer has its unique task to optimize the communication. The advantage of having a layered model is that each layer can be modified and optimized without affecting the other layers. The three layered model can be represented as indicated in figure 2-9 [26].

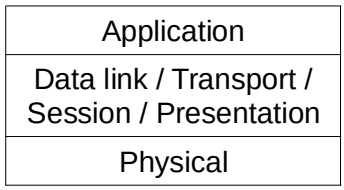


Figure 2-9: A three layered model

This model has been divided into three layers as each layer will be controlled by a separate module or micro-controller. The Physical layer consists of the hardware that will be used. In the case of the USAR robot, this will be the radio modules that will act as the transceivers.

The layer that is a combination of the Data link, Transport, Session and Presentation will be controlled by a single micro-controller, where the RCP was implemented. The

Data link layer is in control of the packets that are being sent, while the Transport and Session layer is responsible for the packet's control and transmission permission respectively. All the received data must be presented in an appropriate format for the computer to interpret. This is accomplished by the Presentation layer.

The Application Layer is involved in the displaying of the information, and with the interaction with the user. This layer is also involved in the output, being the movement of the motors and any other attachments of the robot. This layer will be controlled by a micro-controller which could be attached to other micro-controllers or modules, depending on the complexity of the attached module.

2.1.4 Procedures for Micro-controller Code

The code for the micro-controllers must be in a certain order. The code procedure and explanation below is in all probability not the only manner that this code can be presented. An explanation is given of the reasoning as to why these exact procedures were used.

- The declaration of arrays is stored in EEprom, as these arrays consume large amounts of memory and the RAM is limited.
- The sub-routine for the reception from the Physical layer is initiated when data is received. Should the character that is received be a start character, then data is stored from the beginning of the array, alternatively data is stored in an incremented position in the array. When the position in the array has reached the tenth position, then a confirmation is initiated as to whether the packet is intended for the station. If the packet is not for the station, there is no reason to continue receiving the data, and a delay for the time period that is indicated in the receiving packet is initiated. This prevents the station from receiving data that is not be appropriate for it. Once the end character is received, a checksum of the packet is verified and, should this be accurate, a flag is set to indicate that the packet is valid.
- The main program commences with the copying of the receiving packet into a different array. This allows the micro-controller to analyze a current packet even though the above sub-routine is initialized and data is currently being stored into the receiving array. Furthermore, the transmission array is cleared. A brief description of the main program structure is shown in figure 2-10.
- There are two types of conditions that may occur when data is sent. The first is that data is sent in response to receiving data from another station. This type of data is vitally important and must be sent as soon as possible. Any data that needs to be sent to the application layer is sent when it is received. The other type of packet that will be sent is a RTS packet, which the micro-controller will request from the application layer should it want to send data, in which case the RTS packet will be sent. In both of these circumstances, before any transmission can occur, the micro-controller must first verify that there is no transmission from other stations. This waiting for transmission will continue until no further transmission is noted.

This process will continue as the executing code is in an infinite loop. As the layers below the application layer are more important, the application layer will halt and receive data from these layers immediately, and then continue executing the application it is performing.

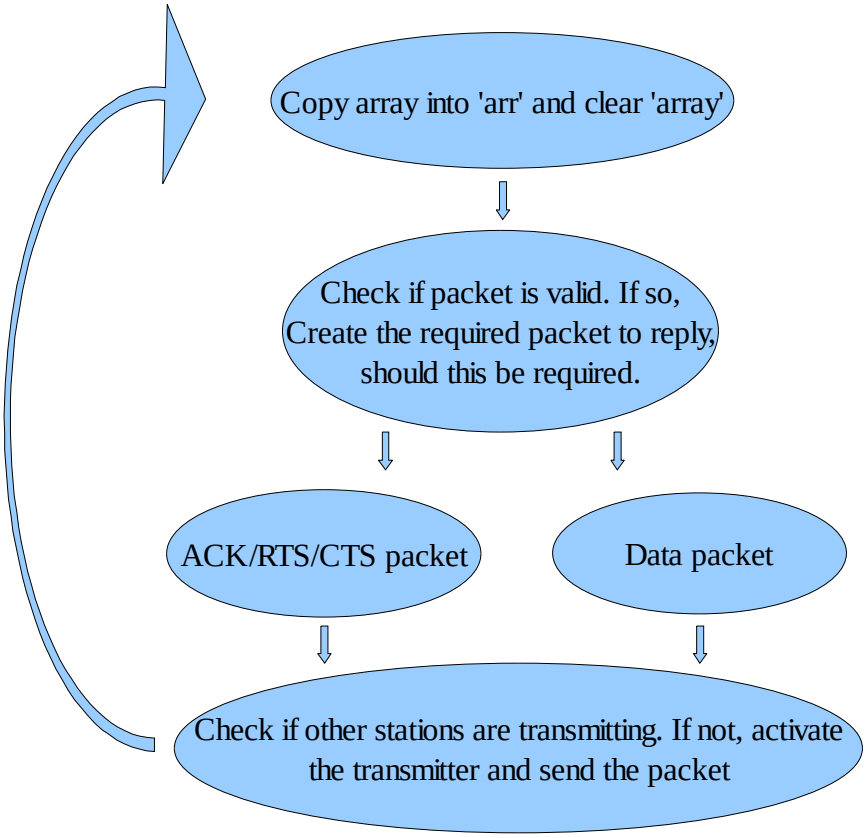


Figure 2-10: Brief description of the main program structure

2.2 Voice Communication

Voice communication between the robot and the rescuers is essential for the rescuers to get information from survivors. Rescuers can also calm the victims when the robot approaches and notify victims that help is on the way as well as possible ways to save themselves. The voice communication between the robot and the rescuers will be achieved through the video communication link as the video transmitters are capable of transmitting both audio and video. Communication between the rescuers and the robot is still required.

Two radios are needed for this communication to occur. Since one of the assigned frequencies is used for the data communication, the other assigned frequency is to be used for the voice communication. It was thus decided to use Amateur radios for this communication because of an acquired license that enabled the purchase otherwise restricted due to the radio regulations.

The decision was to use the Yaesu VX-7R and VX-3E transceivers. These radios can be modified to operate on these emergency bands and have different useful features. Diagrams of the Yaesu VX-7R [27] and VX-3E [28] are shown in figure 2-11.

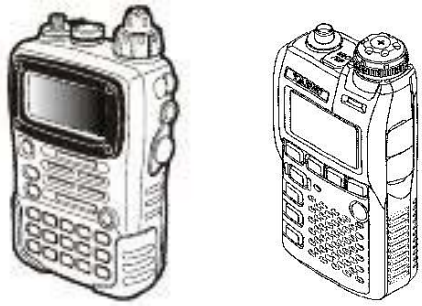


Figure 2-11: Diagram of the Yaesu VX-7R (left) and VX-3E (right) radios

The Yaesu VX-7R has the following features:

- Operation on UHF bands
- Supply Voltage: 10 – 16 V DC
- Current Consumption: 200 mA (Mono Band Receiver)
  1.9 A (UHF band, 5 W Transmission)
- Antenna Impedance: 50 Ω
- Operating Temperature: -20 ºC to 40 ºC
- Case size (W x H x D): 60 x 90 x 28.5 mm
- Weight: 260 g
- Microphone Impedance: 2 kΩ
- Receiver: N-FM, AM: Double-Conversion Superheterodyne
  W-FM: Triple-Conversion Superheterodyne
- AF output: 200 – 400 mW @ 8 Ω

The Yaesu VX-7R is used in the control unit. It has the useful characteristic of a keypad, allowing the rescuers to tune into frequencies other than those used for the robot, if so required. With this radio it is also possible for the rescuers to tune into the audio frequency of the video transmission from the robot, should the sound from the television be unclear.

The Yaesu VX-3E has the following features:

- RX: 420 – 470 MHz
- Antenna Impedance: 50 Ω
- Supply Voltage: 5 – 7 V
- Current Consumption: 120 mA (Receive)
- Operating Temperature: - 20 ºC to 60 ºC
- Case size (W x H x D): 47 x 81 x 23 mm
- Weight: 130 g
- receiver: AM, NFM: Double-Conversion Superheterodyne
  WFM: Triple-conversion Superheterodyne
  AM Broadcast / FM Broadcast: Single-Conversion Superheterodyne
- AF Output: 100 mW @ 8 Ω

The Yaesu VX-3E is primarily used for reception of audio at the robot. The useful characteristics of the Yaesu VX-3E is that it is small in size, light weight and can operate at temperatures that could possibly occur at the robot.

2.2.1 Yaesu VX-7R Modifications

The Yaesu VX-7R needs to be modified to operate in freeband mode. The following steps for the modifications should be performed [29]:

1. Remove the sticker that is found behind the battery pack.
2. A row of eight jumpers are found numbered 1 to 8. Jumpers 2, 4 and 5 are soldered.
3. Remove solder jumpers 4 and 5.
4. Reset the radio microprocessor (Press and hold in the 4, band, and v/m keys while turning the radio on.)

After this modification had been performed, transmission was available on 40 to 224.995 MHz and 300 to 579.995 MHz frequencies.

2.2.2 Microphone and Speaker Adapter

The audio input to the video transmitter, needs to have an impedance of 600 Ω and a maximum voltage of 1 V_{P-P} or 0.775 V_{RMS}. A 600 Ω dynamic microphone was initially connected to the input of the audio as there was no verification as to whether the video transmitter had a built in preamplifier. This did not appear to work, and a

mono microphone preamplifier was consequently used to amplify the signal from the dynamic microphone. While the preamplifier was connected to the transmitter, the preamplifier output was tested on an oscilloscope and the gain was altered to get a maximum output of 1 V_{P-P}. The schematic of the microphone preamplifier is shown in figure 2-12 [30].

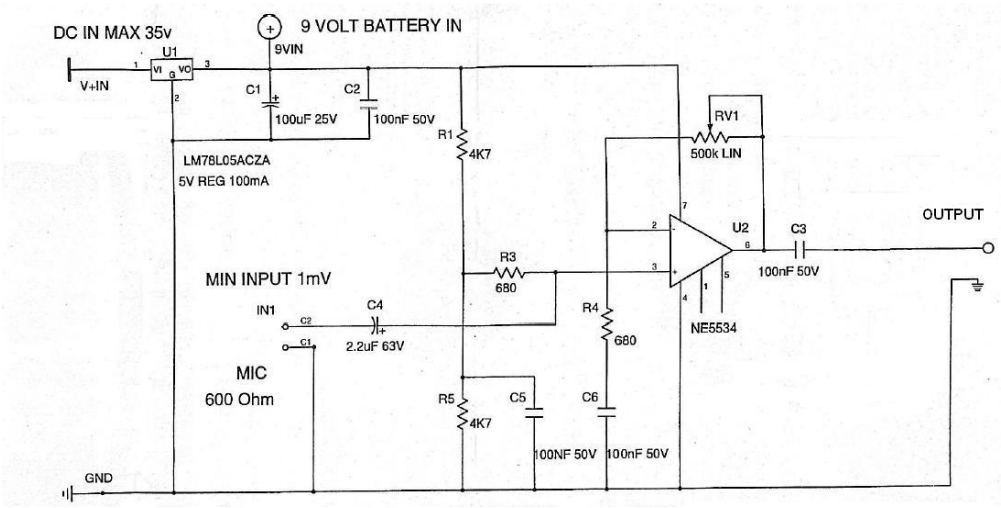


Figure 2-12: Schematic of the microphone pre-amplifier

The dynamic microphone used was manufactured from plastic which could result in a problem at high temperatures. Research had to be performed to determine the availability of high temperature microphones, but the research proved unsuccessful. Hence it was decided to continue using the plastic microphone to enable the testing of the principles to proceed.

An ear piece with a microphone was connected to the Yaesu VX-7R radio to allow the controller to communicate with any victims. The VOX-activation function could be set to allow transmission of spoken voice.

A 1.5 mm earphone plug connected to an 8Ω speaker was used for the Yaesu VX-3E radio. Enquiries were made to determine whether speakers were available that would be able to resist the high temperatures, but proved unsuccessful. An ordinary speaker was therefore used to prove the principle.

2.3 Video Communication

The video is from the FLIR PathFindIR thermal camera shown in figure 2-13 [31]. It has the following specifications:

- Size: (58 mm x 57 mm x 72 mm)
- Input Voltage range: 6V – 16 V
- Power dissipation: Less than 2 W

- Operating Temperature: -40 °C – 80 °C
- High Impact Resistance with heating element
- Weight: less than 0.4 kg

Figure 2-13: FLIR PathFindIR thermal camera

The PathFindIR is ideal for this robot, as it is small and is affordable compared to other available thermal cameras. It has a low power dissipation and can operate from -40 °C to 80 °C. Should the temperature decrease below -40 °C, the heating element is switched on, therefore allowing images to be transmitted in cold environments. A normal camera could be used for daylight scenarios, but most search operations are performed under rubble, smoke environments, mudslides or mines.

The video signal from the PathFindIR needs to be transmitted. ICASA (Independent Communication Association of South Africa) and Sentech have granted permission to use channel 54 (735 MHz) for video transmission, on condition that the output power is less than 1 W, and the transmitter is calibrated by one of their approved dealers.

A modulator and IF converter was used to generate the video on the required frequency. This signal was then amplified to 1 W using the amplifier shown in figure 2-14 [32].

Figure 2-14: 1 W UHF amplifier

These modules can operate between 470 – 862 MHz. It was confirmed that for search and rescue purposes, the minimum output power for communication should be 5 W [33]. As there was a restriction for the video output power, 1 W was used to prove the concept for this robot. It is suggested that a separate video frequency is assigned for search and rescue purposes so that the output power can be increased to 5 W.

A block diagram of the interconnection between the PathFindIR, converter/modulator, microphone, audio preamplifier, video amplifier and antenna is shown in figure 2-15.

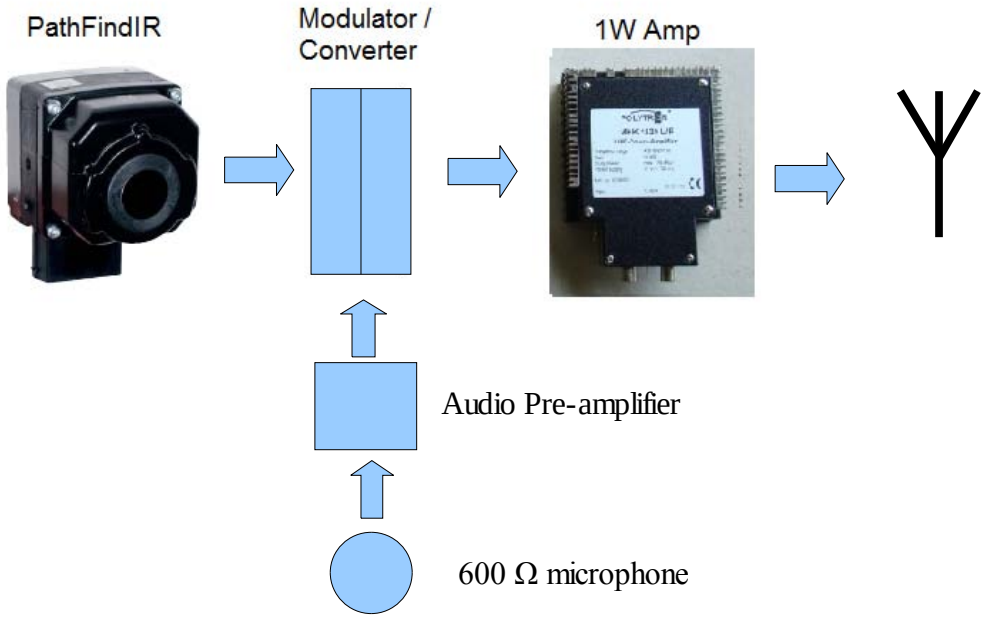


Figure 2-15: PathFindIR connected to the modulator/converter, 1 W UHF amplifier, audio preamplifier and antenna

A Teac 7” flat screen television was used for the reception of the video transmission. Any television could be used, but a television size that would be easy to install at the control station was used. The antenna of the television had a very poor performance and it was replaced with the eggbeater antenna.

With the above improvements and modifications, two-way audio communication was made possible with the use of dedicated emergency UHF frequencies and 5 W transmission power that penetrates building material, which was impossible with previous USAR robots. Previous USAR robots did not have thermal cameras fitted and in certain cases external equipment was constructed on the robot if there was space available and if the robot was able to carry the extra payload [3].

2.4 Antennas

Antennas are the source of transmission into the medium of air and the absorber of signals from the medium of air. Different antennas have different properties of radiation patterns and polarization. This is a topic in communication that is often neglected, but the antenna used has an effect on the performance in respect of the transmission and reception of signals. The antenna of a radio can influence the transmission and reception of signals. The calibrating and selecting of an antenna influences the efficiency of output power and of the signal strength that will be radiated from a radio.

The range of antenna types available was investigated [34]. The orientation of the antenna effects the polarization of the transmitted waves. It would be ideal to have vertical and horizontal polarization. The best antenna for this purpose is the egg-beater type. It gives vertical and horizontal polarization, but it has the disadvantage of being relatively large, which is not ideal, as one of the objectives of a USAR robot is that it should be as compact as possible.

Vertical antennas were investigated and a problem encountered was that the base plane shielded the signals from being transmitted through it. Different fractions of the wavelength antennas have different properties. A ½ wavelength antenna has radiation lobes that are perpendicular to the antenna, while the ¼ wavelength antenna has radiation lobes that are at an angle of about 45 degrees. Testing established that a property of the ¼ wavelength antenna, having a degree of output power directed towards the end point of the antenna, would work well. The only disadvantage of this type of antenna was that there was no radiation beyond the base plane.

This problem was solved by removing the base plane and replacing it with a piece of coaxial cable that was longer than the ¼ wavelength. The reason for the necessity of the base plane or coaxial cable was that it produced the negative part of the modulated sine wave. With the removal of the base plane, tests performed with a spectrum signal analyzer indicated that the radiation from the antenna was relatively isotropic, with low radiation towards the end points of the antenna. The antenna was then seen as a ½ wavelength dipole antenna. This isotropic radiation pattern was caused by the minor lobes that were permitted to be radiated next to the main lobe. In a scenario that the robot should be a number of wavelengths above the ground, the radiation pattern will become more isotropic because of more lobes, and will lower the elevation angle of the lowest angle lobe [35]. This antenna has the disadvantage in that it is not being vertically and horizontally polarized. This was solved by using an egg-beater type antenna that was scaled in size at the receiving unit. It was then able to receive any polarized signal.

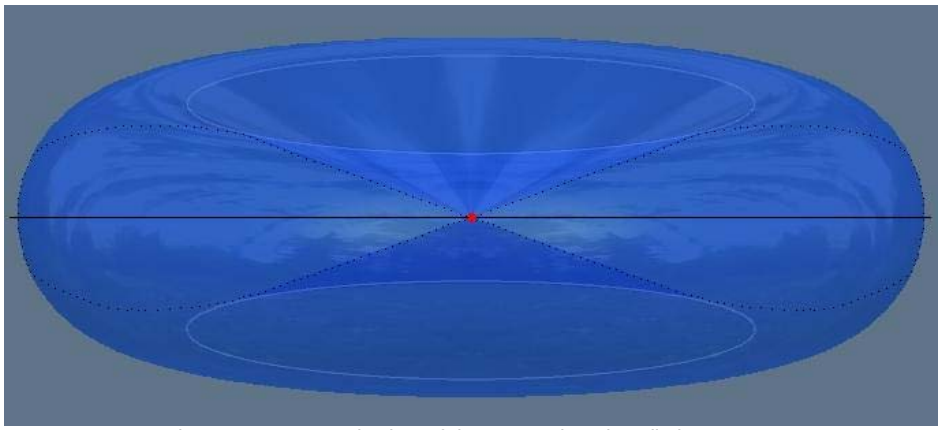


Figure 2-16: Isometric view of the ½ wavelength radiation pattern

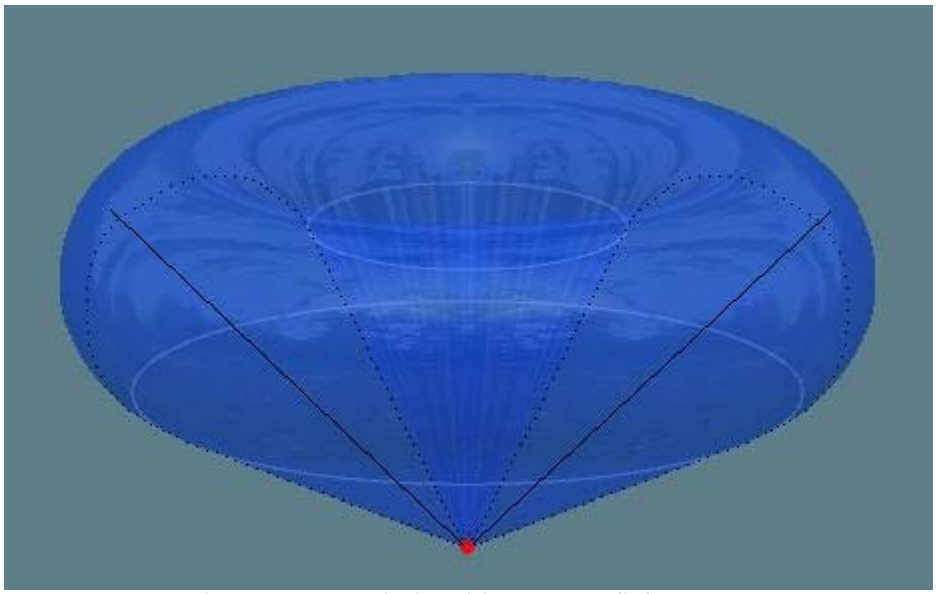


Figure 2-17: Isometric view of the ¼ wave radiation pattern

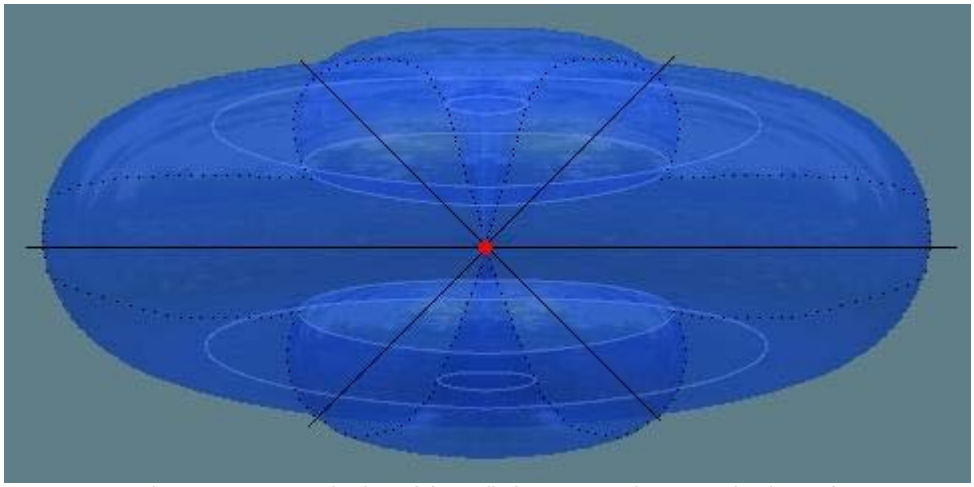


Figure 2-18: Isometric view of the radiation pattern of antenna that is used

Communication was improved with the use of UHF frequencies because, the penetration of the signal was increased. The ¼ wave antenna is approximately 167 mm and the transmission efficiency is still acceptable. With the use of a dipole antenna that has coaxial cable for the ground plane, the radiation pattern is increase by 100 % in terms of direction compared to an antenna that has a base plane. The radiation distance decreases as the output energy remains the same and is spread over a larger angle. The polarization of the radiated waves are in the same orientation as the antenna's orientation and can be received with an egg-beater antenna that is capable of receiving any polarized signal.

2.4.1 Quarter-wave Antenna Design

The length of the full wave antenna in free space is calculated from equation 4. This equation is valid for transmission in free space.

$$\lambda = \frac{300}{f} \qquad \{4\}$$

where: λ = wavelength in meters

f = frequency in MHz

The surrounding air and the environment has an effect on the antenna, so a factor η, which represents the discrepancy of the surroundings and has a value between 0 and 1, is multiplied with equation 4. The value of η is variable and depends on the antenna's surroundings. As the robot will be operated in conditions of smoke, heat and with various objects surrounding it, the value of η would vary.

SWR is the ratio of the forward and reflective power. Power is reflected back into the transmitter when the load does not have a matching impedance to that of the characteristic impedance. The SWR of a specific load can be calculated using equation 5 [36].

$$SWR = \frac{1+\sqrt{P_R / P_F}}{1-\sqrt{P_R / P_F}} \qquad \{5\}$$

where P_R = Reflected power

P_F = Forward power

From equation 5, it is seen that as P_R decreases, the SWR will tend to 1. To determine the SWR of a specific antenna, the meter is calibrated so that there is maximum deflection for the forward transmission of a signal, and then the reflected signal back into the system is read. This reading was performed every time an alteration of the antenna is made, until the SWR is close to 1:1. The ideal situation is to have a SWR of 1:1, but there are many factors that can influence this reading, such as surrounding objects.

Equation 4 was used to calculate the wavelength of the antenna. This length of

antenna wire was then cut and connected to the radio with a Standing Wave Ratio (SWR) meter, which was connected in series with the feed line. Millimeters of the antenna were trimmed away until the SWR value was very close to 1:1.

An antenna tuned for a frequency in the UHF band is generally viable for other frequencies in the UHF band. This characteristic is used to tune the antenna for a frequency of 450 MHz. With the use of equation 4, the antenna wavelength was calculated as:

$$\lambda = \frac{300}{450} = 667mm \quad \{6\}$$

The full wavelength was 667 mm, but since a quarter wavelength antenna was to be used, the antenna length required would be 167 mm. From this length, the antenna was trimmed until a SWR of 1:1 was obtained. This was necessary as the antenna was operated in an environment that was not the same as free space. The final antenna length was 170 mm, which consisted of a straight piece of copper wire.

2.4.2 Eggbeater Shaped Antenna Design

Different forms of the eggbeater antenna design were considered. The testing of the antennas was performed using an RF generator and a SWR meter. A receiver with a horizontal antenna was set up. The strength of the signal received by the receiver was then displayed on a signal analyzer.

2.4.2.1 Loop Shaped Folded Dipole Antenna

A folded dipole antenna was initially considered. This is a dipole antenna that is bent into a loop, bringing the ground and live point to each other, but not touching each other. The same configuration was used for another folded dipole antenna that was placed 90 degrees to the first one. The two loop antennas were separated with a quarter wave stub, so that the transmission between the two loops was 90 degrees out of phase and therefore prevented cancellation. The quarter wavelength coaxial cable stub must be shortened depending on the velocity factor of the transmission line. The velocity factor is the speed that a signal travels through a medium compared to the speed of light. This value typically varies between 0.6 and 0.7 for coaxial cables. The velocity factor of a RG-174 coaxial cable is 0.66. This quarter wave coaxial cable length can be calculated by equation 7 [36].

$$\frac{1}{4}\lambda = F\frac{75}{f} \quad \{7\}$$

where F is the velocity factor of the coaxial cable.

It is very difficult to determine the exact length of the coaxial cable stub, as the theoretical value does not correlate to that of the practical assessment. Therefore a Dip Meter was used to cut the exact length of the coaxial stub. The Kenwood DM-81 Dip Meter was used for this. A photo of this Dip Meter is illustrated in figure 2-19.

Figure 2-19: Kenwood DM-81 Dip Meter

The Dip Meter has a connection for a coil for the required frequency. A coil for a harmonic of 450 MHz was used. As the dial of the Dip Meter is not very accurate, the frequency counter that is on the Yaesu VX-7R was used to get the resonating frequency close to 450 MHz.

The Dip Meter was calibrated at the resonated frequency and a portion of coaxial cable was then placed next to the coil. A single loop then was made from a piece of wire and was soldered between the center conductor and the outside braid. Initially this loop was placed around the coil to get a broader band reading. The dial was cautiously turned, as it is very sensitive and requires fine tuning, until the Dip Meter was at full deflection. With this configuration, 2 mm pieces were cut from the coaxial cable, until it was detected that the Dip Meter was deflected towards zero. This was an indication that the coaxial cable being tested was absorbing most of the power at that frequency and that the coaxial cable was exactly a quarter wavelength with the velocity factor included.

The tests proved that the antenna were relatively omni-directional. There were a couple of cancellations of the signals because of the parallel sections of the two perpendicular antennas.

2.4.2.2 Eggbeater Antenna

The eggbeater antenna was thus considered for the communication between the control station and the robot. The eggbeater antenna consists of two loops that are perpendicular to each other. A quarter wavelength stub was placed between the two loops to cause the transmission between the two antennas to be 90 degrees out of phase. The tests confirmed that this type of antenna was more omni-directional than the loop shaped folded dipole antenna and had relatively rare areas of signal cancellation in the radiation pattern. There were occasional dips in the signal strength rather than complete cancellations.

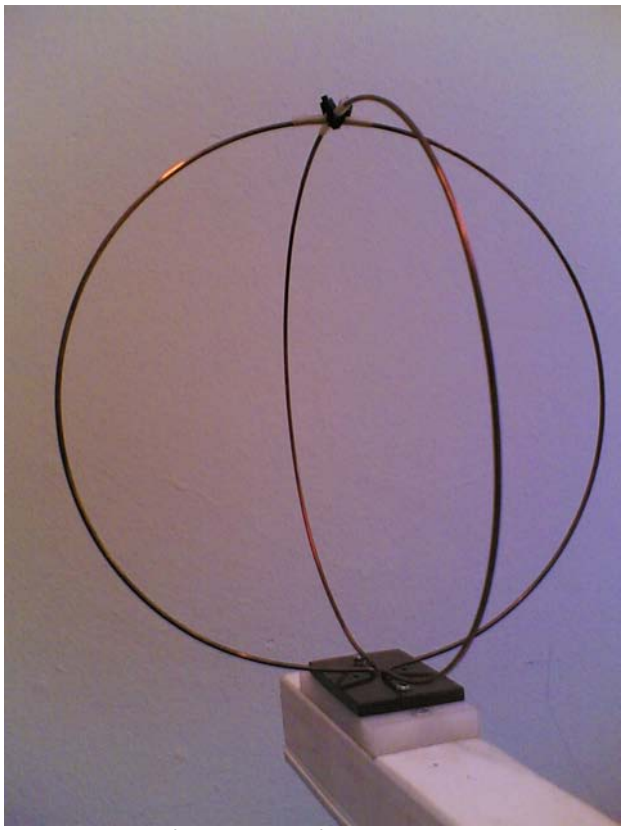


Figure 2-20: Eggbeater Antenna

The problem with this type of antenna is that the loops must have a full wave length circumference, making the diameter of the loop relatively large. This caused a space problem in the robot casing for this type of antenna (at 450 MHz). The loop can be made smaller, but then higher frequencies must be used. Since the UHF frequencies were preferred, it would not have been ideal to use smaller loops.

As UHF frequencies were desired, the decision was to use the eggbeater antennas in the control unit where space was not a constraint. Should the robot have contained an antenna that was polarized in a single direction, then the eggbeater antenna (that has horizontal and vertical polarization) would be able to receive the transmitted signal. The tested eggbeater antenna resonated between 440 MHz and 490 MHz, which was ideal for the available frequencies.

Audio, video and data communication was possible for an omni-directional communication with the use of the eggbeater antennas and the quarter-wave antenna with no base plate. Communication is possible even if the the antennas of the robot and the control station were not in the same plane. This improvement in communication has not been implemented before for USAR robots.

2.5 Summary

The Radiometrix TR2M modules with its related features, and the programming of the modules were discussed. Protocols and the basic procedure of the IEEE 802.11 protocol were explained, and a new robotic communication protocol, with its procedures of operation, was explained. The Robotic Communication Protocol has a decreased size of 33 % and 38 % compared to IEEE 802.11 and hard-wired computer protocols respectively. The problems experienced before with command corruption between the control station and the USAR robot was solved.

Voice communication with the Yaesu VX-7R and VX-3E radio's specific features was explained. The modification of the Yaesu VX-7R was also addressed. Voice communication was one of the features that were required with previous USAR robots used at the World Trade Center [7].

The microphone and speaker connections were discussed. Further research and development is needed regarding the development of microphones and speakers that are able to withstand high temperatures. The video communication between the thermal camera and a television receiver, for a successful observation of the robot's surrounding environment, was explained. The incorporation of the thermal camera in the robot allowed for thermal sensing of the environment without installing a separate thermal camera unit as was previously required at the World Trade Center [3].

Radiation properties of feasible antennas were discussed, and the advantages and disadvantages of the vertical and eggbeater antenna were clarified. The testing procedures and verification of the different antennas and the radiation performance of the chosen antennas were also addressed.

CHAPTER 3 – BODY DESIGN & CONSTRUCTION

The important factors that required attention with regard to the body of the robot were the durability and transformability. Different designs and concepts were considered, but essentially the need was for a robot that would be able to move over obstacles as well as in confined spaces. A tracked robot was therefore considered as the pulling force of tracks was 30 % more than that of standard wheels [23]. This factor resulted in a decision to conduct research to construct a transformable robot with the ability to change its shape to enable it to enter confined spaces and with the ability to maneuver over recurring obstacles.

3.1 Chassis Design

Three feasible concepts and approaches were considered regarding the design of a USAR robot. One of the possible designs is presented in figure 3-1.

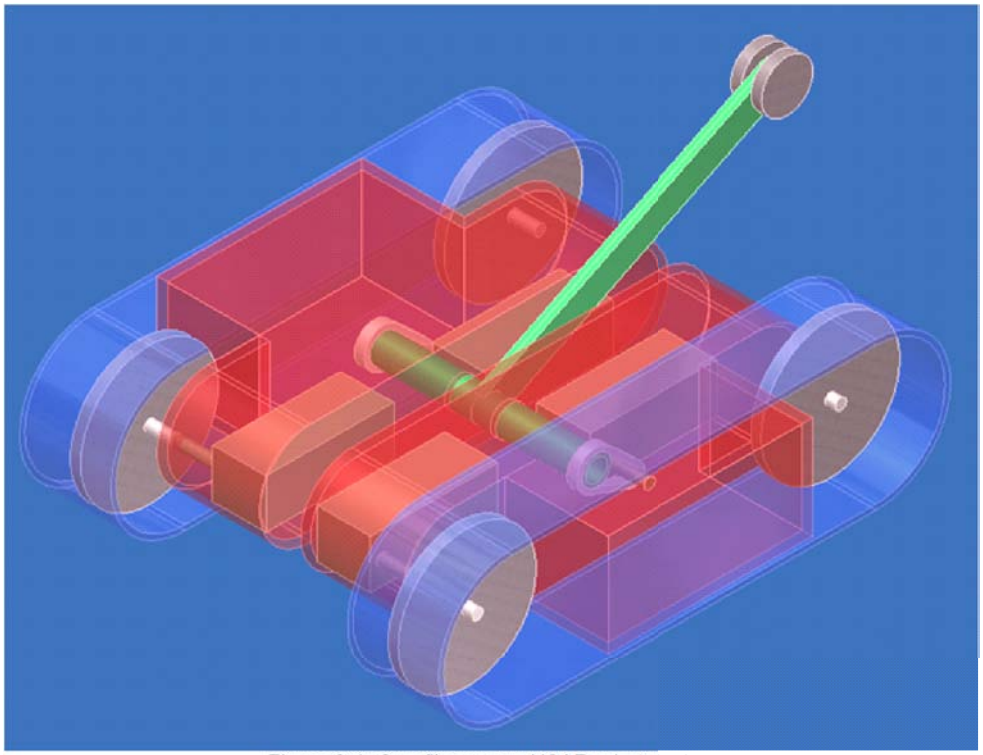


Figure 3-1: One flipper arm USAR robot design

In this concept the flipper arm used two motors that allowed it to rotate 360° with respect to the body, assisting the robot while climbing over obstacles. The body consists of two halves with the left hand side independent of the right hand side. This allows the robot to adapt its shape depending on the terrain it is maneuvering over. The power and data cables extend and protrude through the central shaft allowing the two halves to communicate with each other. This robot is steered by using differential control of the tracks.

This design was rejected as there were concerns that the flipper arm could cause the robot to become unbalanced and therefore obstruct the robot as it travels through the terrain.

A transforming USAR robot was considered which had different compartments that rearranged to change the robots shape. The Houdini Rescue Transformer Robot, which was developed by UKZN MR^2G, was able to change both its height and width by 200 mm and was named HRTR22 [37]. A photo of this robot is shown in figure 3-2.

Figure 3-2: HRTR22 USAR robot

The HRTR22 was built to hold all the components that were required. It was able to maneuver over obstacles of 30 mm, which was sufficient to move over gravel areas but not sufficient to climb stairs. This transforming ability allowed the robot to enter into a range of gaps with different dimensions. The transformation process took a duration of 1 minute and 47 seconds, which was too time consuming considering that the USAR robot should be able to locate possible victims speedily and would required to transform a few times before reaching a survivor. It was then determined that a robot body design was required that would keep the structure as small as possible yet allowing for transformation to assist in maneuvering across obstacles.

Instead of the single central flipper arm, two small flipper arms are used in the front and the back of the vehicle, causing the robot to be stable while transforming. This resulted in a decision to use arms with tracks that would elevate the robot over obstacles as well as allow the robot to have a greater surface contact when climbing stairs. The primary advantage of the Flipper Arm configuration is that it allows the robot to be invertible (i.e. the robot can be placed upside down and would still function), and reversible (i.e. the robot can operate in a reverse direction just as capably as it would in the forward direction). A diagram showing the concept is shown in figure 3-3.

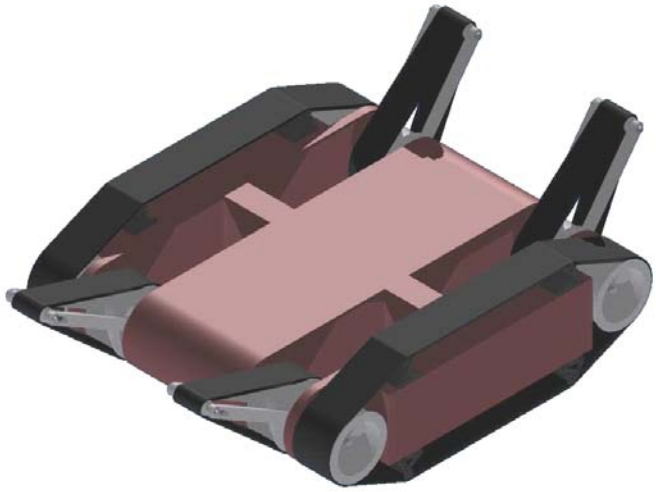


Figure 3-3: Conceptual design of the USAR robot, consisting of two tracked flipper arms at the front and two at the back of the vehicle

The robot design was modeled and different configurations for transformation were investigated. Different configurations and views of the modeled robot are displayed in figure 3-4.

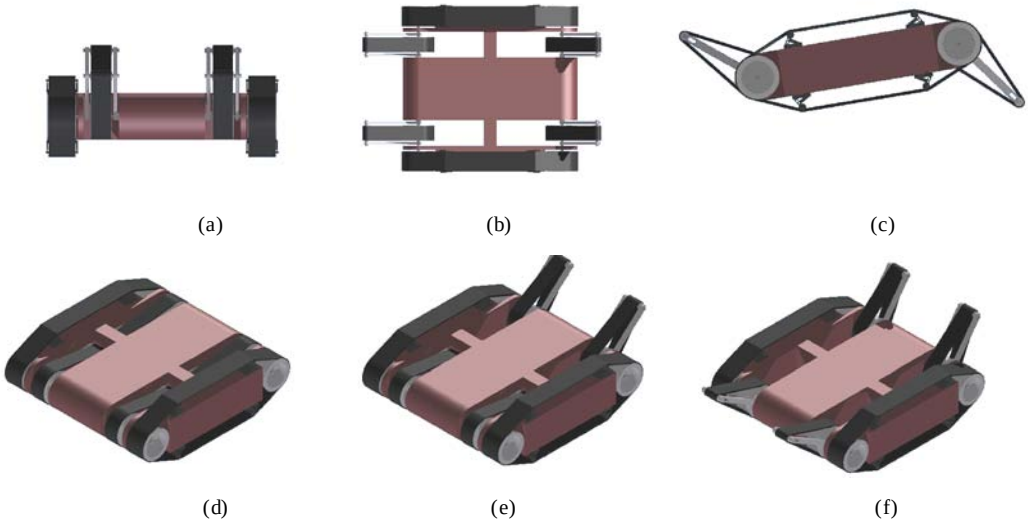


Figure 3-4: Different configurations and views of the robot. (a) Front view, (b) Top view, (c) Side view showing the arms lifting the robot, (d) Isometric view with the arms contracted, (e) Isometric view with the back arms contracted, (f) Isometric view with the front and back arms extended

The chassis dimensions were determined by the size of the different modules that were needed for the performance of the robot. These were modeled in Autodesk Inventor and inserted into a draft diagram to minimize the robot's size. A diagram of the module configuration is shown in figure 3-5.

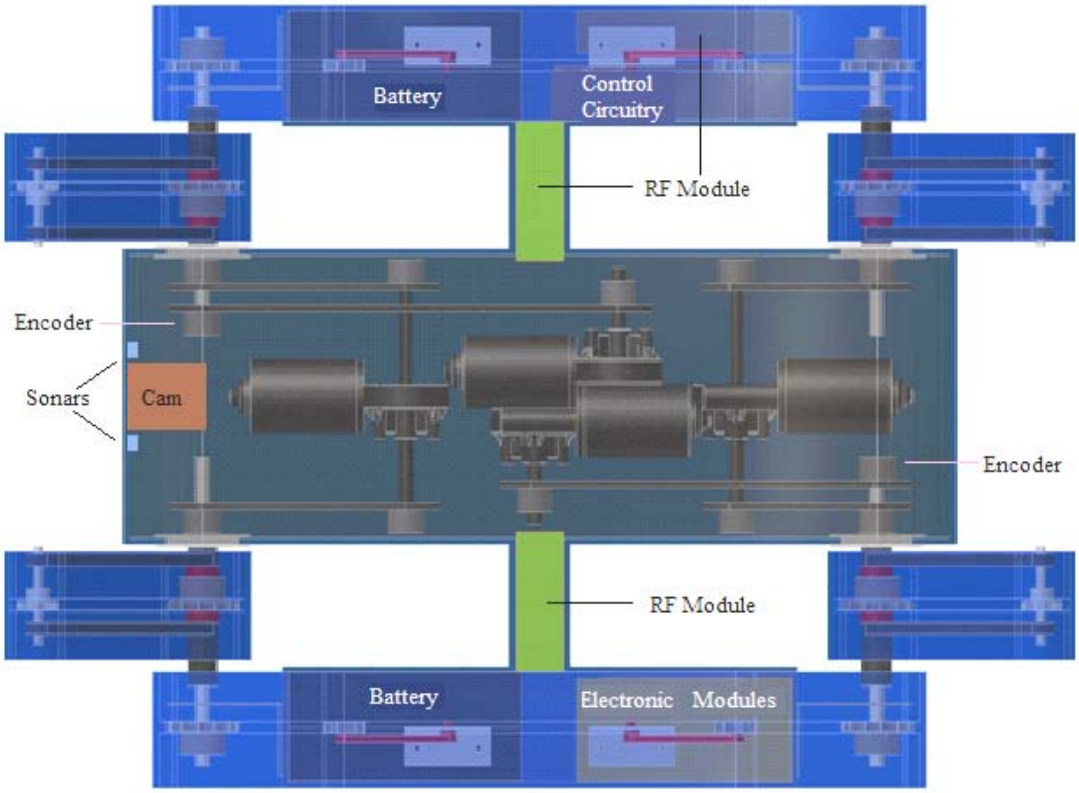


Figure 3-5: Modules configuration within the robot as seen from the top

The dimensions of the complete robot design are the following:

- Height of composite chassis: 150 mm
- Length of robot excluding extended flipper arms: 730 mm
- Length including the extended arms: 1090 mm
- Width: 700 mm

A simulation of the robot in a search and rescue scenario was generated to determine the size ratio of the robot compared to the rubble in a disaster area. An example of a disaster scenario simulation is shown in figure 3-6. The simulation verified the size of the rubble that the USAR robot would have to negotiate. This provided information on how the robot would maneuver across such terrain. Further modeling of the robot was performed to evaluate how it would react in certain scenarios and conditions. The conditions that were considered were the maneuvering over obstacles such as climbing stairs, and transforming to be able to cross rough terrain. The specifications and requirements were taken into account. Once the performance in the modeling environment was performed and the modeled robot was able to maneuver over and through the simulated obstacles, the construction of the robot was started [38].

Figure 3-6: Scenario simulation of the robot in a disaster area

The design considered allows for maneuverability over uneven terrain and over obstacles that might be encountered. At the same time, it allows for transfomability that will decrease the robot's size, therefore giving it the ability to go through concealed and tight regions. The target dimensions of gaps that the robot has to pass through is 1 m wide and 500 mm high. This was not possible with previous USAR robots that had all the required modules and components [4]. It was decided to name the robot CAESAR, which is an acronym for Contractible Arms Elevating Search And Rescue, as it has flipper arms that are able to contract when needed.

3.1.1 Materials Selection and Design

The decision was taken to fabricate the body of the robot from composite materials to save weight and to control the thermal transmission gradient between the operating environment and that of the inner electronic components. The alternatives (constrained by availability), namely metals, have high thermal transmission coefficients and high density and would therefore be less suitable.

The reason for considering composite materials was determined by their strength properties. The stiffness $K = ES/l$ characteristic is given to the mechanical performance of a beam. The ratio of bending stiffness of two arbitrary materials 1 and 2 is calculated from [39]:

$$\frac{K_1}{K_2} = \frac{E_1 S_1}{E_2 S_2} \cdot \left(\frac{l_2}{l_1}\right) \qquad \{8\}$$

where E_x = Young's modulus

S_x = Cross-sectional area

l_x = Length of the material

The ratio of the weights of the beams is:

$$\frac{m_1}{m_2} = \frac{E_1 l_1}{E_2 l_2} \cdot \frac{\wp_1}{\wp_2} \qquad \{9\}$$

where ρ_x = Density of the material

m_x = Mass of the material

Combining equations 8 and 9,

$$\frac{K_1}{K_2} = \frac{E_1 / \wp_1}{E_2 \wp_2} \cdot \frac{m_1}{m_2} \cdot \left(\frac{l_2}{l_1}\right)^2 \qquad \{10\}$$

With the comparison of mechanical performance of two materials, the same length and weights must be used. Therefore;

$$\frac{K_1}{K_2} = \frac{E_1 / \wp_1}{E_2 \wp_2} \qquad \{11\}$$

From equation 11 it was determined that the best material would be the one with the highest specific modulus (E/ρ). During comparisons it was established that Kevlar had a specific modulus value of 87 MN m/kg, which was higher than that for steel, aluminum alloys and tungsten [39].

Phenolic resin is a thermosetting resin and was used as it had the following characteristics [39]:

- excellent dimensional stability
- good thermal stability
- good chemical resistance
- low shrinkage
- good mechanical characteristics
- low cost
- dark colors of the resin

Kevlar has the properties of high impact resistance and damage tolerance as it causes the absorption of energy by widespread delamination and splitting [39].

A stress and deformation analysis of the robot was performed with an initial robot

body weight of 25 kg. The complete weight of CAESAR is 56 kg, but only the weight of the body and internal components is considered in this situation. An indication of the results are shown in figure 3-7.

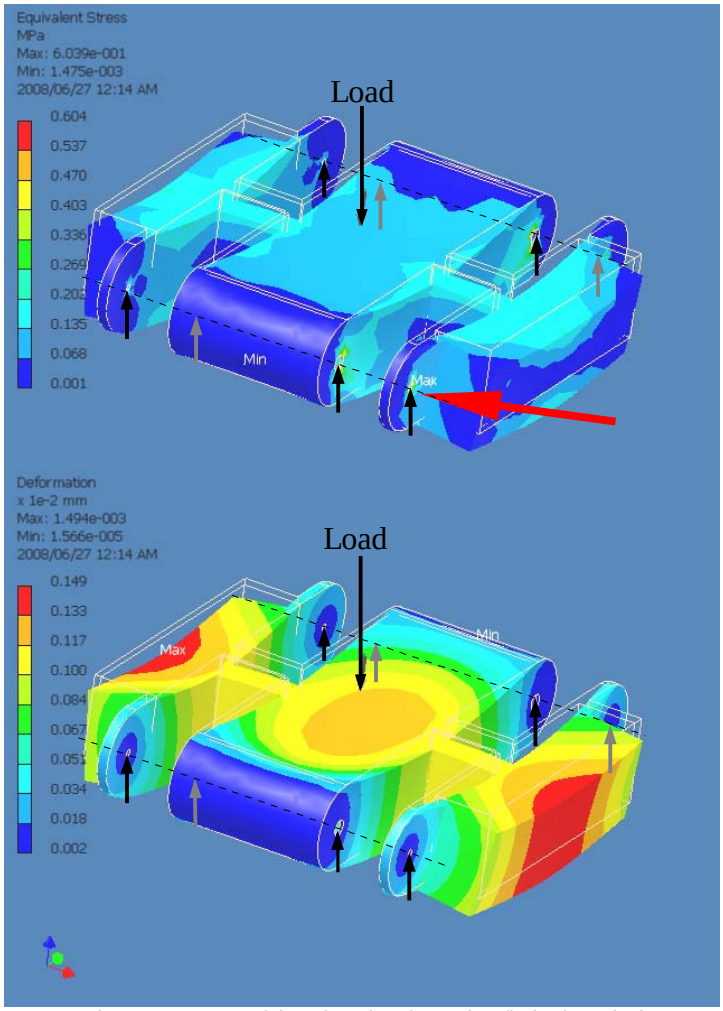


Figure 3-7: Stress (above) and Deformation (below) analysis

As observed from the stress analysis, the maximum stress is 0.604 MPa (indicated with the red arrow), which occurs mostly along the axis areas. Taking into account that the maximum shear stress for mild steel is 210 MPa, the shaft design is safe from shearing. The maximum deformation of the robot body is 0.001494 mm, which is an acceptable value.

Bending and tensile tests were performed to determine whether the composite material that the body was manufactured from was sufficiently strong for the environments it would be exposed to [37].

The bending test were performed with an Instron 5500R. These tests were performed according to ASTM D790 standard, which required a three point loading test. The Instron 5500R was set to a constant cross speed of 12 mm per minute at

room temperature. The pitch was set at 176.5 mm. The test specimen's dimensions were captured by the Instron software and were the following:

Specimen 1: 35 mm x 290 mm x 5 mm (syntactic soric foam core thickness: 1 mm)
Specimen 2: 30 mm x 290 mm x 8 mm (syntactic soric foam core thickness: 4 mm)
Specimen 3: 40 mm x 290 mm x 3 mm
Specimen 4: 35 mm x 290 mm x 3 mm
Specimen 5: 30 mm x 290 mm x 3 mm

Figure 3-8 shows the results that were obtained.

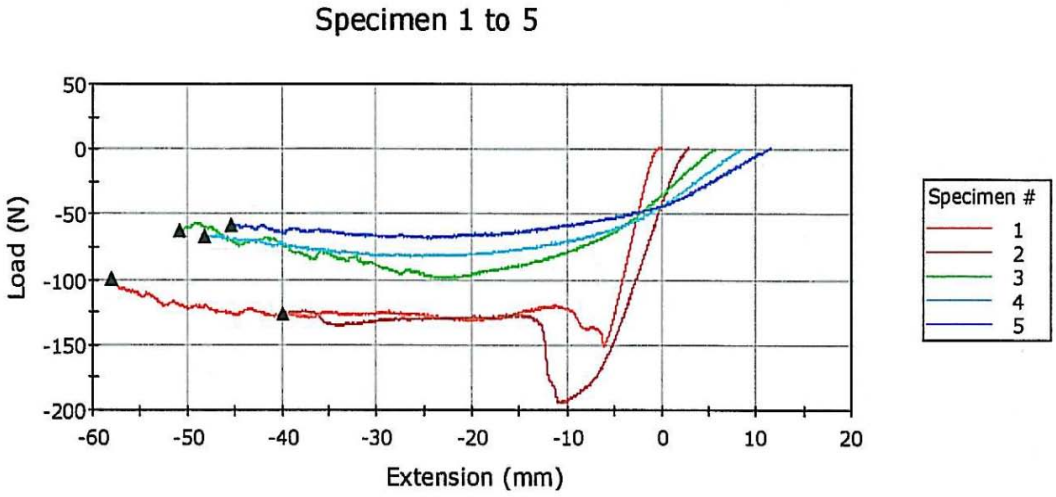


Figure 3-8: Bending test results

Specimen 1 and specimen 2 each consisted of a different thickness syntactic soric foam core. As seen from the graphs, specimen 2 which had a thicker syntactic soric foam core, was able to withstand a greater force and an extension of about 6 mm. Specimens 3, 4 and 5 had no syntactic soric foam core and these appeared to be less able to withstand force. Results indicated that the substrate could absorb more energy with the syntactic soric foam core than without it. This is a result of the load distribution across the substrate. The three modes in which failure occurred is matrix failure, fiber failure and delamination. These results shown are only that of a straight piece of specimen. The composite body consists of curves and of two halves, therefore increasing the strength and enabling it to resist a greater force before disintegrating.

The bending stress can be calculated from equation 12.

$$\sigma_f = \frac{My}{I} = \frac{F.\frac{L}{2} x \frac{t}{2}}{\frac{w.t^3}{12}} = \frac{3FL}{wt^2} \qquad \{12\}$$

$$= \frac{3 x 150 x 176.5}{30 x 8} = 330.938 MPa$$

where σ_f = bending stress (MPa)

M = Moment (Nmm)

y = Distance from cross-sectional neutral point to point of maximum tensional stress (mm)

I = Moment of Area (mm^4)

F = Force (N)

L = pitch length (mm)

t = thickness (mm)

w = width (mm)

Tensile testing were also performed on a specimen of which the dimensions were 17 mm x 290 mm x 5 mm. This allowed for investigation of the composite materials performance in a situation should bending have occurred. The results from this test is recorded in figure 3-9.

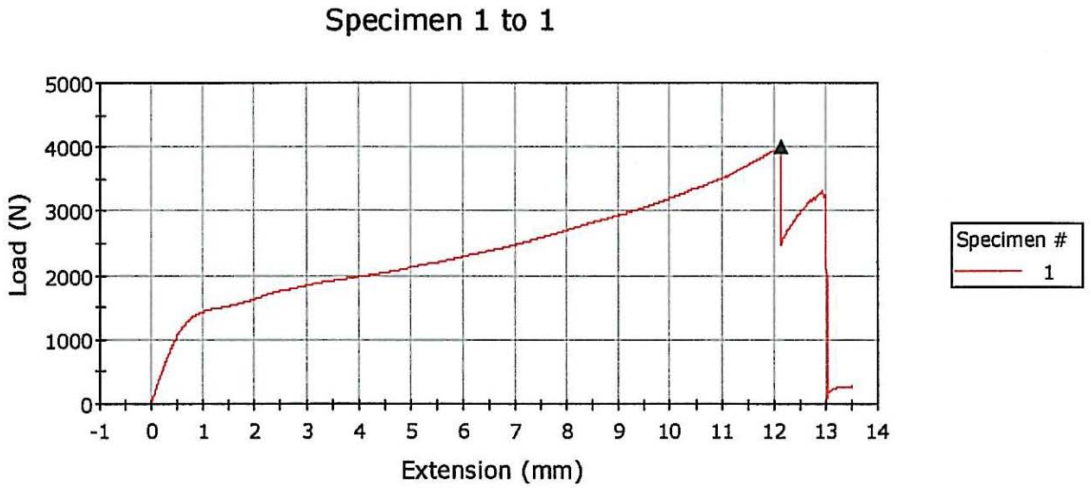


Figure 3-9: Tensile test results

These test results indicate that the specimens extended linearly as the load increased. The point where the linearity deviated is about 3000 N. Failure occurred at about 4000 N.

From this the shear stress can be calculated as shown in equation 13.

$$\sigma = \frac{F}{A} = \frac{F}{wt} \quad \{13\}$$

where: σ = shear stress (MPa)

F = Force (N)

A = Area (mm^2)

w = width (mm)

t = thickness (mm)

therefore, $\sigma = \frac{3000}{17\,x\,5} = 35.294\,MPa$

After the materials selection and design had been confirmed, it was possible to initialize the construction phase.

Tests were performed to verify whether the shell of CAESAR is able to withstand temperatures of 200 °C. The composite material test pieces were inserted into an oven and left for a time period of an hour. The composite material did not disintegrate. The phenolic resin changed to a red color as it cured. This strengthened the composite structure.

3.1.2 Construction

Once the final dimensions were determined a diagram of the mold was developed. An extra 5 mm was added along the border of the mold to compensate for the thickness of the composite materials. From these dimensions, the mold was constructed using hardboard, chipboard and a wooden base. The wood was nailed into position and plaster-of-paris was used to round the front and back corners of the robot. A layer of epoxy resin was applied to seal the wood and to smooth all cracks, irregularities and sharp corners. This is shown in figure 3-10. After the varnish had dried it was ready for the application of the release agent.

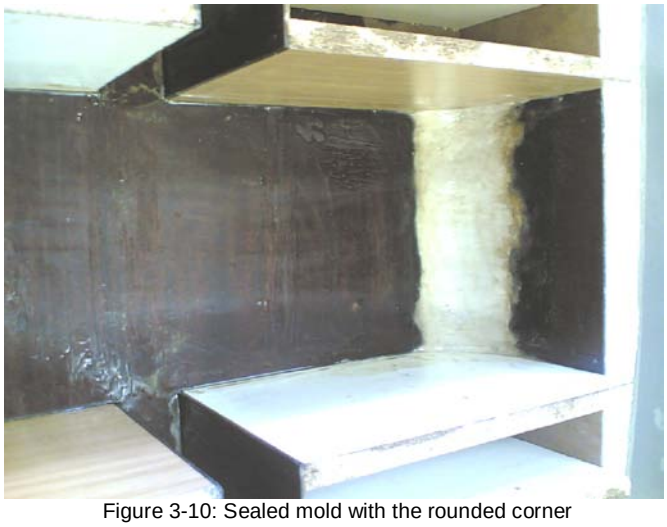


Figure 3-10: Sealed mold with the rounded corner

Different release agents were tested to determine which would release the mold from the phenolic resin. The suppliers suggested many products which were recommended from the manufacturers, but these did not function as required. A piece of wood was taken and treated similarly as to the mold. The following products were tested for use as release agents:

- Ram wax
- Release Agent 827
- Moldwiz F75
- Aerosol Oil
- Elastrosil M4641 silicone

Most of these products did not release the phenolic resin. The best releasing agent proved to be Elastrosil M4641 silicone and this was therefore used. After the Elastrosil M4641 was applied to the mold, it was left to dry. Before the application of the composite materials and phenolic resin an aerosol oil was applied to the surface.

The Phenolic resin was mixed with the catalyst, Phencat 10. Less catalyst than suggested was used, so that the phenolic resin would harden at a slower rate. A thin layer of phenolic resin was applied to the mold and a pre-cut Kevlar layer was rubbed onto the surface until the resin was absorbed. Kevlar has the strength property that prevents objects penetrating the body should anything fall onto it. Phenolic resin was applied with a brush on top of the Kevlar so that the air bubbles were removed. This was then left to dry.

The composite shell in the mold with the identification code [90/45/0/90] is shown in figure 3-11. Each value within the identification code represents the angle of each

layer with reference to the 0°. This angular configuration allows the mechanical properties of the Kevlar to be applied at different angles. After the first layer had dried, resin was applied to the previous layer (which is considered to be at 90°) and another layer of Kevlar was placed into position at 45°. A layer of fiberglass was placed after this layer (at 0°) and then a layer of solid syntactic soric foam core was applied to give the chassis shell strength as it combines lightness and flexural stiffness [39]. Kevlar has a thermal conductivity of 0.04 W/m/K [40], which is the same for fiberglass [41]. Phenolic resin is able to withstand temperatures of over 300 °C and has a thermal conductivity of 0.4 W/m/K [42]. The composite shell was completed with another layer of Kevlar placed at 90° to the fiberglass. This was left to dry. A layer of glass wool packed between the components and the chassis would provide more insulation as it has a thermal conductivity of 0.04 W/m/K [41].

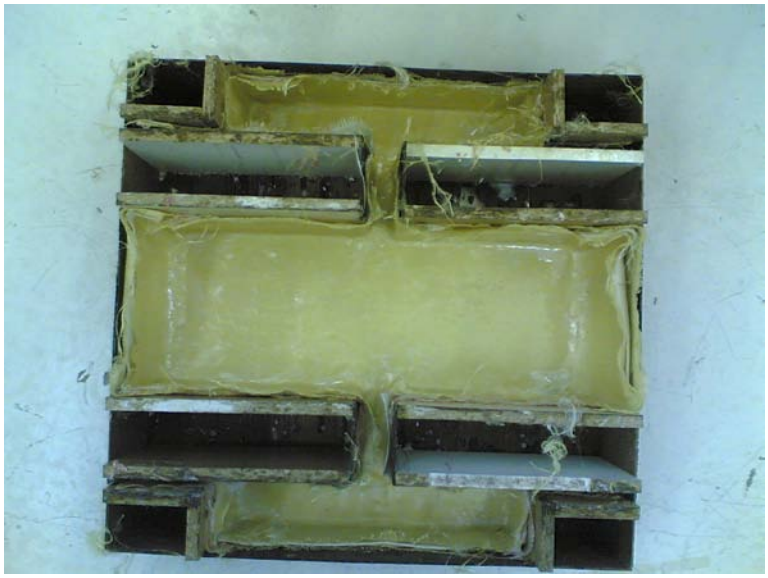


Figure 3-11: Composite shell inside mold

After the composite shell had dried, it was removed from the mold. This is shown in figure 3-12.

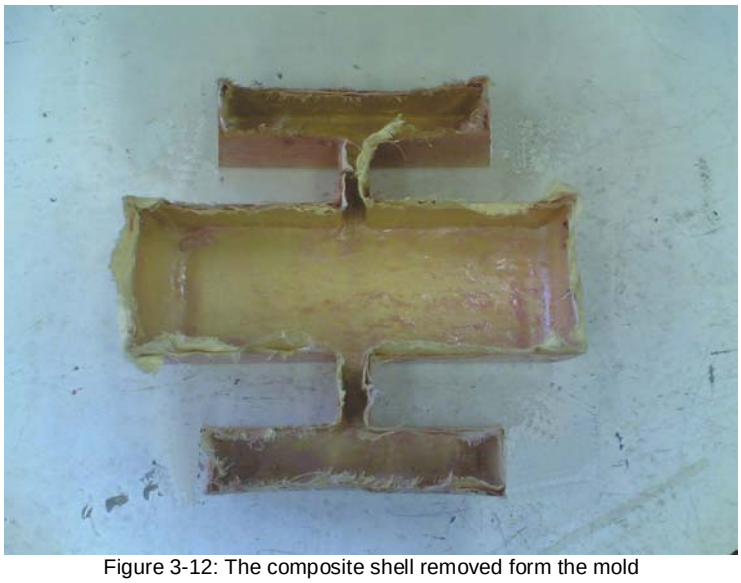


Figure 3-12: The composite shell removed form the mold

The same mold was again prepared and the Elastrosil M4641 silicone and aerosol oil and the composite application process was repeated for the other half of the robot chassis.

These composite shells were made higher than the design so that they could be cut to the correct height. This gave a smooth edge and a equal height of 75 mm around the edges as shown in figure 3-13.

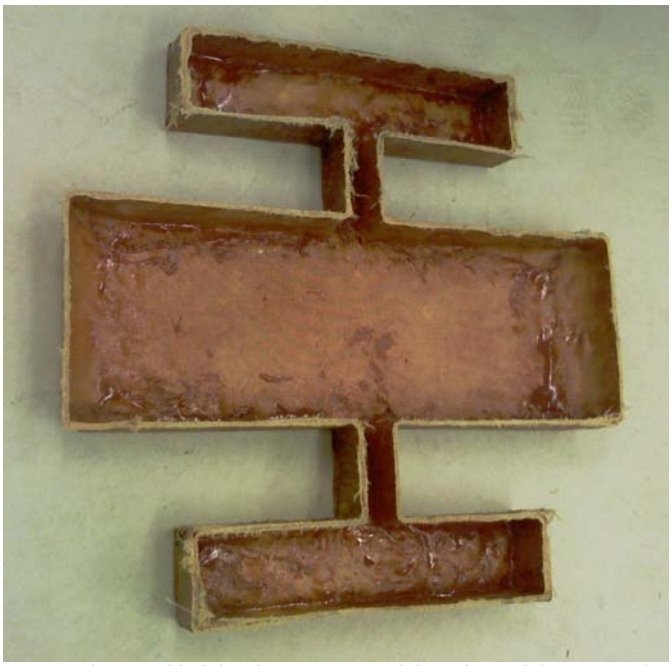


Figure 3-13: The equal height of 75 mm around the edges of the composite shell

After each half of the robot body was cut to equal height and a 0.8 mm mild steel

plate bent to create a 8 x 8 mm C-channeling. This channeling was used for a rim around the edges to allow better sealing between the halves. A 45° angle was cut on the C-channeling at the corners of the chassis, and bent to follow the curve. This C-channeling was kept in position with phenolic resin.

Straps were made from 40 mm strips of 0.8 mm mild steel. These were spot-welded together to allow a tight fit over each half of the chassis. These straps assisted in pulling the two halves of the chassis towards each other with clips.

Elastrosil LR 3001/55 silicone which can withstand temperatures of 230 °C and is flexible at -55 °C, was used to seal the opening between the two halves of the composite body. This helped prevent moisture and heat entering into the body of the robot.

After the complete assembly of the robot, aluminized Kevlar was glued to the body of the robot. The aluminized Kevlar added to the strength of the robot. A test was performed on the aluminized Kevlar with a blow torch. The aluminized Kevlar was able to withstand the flame for periods of about one second, which is similar to a flash flame duration in a disaster scenario. A flash flame is caused in a scenario when a fire is burning with difficulty in an oxygen deficiency room. As a door is opened and oxygen is made available to feed the fire, the fire will cause a blast out of the door until the room is filled with enough oxygen for the fire to continue burning in it's previous environment.

The mechanical properties of the materials used give the design a significant advantage compared to previous USAR robots, in that it is able to withstand high temperatures of 200°C, and to protect the electronics components and modules within the chassis from the falling debris [19]. The strength of the body also allows for the transportation of medical equipment that might be needed for the rescue operation, which is not possible with other medium-sized USAR robots. CAESAR's construction is compact in size and lighter in weight by 14 % and 33 % respectively, in comparison to other medium-sized USAR robots such as the ATR-X-50 [18].

3.2 Leverage / Flipper Arms

In the conceptual design phase it was opted to allow the flipper to rotate 360° around the main shaft. This would increase the effectiveness of the flippers to assist in troublesome situations. To allow this freedom, each arm required slots in the body through which to move.

The design of the flipper mechanism proved to be the most challenging part of the mechanical design. The front and rear flippers are each independently powered by a 20 W motor. Each mechanism had to be constrained to move only in rotation and not to slide along the axis of the main drive shafts. A secure mechanism was conceived as indicated in figure 3-14.

Two 08B and one 06B sprockets were solidly attached to each main drive shaft. This provided supports against the translation of the shaft along the turning axis.

As is seen in the exploded diagram, two flipper arms rotate around a supported axis provided by the main 12 mm shaft. Each flipper rotates on a needle roller bearing and is held against a needle thrust bearing by a 8 mm shaft which holds the two flippers together with four nuts. The needle thrust bearings allow freedom of rotation of the flipper with respect to the mounted sprocket and the main drive shaft simultaneously. Teflon could also have been used for this application, but was not used due to to costs and the quantity that had to be ordered. Diagrams of the needle roller bearing, needle thrust bearing and ball bearing are shown in figure 3-15. Their selection were determined by availability, and strength properties of the required shaft and flipper arms, derived from the analysed simulations.

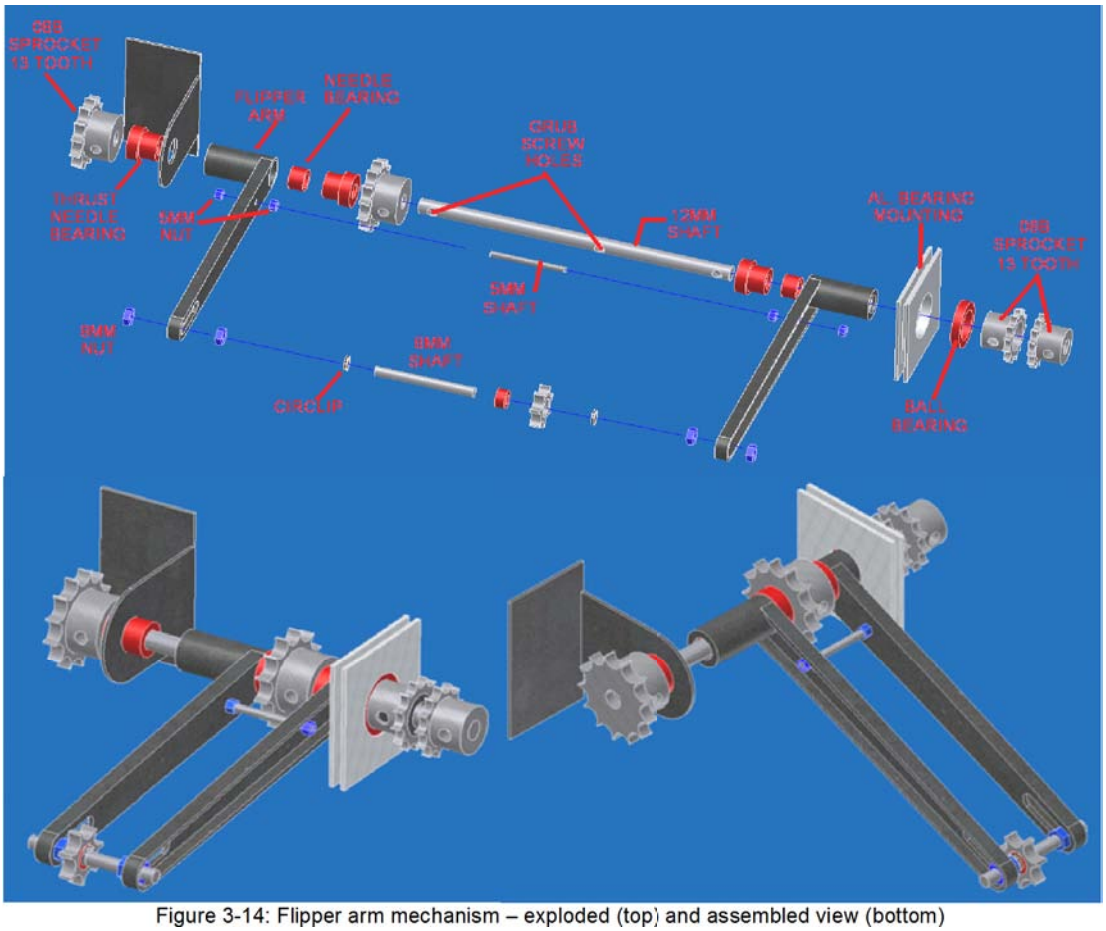


Figure 3-14: Flipper arm mechanism – exploded (top) and assembled view (bottom)

A bracket is attached to the body at the outer ends of the shafts and supports the shaft with another thrust needle roller bearing. The bearing is held against the outer 08B sprocket to constrain translation along the axis of rotation.

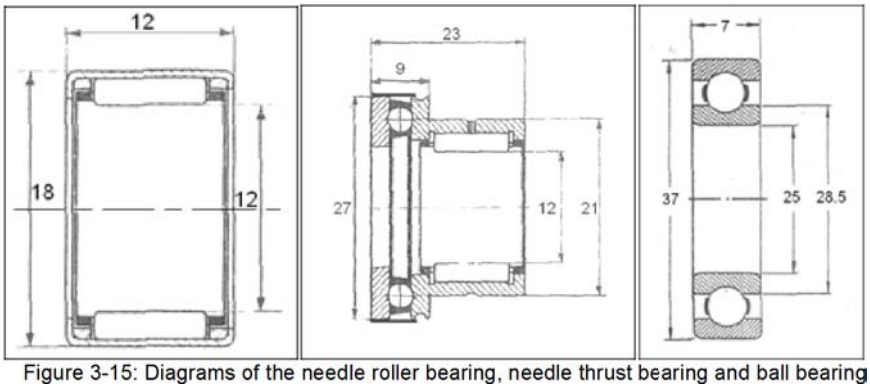


Figure 3-15: Diagrams of the needle roller bearing, needle thrust bearing and ball bearing

The shaft attached to the flipper arms, which extends through the body, can rotate within a normal ball bearing. The ball bearing is fastened in a machined bracket which is specially made to slide into both the top and the bottom half of the body. To keep the flipper stationary, a 06B sprocket was attached to the flipper via a grub screw which would thrust against the lower portion of the ball bearing.

At the end of the flipper arms a 8 mm shaft provides support for the 08B sprocket. The shaft is held by four nuts stationary with respect to the flipper in rotation and translation. The sprocket rotates on a press-fitted ball bearing constrained by two cir-clips.

The flipper arm was made by cutting a square mild steel tube in half to form two U-beams. These U-beams were then welded to a cylinder machined to house the bearings. Stress analysis of the arms was performed to determine if it is able to withstand the weight of the robot. As the front and back each is fitted with two arms, each arm could carry half of the weight on it. With an initial estimation of the body weight of 25 kg, and taking a safety factor of at least 5.5, the force applied to an arm is 674 N. This analysis is found in figure 3-16. A safety factor of 5.5 was considered, so that should a single flipper arm have to carry the full load, the flipper arm will not deform due to the strain. The safety factor must also consider the force of a payload that might be on the robot.

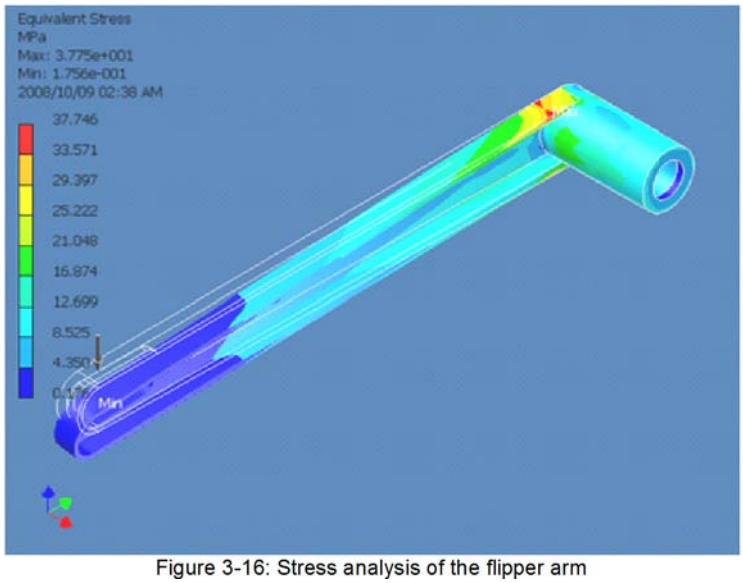


Figure 3-16: Stress analysis of the flipper arm

This analysis indicates that the highest stress concentration is at the welded section with a value of 37.8 MPa.

3.3 Drive / Actuator System

Motor vehicle windscreen wiper motors were selected to be the drive units due to their low cost, high torque and reliability. Each wiper motor is capable of delivering a maximum of 12 Nm and a maximum speed of 30 RPM at the output shaft. The motor's stall torque was 20 Nm, which was the torque applied to the motor at which the worm gear mechanism would start to rotate.

A decision was taken to utilise a differential steering mechanism where the left and right drive chains of the robot were controlled individually by a wiper motor.

To decrease the overall dimensions of the robot it was decided to group the front flippers, and to control both at the same time with one motor while doing the same for those at the rear.

This setup therefore required a total of four wiper motors which were mounted in the configuration as indicated in figure 3-17. Motors 1 and 4 were used for flipper control while motors 2 and 3 were used for the differential drive system.

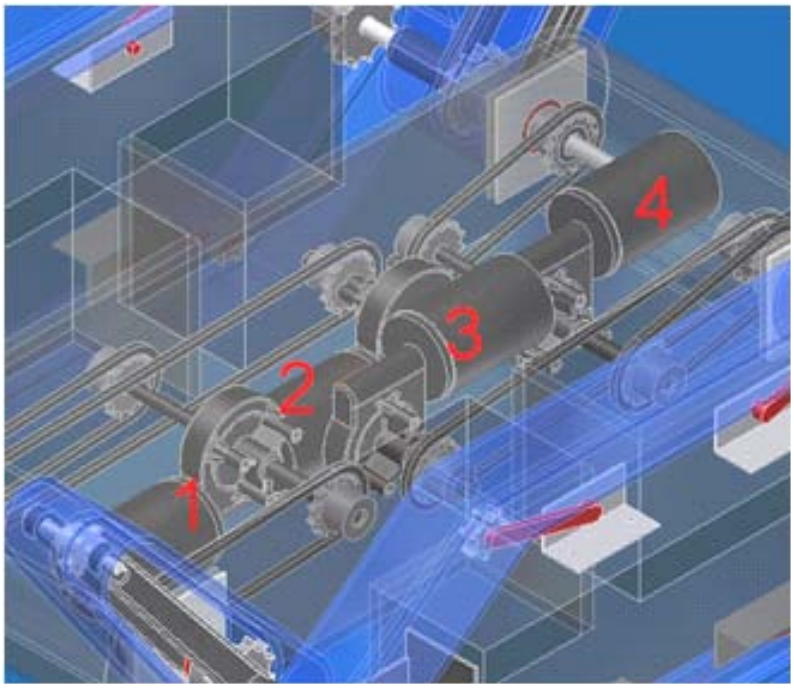


Figure 3-17: Motor and chain-sprocket configuration

Motor 1 and 4 required to be retrofitted with a fully protruding shaft. The worm gear mechanism was disassembled and the shaft was replaced by a longer one. The drive configuration of each motor is shown in figure 3-18.

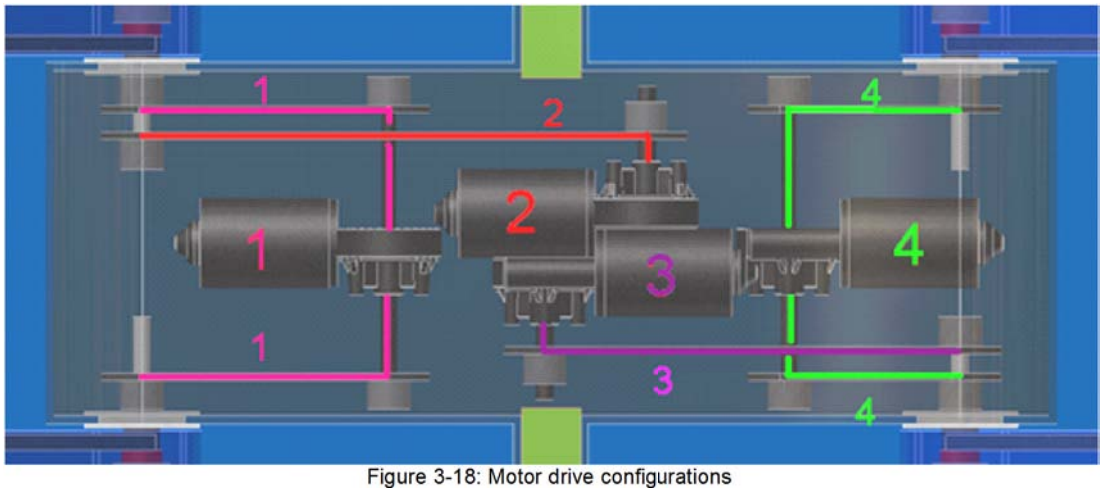


Figure 3-18: Motor drive configurations

In order to decrease the space occupied by the motors, it was decided not to mount the motors to the body until after the assembly of the flipper mechanism. This was necessary to make allowance for the needs of the thermal camera which was to be mounted in the front of the central cavity.

As seen in figure 3-18, the shafts of motors 1 and 4 ran between the chains of

motors 2 and 3 respectively. In the simulation, a mounting arrangement was found to comply with this constraint. The flexible mounting arrangement made allowance for any fine tuning of the motor configuration.

It was decided to use a chain and sprocket mechanism as a power transfer mechanism in order to decrease the number of encoders needed to sufficiently track the motion of the robot. The alternative would have been a pulley mechanism which under certain scenarios could result in slippage and therefore a deviation between the actual and the tracking position and speed could result.

For the power to be transferred from the motors, a 06B chain-sprocket group was used for the drive chains. A 08B chain-sprocket group was selected for the outer track chains in order to provide for the attachment links available. The differentiation between the chains used is shown in figure 3-19.

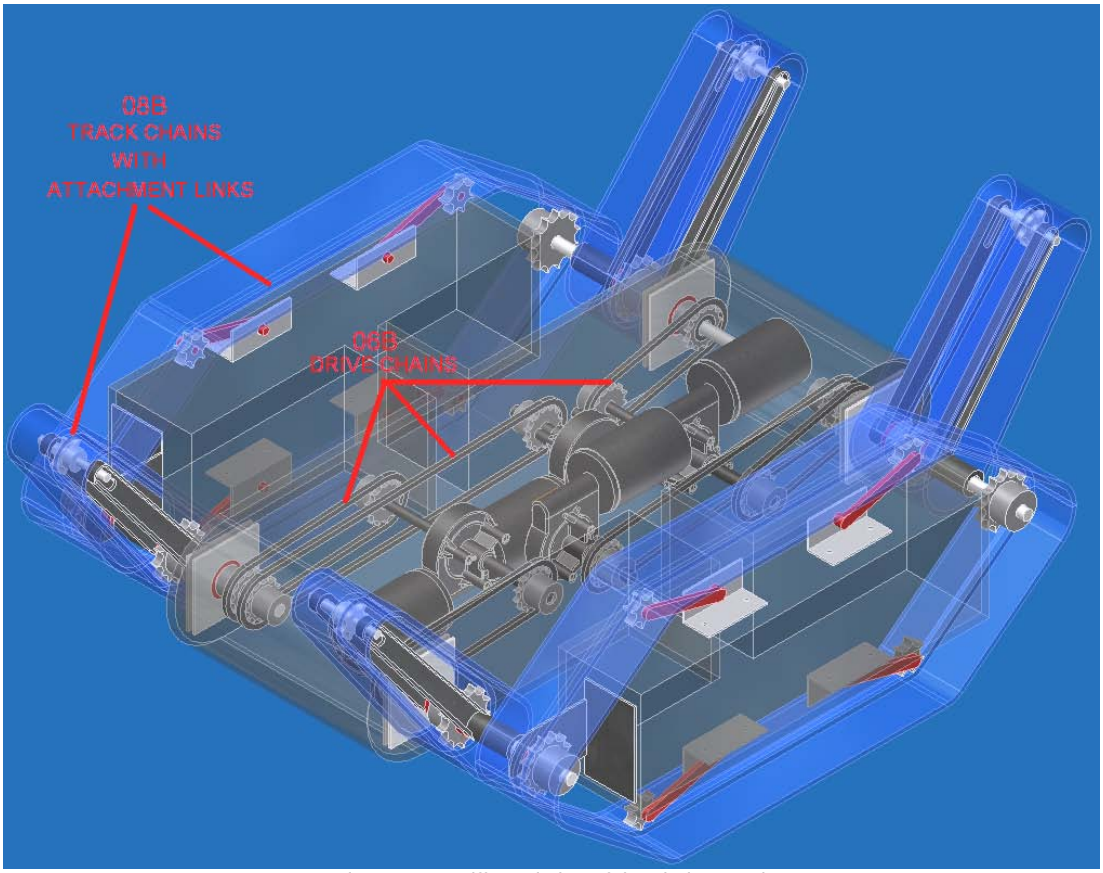


Figure 3-19: Differentiation of the chains used

The smallest sprockets available, which is the 06B 13 tooth, were used on the inside of the body. The following factors had to be considered to determine whether a sprocket ratio is required before calculations could proceed.

The factors that had to be considered are:

- Each motor is capable of supplying a torque of 12 Nm and a maximum speed of 30 RPM.
- The maximum flipper arm length (from centre of main shaft to flipper end) is roughly 228 mm.
- It would be preferable to have a 1:1 sprocket ratio between the motor and the flipper, as the speed would not be affected.
- The maximum sprocket diameter is limited by body dimensions to 100 mm. This makes allowance for chain width as well as other spacing allowances.

Using the scenario of the torque diagram in figure 3-20, the maximum force on the flipper arm can be calculated as shown in equation 14.

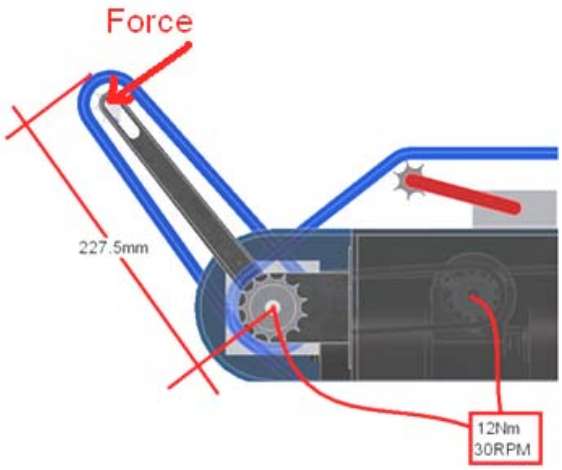


Figure 3-20: Torque diagram of the flipper arm

$$Max\ Force = \frac{Torque}{Radius} = \frac{12\text{Nm}}{0.228\text{m}} = 52.632\ N \quad \{14\}$$

An increase in torque would assist in the functionality of the flipper arm. The sprocket ratio is calculated by:

$$\frac{Force\ that\ \ flipper\ arm\ must\ supply}{Force\ the\ \ flipper\ arm\ can\ supply} = \frac{280\text{N}}{52.634\text{N}} = 5.320 \quad \{15\}$$

Each flipper arms set at the front or back will carry half the weight of the robot, this being 28 kg. The size of the chassis restricts the sprocket size to a ratio of 1:2.5. The stress and deformation analysis of the flipper arms were performed to determine whether these arms were able to withstand the weight of the robot for the different scenarios. This stress analysis is shown in figure 3-16 and the deformation analysis in figure 3-21.

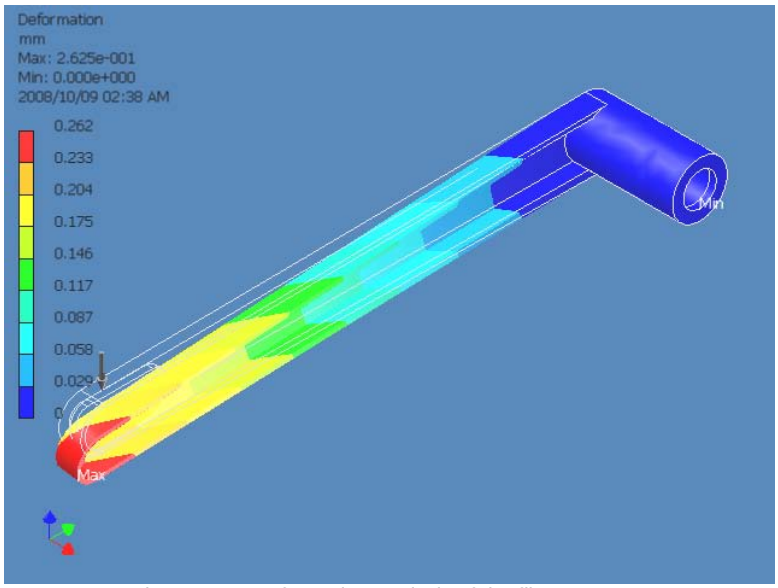


Figure 3-21: Deformation analysis of the flipper arms

3.4 Tracks

Different approaches were considered regarding the tracks. A rubber type material would be preferable for the tracks as it assists with traction on smooth surfaces, but rubber has a low melting point.

Elastrosil LR 3001/55 silicone can withstand temperatures of 230 °C and is flexible at -55 °C. It is fire resistant and meets with UL 94 classification. It is also resistance to steam and mineral oils ASTM no. 1,2,3 [43].

A problem found with the Elastrosil LR 3001/55 silicone was that it tore very easily when a small cut was made. This was solved by laminating Kevlar with the silicone. Another problem experienced was that this silicone was thick in its liquid form and this made it difficult to mold.

It was then decided to use a chain for the tracks and the 08B-1 chain was chosen [38]. Traction pads were required on the chain to prevent slippage. To implement this, the head of each pin that joined the chain was carefully removed with a grinder and the pins were removed with a press. The chain was reassembled with the 08B-1M K1 link replacing every second link. The 08B-1M K1 links have feet attached allowing an object to be bolted to the chain as shown in figure 3-22.

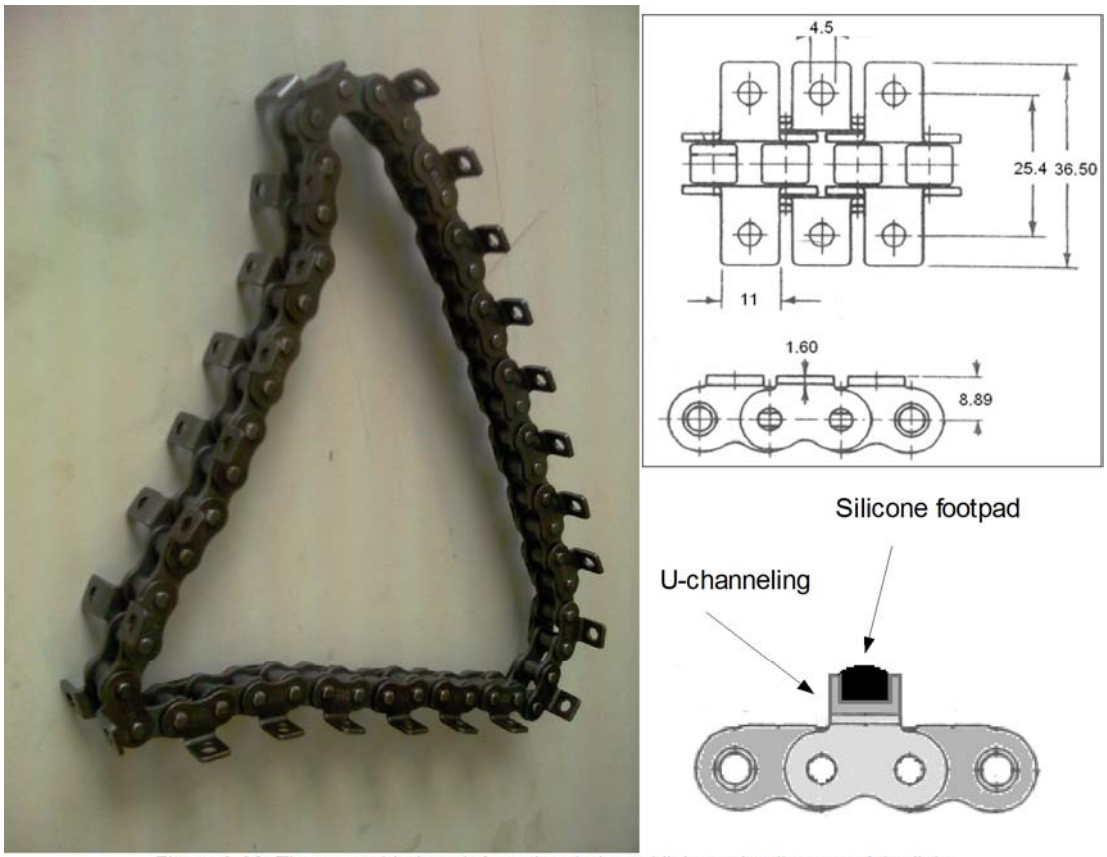


Figure 3-22: The assembled track from the chain and links and a diagram of the links

Traction was improved by fastening U-channeling to each of these chain feet. The U channeling allowed traction over rough terrains, but there still was a possibility of slippage on smooth surfaces. To prevent this, each U-channel piece was filled with Elastrosil LR3001/55, giving a small elevation of silicone, as shown in figure 3-22.

The traction system implemented allows for mobility on smooth and rough terrains of both high or low temperatures. The terrains could be rubble, oil spills and wet floors. This were not always possible with previous USAR robot traction systems [9]. In the event that CAESAR should tip over, it can change the flipper arms orientation and still continue moving.

3.5 Assembly

To assemble the robot, many aspects of the modules had to be designed for the final construction.

3.5.1 Camera Lens

A lens was needed in front of the thermal camera to protect it from high external temperatures. The reason for this lens was to prevent heat from entering the robot chassis and to protect the thermal camera from high temperatures.

Perspex material was considered, but this would melt at high temperatures. Treated glass and was also a possibility, but the glass could only be treated as a large pane. Glass also has the problem that it could shatter should water spill onto the heated glass due to the environmental temperatures.

After extensive research on the properties of different materials that could be used for this application, it was found that the Oakley lenses would be a suitable option. Soldiers have had their eyes and faces protected from explosions and shrapnel as they were wearing Oakley sunglasses [44]. People's eyes and faces were also protected after motorcycle accidents and a broken circular saw accident [45]. Further tests would have been required would the Oakley lenses be considered. Tests performed with the Oakley sunglasses indicated that the infra-red (IR) radiation was blocked out, therefore allowing no thermal image to be displayed.

The Hawk I.R. C-Range sightglass was examined for possible use as material for the lens. This allowed IR radiation through and was arc-resistant as it had a CLIRVU coated crystal optic. The Arc-Flash test complied with both IEC62271 and ANSI C37 standards. It was also compliant with UL impact test requirements and tested IP65 to EN60529 and Type 3/12 to UL50/NEMA, allowing all weather capability [46]. The maximum continuous operating temperature was limited to the gasket material which deformed at 250 °C, and it had a barrier that could withstand 1.676 MJ/m^2 which was compliant to the highest safety level of NFPA70E Hazard/Risk category [47].

A hole was made into the front of the chassis, to allow a viewpoint for the thermal camera. A 0.5 mm plate was attached over the hole, to allow a flush point of attachment of the lens. Holes were drilled around lens casing and corresponding with the same places on the chassis, to allow a firm and sealed fitting.

3.5.2 Body Assembly

Different issues were discovered as development and assembly of the body construction continued. Bearing casings were made that would allow the shafts to protrude into the composite body. This was done by taking a block of mild steel and machining out a hole so that the bearing would fit tightly in it. A slightly larger hole was machined so that a cover could be placed over the bearing, to keep the bearing in place. Four holes were tapped to keep the cover in place. A bearing in its casing is shown in figure 3-23.

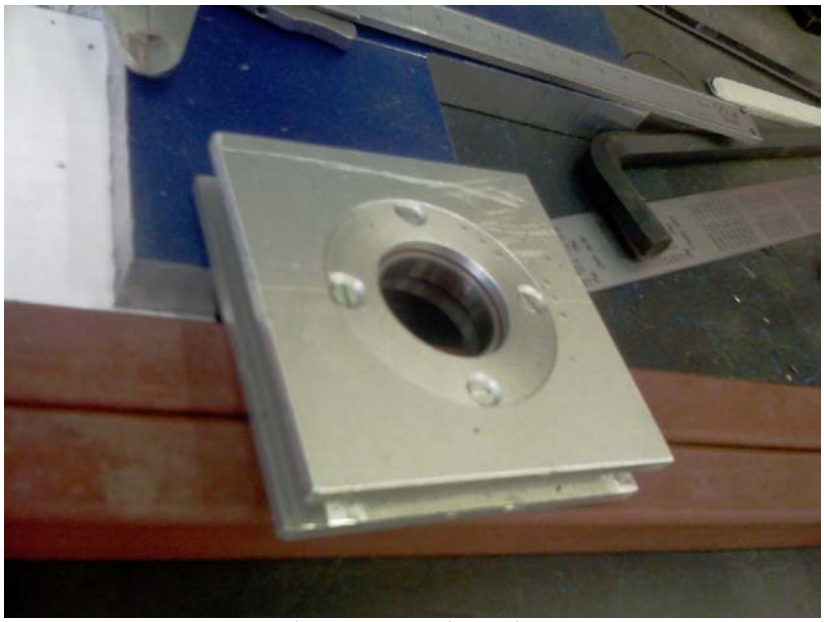


Figure 3-23: Bearing casing

After the composite body was completed, measurements were made and sections were cut out so that the bearing casings could be inserted in the front and back of the robot. After these cuts were made, the bearings were tightly inserted. These casings were glued in place with phenolic resin. This is shown in figure 3-24.

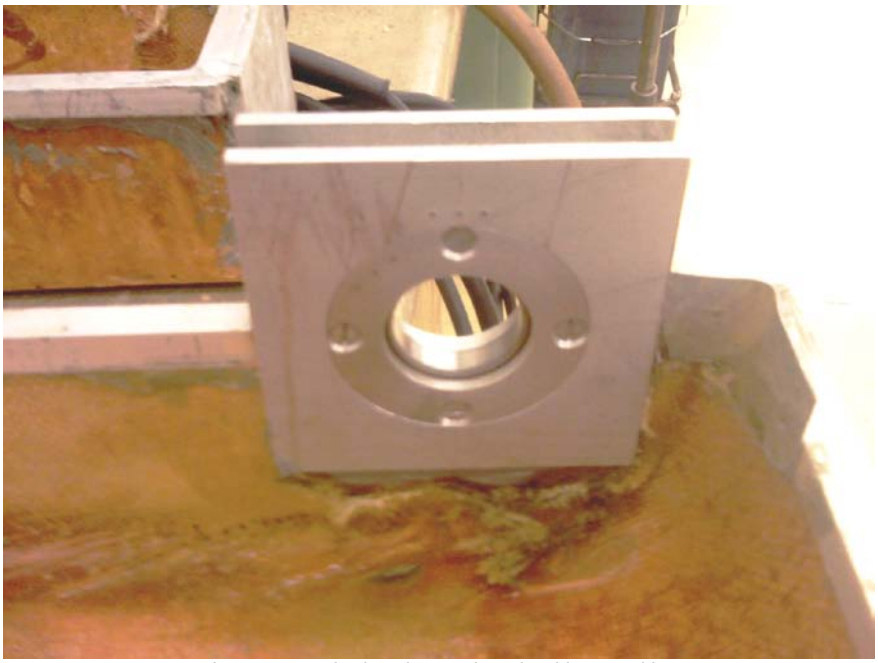


Figure 3-24: The bearing casing glued into position

The flipper arms were made from square tubing cut in half. These were then milled to the correct width to give the shape of the C-channeling. A slot was also cut into this to allow tensioning of the track by adjusting the sprocket position.

Round shafts were turned to be inserted into the bearings. Holes were also machined out of these shafts to allow bearings to be inserted for shafts. The same shafts were used as part of the flipper arms. To allow the welding of these round shafts to the arms, dummy bearings were turned and inserted to prevent it from distorting. After the welding was performed, these dummy bearings were removed and the needle and thrust needle bearings were inserted where needed.

The shafts were made from silver steel. Screw thread was turned into the sides of the shafts to allow nuts to be fastened where required. The sprockets were milled to be the desired thickness, so that the tracks could run in the center of the flipper arms. Washers were turned from brass to assist with this centering. The smaller sprocket's pivot hole was enlarged to allow the insertion of a needle roller bearing for ease of movement. This assembly is shown in figure 3-25.

Figure 3-25: The initial assembled flipper arm, showing the shafts welded to the arms and the sprocket-tracks connectivity

The motors were fastened with a bracket and bolts. It was decided to make the brackets from mild steel plates. Other brackets were needed to support the motor shaft. Brass was turned and inserted into a threaded rod, which allowed the bush to be bolted with two large nuts to the bracket as shown in figure 3-26.

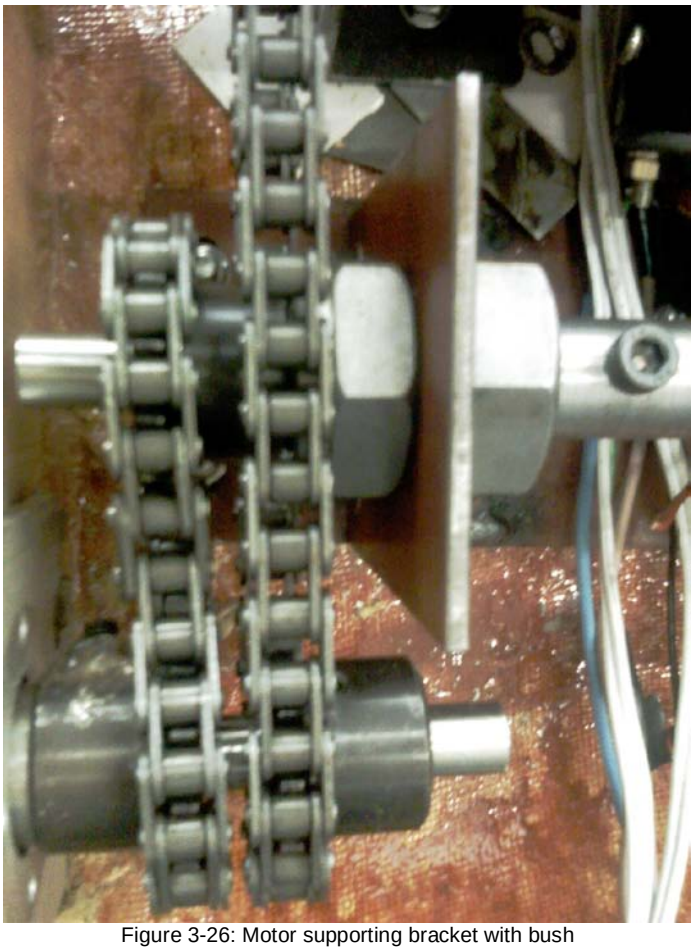


Figure 3-26: Motor supporting bracket with bush

Tests of the flipper arms movement, traction and motion were performed as the robot was constructed and assembled. A figure of the CAESAR robot under testing is shown in figure 3-27.

During the testing process it was established that the motors did not have enough torque to pull the CAESAR robot over obstacles. Increasing the sprocket ratio to 1:2.5 made it possible for the CAESAR robot to climb over an obstacle more easily, but the speed of the robot decreased to about 1 inch per second. This speed could be improved by using faster, higher geared and more expensive motors that would supply the power requirements. As the budget for this research was limited, less powerful motors were considered and the body dimensions were determined from the dimensions of all the internal components that were to be used.

A problem noted was that the body scraped on the obstacles when the CAESAR robot had to climb stairs. The reason being that the flipper arms were narrow and the obstacle first made contact with the chassis rather than with the tracks. Further improvements were required for the flipper arms. This was achieved by having extensions at right angles to the arms, allowing the tracks to move in a diamond shape. Should the CAESAR robot approach an obstacle, the tracks from the flipper

arms would push against the obstacle as there would be a larger traction area and prevent scraping to the chassis. Such an arrangement is shown in figure 3-28.

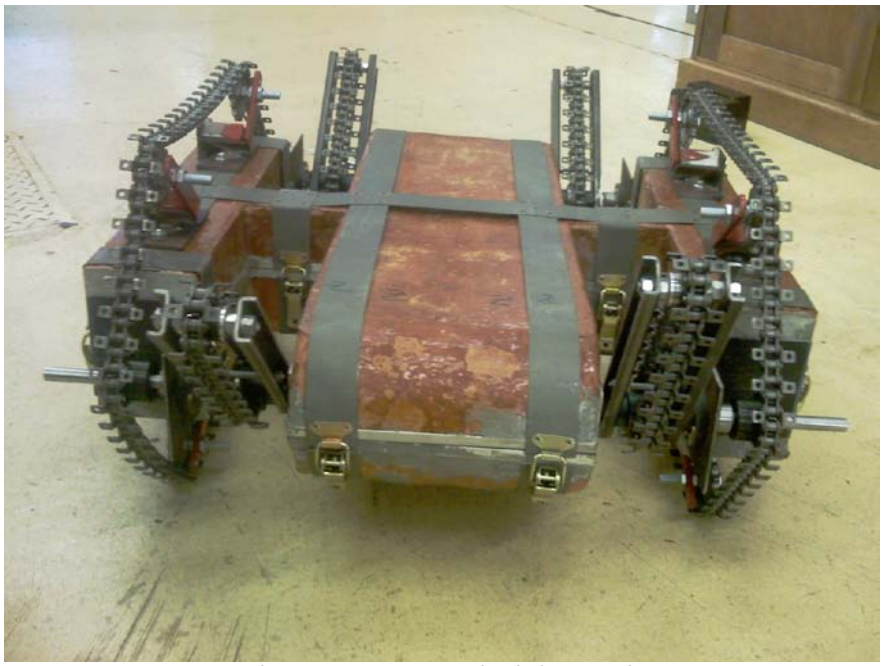


Figure 3-27: CAESAR robot being tested

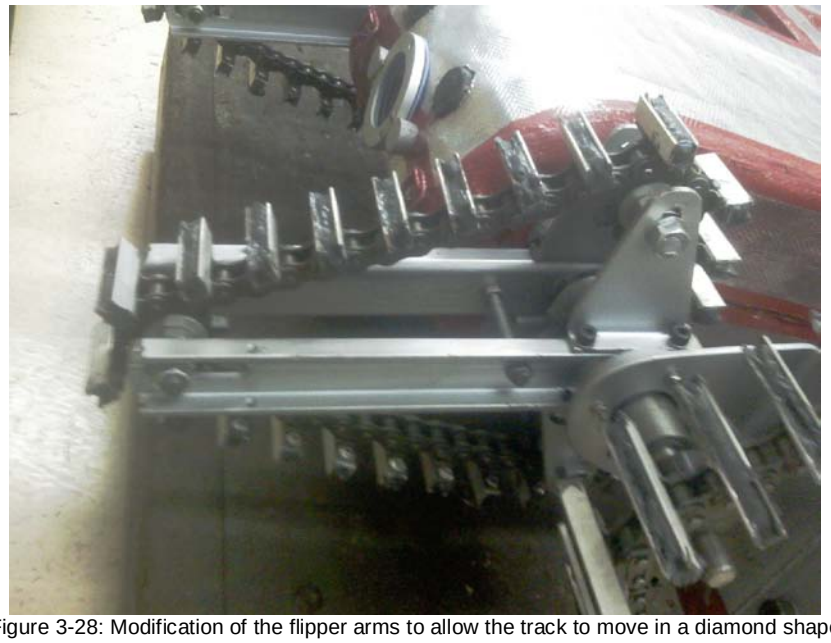


Figure 3-28: Modification of the flipper arms to allow the track to move in a diamond shape

Once the CAESAR robot was moving satisfactory and to requirements, the final holes were drilled for all the input and output devices. All the straps were securely positioned in place with phenolic resin and a final covering of aluminized Kevlar. The other metallic components that could not be covered with the aluminized Kevlar were covered with silver high-temperature spray paint, which also allowed for the reflection of heat.

A gasket was made with 'Siltech 300 Silicone HT' high temperature acetoxy silicone sealant around the rim of the casings. 'Siltech 300 Silicone HT' has the properties of being resistant to UV radiation, moisture, extreme temperature fluctuation and tensile compression stresses and can withstand a maximum temperature of 300 °C. This allowed the two joined halves of the casings to be watertight and kept heat out. All joints and inserts were sealed with this silicone to make the robot interior watertight and to prevent internal explosions that might result in flammable gases.

3.6 Summary

The reason for the design of the robot has been explained as well as the maneuverability and the environmental conditions that the robot would operate in. Size criteria of the robot were stated to house all the modules. Modeling of the robot was undertaken to determine if the design was reasonable.

The composite body design and construction has been explained. The different tests performed to determine the best release agent were analyzed, and the different composite materials that have strength and thermal properties were investigated. The phenolic resin selection was chosen for its heat resistant properties. Further construction procedures with stress and deformation analysis were explained.

Discussion was given as to the leverage / flipper arms with the sprocket, chain and tracks selection. Different bearings used for the assembly of these arms were demonstrated. A stress and deformation analysis with a maximum force calculation was addressed to determined whether the strength of these designed arms were efficient for the conditions it would be exposed to.

The composite body allows for the protection of the electronic modules within the robot from heat and falling rubble as was previously suggested [17]. With the incorporated flipper arms in the body design, it was possible for the robot to climb over obstacles. It is also possible for the flipper arms to contract when the robot requires entry into a confined space.

The motor and actuator system was shown with the motor, chain and sprocket configuration, explaining the different sizes that were used for the respective purposes.

The design and construction of the tracks have been presented with the use of the links that allow grip for slippery conditions. This prevents slippage on rubble and smooth terrains which was previously a problem experiences with USAR robots [9]. A decrease of traction malfunction occurs as materials used are capable of withstanding the high temperature environments.

The complete assembly of the robot construction was discussed with the assigned position of the different components to optimize the shape and size of the robot.

CHAPTER 4 – ELECTRONIC DESIGN

The electronics within the CAESAR robot allows for the interaction between the controlling and mechanical aspects of the robot. Batteries supply power to the different electronic modules. Sensors supply the robot with local intelligence and environmental information to the rescuers. These components and modules interconnections are discussed.

4.1 Power Supply

It is of vital importance that the USAR robot is not tether connected so the vehicle needs its own on-board power supply. Different types of batteries, such as lead-acid, nickel-cadmium and lithium-ion batteries were considered. The following had to be considered:

- weight
- energy
- rechargeable
- inherently safe
- battery memory effect

Lithium-ion Phosphate batteries proved to be best for this application. The advantages of Lithium-ion Phosphate batteries are [48]:

- less weight for same size
- more energy
- inherently safe
- rechargeable
- lithium-ion chemistry with no thermal runaway
- remains thermally stable
- virtually zero maintenance over the service life of the battery
- no battery memory effect
- low Peukert's Effect – low capacity loss as discharge rates increases
- low self-discharge
- no sudden death syndrome
- no explosive hydrogen gases (vital for the environment that the robot will operate in)
- no corrosion on battery terminals
- no need to refill electrolyte

- maintains higher voltage / power even towards the end of discharge
- environmental friendly

Powerstream's 12 V, 55 Ah Lithium Ion batteries [49] were investigated were found that to be the best for a search and rescue robot. The advantages of these batteries above other Lithium-ion batteries are:

- laser welded stainless steel cases
- no polymer, rubber, or plastic seals

The body of the USAR robot was so designed that it could accommodate these battery packs. These batteries were very costly and due to a limited budget the decision was made to use normal lead acid batteries to test the prototype of the robot. With the same available space, it was possible to use two 12 V, 7 Ah lead acid batteries. This gave a total of 14 Ah of energy capacity that would be available.

4.2 Control Unit

Two types of control station configurations were considered. The first consisted of a unit with a large joystick attached to it as well as an LCD with buttons. Different commands are sent to the robot as the different buttons are activated. The LCD displays information that is sent from the robot.

The advantage of this configuration was that the joystick control made it possible to effectively control the robot even while wearing gloves. The disadvantages of this configuration was that the robot and control station would be limited in Artificial Intelligence (AI) due to the micro-controller's limited processing power. This configuration also had an “outdated” look and the control methods and AI decisions could not be updated with future research, as the LCD available are limited in size which affects the amount of information that can be shown.

The other configuration that was considered included a notebook computer. The disadvantage of this type of configuration was that the control was done with the mouse and an external joystick could be added while wearing a glove. Most AI decisions are determined by the computer. Future research could be performed on the AI of the data received from the robot and the computer could be upgraded if needed. This researched AI could autonomously control the robot remotely thus solving the restriction of having a powerful on-board computer on the robot. This option is the most efficient one of the two described.

This type of configuration also allowed for easier implementation permitting more time to focus on other contributing factors. It was therefore decided to implement this latter configuration.

4.3 Sensors

Sensors for the detection of objects, gases and flipper orientation were investigated.

4.3.1 Ultra Sound Sensors

Object detection for the required application was provided by means of sonar sensors. Sonar sensors operate on the principal of echo location (used by Bats). The sensors comprise of either a single or double transducer connection incorporated on an integrated circuit. The circuit enables the transducer to vibrate at a certain frequency (between 20 kHz and 60 kHz) causing the transducer to emit a pulse/echo at a certain cone angle, which upon collision with an object (in front of sensor, at a specified distance, within the angle scope) is reflected and received by the alternate transducer or same transducer in the case of a single transducer connection [37]. Ultra sound sensors also have problems with the detection of objects. Specular reflection is caused when the signal bounces off an object which is at an angle, resulting in the reflection not returning to the transducer. An example of this could be when approaching corners, as there is no parallel area with the transducer to reflect the signal back to the receiver. Another problem that ultra sound sensors could have is that objects could absorb the signal instead of reflecting it back to the transducer.

An ultra-sound system (shown in figure 4-1 [50]) was used for the object detection. The 40 kHz transducers were used as they were easily available. This was used for the artificial intelligence explained in chapter 5. The ultra-sound system has a double transducer, which detects an object within a range of approximately 50-1500 mm. The sensor is compact and uses an appropriate supply voltage of 12 V with a light and buzzer as the output. The light and buzzer is activated when an object is detected a certain distance away. The sensor transmission of an echo/pulse is sufficient for object detection over a wide field.

Figure 4-1: Ultra-sound system used on CAESAR

The ultra-sound system has the following characteristics:

- Supply voltage: 10 – 15 VDC
- Range: 50 – 1500 mm
- Output: 12 V / 0 V voltage
- Angle: 40^{o} cone angle
- Size: Sensor PCB: 28 x 95 mm
  Base PCB: 48 x 128 mm

The CAESAR robot needs to identify when to lift its flipper arms to assist it to climb obstacles. This transformation is only needed when the detected obstacle is in close proximity. With the ultra-sound system it is possible to configure it to give an output voltage for detection at a desired distance.

The buzzer was removed to use the output as an input to the micro-controller. Appropriate signal conditioning was required to reduce this voltage output to approximately 5 V since the micro-controller used operates on a 3 – 5 V range as input to the pins [37].

This reduction in voltage was made possible by means of a potential divider circuit, which consisted of a 10 kW and a 15 kW resistor.

The ultrasonic transducers has an aperture of 40°. The transducers had to be moved further away from each other, as they were mounted on either side of the thermal camera lens. Since the distance between the positions of the transducers have to be moved, it was necessary to determine if they would still be able to sense objects in their new positions and whether the flipper arms might cause obstructions, causing inaccurate readings. Figure 4-2 is used for the calculations that follow.

Referring to figure 4-2, x indicates the length of the flipper arms that will be beyond the front of the vehicle, y the distance from a sensor to the closest flipper arm, and z is the distance between the ultrasonic transducers.

The minimum distance from the ultrasonic transducer to the closest flipper arm needs to be determined to prevent the robot from detecting the flipper arm as an obstacle. This distance is referred to as $Dist_{sf}$.

$$Dist_{sf} = x\tan\left(\frac{\theta}{2}\right) = 160\tan(20) = 58.235\,mm \quad \{16\}$$

where x = length of the flipper arm beyond the front of the robot = 160 mm

θ = aperture of the ultrasonic transducer = 40°

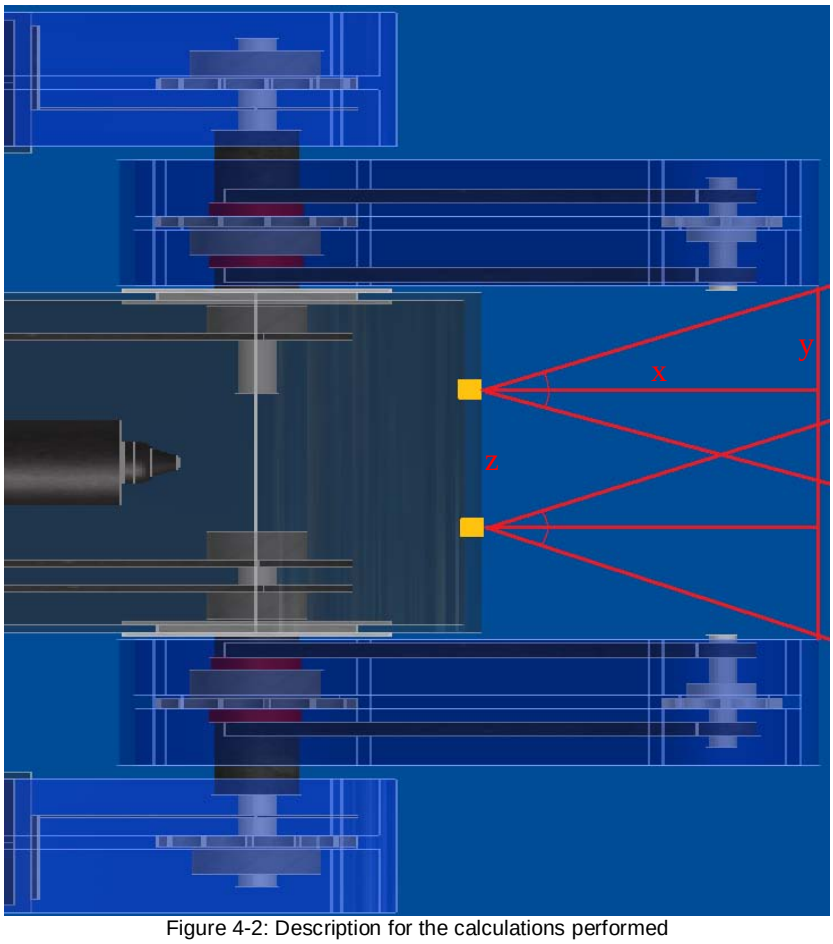


Figure 4-2: Description for the calculations performed

From equation 16, the minimum distance from the ultrasonic transducer and the flipper arm is 58.235 mm, which is less than that of the current dimensions on the robot, being y = 70 mm.

As the distance between the transducers had been moved, it is necessary to determine the furthest distance the ultrasonic transducers would detect. This distance is referred to as $Dist_{max}$.

$$Dist_{max} = \frac{z+y}{\tan(\frac{\theta}{2})} = \frac{146}{\tan(20)} = 401.132\,mm \qquad \{17\}$$

where z = distance between the sensors = 76 mm

From equation 17 the distance of 401.132 mm is more than the distance that the flipper arms protrudes from the robot body and sufficient for the desired detection of obstacles. The distance of 400 mm is sufficient for the detection of an obstacle, allowing the CAESAR robot to transform as required.

The shortest possible distance for detection by the ultrasonic transducers must be determined. This distance is referred to $Dist_{min}$.

$$Dist_{min} = \frac{\frac{w}{2}}{\tan(\frac{\theta}{2})} = \frac{38}{\tan(20)} = 104.404\,mm \quad \{18\}$$

From equation 18, it was found that the minimum distance that the ultrasonic transducers would detect an object was 104.404 mm, which was shorter than the distance that the flipper arms protrude from the robot. This was an acceptable distance as the CAESAR robot would be able to detect an obstacle that was further away than the flipper arms.

4.3.2 Flipper and Body Orientation

Avago Technologies HEDS 5500 encoders were used on the flipper arms shafts to determine the angle of the front or back arms. This enabled feedback to the micro-controller to verify the arms angle and orientation. A cup system connected to a shaft was developed to allow the encoder to turn coincident with the turning of the external shaft of the flipper arms. This is shown in figure 4-3.

Figure 4-3: Flipper arm orientation encoder system

It was useful to be able to determine the orientation of the robot with respect to the pitch and roll of the chassis. This indicated to the controller and the robot at what incline there was being ascended and also whether the robot was upside down or in the correct orientation.

To achieve this a weight shaft was developed that was inserted into the encoder. The weight is always pulled towards gravity. As the robot climbs an incline, the weight would moved from its zero position, and the angle could be determined. This configuration is shown in figure 4-4.

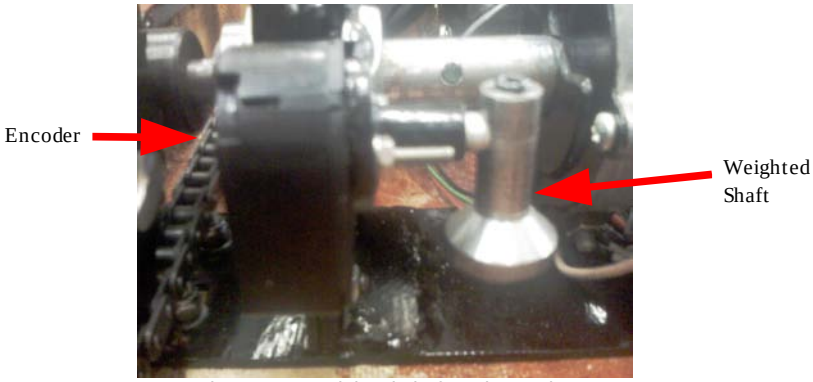


Figure 4-4: Weighted shaft and encoder

The encoders had two channels, channel A and channel B. The out of phase signals are shown in figure 4-5.

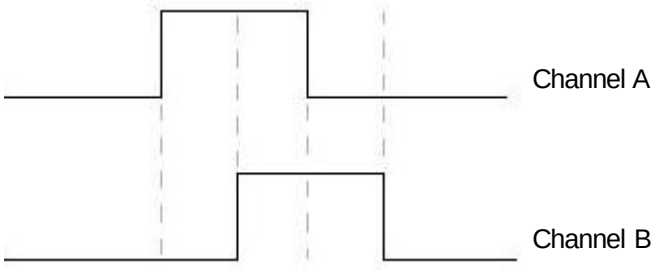


Figure 4-5: Channel A leading channel B

In the event that the shaft turned in the one direction, channel A lead channel B, while it is visa versa should the shaft have turned the other direction.

The Atmel ATMega32L had three interrupts. One interrupt was used for the angle of the front flipper arms, while another interrupt was used for the angle of the rear flipper arms. The angle of the robot could be determined with the last interrupt. As the interrupt was initiated when a falling edge was detected on channel A, channel B was probed to determine if it was a high or low state. Should it have been a high state the angle increased in one direction, but should it have been a low state the angle decreased in that same direction.

With the detection of robot orientation, it allows the controller to know the incline that is being climbed, which is difficult to determine from the video feedback. Robot orientation was not successfully detected at the World Trade Center disaster operation and NIST [12].

4.4 Micro-controller Configuration

As the RCP module is a router for communication between the radio transmitter and the application modules, it must be able to switch the serial connections to the

appropriate module. This was achieved with AND gates, that selected the serial connection to the module by a switching mechanism from the RCP module. This is indicated in figure 4-6.

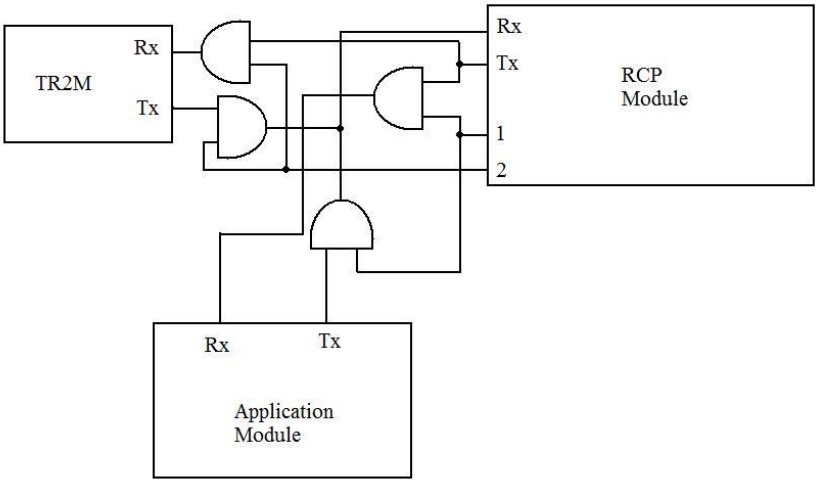


Figure 4-6: RCP switching circuitry using AND gates

More application modules can be added with the inclusion of two AND gates and an output pin from the RCP module. A combination of outputs generating a specific binary number is used to switch between communication modules.

4.5 Micro-controller Code

The code of the micro-controller determined the actions that the robot would perform. Factors that affect the different actions were the feedback encoders and commands from the control station. The CAESAR robot had two interfaces that apply to the application layer, being the sensor controller and the motor controller.

In the event that the controller sent the instruction for information from a specific sensor, the sensor controller replied with the details. On the other hand, should the controller have sent an instruction to the motor controller, it responded by turning the motors as required.

4.5.1 Motor Controller

The CAESAR robot has a manual control and an autonomous control with respect to the flipper arms. Motion control is still achieved via the control station such as the movement to the left, right, forwards and backwards.

The manual control commands and the corresponding motion steps that have to be performed are shown in table 4-1. The GUI control allows for the hovering of the cursor over the direction buttons, which in turn sends the instructions to the robot. Upon receiving the instruction, the motor controller reacts and controls the required motors.

Table 4-1: Commands and corresponding motor motion

Command	Action	Motor Motion
ff	Move forward	Both motors forward
bb	Move backwards	Both motors backwards
ll	Move left	Left motor reverse, right motor forward
rr	Move right	Right motor reverse, left motor forward
fl	Move in forward-left direction	Left Motor stop, right motor forward
fr	Move in forward-right direction	Right motor stop, left motor forward
bl	Move in reverse-left direction	Left motor stop, right motor reverse
br	Move in reverse-right direction	Right motor stop, left motor reverse
zf	Move front flipper arms forward	Front flipper arms motor forward
zb	Move front flipper arms backwards	Front flipper arm motor reverse
wf	Move rear flipper arms forward	Rear flipper arm motor forward
wr	Move rear flipper arms backwards	Rear flipper arm motor reverse
zc	Contract front flipper arms	Reverse front flipper arm motor till 0°
wc	Contract rear flipper arms	Reverse rear flipper arm motor till 0°

Since the sprocket ratio between the motors and the flipper arm shafts is increased, the output speed is relatively slow to create the higher output torque. The track speed is estimated to 25 mm a second. The maximum speed from the motors can be utilized, therefore allowing no need to control the speed of the motors, but only the direction. This allows for a simplified current amplification from the micro-controller to the motors, which consists of a transistor and a double switch relay, as shown in figure 4-7.

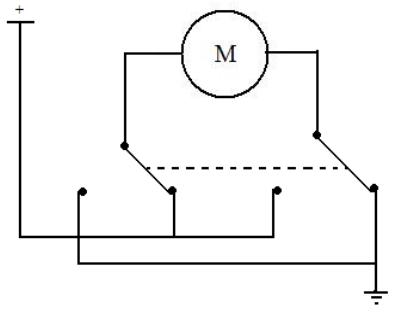


Figure 4-7: Motor current amplification schematic

At a default state the motor will turn in one direction due to the polarity, but when the relay is switched on, the polarity is inverted and the motor turns the opposite direction. With this type of simplified current amplification circuit, the size of circuitry is minimized compared to using an H-bridge. The minimizing of size allows space inside the robot for other modules to be inserted, should there be a space constraint.

4.5.2 Sensor Controller

The AT-Mega8L micro-controller is used to read the signals from the different

sensors. As this micro-controller has 6 A/D converters, it allows up to six sensors with a voltage output to be connected to it. Other sensors that might use other protocols such as Inter-Integrated Circuit (I^2C), could also be connected to this micro-controller should it be required. A LM-37 temperature sensor was connected to one of the A/D converters, to allow for the monitoring of the internal electronic module temperatures.

4.6 Summary

The power supply used for the CAESAR robot, with the advantages of having Lithium-ion batteries are explained. Reasons why the control station was developed with a computer interface were explained.

Sonar sensors were used for object detection. The configuration of the sensors with the derivative of sensor location and detection range was viewed and calculated. Flipper arms and body orientation was determined with the encoders, and the corresponding fixtures that were designed and developed were shown.

The object detection sensors are used for autonomous transformation, allowing the controller to focus on not only vehicle control but also victims and danger identification. Previous USAR robot controllers had the problem of knowing the vehicle's orientation, as this was not clearly identified from the video feedback [12]. Body orientation feedback allows for local autonomy for the best flipper arm orientation and feedback of the angle of inclination to the controller.

The micro-controller configuration with the explanation of the code and connections of motors and sensors were given. This enabled the control station to successfully send data to each module with the RCP module acting as a router. Similarly, the CAESAR robot could send requested data back to the control station.

CHAPTER 5 – ARTIFICIAL INTELLIGENCE

The Artificial Intelligence of the CAESAR robot [51] is for a semi-autonomous control system. Rescuers prefer to be able to control the robot manually, as they can direct it to specific areas that they are interested in. The robot must be able to transform into a desired shape to allow it to maneuver over obstacles. With the future development of the AI, the PC is able to warn the user of dangers such as unstable constructions and notifying the robot to make decisions and exit if needed.

Different strategies and command integrations used before in field robotics were investigated. The goal was to minimize the size of all modules, which included the boards for the AI processing. Considering the fact that the on-board AI had to use limited processing power and memory from the available micro-controllers, resulted in a limitation of localization methods. Due to these limitations, systems using image processing could not be performed on-board but could be added in future research.

5.1 Localization Methods

Mapping the environment is not feasible for a USAR robot, as the environment can continuously change during a disaster. The environment consists of explosions, the ignition of fire and the collapsing of building materials. Different localization sensing methods have been investigated. These include, Global Positioning System (GPS), Differential GPS (DGPS), Indoor GPS, wireless Cell Of Origin (COO) networks, triangulation, imaging methods and Simultaneous Localization And Mapping (SLAM). Most of these localization methods have their problems.

GPS will not work as the satellite signals do not penetrate through building materials. Indoor GPS will not be reliable as buildings will not necessarily have these units built in them, and the transmitters location could vary as the building collapses. Wireless COO networks could be an alternative, but there are not necessarily cellular networks at suitable positions to assist with the robot location. Imaging methods are also not reliable as the imaging system could malfunction during the collapsing of the building. SLAM is not reliable as the environment changes rendering the mapping data ineffective.

Triangulation is an option to identify a 3 dimensional location within the building, but will not possibly always correlate with the floor plans of the building. The reason being that there are not always floor plans available and the mapping of the collapsing building will not necessary correctly correlate with these. The only reason that the triangulation would be useful is to derive an estimation of the robot position to notify the rescuers as to the position of possible victims.

To be able to use a triangulation method, at least three transmitters would be needed to be located at strategic points around the building, each with its own ID. The intersection of the signals will indicate a virtual position of the robot, which could be transmitted to the control station. This system works on the same principle as

GPS and is referred to lateration. [52]

Two transmitters could be used for the localization of the robot, but then the known location transmitting stations must be completely stationary, be perpendicular to the horizon as a reference, and have 180° directional antennas that are able to move in the vertical position. This method which is also known as angulation [52], measures angle or bearing relative to the two transmitters of known separation. This would allow for the robot location in the x and y co-ordinates, while the angle of the antennas indicate the z co-ordinate, which identifies the altitude of the robot.

The localization with two or three transmitters could be viable methods should the rescue attempts endure for numerous hours. This is not usually the case, as rescuers have limited time to locate victims in a disaster scenario. These methods could be investigated in future research for rescue attempts that happen over a lengthier time period.

5.1.1 CAESAR Localization Method for Semi-Autonomy

AI similar to RatSLAM was considered for the CAESAR robot. RatSLAM is the localization method that rats use. Rats know their local surroundings and react on this information. An example of this will be a rat running down a corridor until it approaches a wall. It will identify the wall and turn as required. The rat could also identify a hole in the wall and thus continue running to enter the hole.

The CAESAR robot is remotely controlled and should relieve the controller from deciding whether the obstacles in front of it are large enough for transformation to be required. The sonar sensors are used for this.

With the CAESAR robot in automatic mode these transformations occur automatically. There could possibly be times that the user would desire to override the autonomous transformation and thus would prefer to transform at their own discretion.

The CAESAR robot detects objects via the ultrasonic distance sensor up to a distance of about 400 mm. The reference of the flipper arm angles are calculated from the point where the flipper arms are contracted within the body on a longitudinal plane, as shown in figure 5-1. The reference to the front of the CAESAR robot refers to the side that is traveling forward, while the rear side is considered to be the opposite side.

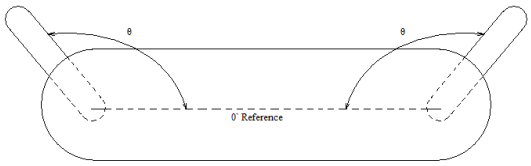


Figure 5-1: Diagram indicating the reference plane of the flipper arms

There are different scenarios to be considered. Should the CAESAR robot be traveling in a horizontal position, the default position for the front and rear flipper arms will be at 150°. This allows for stabilization as it maneuvers over a rough terrain. Should there be a small undetected object in front of the robot, or a small ditch that it has to cross, the angle of the flipper arms will assist it to continue with movement as the traction of the tracks are increased.

In the event that the CAESAR robot is traveling in a horizontal position and approaches an obstacle, the front flipper arms will move to 135°, while the rear flipper arms will move to 150°. This allows the CAESAR robot to move over the obstacle, while supporting the rear side as it climbs the incline.

While the CAESAR robot is climbing an incline and no object is detected, the front flipper arms will be at 150°, while the rear flipper arms will be at 135°. Should an object be detected while on the incline, the front flipper arms will move to 135°.

Should the CAESAR robot be on a decline and no object is detected, the front flipper arms will move to 135° and the rear flipper arms will move to 150°. This will support the front of the CAESAR robot should it approach or slide to an object. It would appear as if an object is being detected when the CAESAR robot approaches a horizontal terrain. With the front flipper arms at 135°, it will assist it to stabilize on the horizontal terrain.

A summary of the flipper arm orientation for the CAESAR robot in the normal orientation or upside down orientation is shown in table 5-1.

Table 5-1: Flipper orientation with the corresponding robot orientation

Angle	Object Detection	Action
- 10° to 10°		Front and rear flipper arms @ 150°
-170° to 180° or 170° to 180°		Front and rear flipper arms @ 240°
-10° to 10°	√	Front flipper arms @ 135°, rear flipper arms @ 135°
-170° to -180° or 170° to 180°	√	Front flipper arms @ 225°, rear flipper arms @ 225°
10° to 90°		Front flipper arms @ 150°, rear flipper arms @ 135°
-90° to -170°		Front flipper arms @ 240°, rear flipper arms @ 225°
10° to 90°	√	Front flipper arms @ 135°, rear flipper arms @ 135°
-90° to -170°	√	Front flipper arms @ 225°, rear flipper arms @ 225°
-10° to -90°		Front flipper arms @ 135°, rear flipper arms @ 150°
90° to 170°		Front flipper arms @ 225°, rear flipper arms @ 240°

5.2 Gas Concentration Decisions

The gases that are of primary importance in a search and rescue event is carbon dioxide, carbon monoxide, hydrogen sulphide, methane and oxygen [22]. Other gases and fumes could also be detected, which could be considered in future research. Most sensors give an output of gas concentration which is measured in parts per million (ppm). This data may be meaningless to the controller as it might hold no threat. An example will be the detection of methane gas. Should the robot detect 1 ppm of methane, this could possibility not be dangerous as it could be either a natural gas in the environment or of such a small quantity that it will not cause an explosion.

Previous USAR robots did not have the ability to detect the gas concentrations and to immediately notify the rescuers. Furthermore, at the World Trade Center scenario, the rescuers were not able to immediately determine the environmental dangers that the gas concentrations had on victims, rescuers and the robots. [3]

5.2.1 Gas Analysis

Analysis of the gases and their respective properties needed to be evaluated [53]. Different aspects of the gases were analyzed to determine the concentrations that would be considered dangerous.

The Immediately Dangerous to Life or Health (IDLH) levels are used for non-emergency and emergency scenarios [54]. These IDLH concentrations are determined with the following considerations [55]:

- People must be able to escape the danger without the loss of life or irreversible health effects that could happen after exposure to that environment for a time period of 30 minutes.
- Prevention of severe eye or respiratory irritation which will prevent a person from escaping the dangerous environment.

The compiled properties of the gases are represented in table 5-2.

Tables 5-2: Properties of gases of interest.

Substance	IDLH (ppm)	TLV (ppm) *	Color
CO2	40 000 [54]	5000 [56]	Colorless / black – sublimation [57]
CO	1 000 [58]	25 [58]	Colorless [57]
H2S	100 [54]	10 [57]	Colorless [57]
Methane	5000 ***	1000 [59]	Colorless [60]
Oxygen	**	**	Colorless

Substance	Smell	Flammability Percentage	NFPA Health / Flammibility / Reactivity
CO2	Odorless [57]	Non-flammable [57]	3 / 0 / 0 [57]
CO	Odorless [57]	12% - 75% [57]	3 / 4 / 0 [57]
H2S	Rotton Egg [57]	4.3% - 46% [57]	4 / 4 / 0 [57]
Methane	Odorless [60]	5% – 15% [60]	1 / 4 / 0 [60]
Oxygen	Odorless	Non-flammable **	N/A **

* Threshold Limit Value
** Oxygen is not flammable, but assist with combustion Oxygen level that are required to function mentally is 19.5% [55]. Higher concentrations of Oxygen does not have serious effect on a person, but could cause sever explosions.
*** As methane is an asphyxiant, there is no IDLH data available [61]. A value for IDLH of five times that of the TLV is used.

Concentrations of the gases up to the level of the Threshold Level Value (TLV) are considered to be safe. Any gas concentrations between the TLV and the IDLH are considered unsafe, while any concentration above the IDLH are dangerous. Table 5-3 shows a combination of all these properties in percentages for an environment. The $Unsafe_{human}$ is the TLV, while $Dangerous_{human}$ is the IDLH value.

Table 5-3: Gas properties in percentages

Substance	$Unsafe_{human}$(%)	$Dangerous_{human}$(%)	Flammable (%)
CO_2	0.5	4	Non-flammable
CO	0.0025	0.1	12 % - 75 %
H_2S	0.001	0.01	4.3 % - 46 %
CH_4	0.1	0.5	5 % - 15 %
O_2	< 19.5	< 16.9	Non-flammable

The gas concentration in table 5-3 needs to be converted to a ratio with respect to 1 million. As it becomes more dangerous for humans when the oxygen decreases, the values required are subtracted from 1 million. This allows for monitoring values that will be increasing throughout the table. The measurements for the oxygen concentration will also need to be deducted from 1 million to get an accurate decision. This is shown in table 5-4.

Table 5-4: Gas properties in ratio with respect to 1 million

Substance	$Unsafe_{human}$(ppm)	$Dangerous_{human}$(ppm)	$Flammable_{min}$ (ppm)
CO_2	500	4000	Non-flammable
CO	25	1000	120 000
H_2S	10	100	43 000
CH_4	1000	5000	50 000
O_2	805 000	831 000	Non-flammable

Using the above data, it is possible to alert the rescuers to possible conditions that could occur in the environment. These conditions could either be considered dangerous for humans or for the robot. A dangerous level for the robot is the gas flammable value. With this information the rescuers could determine whether to risk their lives or the robot to enter a room with this environmental conditions.

5.2.2 CAESAR PC AI for Gases

Fuzzy logic is a way to work with logical expressions that are neither true or false. This type of reasoning is used to determine the unsafe, danger and flammable possibilities. The standard rules for fuzzy truth (T) are the following [62]:

$$T(A \wedge B) = min(T(A), T(B)) \quad \{19\}$$

$$T(A \vee B) = max(T(A), T(B)) \quad \{20\}$$

$$T(\neg A) = 1 - T(A) \quad \{21\}$$

where $\wedge$ is AND

V is OR

$\neg$ is NOT

This form of expressions was selected as it allows for the determining of the unsafe, danger and flammable ranges. For the above rules to be applied to the gas concentrations, they needed to be associated with a relationship referenced to the percentages in table 5-2, which were the boundaries. The value g_n which is the specific gas concentration read from the sensors, is a value per million. This value has to be normalized with respect to 1 million to get a ratio. The unsafe value for humans (u_h), dangerous value for humans (d_h) and flammable value (f) are also normalized with respect to 1 million to get a ratio for the boundary values. Equations 22, 23 and 24 are used to determine A, B and C respectively.

$$A = g_n - u_h \quad \{22\}$$

$$B = g_n - d_h \quad \{23\}$$

$$C = g_n - f_{min} \quad \{24\}$$

where A, B and C are the variables to be substituted into equations 19, 20 and 21.

In the event that the values of A, B and C are negative, the environment is safe for humans. Should any of the values become positive, it indicates that the gas concentration is either unsafe, dangerous or flammable.

Using equation 20 and equation 21, and only the positive numbers of A, B and C, (denoted by pos()), then

$$\begin{aligned} & T(pos(\neg A) \vee pos(\neg B) \vee pos(\neg C)) \\ = & max(T(pos(\neg A)), T(pos(\neg B)), T(pos(\neg C))) \end{aligned} \quad \{25\}$$

Combining equations 22, 23 and 24 with equation 25, results,

$$\begin{aligned} & T(pos(\neg(g_n - u_h)) \vee pos(\neg(g_n - d_h)) \vee pos(\neg(g_n - f_{min}))) \\ = & max(T(pos(\neg(g_n - u_h))), T(pos(\neg(g_n - d_h))), T(pos(\neg(g_n - f_{min})))) \end{aligned} \quad \{26\}$$

Let equation 26 relate to a function (K) that returns a solution to the logical expression. To determine if the gas concentrations are unsafe, dangerous or flammable, the boundaries u_h, d_h and f_{min} are compared to ¬K, which concludes the possible decision (D). With this model, it specifies P(Safety of the environment | specific gas concentration).

Different solutions of safety levels are expected from each gas analysis. All the solutions from the different gases are required to determine the safety of the environment. As the order of safety (being unsafe, dangerous and flammable) decreases and the gas concentration increases, the final decision is considered in respect of the worst case solution from the different gases. This is the maximum value of D_n. Should one gas concentration be flammable but another only unsafe for humans, then the flammability of the gas takes priority.

Should the gas concentration be between $Unsafe_{human}$ and $Dangerous_{human,}$ the robot then could continue to search for victims. In the event that the gas concentrations is higher than the $Dangerous_{human}$ level, the possibility for humans to survive in these

conditions is decreasing, and the rescuers must decide about entering the environment depending on other safety issues. These safety issues could be falling debris or unstable surfaces. As the gas concentration for the $Flammable_{min}$ condition is much higher than that of the $Dangerous_{human}$ levels, it could imply that humans would not survive in these environments. The robot could make the decision to evacuate the environment and possibly save itself from an explosion or search for survivors in other areas of the disaster. These logical decisions will be determined using the weighting table shown in table 5-5.

Table 5-5: Gas weighting factors

Substance	**$Unsafe_{human}$**	**$Dangerous_{human}$**	**$Dangerous_{robot}$**
CO_2	1	2	0
CO	1	2	1
H_2S	1	2	1
CH_4	1	2	1
O_2	1	2	0

There are two types of warnings that have to be considered. The unsafe / danger factor for humans and the danger factor for the robot. A model is required to determine the danger for humans. This is achieved with equation 27.

$$Danger = \frac{100}{2n}.(w_u.p + w_d.q) \quad \{27\}$$

where n = number of gases being considered

p = number of gases that give an unsafe warning

q = number of gases that give a danger warning

w_u = unsafe human weighting factor

w_d = Dangerous human weighting factor

The above models give a percentage level for danger for humans. As the number of unsafe and dangerous factors for humans increases, the model increase the percentage value.

A model is also required for the danger of the robot. As seen in table 5-5, carbon dioxide and oxygen does not have a weighting factor as these gases are not flammable. This danger for the robot is expressed by the model shown in equation 28.

$$Danger_{robot} = Gas_{HighestConcentrationPercentage} + \left(\left(\frac{100}{m}.\sum_{n=1}^{n=m}\frac{x_n}{D_n}\right)\left(\frac{100 - Gas_{HighestConcentrationPercentage}}{100}\right)\right) \quad \{28\}$$

where m = number of gases not giving Danger warnings and that $w_n \neq 0$

D_n = danger that gas n has on robot ($flammable_{min}$)

Should any one gas have a concentration higher than $flammable_{min}$, the environment is considered to be 100 % dangerous for the robot. The danger for the robot could increase as other gas concentrations increase, but it will never decrease below the highest danger percentage.

Equation 28 could be used to determine the danger or unsafe value for humans, but D_n will be the danger or unsafe value that gas n has on humans. This is a more accurate result and desirable to use, compared to equation 27, which only monitors the limits of the gas concentrations.

The models shown give a probability that humans would be able to survive in the surrounding environments and the probability that the robot is in a dangerous environment. All the decisions described above are performed by the control station as it randomly requests for data on environmental status. The CAESAR robot responds to the request and awaits its next instruction. The control station evaluates the procedures that the rescuers and the robot must follow from the information received.

5.2.3 CAESAR Gases Detection

Sensors are required to determine the gas concentrations. The decision was for the Figaro range of gas sensors, as these sensors have a low price range compared to that of other available sensors. Many of the other units available are expensive or are built with circuitry for a specific task.

The Figaro sensors were used in CAESAR. The manufacturers developed these sensors but were not able to relate the sensor properties to the gas concentration being measured. This problem was solved by determining equations that could estimate close readings of the gas concentrations that are of interest. The following gas analysis were performed.

5.2.3.1 Carbon Dioxide (CO_2)

The TGS 4162 CO_2 sensor has the following specifications [63]:

- High selectivity to CO_2
- Compact Size
- Low dependency of humidity
- Long life and low cost
- Typical Detection: 350 – 10 000 ppm

- Response time: approximately 1.5 minutes

The sensitivity characteristic curve of CO_2 compared to other gases such as CO, ethanol and hydrogen is shown in figure 5-2 [63].

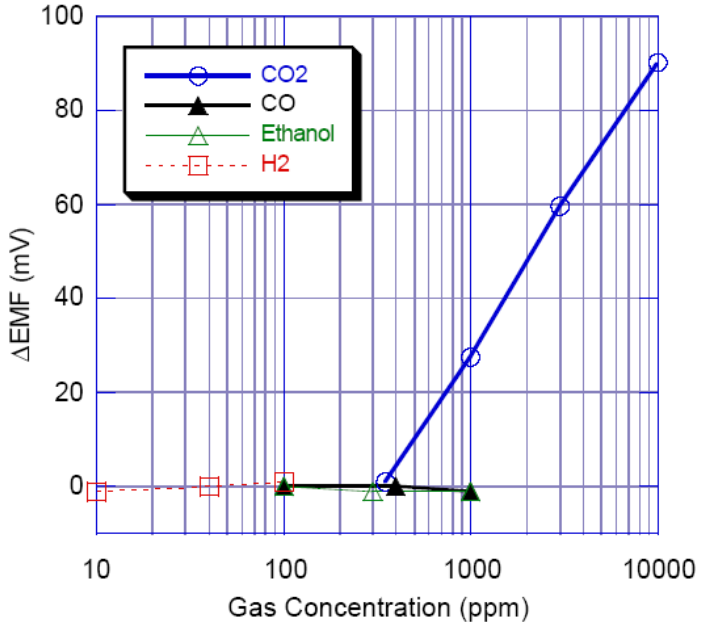


Figure 5-2: Sensitivity characteristic curve and gas comparison with CO_2

The amplification stage could consist of an operational amplifier to create a high impedance (>100 GΩ) and a bias current of less than 1 pA. This is shown in figure 5-3 [63].

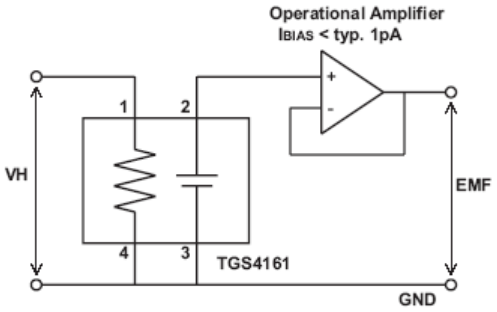


Figure 5-3: Possible amplification stage between the CO_2 sensor and the micro-controller

VH is a 5 V supply connection. This permit the sensor to be heated to a specific temperature for an optimal reading. The reading will be inaccurate in the event that the environment has a high temperature, but it could be useful to measure the gas concentrations in environments that do not have high temperatures but have dangerous gas concentrations.

The Electro Motive Force (EMF) for gas concentrations of 350 ppm is 0V. There is a relatively linear comparison between the gas concentration and the EMF. The calculation of gas concentration g_n is described by equation 29. The constants of equation 29 were calculated by knowing two points on the graph and substituting them into a straight line graph equation.

$$g_n = \frac{1}{k}.10^{\frac{EMF+0.163579}{0.0643}} \quad \{29\}$$

where k = the multiplication of the amplification stage

EMF = measured voltage with respect to mV

5.2.3.2 Carbon Monoxide (CO)

The TGS 2442 CO sensor has the following specifications [64]:

- Low power consumption
- High sensitivity to CO
- Miniature size
- Low sensitivity to alcohol vapor
- Long life and low cost
- Low humidity dependence
- 30 – 1000 ppm

The sensitivity characteristic curve of CO compared to other gases such as H_2, ethanol and air is shown in figure 5-4 [64].

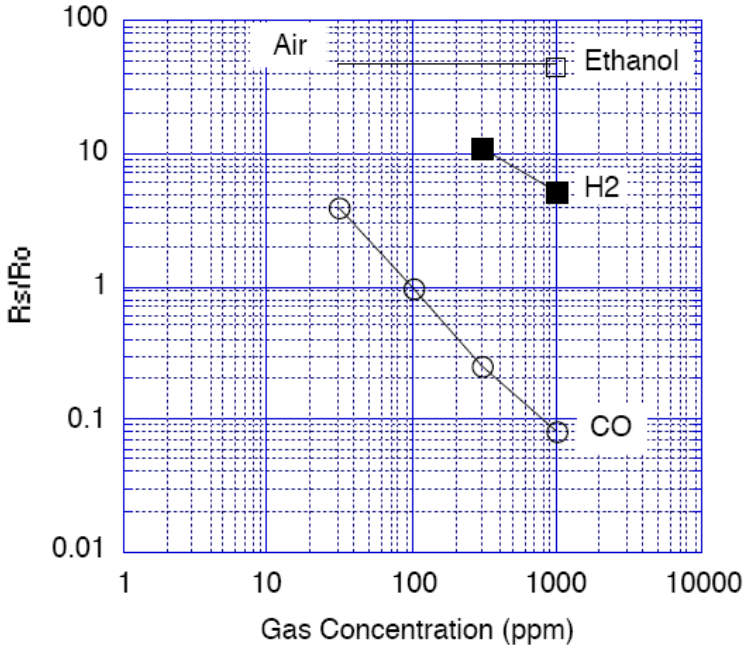


Figure 5-4: Sensitivity characteristic curve of CO gas in comparison with other gases

The circuitry that is required for the CO sensor is shown in figure 5-5 [64].

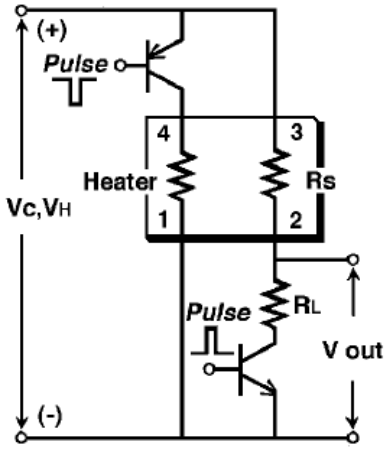


Figure 5-5: Circuit diagram required for CO sensor

The process of reading a voltage is produced by creating a pulse on the heating circuitry for 14 ms, and then switching the heating circuitry off for 986 ms. The rest of the circuitry must simultaneously be switched off for 995 ms and then switched on for 5 ms, during which time the measurement of voltage must be taken. This is shown in figure 5-6 [64].

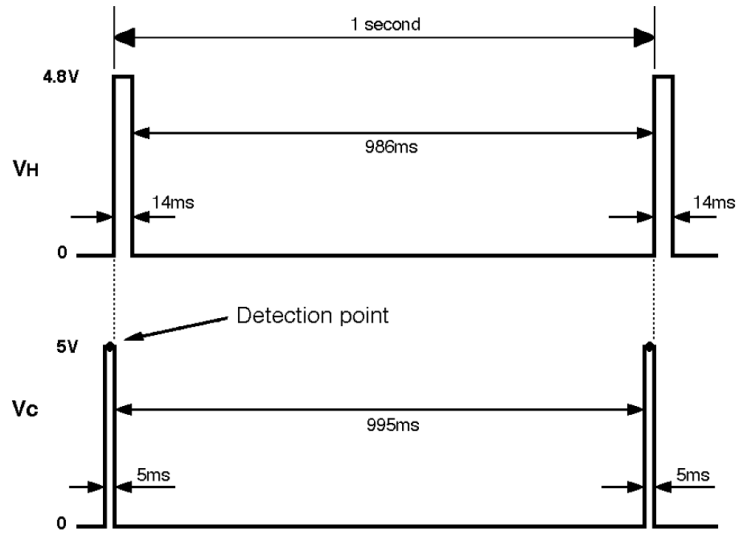


Figure 5-6: Timing diagram for the CO gas measurements

The graph shown in figure 5-4 is relatively linear, so gives equation 30. The constants of equation 30 were calculated by knowing two points on the graph and substituting them into a straight line graph equation.

$$\frac{R_s}{R_0} = -1.1156305 \log(g_c) + 2.24998 \quad \{30\}$$

The voltage dividing circuit with R_S and R_L achieves V_{out}, which results equation 31.

$$R_S = R_L\left(\frac{5}{V_0} - 1\right) \quad \{31\}$$

Substituting equation 31 into equation 30,

$$g_n = 10^{\frac{\log\left(\frac{R_L}{R_0}\left(\frac{5}{V_0}-1\right)\right) - 2.24998}{-1.1156305}} \quad \{32\}$$

where: R_0 = Sensor resistance at 100 ppm CO = 13.3 kΩ – 133 kΩ

As the minimum value for R_L is 10 kΩ, this resistance is used.

5.2.3.3 Methane (H_2S)

The TGS 825 H_2S sensor has the following specifications [65].

- High sensitivity to H_2S
- Good repeatability of measurements
- Uses simple electrical circuit

The sensitivity characteristic curve of H_2S compared to air is shown in figure 5-7 [65].

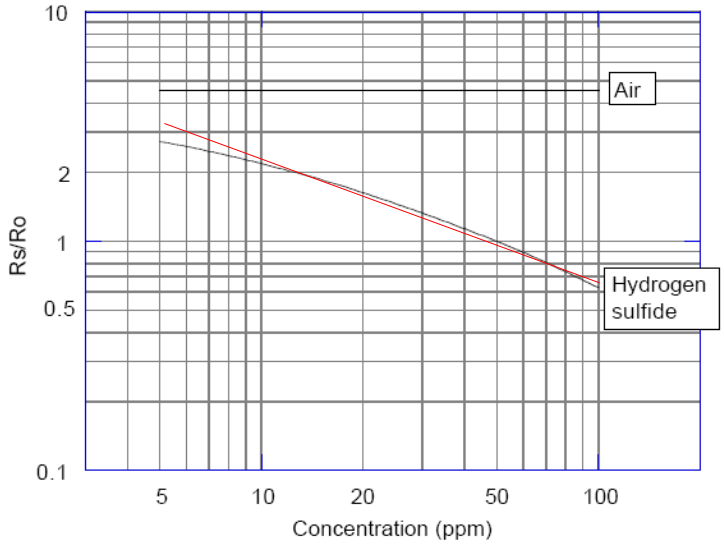


Figure 5-7: Sensitivity characteristic curve of hydrogen sulphide gas in comparison with air

To determine the gas concentration from this sensor, a linear graph is used as shown by the red line in figure 5-7. The values are very close between the 10 ppm and 100 ppm which are the important concentrations.

The circuitry that is required for the H_2S sensor is shown in figure 5-8 [65].

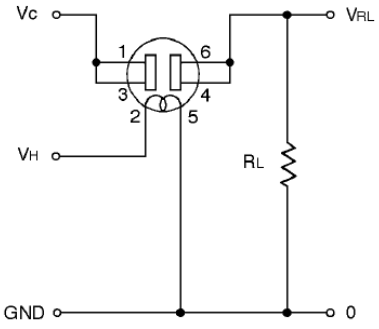


Figure 5-8: Circuit diagram required for H_2S sensor

The minimum value for R_L is 450 Ω, so a 470 Ω resistor is used.

With the linear log graph shown in figure 5-7, it can be expressed by equation 33. The constants of equation 33 were calculated by knowing two points on the graph and substituting them into a straight line graph equation.

$$\frac{R_S}{R_0} = 10^{-0.507\log(g_c)+0.836} \quad \{33\}$$

substituting equation 31 into equation 33 gives equation 34:

$$g_n = 10^{\frac{\log(\frac{R_L(\frac{5}{V_0}-1)}{R_0})-0.836}{-0.507}} \quad \{34\}$$

where: R_0 = Sensor resistance at 50 ppm hydrogen sulphide = 3 kΩ – 30 kΩ.

5.2.3.4 Methane (CH_4)

The TGS 2611 - E00 Methane sensor has the following specifications [66]:

- Low power consumption
- High sensitivity to Methane
- Long life and low cost
- Uses simple electrical circuit
- Contains a filter to prevent interference of other alcohol gases

The sensitivity characteristic curve of CH_4 compared to air, ethanol, iso-butane and hydrogen is shown in figure 5-9 [66].

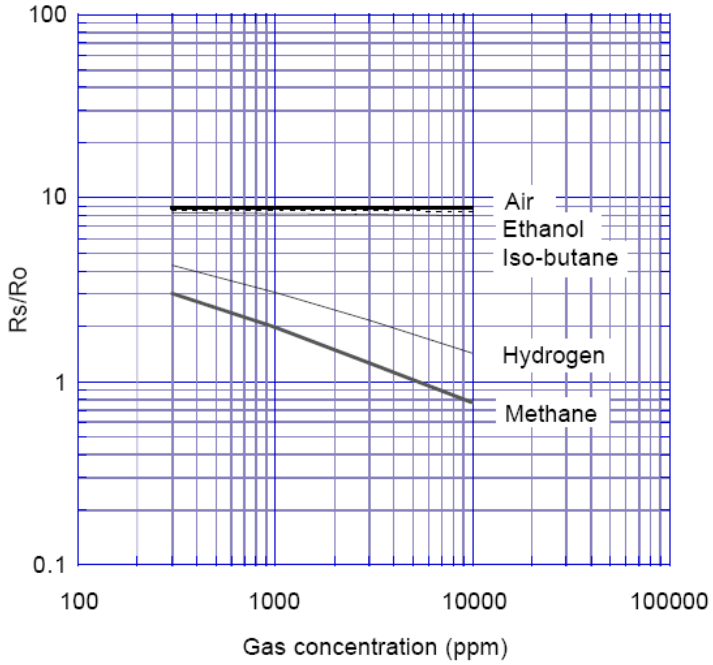


Figure 5-9: Sensitivity characteristic curve of methane gas in comparison with other gases

The circuitry that is required for the CH_4 sensor is shown in figure 5-10 [66].

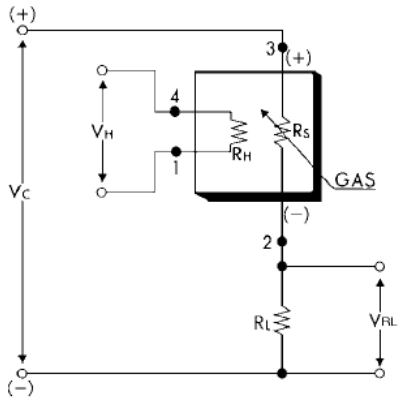


Figure 5-10: Circuit diagram required for CH_4 sensor

The minimum value for R_L is 450 Ω, so a 470 Ω resistor is used.

With the linear type of log graph shown in figure 5-9, it can be expressed by equation 35. The constants of equation 35 were calculated by knowing two points on the graph and substituting them into a straight line graph equation.

$$\log\left(\frac{R_s}{R_0}\right) = -0.39794 \log(g_c) + 1.47664 \quad \{35\}$$

substituting equation 31 into equation 35 gives equation 36:

$$g_n = 10^{\frac{\log\left(\frac{R_L\left(\frac{5}{V_0}-1\right)}{R_0}\right)-1.47664}{-0.39794}} \quad \{36\}$$

where: R_0 = Sensor resistance at 5000 ppm methane = 0.68 kΩ – 6.8 kΩ.

5.2.3.5 Oxygen (O_2)

The SK-25 Oxygen sensor has the following specifications [67]:

- Virtually no influence from CO_2, CO, H_2S, NO_x and H_2
- Good linearity
- Stable output signal
- No external power supply needed to operate sensor
- No warm-up time required

The sensitivity characteristic curve of O_2 is shown in figure 5-11 [67].

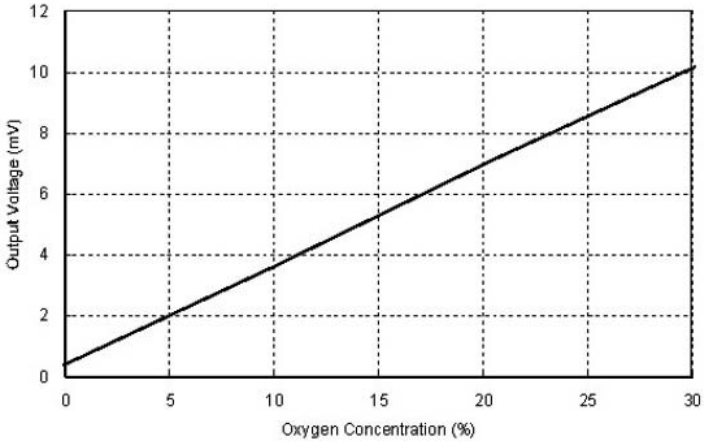


Figure 5-11: Sensitivity characteristic curve of oxygen

The linear type graph shown in figure 5-11 can be expressed with equation 37. The constants of equation 37 were calculated by knowing two points on the graph and substituting them into a straight line graph equation.

$$EMF = 0.32\,g + 0.4 \quad \{37\}$$

where: EMF = Output voltage (in mV)

g = Oxygen concentration (%)

As the voltages are in millivolts, the voltage needs to be amplified by a gain of k, as the A/D converter on the micro-controllers can measure the EMF in steps of 4.9 mV. This results in equation 37 needing to be divided by the amplified gain k, resulting in equation 38.

$$EMF = \frac{1}{k}(0.32\,g + 0.4) \quad \{38\}$$

5.3 RCP Decisions

The RCP module is an intelligent router. A series of different decisions must be considered to ensure the protocol is followed. A description of the flow of decisions is as follows.

5.3.1 Serial Data Received Interrupt

After the character is received, it must firstly be determined from which module the data is received. Should it be from the default module, the following steps are followed:

1. Is the character a “#”. If so, then the first character of the receiving array is set to be the received character and the array position is set to zero.
2. In the event that the character is not a “#”, the array position is incremented and the received character is stored in the receiving array.

3. Should the position in the receiving array be equal to 10, then it must be verified whether the packet is meant for the receiving station. If not, then the reception of data is delayed for the time period indicated in the duration fields.
4. When a “!” is received, the checksum of the packet is validated. If it is equal to the received checksum value, the packet is accepted and the module will respond if needed.

5.3.2 Packet Analysis and Transmission Packet

The RCP module will continuously analyze a new received packet and create a transmission packet to be sent either to the other station, or to the application modules. The following steps are followed:

1. Has a new packet been received and is it from the other valid station. If so,

   a) The first transmission character is a “#”

   b) Analyze the second character. In the event that it is a:

      1. “0” then the second transmission character is a “1”
      2. “1” then the second transmission character is a “3” when the sent data is a RO packet, or a “4” when the sent data is a RC packet.
      3. “4” then the second transmission character is a “2”

   c) If the application module wants to send data, then the second transmission character is a “0”

   d) Should the application module want to send data, then calculate the duration of communication, and set the duration fields to this value. If a packet was received from another station, recalculate the duration of communication after deducting the duration of past communication for the current session. Validate whether the values in the duration fields are printable and valid characters. If not, modify the duration fields to abide to the rules of the protocol.

   e) Insert the callsign of the desired station that the packet has to be sent to and the callsign of the local station.

   f) In the event that the transmitted packet has a Type value of “3” or “4”, add the data from the application layer to the transmission packet.

   g) Generate a checksum value and validate if the checksum value is a printable and valid character. Add this to the transmission packet. The code used to generate the checksum byte is:

```
 cksmT = 35;  //set the initial value of cksmT to 35
//evaluate the cksmT value is the packet contains data
   if ((Txarr[1] == 51) || (Txarr[1] == 52))
      {
         // continue evaluating the cksmT value if it is a “#”, “!” or non-printable character
         while ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1))
```

```
    {
    cksmT = 0;
    for (i = 0; i <= 15 + Datalength - 1; i++)
      {
       //chksmT value is modulate 94
       cksmT = fmod((Txarr[i] + cksmT),94);
       }
       // 32 is added to the cksmT value to ignore the non-printable characters
       cksmT = cksmT + 32;
     // if the cksmT value is a “#”, “!” or non-printable character, increment the duration
          fields slightly
    if ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1)) Txarr[2]++;
    }
}
// if the packet dows not contain data
else if ((Txarr[1] == 48) || (Txarr[1] == 49) || (Txarr[1] == 50))
    {
    // continue evaluating the cksmT value if it is a “#”, “!” or non-printable character
    while ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1))
       {
       cksmT = 0;
       for (i = 0; i <= 15 ; i++)
         {
          //chksmT value is modulate 94
          cksmT = fmod((Txarr[i] + cksmT),94);
          }
          // 32 is added to the cksmT value to ignore the non-printable characters
          cksmT = cksmT + 32;
       // if the cksmT value is a “#”, “!” or non-printable character, increment the
          duration fields slightly
       if ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1)) Txarr[2]++;
       }
    }
```

where: cksmT is the checksum value that is being generated

Txarr is the transmission array

Datalength is the length of data that the application module wants to send

h) End the packet with a “!” character.

i) Transmit the packet to the other station.

2. If data is received from the other station that must be sent to an application module, then transmit the data to the specific module indicated within the data field. If the data field requires a response from the application module, then wait until it is received, so that preparation can be made for the data to be sent back to the other station.

5.4 Conclusion

Different localization methods are explained with their advantages and disadvantages. The CAESAR localization methods that allow it to transform autonomously or manually if required by the control station, are discussed.

Decisions were made from the analysis of gases and their concentrations in the environment. Models were developed that enabled this analysis, determining if the environment was dangerous to humans or for the robot. These models incorporate fuzzy logic rules and different weighting factors depending on the concentration of the gases. This would assist rescuers in determining whether it is safe or worthwhile to risk their lives to enter the disaster. The immediate gas analysis and danger factor of the environment has not been obtainable with previous USAR robot operations [3].

The gas sensors used have been analyzed and a relationship has been determined between the gas concentration and the output voltage which would be dependent on the resistances in the potential divider circuitry. This relationship was not available or determined by the sensor manufacturers.

The decisions that the Robotic Communication Protocol requires for a reliable communication network were also explained. This is achieved by analyzing the packets received and responding with the necessary packet. Incorporating an intelligent router that operates as the RCP prevents incorrect packets being sent to the wrong station and resulting in a robot following instructions that was not intended for it, as was experienced at the World Trade Center [16]. The RCP also determines if the received packet has any errors in it, which is caused by interference from surrounding equipment as experienced previously.

CHAPTER 6 – TESTING AND VERIFICATION

6.1 Communications

Tests were performed using the RCP protocol in the laboratory environment and this was found to be reliable for the mobile robot application as instructions were followed correctly by the appropriate station. Packets were rejected when they were not meant for that specific station or when the checksum of the packet was not correct. It responded with the appropriate packets and routed the data to the application modules and visa versa. Due to the limited memory of micro-controllers the EEPROM was used for the storing of arrays. It was determined that, depending on the amount of memory available, the length of the data field was proportional.

With the UHF frequencies used, a limited reflection of the radio waves by building materials was experienced. The useful advantage of this is that it allowed the radio signals to be reflected between stations where previous contact would not have been possible. This permitted further beyond line of sight communication, but cannot be relied on.

The output power of the RF amplifier was dependent on the drive stage's connection to the positive voltage via a resistor. The resistance was varied to achieve the output power required. This relationship is shown in figure 6-1.

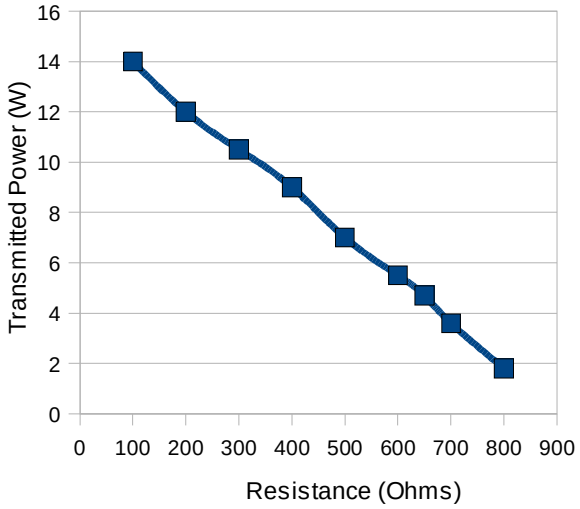


Figure 6-1: Transmitted power and drive resistance relationship

Figure 6-1 indicates that the resistance was inversely proportional to the transmitted power. With a resistance value of 680 Ω, an output power of 4.7 W was achieved. This output power was acceptable as it was not above the 5 W restriction.

The observation was that with the increase of RF power, the electronics within the control station did not behaved as required. After numerous tests it was noted that the radiation emitted from the amplifier stage caused interference with the rest of the system. A Faraday cage was developed around the RF circuitry to shield the emitted radiation. Figure 6-2 shows the control station comparison with the Faraday cage.

Figure 6-2: Control station with the Faraday cage on the right hand side

Using the egg-beater antennas proved to give a lower performance compared to the communication between two vertical antennas. This lower performance was expected, as the signal interception with the loop of the eggbeater antennas was less, but it did allow for video, data and audio communication irrespective of the orientation of the antennas in the robot.

Video transmission tests were also performed and a good signal of audio and video was received. The thermal camera also revealed heated areas and it made it possible to identify human features in dark environments.

6.2 Ultrasonic Distance Sensor Testing

Voltage at the output of the range sensor had been stated to be approximately 11-12 V. Verification of this voltage output over the specified range had initially been carried out before the implementation of the potential divider circuitry by means of a test procedure: A flat plate had been held perpendicular to the face of the sensors at various distances, with the voltage output measured using a Digital Multimeter. Figure 6-3 shows the results obtained.

The graph confirmed the expected voltage over the specified range allowing the assumption that this voltage output would pass through when the additional circuitry had been connected to that of the sensor output.

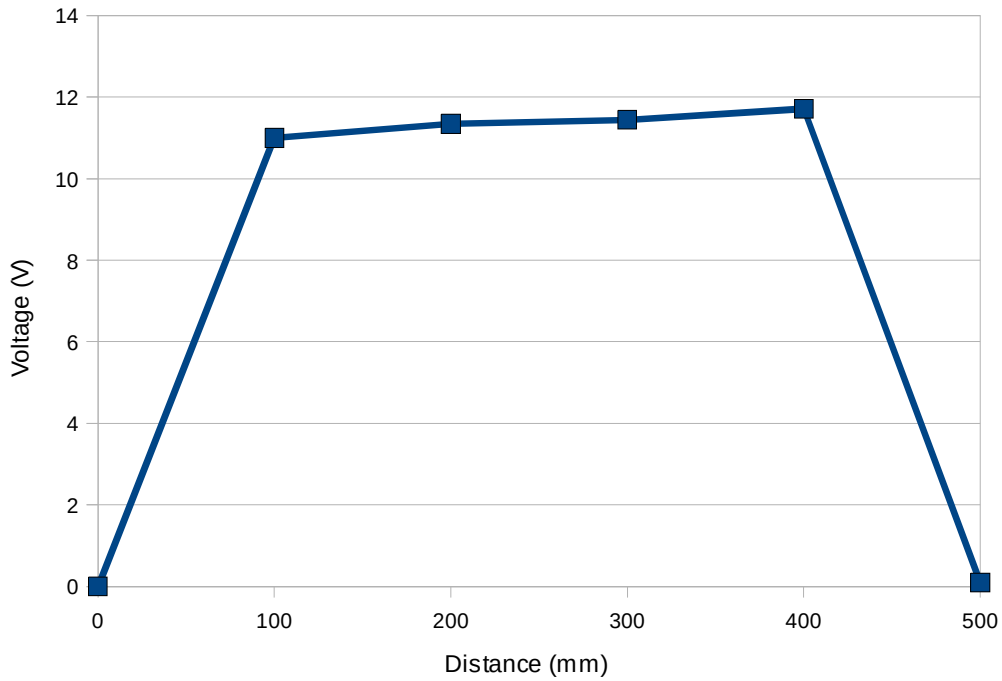


Figure 6-3: Voltage output vs distances for the configuration of a distance detection of 400 mm

The potential divider circuit was also tested. This was conducted by means of the same experiment mentioned above. Measurements were taken from the output of the circuit connected directly to the micro-controller. The circuit had been expected to reduce the voltage output of the sensor to approximately 4.9 V. Figure 6-4 shows the results obtained.

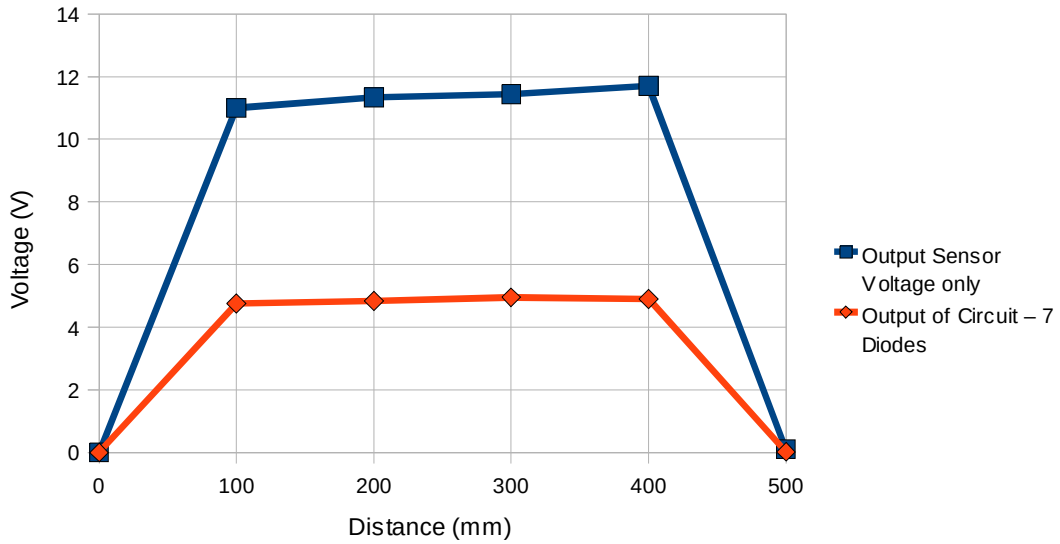


Figure 6-4: Output voltages measured before and after the interconnection circuitry vs the distance of detection

The tests revealed that the output required for the micro-controller to use as an

input, is the required voltage to initiate transformation.

6.3 Gas Sensory System / Control System

During testing the gas sensors detected within 1 second, an increase of the gas concentration. Stabilization of readings occurred after a time period of about a minute.

The gas sensors were monitored to determine the consistency of the output in a nonfluctuating gas environment. This was achieved by performing the tests in a sealed container. It is important to perform this type of testing, to evaluate if the sensors are consistent. Calibrated sensors available from the fire department allowed for the verification of the gas concentrations. These monitored concentrations are shown in figure 6-5.

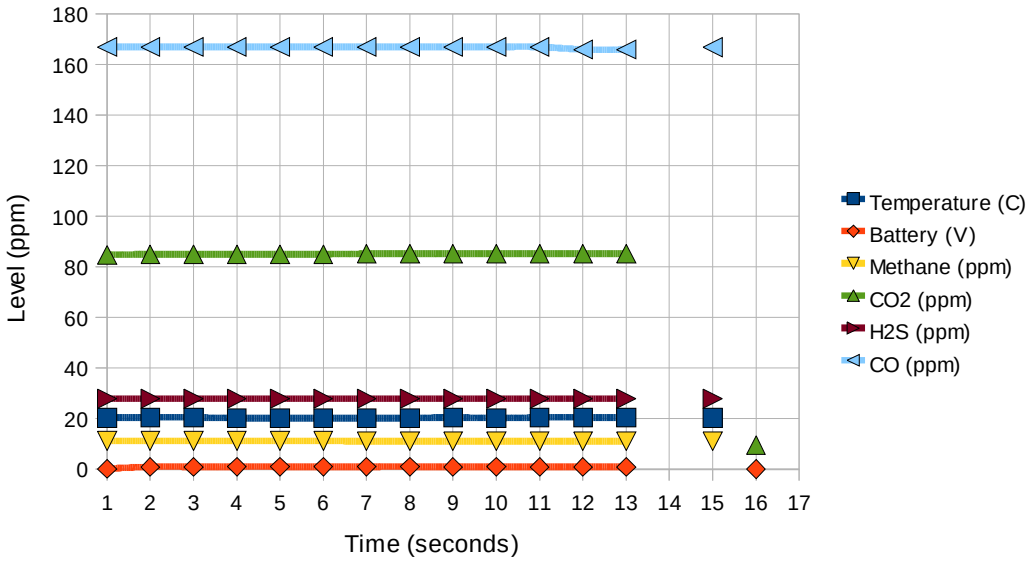


Figure 6-5: Gas concentration levels indicated by the gas sensors in a constant gas environment

In figure 6-5 the temperature and battery monitoring are also displayed. These sensors showed a constant output. CO_2 gases indicated a concentration of 165 ± 1 ppm, while H_2S indicated 40 ± 1 ppm. CO indicated a concentration of 33 ± 3 ppm while methane indicated 28 ± 0 ppm. The gas concentration levels are viable as about 0.2 % of the atmosphere consists of CO_2 and other gases. Chauvenet's criterion statistical analysis [68] was performed on the readings to determine the validity. All readings were acceptable.

The control station consisted of a Graphics User Interface (GUI) on a computer system. This allowed easy control of the robot, video capturing and the ability to perform mathematical calculations due to the processing power. This GUI is developed to allow multi-agent robots to be controlled from a single system. Rescuers do not need to be familiar with different interfaces, as all the robot's controls are similar with a selection of unique instructions that are only feasible for a

robot.

This interface was also developed for the First Encountered Assisting Robot (FEAR) [69]. Figure 6-6 shows the GUI for the CAESAR robot. The gas concentration levels are shown with battery and internal temperature data. Flipper arm orientation selection is accessible, depending on whether the manual or automatic mode is selected.

(a)

(b)

Figure 6-6: (a) GUI for the CAESAR robot with the video from the thermal camera on the right (b) Example of color indication for warnings

The AI that was incorporated with the GUI system, designed to determine if the conditions were unsafe or dangerous for humans. A danger probability for the robot was also indicated. The environment danger information assists the rescuers with decisions for further rescue attempts.

6.4 Traction System and Transformability

CAESAR has an overall weight of 56.5 kg, which is significantly in excess of the target of 25 kg. The weight at initial tests were 25 kg, but the addition of the overall payload, the protective layer and the phenolic resin to bond the protective layer added the additional 31.5 kg. Weight could be reduced with the layering and manufacturing of the composite material in a single stage, as the amount of phenolic resin used for bonding is then reduced.

The CAESAR robot was tested on different terrains to determine the feasibility of the traction. The silicone pads decreased slippage on smooth and oily surfaces while the chain-type tracks allowed grip for climbing over obstacles.

The efficiency of the traction system is required to determine the difference of the actual output compared to the theoretical output. This decrease in the output value is due to friction caused by the load of the components used by the system.

The sprocket configuration of the side tracks are shown in figure 6-7.

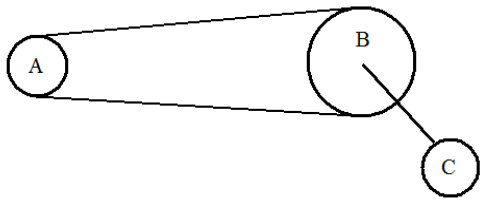


Figure 6-7: Sprocket / chain configuration of the main tracks

The motor is connected to sprocket A, which drives sprocket B. Sprocket C is joint to sprocket B via a shaft. The specifications of A is:

$Torque_A$ = 12 Nm

N_A = 30 rpm

T_A = 13 teeth

Radius: 0.02 m; Diameter: 0.04 m

Sprocket B has the following specifications:

T_B = 21 teeth

Radius: 0.0325 m; Diameter: 0.065 m

Sprocket C has the following specifications:

T_C = 13 teeth

Radius: 0.04 m; Diameter: 0.08 m

The turn rate of sprocket B is required. This is determined from:

$$N_A T_A = N_B T_B$$

$$\rightarrow N_B = 30\, x \frac{13}{21} = 18.571\, rpm \qquad \{39\}$$

Similarly,

$$Torque_A T_A = Torque_B T_B$$
$$\rightarrow Torque_B = 12\ x\ \frac{21}{13} = 19.385\ Nm \quad \{40\}$$

As $N_B = N_C$, the velocity of C with diameter of 0.08 m, determines the velocity of the tracks, by:

$$v = \pi D n = \pi\ x\ 0.08\ x\ \frac{18.571}{60} = 0.078\ m/s \quad \{41\}$$

Comparing this theoretical velocity to the actual velocity of the tracks of 0.074 m/s (without the weight of the robot), the efficiency can be determined as shown in equation 42.

$$Efficiency = \frac{velocity_{actual}}{velocity_{theoretical}}\ x\ 100 = \frac{0.074}{0.078}\ x\ 100 = 95\ percent \quad \{42\}$$

The loss in efficiency is due to friction of bearings and sprockets.

Similar calculations can be performed for the efficiency of the flipper arms. The flipper arms has a configuration as shown in figure 6-8.

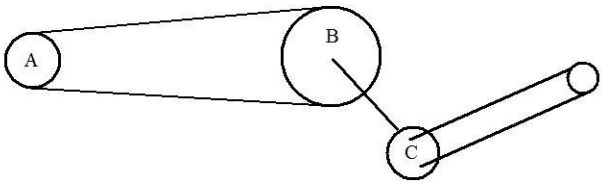


Figure 6-8: Flipper arms gear ratio

Sprocket A has half the torque, as this is split between two arms. Sprocket A has the following specifications:

$Torque_A$ = 6 Nm

N_A = 30 rpm

T_A = 13 teeth

Radius: 0.02 m; Diameter: 0.04 m

The known specifications for sprocket B is:

T_B = 30 teeth

Radius: 0.046 m; Diameter: 0.092 m

The turn rate of sprocket B is determined by:

$$N_A T_A = N_B T_B$$
$$\rightarrow N_B = 30\, x\, \frac{13}{30} = 13\, rpm \quad \{43\}$$

Similarly,

$$Torque_A T_A = Torque_B T_B$$
$$\rightarrow Torque_B = 6\, x\, \frac{30}{13} = 13.846\, Nm \quad \{44\}$$

As N_B = N_C, 0.231 revolutions will occur in a second. Therefore, one revolution will occur in 4.333 seconds.

When comparing this theoretical time to the actual time of the flipper arms to rotate, the efficiency can be determined as shown in equation 45.

$$Efficiency = \frac{time_{theoretical}}{time_{actual}}\, x\, 100 = \frac{4.333}{5}\, x\, 100 = 86.67\ percent \quad \{45\}$$

The loss in efficiency is due to friction on the bearings and sprockets as there is a load on the system.

The accuracy of the CAESAR robot turning 360° was tested. One hundred tests were performed on smooth concrete and the results are shown in figure 6-9.

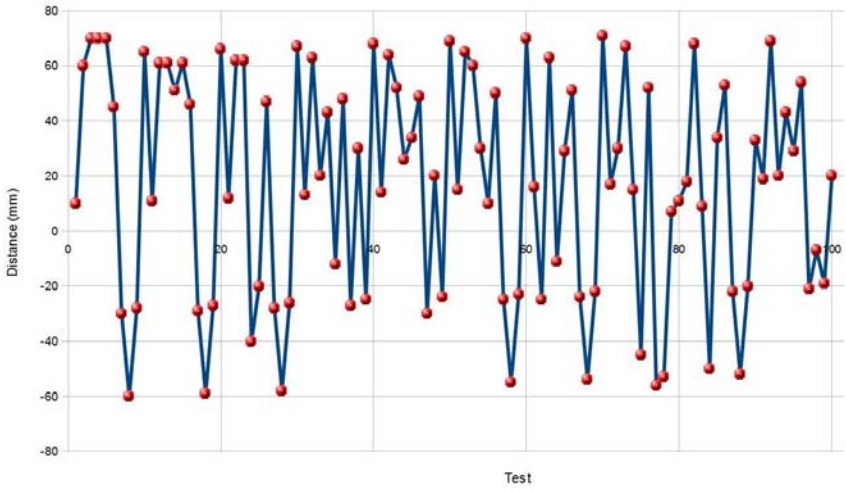


Figure 6-9: Accuracy of 360° turns

As observed from figure 6-9, the absolute accuracy was less than 70 mm. This would indicate that a 70 mm space is required around the CAESAR robot to allow for a successful turn in an concealed area. Chauvenet's criterion statistical analysis [68] was performed on the readings to determine the validity. All readings were acceptable.

Different weights were placed onto CAESAR's chassis and the velocity was observed. The results of the tests are shown in figure 6-10.

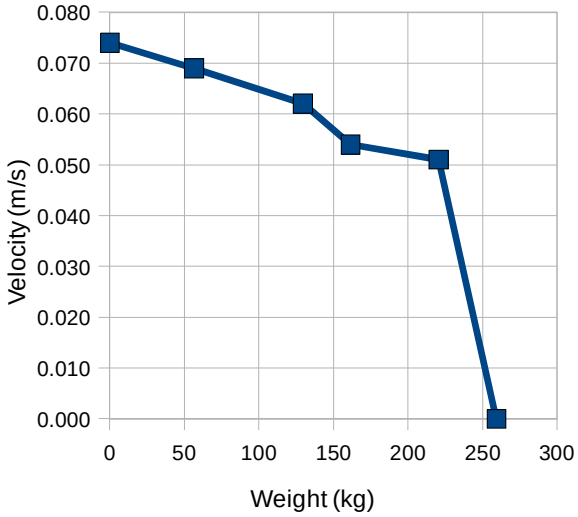


Figure 6-10: The velocity vs weight relationship

The initial velocity was measured for the weight of the robot chassis, which is 56.5 kg. As the weight increased, the velocity decreased. Traction was still possible up to 164 kg, after which the weight prevented the chain tensioners to hold the chain under tension, and therefore slippage occurred at the sprockets. The design and construction of previous USAR robots made it difficult to carry equipment that might be required into the disaster area [3], which is possible with CAESAR.

The flipper arms were also tested with different weights on the robot. Elevation is possible with the weight of the robot chassis, but anything more than this was not possible.

It is suggested that not more than 50 kg payload be added to CAESAR's chassis, as this could cause traction failure due to the force on the tensioners. This payload was obviously only needed to move over a terrain where elevation with the flipper arms was not needed due to the restriction of the flipper arms strength. The ability for an addition of a payload is useful as it allows the rescuers to send additional equipment into the disaster scenario. Additional equipment could either assist the rescuers or the victims when located.

Transformation with the flipper arms assisted the CAESAR robot maneuver over an obstacle that had a gradient of 30° or less for long distance climbing, while 45° slopes were climbed for shorter distances. This gradient for the respective distances was considered acceptable for the environment where such rescues would occur. A comparison of speed vs the angle of inclination is given in figure 6-11. This analysis is dependent on the terrain that CAESAR is on, as loose obstacles can cause slippage for short time periods.

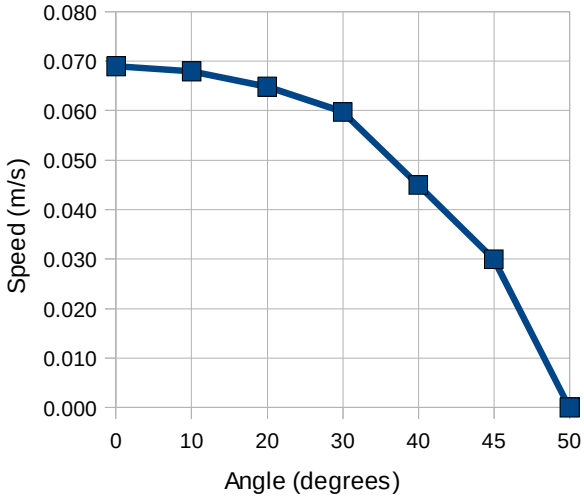


Figure 6-11: Angle of inclination vs the Speed of motion

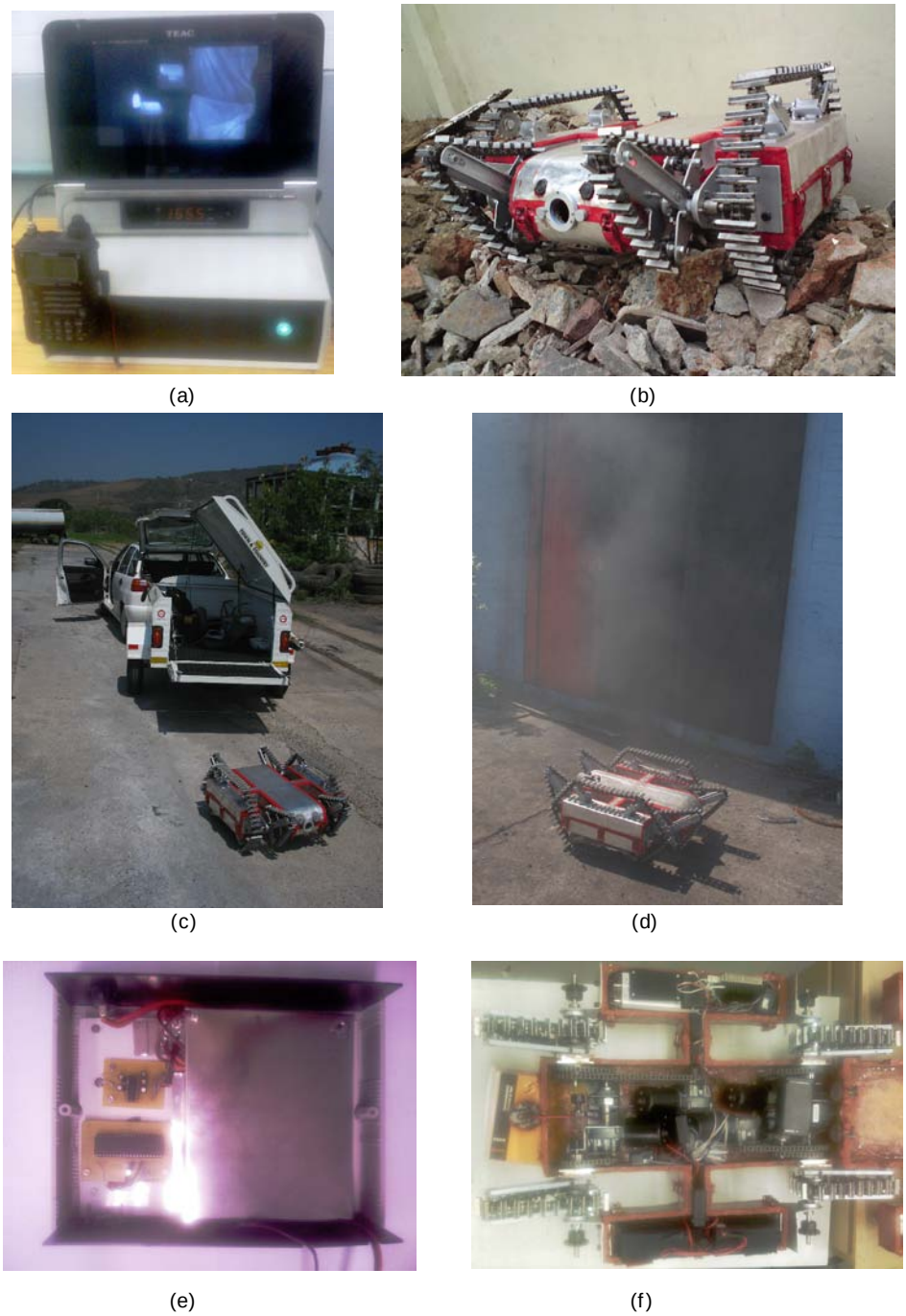
These flipper arms were also able to assist the CAESAR robot by pushing it over an obstacle by rotating the arms down under the chassis before contracting them to the homing position.

The overall dimensions of the CAESAR robot to determine the confined space it could enter, is:

- Height of composite chassis: 150 mm
- Length of robot excluding extended flipper arms: 730 mm
- Length including the extended arms: 1090 mm
- Width: 700 mm
- Height with arms at 90°: 395 mm
- Height including the side tracks: 364 mm
- Height from the ground to the top of the composite body: 250 mm

6.5 CAESAR Prototype

Photos of the first prototype of the CAESAR robot and the control station, with photos of the interior configurations are shown in figure 6-12.



(a) (b) (c) (d) (e) (f)

Figure 6-12: (a) Control station with radio and video reception (b) CAESAR being tested on rubble terrain (c) CAESAR being deployed from the Search and Rescue Division trailer (d) CAESAR entering the smoke-filled training facility (e) Interior configuration of the control unit (f) interior configuration of the CAESAR robot

6.6 Field and Scenario Testing

After the separate tests proved to be successful, a series of integrated system tests were performed. These consisted of creating similar scenarios to what could be encountered as well as a test at the Ethekwini Fire Department Training facilities.

Upon arrival at the site the immediate setup for the robot usage was initiated. The human to robot ratio for the loading and carrying of the CAESAR robot is 2:1, but the human to robot ratio to control the robots with the GUI is 1:n, where n is the number of robots that need to be controlled. The weight and size of the CAESAR robot is more than the pursued goal, however it is able to contain all the modules and components required. In total the setup time came to 2 minutes and 34 seconds, of which 52 seconds were utilized on waiting for the PC to boot up and the software to load.

Five Watts transmission power was sufficient for communication with the data and audio modes. It was possible at this power level to send data to the robot in the built-up environment. Interference was reduced with dedicated UHF frequencies and the designed Robotics Communication Protocol. Audio reports from the actors playing the victims were stated to be clear with the readability as good. The audio feedback from the victim was dependent on the video reception. Using the 1 W transmission power for the video feedback resulted in reduced video quality and signal strength within the built-up area when compared to the 5 W audio transmission from the control station.

The egg-beater antenna provided reliable communication between stations irrespective of the orientation of the robot. Transmitters with 5 W output allowed Line Of Sight (LOS) and Beyond Line of Sight (BLOS) communication. It was found that the 1 W transmission power for the video communication limited the distance for reception through the built-up area. The egg-beater antennas were mounted to the trailer with fishing rod suction cups. Higher elevation of the antennas will increase the distance of communication.

The traction system allowed for traction on rough and smooth surfaces. By means of the contracting arms it is possible to climb inclines and go through confined spaces where rescue workers are not able to enter. This is seen in figure 6-13.

The flipper arms assist the CAESAR robot to have 4-degrees of freedom (DOF). These DOF allows the CAESAR robot to propel itself over obstacles if required, while the tracks allowed movement in different directions. These directions allow for a gradual or sharp turn and assistance with easy maneuverability. With the utilization of lead-acid batteries, CAESAR was able to lift itself for a time period of about 10 minutes. Following this time period, CAESAR still had one hour of power to climb over obstacles and move across the terrain.

Figure 6-13: CAESAR robot climbing obstacles and going through confined spaces

It was perceived that the composite body construction allowed for protection of the internal components when objects fell on the CAESAR robot due to unsecured structures. Tests performed in the fire environment proved that the heat was reflected away from the chassis and therefore prevented internal temperatures from increasing unnecessarily. The chassis and tractions system did not soften or malfunction in the heated environment.

Video from the thermal camera displayed information and detail about the environment that the normal camera was not capable of revealing. Figure 6-14 shows a video comparison of the fire ignited. The thermal camera image reveals the rising of the heated gases.

Figure 6-14: Video comparison of ignited fire

With the increase of smoke it was not possible to view the lighted exit areas. Eventually the flame was not visible with the normal camera, but the thermal camera was still able to identify the flame, as seen in figure 6-15.

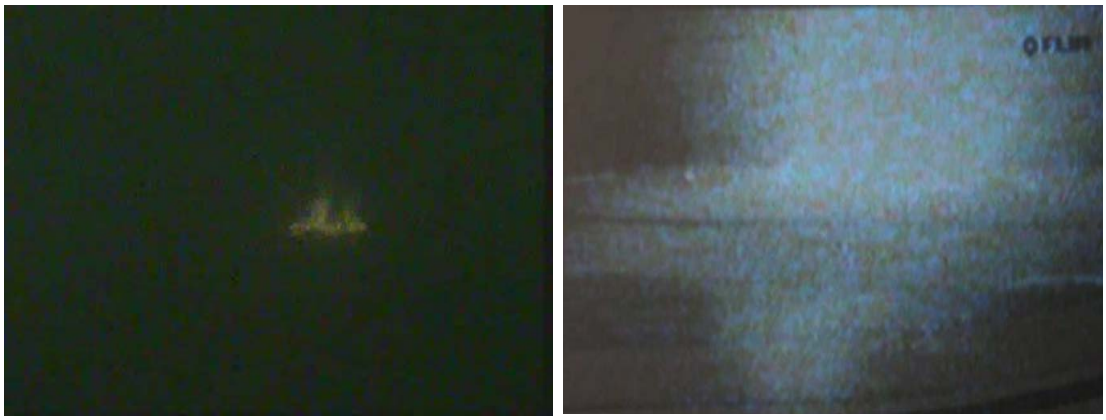


Figure 6-15: Video comparison of the smoke-filled room

As the CAESAR robot was proceeding in the high dense smoke environment, the normal camera's display was limited but the thermal camera was able to reveal objects in the surrounding area. Only the thermal camera indicated heated walls and areas that could be hazardous for the rescue personnel. These thermal images are seen in figure 6-16.

Figure 6-16: Thermal images of objects in the surrounding area (left) and the heated danger area (right)

The thermal camera also made it possible to locate victims as shown in figure 6-17.

Figure 6-17: The moving head of an injured victim was noticeable with the thermal camera

Smoke rises as it is generated and therefore the visibility is better towards the ground. As CO_2 is heavier than air, it descends. The CAESAR robot immediately

detected an increase of CO_2 which gradually increased as the smoke permeated the room. Figure 6-18 shows the gas concentration levels over the testing period in a room 5 x 10 x 3 m.

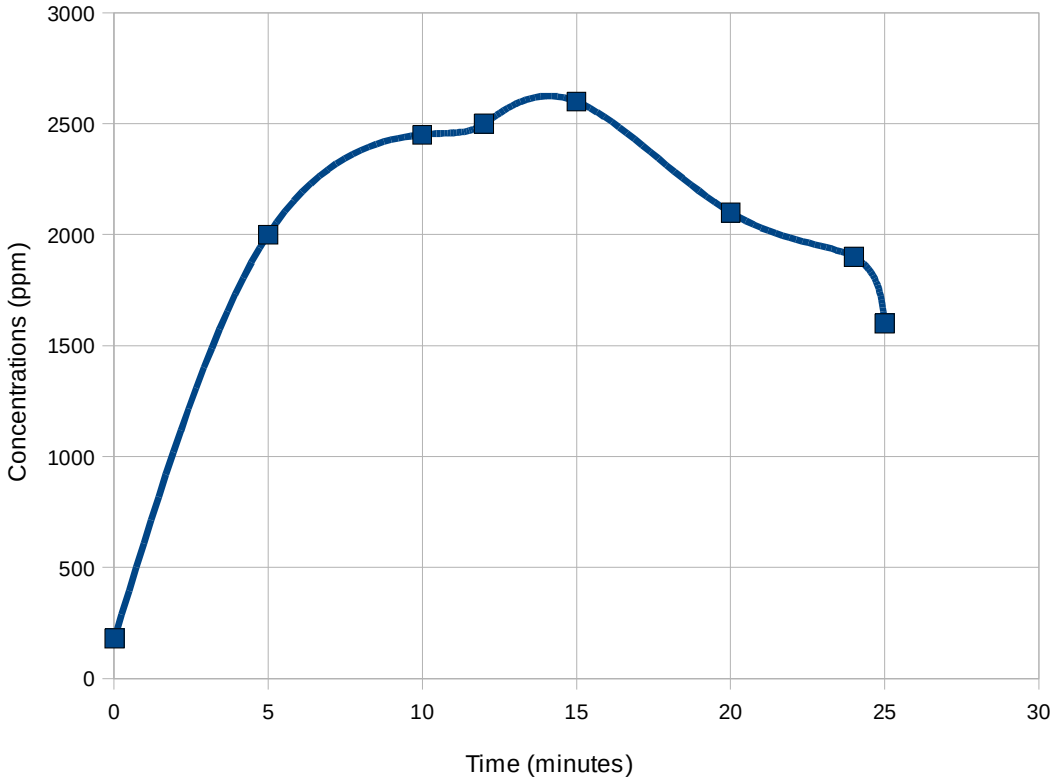


Figure 6-18: CO_2 concentration vs time

The CO_2 concentration increased drastically. After 10 minutes the concentration level increased more gradually as the room was filled with smoke and the fire was decreasing in size. From a time period of 15 minutes, the observation was that the CO_2 concentrations decreased as the fire died. After 24 minutes the concealed room was opened and the smoke and gases escaped.

Methane levels increased slightly as time passed, which is possible depending on the materials being burned by the fire. The AI implemented by the GUI indicated the unsafe and dangerous levels by changing the background color from green to either orange or red respectively. The danger probability was also indicated and was shown as being unsafe for humans as the CO_2 levels were over 2000 ppm.

6.7 Summary

The research, development and testing of CAESAR indicated that it has significant advantages compared to other USAR robots and their relevant problems previously discussed. These advantages include the contributions of reliable audio, thermal video and data communication, which permits communication that penetrates through building material irrespective of the robot orientation. Traction on smooth

surfaces is possible and the contractible arms allow for maneuverability over unstable terrain. The overall structure strength has proven to be strong enough to protect the sensitive electronic equipment from falling rubble and extreme temperatures. Data fusion of the gas concentrations allows the determining of the safety and danger levels for victims, rescuers and robots. Semi-autonomous control is possible with the on-board local AI, thus relieving the controller to determine the best orientation of the robot in the current situation. A comparison of CAESAR with other USAR robots such as those used at the World Trade Center [70] is shown in table 6-1. The vehicle properties that CAESAR has improved on are indicated in red. Improvements that CAESAR has that the other vehicles don't have are indicated in blue.

Table 6-1: CAESAR compared to other USAR robots

	TALON	**SOLEM**	**VGTV**	**ATR X-50 [19]**	**URBOT**	**PACKBOT**	**CAESAR**
Size (LxWxH)	864 x 572 x 279 mm	508 x 375 x 203 mm	317 x 165 x 65 mm	950 x 650 x 350 mm	838 mm long	687 – 864 x 152 x 435 mm	730 – 1090 x 700 x 364 – 395 mm
Weight (kg)	39	15	Not Available	84	27	18	56
Max Incline angle (°)	Not Available	Not Avaiable	15	15 - 20	Not Available	45 - 55	30 - 45
Speed (m/s)	1.8	0.46	0.08	0.5	0.76	3.7	0.07
Power Supply	4 NiMH batteries	4 NiMH batteries	Tether	On-board Battery	On-board Batteries	On-board batteries	On-board batteries
Communication	2 W	1 W	Tether	2.4 GHz TCP/IP	Wifi	2.4 GHz Wifi	UHF - 5 W
Vision	4 color cameras	Color CCD camera	Color CCD camera	CCD camera	1 zoom camera and 2 cameras on top and bottom	Low light camera / Color CCD camera	Thermal camera transmitted wireless - 1W
Lighting	User selected	User selected	2 halogen headlights	Non specified	2 halogen headlights	2 halogen headlights	None – thermal imaging
Other Sensors	Encoders, 3 axis compass, arm position	Encoder, 3 axis compass, arm position	Encoders	Encoders, compass, infrared rangefinder	Altitude, compass, temperature	Optional – Indigo Alpha FLIR	FLIR Thermal camera, tilt (pitch and roll), gas sensors, arms position encoders, temperature, sonar, audio comms

The effects of the communication improvements compared to previous USAR robots are discussed which involved the use of UHF frequencies, the RCP and egg-beater antennas. These improvements allow for LOS and BLOS communication which was not previously possible. The audio and data transmission were possible within the built-up environment and rubble and fire between the stations. Video communication

was not as successful due to the limited power, but did prove the point.

The traction system and transformability of the CAESAR robot revealed that it was possible to maneuver over obstacles and to go through confined spaces that would not be possible for rescue personnel. This allows the CAESAR robot to enter areas and determine victim location without risk to the lives of firefighters. The combination of the ultrasonic sensors and the control system allows for semi-autonomous and manual control.

Feedback from the gas sensors immediately indicated the rise of danger for human survival, which assists the rescuers to decide about further rescue attempts. This immediate notification of gas concentrations and danger levels were not available with previous USAR robots [3]. The field testing also made it possible to determine the strength and heat protection provided by the chassis. Thermal images give vital information about the environmental dangers, victim location and object location for navigation purposes, which was not previously incorporated within USAR robots [3].

CHAPTER 7 – CONCLUSION

Communication improvements were developed, which resulted in reliability with the use of UHF dedicated frequencies and 5 W transmission power. These improvements permitted LOS and BLOS communication. The developed RCP granted data reliability with a single frequency and multiple stations required to transmit and receive. Corruption of data due to interference was also prevented. Egg-beater antennas made it possible for omni-directional communication irrespective of the position of the robot within a building. Two way audio communication was established, allowing rescue workers to supply victims with critical information and the ability to receive audible information from the surrounding area. Thermal video transmission from the CAESAR robot allowed visibility of the surrounding area. The observing of images of dangerous areas and victims, not possible with a normal camera, was displayed with the thermal camera. The above advantages were required in USAR robots at the World Trade Center disaster [3].

CAESAR's tracks prevented slippage on smooth surfaces. Although slippage was experienced on rougher surfaces, CAESAR was able to proceed forward as the flipper arms assisted with climbing over obstacles. Transformation of the flipper arms did not only allow for gradual lift of the body, but for lifting of either the front or back for better maneuverability. The integration of the motor and drive systems made it possible for the flipper arms to rotate while the tracks were turning. The Kevlar and phenolic resin composite body ensured a strong chassis thus protecting the delicate components and modules inside, which previous USAR robots did not [2]. Another advantage compared to previous USAR robots used at the World Trade Center disaster [3], is that CAESAR can carry a payload into the disaster environment, which could assist victims. Heat was also reflected from the outside body by the aluminized Kevlar, protecting the electronic components from the increasing temperature resulting from the outside environment.

The control unit consisted of a GUI which made it possible to control different robots with a single interface. Though only one robot could be controlled at a given time, it is uncomplicated for one person to switch between robot controls. This would relieve other rescue workers to assist with the disaster scenario, which was not possible at the World Trade Center disaster [7]. The GUI supplied the controls to move the robot, visuals to view the receiving video and information determined from the AI models. Equations had to be developed for estimations of the gas concentrations depending on the voltages received or the resistance ratios as these were not available from the developers. Gas concentration levels for unsafe and dangerous conditions have been used as guidelines to determine the safety of victims and rescue workers. By using these concentration levels, it was possible to develop fuzzy logic models and therefore determine the level of danger in the surrounding environments. USAR robots used at the World Trade Center disaster were not able to immediately notify rescuers of gas concentrations and their dangers [3].

With the use of the encoders it was possible for CAESAR to determine the

orientation of the flipper arms. The ultrasonic sensors made it possible to detect obstacles. With this information, the internal AI of CAESAR could determine the orientation of the flipper arms to allow for the best way to maneuver over the obstacles. Angle of inclination of the robot is possible with the installed orientation sensor, which autonomously transforms the flipper arms orientation for the best maneuverability and feedback to the control station. This orientation feedback allows the controller to know of the inclination angle of the robot which is difficult to determine from the video feedback and which was not possible with previous USAR robots [12].

Field testing verified many of the expected results. The setup time was 153 seconds, which consisted of connecting the different power points from the car to the modules in the trailer. This time period is critical as it could be used to locate victims within a burning building. The PC boot-up also consumed a large portion of the time. This time period could be decreased by using an alternative operating system. This time period could be minimized if a mini-bus is available for the transportation method, as the setup could be completed in advance. All the modules and control systems would be pre-installed, allowing the PCs to be booted while approaching the disaster scene. The CAESAR robot is switched on by a remote control, and is ready for instructions within a second. With the mini-bus it would be possible to have a ramp that the robots could deploy from and therefore eliminate the need to unload it.

The maneuverability of CAESAR was tested in the built scenario structures and at the fire department's training facilities. Gas concentration and danger levels were shown with the GUI. Audio, data and video communication were tested in the built-up environment. With the use of Lithium-ion batteries the overall length of the use of the robot will be increased dramatically.

The minimum requirements required from the fire department were thermal imaging, gas sensing, maneuverability, audio feed, telemetry, thermal shielding and maintaining the weight and size as low as possible. All of these requirements were incorporated within CAESAR.

The inclusion of the above improvements has made it possible to develop a robot that will assist the rescue workers and firefighters in rescue attempts. Victims can successfully be located and dangerous environments can be observed. Confined spaces can be entered and searched without risking the lives of rescue worker unnecessary, when there are no living victims in these dangerous environments.

7.1 Achieved Objectives, Specifications and Requirements

The objectives, specifications and requirements achieved with the research, design, development and testing of CAESAR are summarized below.

- The traction system allowed CAESAR to maneuver over harsh and unstable environments. CAESAR is able to withstand temperatures higher than 200 degrees Celsius, common of flash fires, for short time periods. A thorough research of the material for manufacturing was conducted as this affected the concealing and protection of all the components, making the encloser splash proof and impact resistant. The traction system allowed for traction over the rough terrain and on smooth surfaces that might have been covered with oil or water. CAESAR is able to climb an incline of 45° for short distances, and 30° for longer distances.
- Research was conducted in the construction of an autonomous-transformable robot. CAESAR's construction is compact in size, this being dependable on the module and component dimensions. The robot dimensions are within the specifications of 1 m x 1 m x500 mm. The flipper arms are contractible to decrease the height should the arms restrict in the robot's movement. Should CAESAR tip over, it is able to continue with operation as the flipper arms transform its orientation for the best performance. This compact size enables the entering of confined spaces. CAESAR is able to enter a gap of 1 m wide and 500 mm high. The weight of CAESAR is 56 kg, but use of lighter materials would result in a weaker construction. The lower strength property would prevent CAESAR from carrying the weight of the equipment that might be needed to save trapped victims. A weight of 50 kg of equipment or other smaller robots can be carried by CAESAR if required, which excludes the weight of the internal components. With a speed of 0.069 m/s on a flat concrete surface, the time taken to move over a certain distance is dependent on the terrain of operation and the restriction of movement of the underlying rubble.
- Licensed and emergency frequencies were used. The Robotic Communication Protocol (RCP) allows for reliable data communication between control station and robots with the use of a single frequency. A packet size of 33 % compared to the IEEE 805.11 protocol is possible with the RCP. Five Watts power transmitted from each station and the use of UHF enables successful communication through the building material and rubble. Omni-directional communication is possible with the designed antennas, as the robots can have different orientations in the disaster scenario. Audio communication with victims is possible and video feedback of the environment is achievable with the thermal camera.
- Research was conducted on the telemetry sensory system that is required in the search for survivors. The thermal camera supplies victim localization in the dense smoke environment and dangerous areas that are caused by high temperatures. Gas sensors allow the control station to indicate the concentrations and determine the safety and danger factors for humans, rescuers and the robots. Tests performed indicated that the operation of CAESAR is possible for a time period of at least an hour. A GUI interface allows for multi-robot control station.
- Necessary investigation for autonomous transformation was conducted. This required a robot with omni-directional movement. A tracked system was utilized.
- The required robot needed to operate with a low setup time to prevent any

delays. The operator is able to control the robot with a minimum amount of training. The human to robot ratio needed to control CAESAR is 1:1.

7.2 Future Work

CAESAR has incorporated various aspects which resulted in it being effectual in rescue attempts. There is however the prospect of further possible improvements. These were not incorporated into CAESAR at this stage as the components and modules were either not available or because of other limitations.

Lighter weight and smaller size of the CAESAR robot would be beneficial as technology develops and improves. This will result in components being smaller, therefore decreasing the size and weight.

Motors with high speed and torque will improve the time for the robot to reach victims. The increase of torque and speed of motors generally increase the size and weight of the motors.

Heat resistance speakers and microphones will allow the robots to enter higher temperature environments without the danger of these components melting. Further, heat resistance materials, that are light in weight will also increase the environment temperatures that CAESAR could enter.

International dedicated UHF video frequencies for emergency purposes will allow search and rescue robots to be used in any location around the world. This will also allow 5 W power transmission to be used which will penetrate most building materials. Ways to elevate the egg-beater antennas will improve the distance of communication.

REFERENCES

[1] Mechatronics, http://www.yaskawa.com/images/News/Mechatronics.jpg, 23 July 2007

[2] Bread Crumbs and Robots May Save Your Life, Karen Wiens, SciTini, October 2006

[3] Human-Robot Interaction during the Robot-Assisted Urban Search and Rescue Response at the World Trade Center, Jennifer Casper, Dr Robin Murphy, IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 33, No. 3, June 2003

[4] Search-and-Rescue Robots Tested at New York Disaster Site, Bijal P. Trivedi, September 2001

[5] 9/11 (documentary), Directors: J. Naudet, G. Naudet, J. Hanlon, A Goldfish Pictures, Inc. Silverstar Production, LLC Production, Paramount Pictures, 2002

[6] Better robots could help save disaster victims, Kurt Kleiner, January 2006

[7] Robots in Urban Search and Rescue Operations, David Greer, Philip McKerrow and Jo Abrantes, © ARAA, 2002 Australasian Conference on Robotics and Automation, November 2002

[8] A Standard Test Course for Urban Search and Rescue Robots, A. Jacoff, E. Messina, J. Evans, MIST Special Publication, 2001

[9] Analysis of How Mobile Robots Fail in the Field, Jennifer Carlson, Dr Robin Murphy, Dr Kimon Valavanis, Dr Dewey Rundus, March 2004

[10] Evolution and Field Performance of a Rescue Robot, Mark Micire, International Journal of Field Robotics, Wiley, 2008

[11] Standards Development for Wireless Communications for Urban Search and Rescue Robots, Kate A. Remley, Gelen Koepke, Elena Messina, Adam Jacoff, February 2007

[12] Evaluation of Human-Robot Interaction Awareness in Search and Rescue, Jean Scholtz, Jeff Young, Jill L. Drury, Holly A. Yanco, Mitre.org, 2004

[13] Marsupial and shape-shifting robots for urban search and rescue, R. Murphy, IEEE Intelligent Systems, 2000

[14] Development of a Shape-shifting mobile robot for urban search and rescue, YE Changlong, MA Shugen, LI Bin, Chinese Journal of Mechanical Engineering, 2008

[15] Protocols for robot communications: Transport and Content Layers, Scott Y. Harmon, Douglas W. Gage, 1980 International Conference on Cybernetics and Society, Cambridge MA, 8 October 1980

[16] Long Term Study of a Portable Field Robot in Urban Terrain, C. Lundberg, H.I. Christensen, R. Reinhold, International Journal of Field Robotics, Wiley, 2007

[17] Rescue Robots for Mudslides: A descriptive study of the 2005 La Conchita Mudslide Response, R. Murphy, S. Stover, International Journal of Field Robotics, Wiley, 2008

[18] ATR X-50: Development Of An All Terrain Mobile Robot, Dip N. Ray, A. Maity, S. Majumder, K. K. Mistry, 23rd ISPE International Conference on CAD/CAM Robotics and Factories of the future, 2007

[19] Performance Standards for Urban Search and Rescue Robots, Elena Messina, Adam Jacoff, Intelligent Systems Division, National Institution of Standards and Technology, 2006

[20] Human-Robot Teaming for Search and Rescue, Illah R. Nourbakhsh, Katia Sycara, Mary Koes, Mark Yong, Michael Lewis, Steve Burion, IEEE Pervasive Computing, IEEE CS and ComSoc, January – March 2005

[21] Towards Heterogeneous robot teams for disaster mitigation: Results and performance metrics from Robocup rescue, S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, V. Amos Ziparo, International Journal of Field Robotics, Wiley, 2007

[22] Durban Metro Fire Department Training division Consultant, Alex Glouster, February 2007

[23] Galileo Mobility, 15 March 2007,

http://www.galileo-mobility.com/imgs/Uploads/full%20mov%20low%20eng.wmv

[24] Radiometrix TR2M Narrow Band FM Multi-channel UHF Transceiver data sheet, February 2004

[25] A Technical Tutorial on the IEEE 802.11 Standard, P. Brenner, BreezeCom Wireless Communication, 1997

[26] Robotic Communication Protocol (RCP), R. Stopforth, G. Bright, R. Harley, 2nd Symposium in Mechatronics and Robotics, Bloemfortein, South Africa, November 2008

[27] Yaesu VX-7R operating manual, Vertex Standards Co., Ltd., Japan, 2002

[28] Yaesu VX-3E operating manual, Vertex Standards Co., Ltd., Japan, 2007

[29] Yaesu Modifications, 4 March 2008,

http://www.mods.dk/index.php?ModelId=116&RadioRec=yaesu

[30] ProQuip Sound, Excellence in Audio, Line & Mic Pre-Amplifiers datasheet, January 1998

[31] PathFindIR Brochure, FLIR, 2006

[32] RF Jackël, South Africa, June 2008

[33] City of Orlando Fire Department, Arson/Bomb Squad, Lt. Dale Reynolds, Bomb Squad Commander, July 2008

[34] Required Improvements and Possible Solutions for Urban Search And Rescue (USAR) Robots, R. Stopforth, G. Bright, 2008 International Conference on Automation, Robotics and Control Systems (ARCS-08), Orlando, USA, July 2008

[35] South African Radio League Radio Amateur Examination Manual, Andrew Roos (ZS1AN), 2005

[36] Communication Electronics, Principles and Applications, Third Edition, L. Frenzel, McGraw-Hill Companies, Inc., 2001

[37] Urban Search And Rescue Robot, R. Stopforth, H. Ramsurup, I. Mmutloane, N. Mbhatha, S.C. Cele, University of KwaZulu-Natal, 2008

[38] Experienced Outcomes from the Improvements made to the USAR robot, R. Stopforth, G. Bright, R Harley, 2009 International Conference on Automation, Robotics and Control Systems (ARCS-09), Orlando, USA, July 2009

[39] Composite Materials, Mechanical Behavior and Structural Analysis, Mechanical Engineering Series, Jean-Marie Berthelot, Springer-Verlag, New York, USA, 1999

[40] DuPont Kevlar 49 Aramid Fiber, 16 October 2008, http://www.matweb.com/search/DataSheet.aspx?bassnum=PDUKEV29&ckck=1

[41] Materials / Subtances Thermal Conductivity, 16 October 2008, http://www.engineeringtoolbox.com/thermal-conductivity-d_429.html

[42] Durban Specialty Chemicals J2018L, J2027L and J2042L Product Data Bullitin, November 2003

[43] Wacker Silicones, Elastrosil LR 3001/55 A & B datasheet, version 3.0, February 2006

[44] Oakley US Standard Issue Testimonials, 29 October 2008, https://secure.usstandardissue.com/Page.cfm?id=5

[45] Oakley Community News, 29 October 2008, http://oakley.com/community/news/bloody_eye

[46] HAWK I.R. C-Range – The Worlds only Arc-resistant infrared sightglass, 30 October 2008, http://www.hawksightglasses.com/infrared-sightglasses-c-range.asp

[47] HAWK I.R. Testing – Temperature, 30 October 2008, http://www.hawksightglasses.com/infrared-testing-temperature.asp

[48] Performance of Phosphate Lithium-ion Batteries in Motive Applications, J. Nguyen, Valence Technology, Austin, Texas, USA

[49] Powerstream Lithium Iron Phosphate Battery packs, 25 September 2008, http://www.powertream.com/LLLF-12v.htm

[50] Velleman-kit K3502 Parking Radar, Illustrated Assembly Manual, June 2005

[51] Location Systems for Ubiquitous Computing, J. Hightower, G. Borriello, Location Aware Computing, IEEE, August 2001

[52] Communication and Artificial Intelligence systems used for the CAESAR robot, R. Stopforth, G. Bright, R. Harley, Mobile Robot Navigation, InTeh, Vukovar, Croatia, March 2010

[53] Fuzzy Logic Analysis of environmental threat level based on selected gas concentration, R. Stopforth, G. Bright, 3rd Symposium in Robotics and Mechatronics (RobMech) 2009, CSIR, Pretoria, South Africa, November 2009

[54] Aerotech Environmental Laboratories, http://www.aeroenvirolabs.com/Resources/niosh.asp, 24 February 2009

[55] National Institute for Occupational Safety and Health (NIOSH), Publication number: 2005-100: NIOSH Respirator Selection Logic 2004

[56] Threshold Limit Values (TLV) and Immediately Dangerous to Life and Health (IDLH) values Reference Information, www.mathesontrigas.com, 25 February 2009

[57] Hazardous Materials Chemistry for Emergency Responders, second edition, Robert Burke, Fire Marshal, University of Maryland, Baltimore, USA, Lewis Publishers, 25 November 2002

[58] Material Safety Data Sheet, Prepared by U.S. OSHA, CMA, ANSI and Canadian WHMIS Standards, Document number: 50009, 6 May 2005

[59] Development Methane Safety, Alberta.ca > Agriculture and Rural Development, Prepared by Atta Atia, PhD, Manure Management Specialist, 24 February 2009

[60] Airgas, Material Safety Datasheet, Prepared by U.S. OSHA, CMA, ANSI and Canadian WHMIS Standards, Document number: 001033, 3 January 2001

[61] Stanford TGO Data Tables, http://stanford.edu/dept/EHS/prodresearchlab/lab/tgo/tgodata.html, 25 February 2009

[62] Artificial Intelligence, A Modern Approach, second edition, S. Russell and P. Norvig, Prentice Hall, 2003

[63] Figaro TGS 4161 – for the detection of Carbon Dioxide, Product Information sheet, December 2005

[64] Figaro TGS 2442 – for the detection of Carbon Monoxide, Product Information sheet, July 2007

[65] Figaro TGS 825 – Special sensor for Hydrogen Sulphide, Product Information sheet, November 2004

[66] Figaro TGS 2611 – for the detection of Methane, Product Information sheet, February 2005

[67] Figaro GS Yuasa Oxygen Sensor SK-25, Product Information sheet, May 2007

[68] A Manual of Spherical and Practical Astronomy, W. Chauvenet, 2nd Edition, Philadelphia, 1864

[69] Search And Rescue Robot - First Encounter Assist and Rescue (FEAR), R. Stopforth, W. Elson, R. Badat, S. Mkize, K. Mavundla, University of KwaZulu-Natal, 2009

[70] Survey on Urban Search and Rescue Robotics, B. Shah, H. Choset, Carnegie Mellon University, Pittsburgh, PA, USA, 2003

Appendix A – Tables of Test Results

Table A-1: Transmitted Power and Drive Resistance Relationship

Resistance (Ohms)	Power (W)
100	14
200	12
300	10.5
400	9
500	7
600	5.5
650	4.7
700	3.6

Table A-2: Voltage output vs distances for the configuration of a distance detection of 400 mm

Distance (mm)	Output Sensor Voltage only (V)
0	0
100	11
200	11.34
300	11.44
400	11.71
500	0.1

Table A-3: Output voltages measured before and after the interconnection circuitry vs the distance of detection

Distance (mm)	Output Sensor Voltage only (V)	Output of Circuit – 7 Diodes (V)
0	0	0
100	11	4.75
200	11.34	4.83
300	11.44	4.95
400	11.71	4.9
500	0.1	0.02

Table A-4: Gas concentration levels indicated by the gas sensors in a constant gas environment and the statical analysis

Time (s)	Temperature (C)	(Temperature – mean)^2 /standard deviation	Battery (V)	(Battery – mean)^2 / standard deviation
0	20.4	0.184	11.17	84.805
1	20.5	0.983	11.16	85.021
2	20.5	0.983	11.16	85.021
3	20.25	1.014	11.16	85.021
4	20.25	1.014	11.16	85.021
5	20.25	1.014	11.16	85.021
6	20.25	1.014	11.14	85.237
7	20.25	1.014	11.14	85.237
8	20.5	0.983	11.14	85.237
9	20.25	1.014	11.14	85.237
10	20.5	0.983	11.14	85.237
11	20.5	0.983	11.14	85.237
12	20.5	0.983	11.14	85.237
mean	**20.38**		**11.15**	
standard deviation		**0.13**		**9.6**

Time (s)	Methane (ppm)	(Methane – Mean)^2 /standard deviation	CO2 (ppm)	(CO2 – Mean)^2 /standard deviation
0	28	0.000	167	0.001
1	28	0.000	167	0.001
2	28	0.000	167	0.001
3	28	0.000	167	0.001
4	28	0.000	167	0.001
5	28	0.000	167	0.001
6	28	0.000	167	0.001
7	28	0.000	167	0.001
8	28	0.000	167	0.001
9	28	0.000	167	0.001
10	28	0.000	167	0.001
11	28	0.000	166	0.006
12	28	0.000	166	0.006
mean	**28**		**166.85**	
standard deviation		**7.93**		**152.45**

Time (s)	H2S (ppm)	(H2S – Mean)^2 /Standard deviation	CO (ppm)	(CO – Mean)^2 /Standard Deviation
0	40	0.018	30	0.205
1	40	0.018	31	0.126
2	40	0.018	33	0.030
3	41	0.030	33	0.030
4	41	0.030	33	0.030
5	41	0.030	33	0.030
6	41	0.030	33	0.030
7	41	0.030	33	0.030
8	40	0.018	33	0.030
9	40	0.018	33	0.030
10	40	0.018	33	0.030
11	40	0.018	33	0.030
12	40	0.018	33	0.030
mean	**40.38**		**32.62**	
standard deviation		**20.83**		**12.77**

Table A-5: Accuracy of 360° turns and statistical analysis of data

Test	mm	Test –mean	(Test –mean)^2	di/sigma
1	10	-7.01	49.14	0.17
2	60	42.99	1848.14	1.05
3	70	52.99	2807.94	1.3
4	70	52.99	2807.94	1.3
5	70	52.99	2807.94	1.3
6	45	27.99	783.44	0.69
7	-30	-47.01	2209.94	1.15
8	-60	-77.01	5930.54	1.89
9	-28	-45.01	2025.9	1.1
10	65	47.99	2303.04	1.18
11	11	-6.01	36.12	0.15
12	61	43.99	1935.12	1.08
13	61	43.99	1935.12	1.08
14	51	33.99	1155.32	0.83
15	61	43.99	1935.12	1.08
16	46	28.99	840.42	0.71
17	-29	-46.01	2116.92	1.13
18	-59	-76.01	5777.52	1.86
19	-27	-44.01	1936.88	1.08
20	66	48.99	2400.02	1.2
21	12	-5.01	25.1	0.12
22	62	44.99	2024.1	1.1
23	62	44.99	2024.1	1.1
24	-40	-57.01	3250.14	1.4
25	-20	-37.01	1369.74	0.91
26	47	29.99	899.4	0.74
27	-28	-45.01	2025.9	1.1
28	-58	-75.01	5626.5	1.84
29	-26	-43.01	1849.86	1.05
30	67	49.99	2499	1.23
31	13	-4.01	16.08	0.1
32	63	45.99	2115.08	1.13
33	20	2.99	8.94	0.07
34	43	25.99	675.48	0.64
35	-12	-29.01	841.58	0.71
36	48	30.99	960.38	0.76
37	-27	-44.01	1936.88	1.08
38	30	12.99	168.74	0.32
39	-25	-42.01	1764.84	1.03
40	68	50.99	2599.98	1.25
41	14	-3.01	9.06	0.07
42	64	46.99	2208.06	1.15
43	52	34.99	1224.3	0.86
44	26	8.99	80.82	0.22
45	34	16.99	288.66	0.42
46	49	31.99	1023.36	0.78
47	-30	-47.01	2209.94	1.15
48	20	2.99	8.94	0.07
49	-24	-41.01	1681.82	1.01
50	69	51.99	2702.96	1.27

Test	mm	Test –mean	(Test –mean)^2	di/sigma
51	15	-2.01	4.04	0.05
52	65	47.99	2303.04	1.18
53	60	42.99	1848.14	1.05
54	30	12.99	168.74	0.32
55	10	-7.01	49.14	0.17
56	50	32.99	1088.34	0.81
57	-25	-42.01	1764.84	1.03
58	-55	-72.01	5185.44	1.77
59	-23	-40.01	1600.8	0.98
60	70	52.99	2807.94	1.3
61	16	-1.01	1.02	0.02
62	-25	-42.01	1764.84	1.03
63	63	45.99	2115.08	1.13
64	-11	-28.01	784.56	0.69
65	29	11.99	143.76	0.29
66	51	33.99	1155.32	0.83
67	-24	-41.01	1681.82	1.01
68	-54	-71.01	5042.42	1.74
69	-22	-39.01	1521.78	0.96
70	71	53.99	2914.92	1.32
71	17	-0.01	0	0
72	30	12.99	168.74	0.32
73	67	49.99	2499	1.23
74	15	-2.01	4.04	0.05
75	-45	-62.01	3845.24	1.52
76	52	34.99	1224.3	0.86
77	-56	-73.01	5330.46	1.79
78	-53	-70.01	4901.4	1.72
79	7	-10.01	100.2	0.25
80	11	-6.01	36.12	0.15
81	18	0.99	0.98	0.02
82	68	50.99	2599.98	1.25
83	9	-8.01	64.16	0.2
84	-50	-67.01	4490.34	1.64
85	34	16.99	288.66	0.42
86	53	35.99	1295.28	0.88
87	-22	-39.01	1521.78	0.96
88	-52	-69.01	4762.38	1.69
89	-20	-37.01	1369.74	0.91
90	33	15.99	255.68	0.39
91	19	1.99	3.96	0.05
92	69	51.99	2702.96	1.27
93	20	2.99	8.94	0.07
94	43	25.99	675.48	0.64
95	29	11.99	143.76	0.29
96	54	36.99	1368.26	0.91
97	-21	-38.01	1444.76	0.93
98	-7	-24.01	576.48	0.59
99	-19	-36.01	1296.72	0.88
100	20	2.99	8.94	0.07
mean	17.01			
sigma			40.79	

Table A-6: The velocity vs weight relationship

Weight (kg)	Velocity (m/s)
0	0.074
56.5	0.069
129.5	0.062
161.5	0.054
220.5	0.051
259.2	0.000

Table A-7: Angle of Inclination vs the Speed of motion

Angle	Speed
0	0.069
10	0.068
20	0.065
30	0.060
40	0.045
45	0.030
50	0.000

Table A-8: CO_2 concentration vs time

Time (minutes)	CO2 levels (ppm)
0.02	180
5	2000
10	2450
12	2500
15	2600
20	2100
24	1900
25	1600

Appendix B – Mechanical Drawings

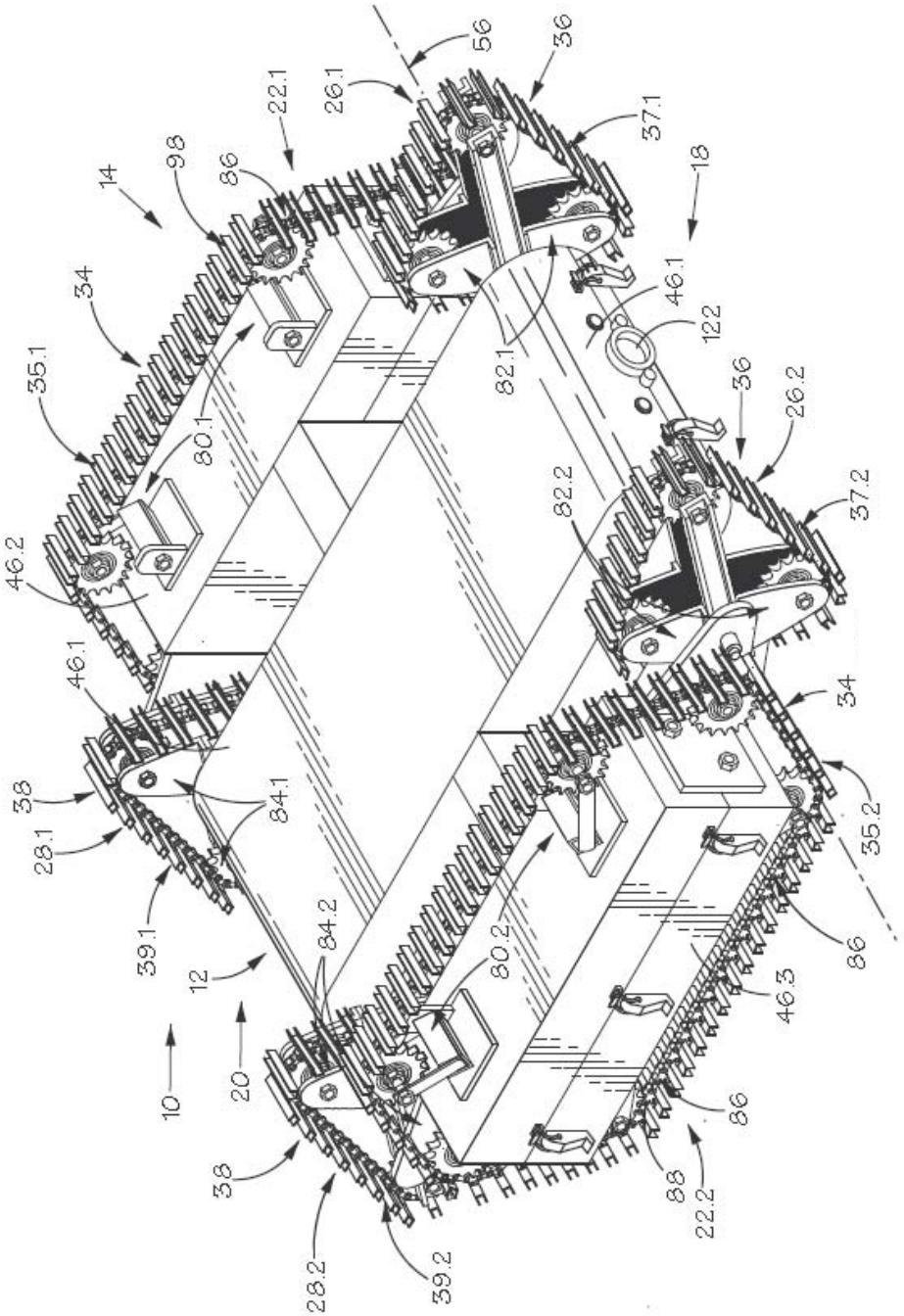


Figure B-1: Assembled Drawing

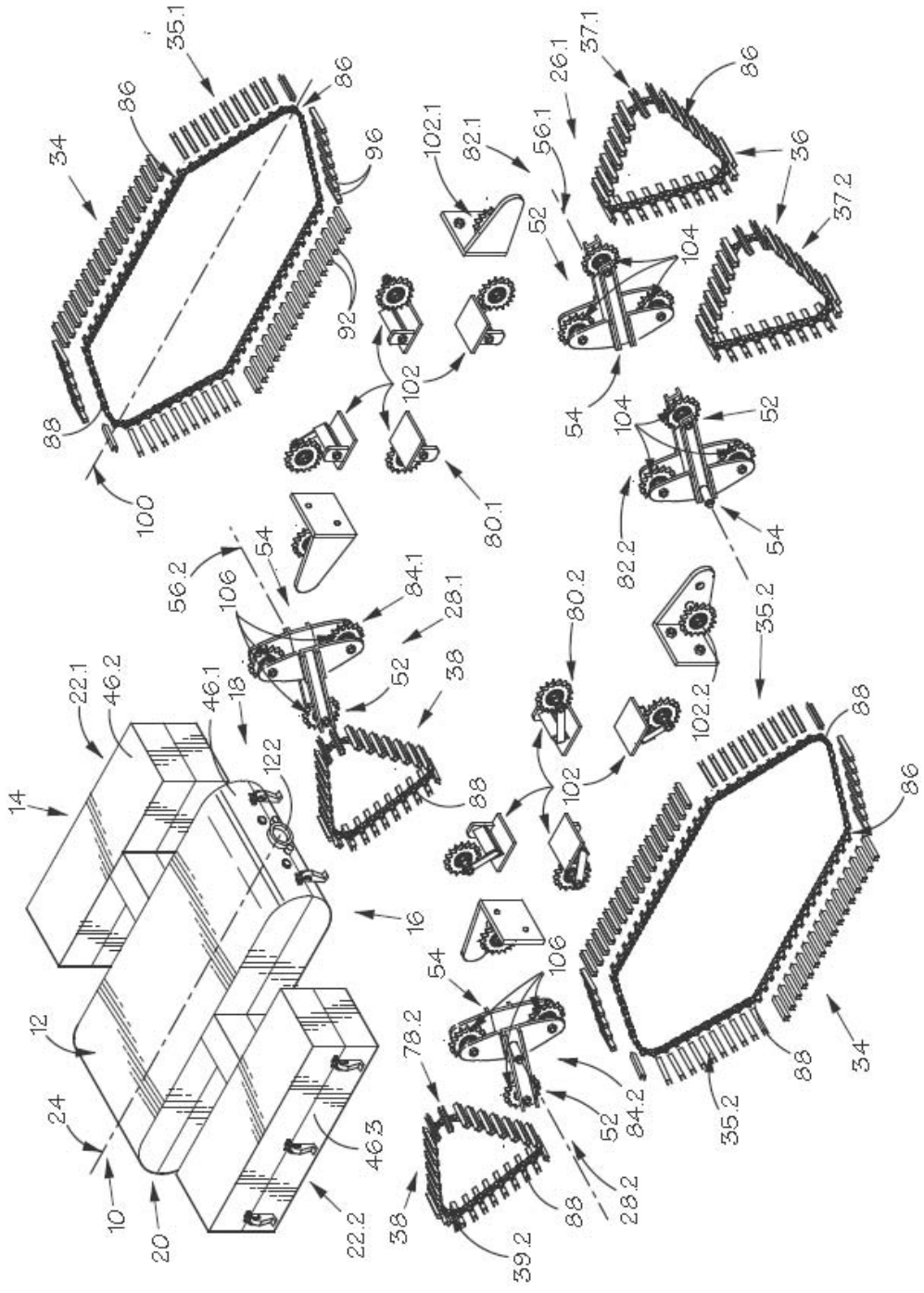


Figure B-2: Exploded View of CAESAR's Components

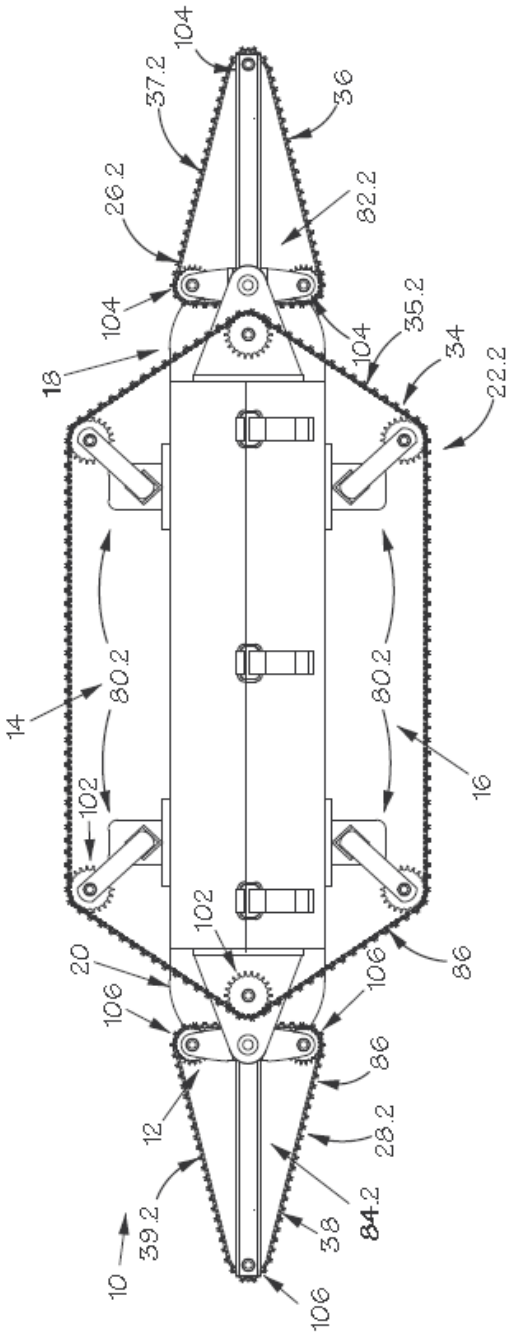


Figure B-3: Side View

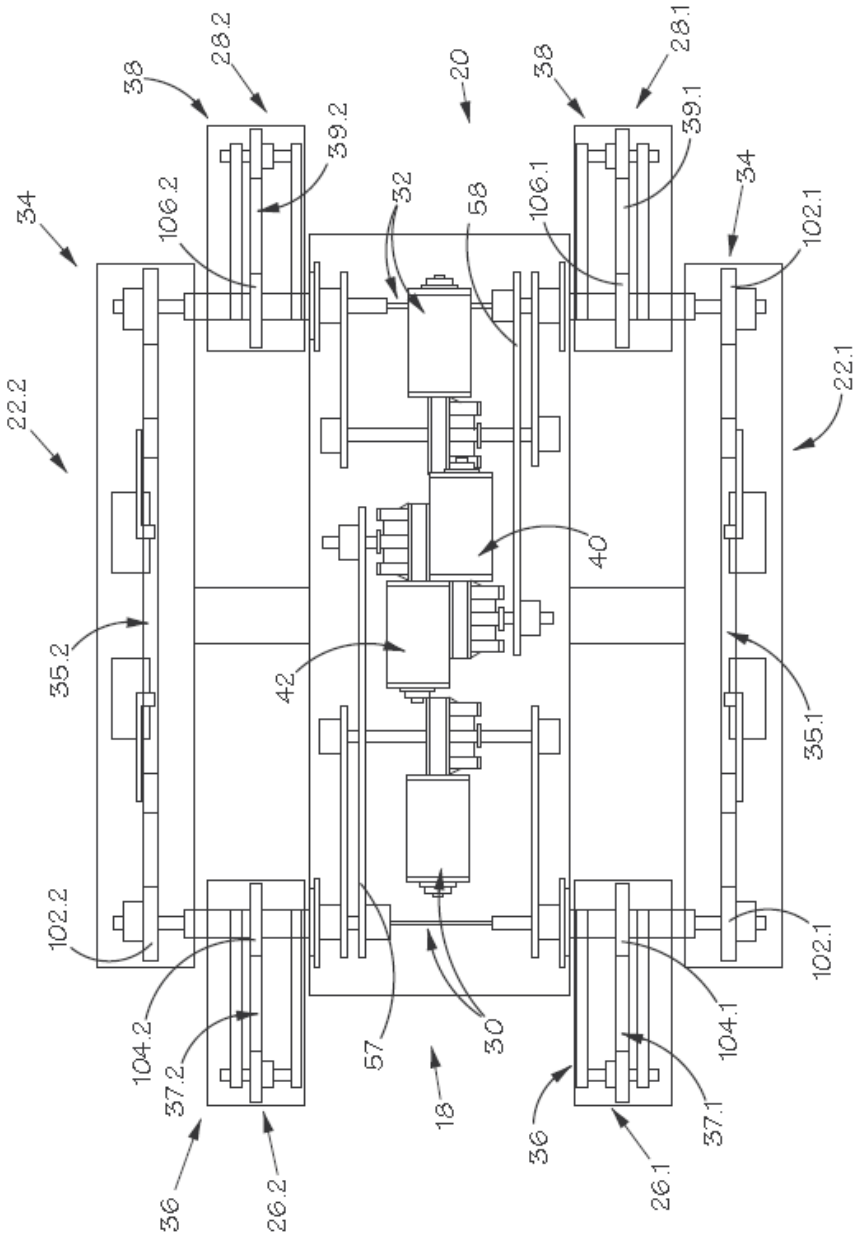


Figure B-4: Mechanical and Motor Interconnection

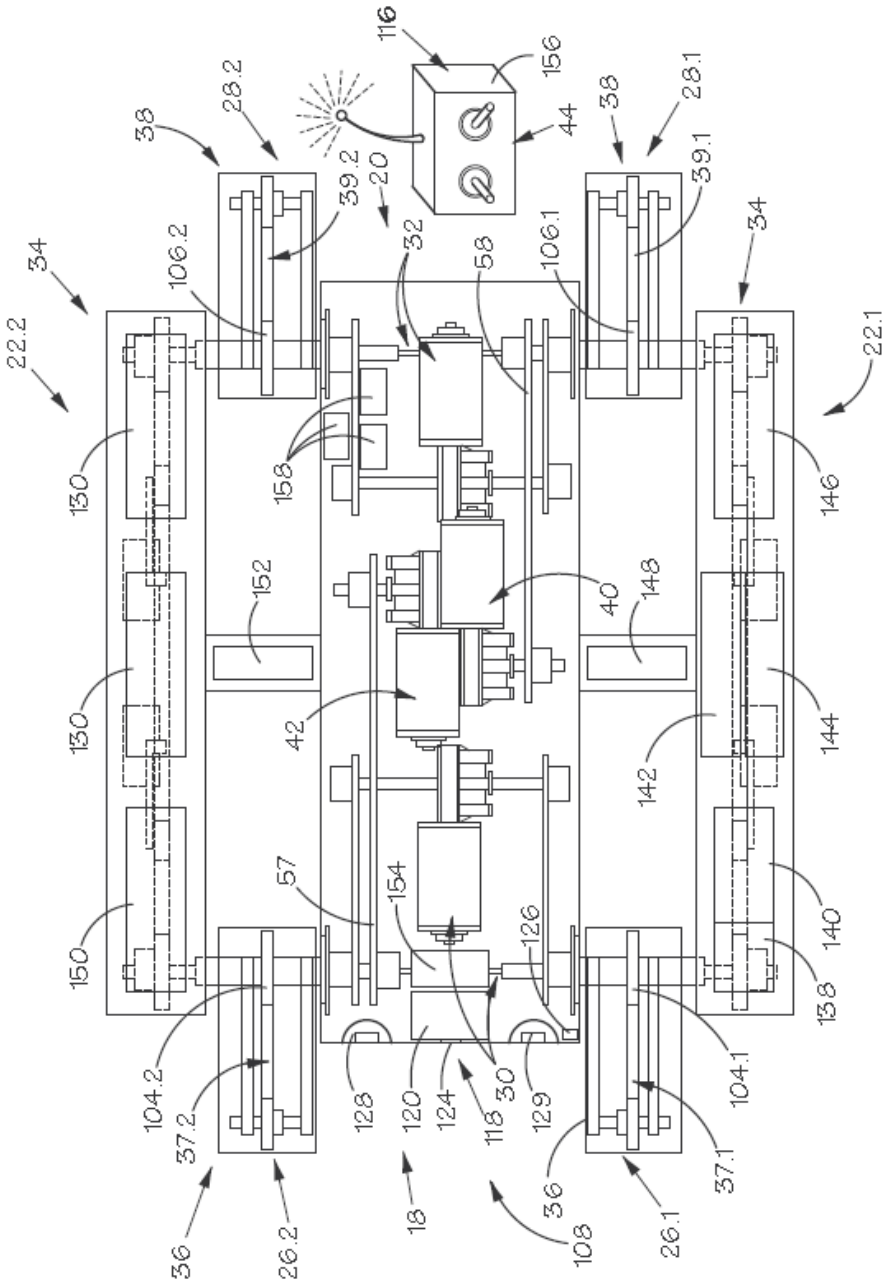


Figure B-5: Mechanical and Module / Components Layout

Appendix C – Micro-controller Code

C.1 Motor Driver Controller Code

```
/*****************************************************
Register generation of this program was produced by the
CodeWizardAVR V2.04.1 Standard
Automatic Program Generator
© Copyright 1998-2009 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : CAESAR Urban Search And Rescue Robot
Version : 4
Date    : 2009/11/02
Author  : Riaan Stopforth
Company : MR2G - Search and Rescue Division - UKZN
Comments: Motor Driver Controller


Chip type               : ATmega32L
Program type            : Application
AVR Core Clock frequency: 3.680000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 512
*****************************************************/

#include <mega32.h>
#include <ctype.h>
#include <stdio.h>
#include <delay.h>
#include <stdlib.h>

int Moutput = 0;
short int pos = 0, am = 0, ro = 0, fa = 0, ba = 0, faangle = 0, baangle = 0, c = 0, wc = 0, zc = 0, m = 0, f = 0, b = 0;
unsigned char arr[10] = {0};

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
//Check encoder interrupt for position of flipper arms
if (PIND.4 == 1)
   {
   fa++;
   if (fa >= 500) fa = 0;
   }
   else
      {
      fa--;
      if (fa <= 0) fa = 500;
      }
}

// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
// Check Encoder interrupt for positions of flipper arms
if (PIND.5 == 0)
   {
   ba++;
   if (ba >= 500) ba = 0;
   }
   else
      {
      ba--;
      if (ba <= 0) ba = 500;
      }
}

// External Interrupt 2 service routine
interrupt [EXT_INT2] void ext_int2_isr(void)
{
// Determine robot orientation
if (PIND.6 == 1)
   {
   ro++;
   if (ro >= 500) ro = 0;
   }
   else
      {
      ro--;
      if (ro <= 0) ro = 500;
      }
}

#ifndef RXB8
#define RXB8 1
```

```
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
     {
     rx_counter=0;
     rx_buffer_overflow=1;
     };
  };

  if (data == toascii(35))  //# - start of packet
   {
   pos = 0;
   arr[pos] = data;
   }
   else
      {
      pos++; //increment position
      arr[pos] = data;
      }
   if ((data == toascii(33)) && (arr[1] == toascii(50)))   // end of packet
      {
      if (arr[2] == toascii(67))
          {
          printf ("#1Ona!");
          }
          else
              {
                  if ((arr[3] + arr[4]) == 140) //forward
                     {
                     PORTA = 0x00;
                     delay_ms (200);
                     Moutput = 0xE0 | (Moutput & 0b00001111);
                     //PORTA.4 = 1;
                     //PORTA.5 = 1;
                     //PORTA.6 = 1;
                     //PORTA.7 = 1;
                     }
                  if ((arr[3] + arr[4]) == 132) //reverse
                     {
                     PORTA = 0x00;
```

```
    delay_ms (200);
    Moutput = 0xB0 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 167) //STOP
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0x00;
    }
if ((arr[3] + arr[4]) == 152) //left
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0xF0 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 164) //right
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0xA0 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 146) //forward left
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0xC0 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 184) //forward right
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0x20 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 142) //back left           *
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0x80 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 148) //back right
    {
    PORTA = 0x00;
    delay_ms (200);
    Moutput = 0x30 | (Moutput & 0b00001111);
    }
if ((arr[3] + arr[4]) == 160) //front arms forward
    {
    PORTA = 0x00;
    delay_ms (200);
    f = 1;
    //if (am == 0)
    if (arr[5] == toascii (33))
       {
       faangle = 600;
       Moutput = 0x02 | (Moutput & 0b11111100);
       }
    else
       {
       faangle = (arr[5]-48)*100 + (arr[6]-48)*10 + (arr[7] - 48);
       if (fa >= 375) Moutput = 0x02 | (Moutput & 0b11111100);
          else Moutput = 0x03 | (Moutput & 0b11111100);
       }
    }
if ((arr[3] + arr[4]) == 156) //front arms backward
    {
    PORTA = 0x00;
    delay_ms (200);
    f = 1;
    Moutput = 0x03 | (Moutput & 0b11111100);
    }
if ((arr[3] + arr[4]) == 157) //back arms forward
    {
    PORTA = 0x00;
    delay_ms (200);
    b = 1;
    if (arr[5] == toascii(33)) // end byte
       {
       baangle = 600;
       Moutput = 0x0C | (Moutput & 0b11110011);
       }
       else
       {
       baangle = (arr[5] - 48)*100 + (arr[6] - 48)*10 + (arr[7] - 48);
       if ((ba >= 375) || (ba < baangle)) Moutput = 0x0C | (Moutput & 0b11110011);
       if (((ba < 375) && (ba >= baangle)) || (baangle >= 375)) Moutput = Moutput = 0x08 | (Moutput & 0b11110011);
       }
    }
if (((arr[3] + arr[4]) == 169) && (arr[5] == toascii (33))) //back arms backwards
    {
    PORTA = 0x00;
    delay_ms (200);
    b = 1;
    Moutput = 0x0C | (Moutput & 0b11110000);
```

```
                }
            if (arr[3] == toascii(65))//auto
                {
                am = 1;
                }
            if (arr[3] == toascii(77))//manual
                {
                am = 0;
                }
            if ((arr[3] + arr[4]) == 189) //front arms zero
                {
                PORTA = 0x00;
                delay_ms (200);
                f = 1;
                //if (am == 0)
                zc = 1;
                }
            if ((arr[3] + arr[4]) == 154) //back arms zero
                {
                PORTA = 0x00;
                delay_ms (200);
                f = 1;
                //if (am == 0)
                wc = 1;
                }
        }
    }

}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
   {
   --tx_counter;
   UDR=tx_buffer[tx_rd_index];
   if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else
   UDR=c;
#asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>
```

```
void main(void)
{

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: On
// INT1 Mode: Rising Edge
// INT2: On
// INT2 Mode: Rising Edge
GICR|=0xE0;
MCUCR=0x0F;
MCUCSR=0x40;
GIFR=0xE0;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 1200
UCSRA=0x00;
```

```
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
//UBRRL=0xBF; //3.68 MHz
UBRRL=0xCF;  //4.0 MHz

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// Global enable interrupts
#asm("sei")
PORTA = 0x00;
while (1)
    {
    if (am == 1) //autonomous flipper arm orientation
     {
     if (((ro >= 375) && (ro < 501)) || ((ro < 125)) && (ro >= 0))
        {
        faangle = 250;
        baangle = 250;
        c = 1;

        if (PIND.7 == 0)
           {
           faangle = 206;
           baangle = 206;
           c = 1;
           }
        }
       else if (((ro < 375) && (ro >= 125)))
           {
           faangle = 250;
           baangle = 250;
           c = 1;
           if (PIND.7 == 0)
              {
              faangle = 294;
              baangle = 294;
              c = 1;
              }
           }

      //determine direction that flipper arms turn with least resistance on the arms
      // This is determined by the current position of the arms
      if ((c == 1) || (zc == 1) || (wc == 1))
         {
         if (zc == 1)
            {
            faangle = 0;
            zc = 0;
            }
         if (wc == 1)
            {
            baangle = 0;
            wc = 0;
            }
         PORTA = 0x00;
         delay_ms (200);
         if (m == 0)
           {
            if (((fa >= 375) || (fa < faangle)) && (f == 0))
               {
               Moutput = 0x02 | (Moutput & 0b11111100);
               m = 1;
               f = 1;
               }
            if ((f == 0) && (fa <375) && (fa >= faangle))
                  {
                  Moutput = 0x03 | (Moutput & 0b11111100);
                  m = 1;
                  f = 1;
                  //printf ("L");
                  }
            if ((b == 0) && ((ba > 375) || (ba < baangle)))
               {
               Moutput = 0x0C | (Moutput & 0b11110011);
               m = 1;
               b = 1;
               }
            if ((b == 0) && (((ba >= baangle) && (ba < 375))))// || baangle >= 375)
                  {
                  Moutput = Moutput = 0x08 | (Moutput & 0b11110011);
                  m = 1;
                  b = 1;
                  }
            }
         }
       c = 0;
      }
```

```
    if (((abs(faangle - fa) < 50) && (f == 1) && (am == 0)) || (((abs(faangle - fa) < 20) && (f == 1) && (am == 1))))
      {
      Moutput = 0x00 | (Moutput & 0b11111100);
      m = 0;
      f = 0;
      }
    if (((abs(baangle - ba) < 50) && (b == 1) && (am == 0)) || ((abs(baangle - ba) < 20) && (b == 1) && (am == 1)))
      {
      Moutput = 0x00 | (Moutput & 0b11110011);
      m = 0;
      b = 0;
      }
    PORTA = Moutput;
    delay_ms(100);
    };
}
```

C.2 Sensor Controller Code

```
/*****************************************************
Register generation of this program was produced by the
CodeWizardAVR V2.04.1 Standard
Automatic Program Generator
© Copyright 1998-2009 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : CAESAR Urban Search And Rescue Robot
Version : 2
Date    : 2009/11/02
Author  : Riaan Stopforth
Company : MR2G - Search and Rescue Division - UKZN
Comments:


Chip type               : ATmega8L
Program type            : Application
AVR Core Clock frequency: 3.680000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega8.h>
#include <stdlib.h>
#include <ctype.h>

int value = 0, rec = 0, pos = 0;
char arr[10] = {0};


#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
```

```
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
     {
     rx_counter=0;
     rx_buffer_overflow=1;
     };
   };
   if (data == toascii(35)) //# - start of packet
    {
    pos = 0;
    arr[pos] = data;
    }
    else
       {
       pos++;
       arr[pos] = data;
       }
    if ((data == toascii(33)) && (arr[1] == toascii(51))) // end of packet
       {
       rec = 1;
       }
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
   {
   --tx_counter;
   UDR=tx_buffer[tx_rd_index];
   if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else
```

```
   UDR=c;
#asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

#include <delay.h>

#define ADC_VREF_TYPE 0x00

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

// Declare your global variables here

void main(void)
{
short int i;

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
```

```
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 1200
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
//UBRRL=0xBF;
UBRRL=0xCF; // 4mhz crystal

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 57.500 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x86;

// Global enable interrupts
#asm("sei")

while (1)
    {
    if (rec == 1)
      {
      if (arr[3] == toascii(55)) // Send all sensor data
            {
            rec = 0;
            printf ("#1O");
            for (i = 0; i <= 5; i++)
                  {
                  value = read_adc (i);
                  if (value < 1000) printf ("0");
                  if (value < 100) printf ("0");
                  if (value < 10) printf ("0");
                  printf ("%d:",value);
                  }
            printf ("%d!",value);
            }
       else   // send the requested sensor's data
            {
            value = read_adc(arr[3] - 48);
            printf ("#1O");
            if (value < 1000) printf ("0");
            if (value < 100) printf ("0");
            if (value < 10) printf ("0");
            printf ("%d!",value);
            rec = 0;
            }
      }

    };
}
```

C.3 Control Station RCP Code

```
/*****************************************************
Register generation of this program was produced by the
CodeWizardAVR V1.23.8c Professional
Automatic Program Generator
© Copyright 1998-2003 HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project : Communication Protocol Layer - control station
Version : 5
Date    : 2009/11/02
Author  : Riaan Stopforth
Company : MR2G - Search and Rescue Division
Comments:


Chip type           : ATmega32L
Program type        : Application
Clock frequency     : 3.680000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 512
```

```
**********************************************/

#include <mega32.h>
#include <stdio.h>
#include <string.h>
#include <delay.h>
#include <math.h>
#include <ctype.h>


const char callsign[7] = "USARC1";
short int cksmR = 0, Rx = 0, i = 0, x = 0, pos = 0;
int time = 0;
short int Rvalid = 0, txagain = 0, csel = 1;
eeprom unsigned char arR[50] = {0},Txarr[50] = {0};

const char othercallsign[7] = "USARR1";
short int cksmT = 0, Datalength = 0;
eeprom unsigned char Rarr[50] = {0}, Ttype, appdata [30] = {0};

// I2C Bus functions
#asm
   .equ __i2c_port=0x15
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];
unsigned char rx_wr_index,rx_rd_index,rx_counter;
// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
#pragma savereg-
interrupt [USART_RXC] void uart_rx_isr(void)
{
char status,data;


#asm
    push r26
    push r27
    push r30
    push r31
    in   r26,sreg
    push r26
#endasm
status=UCSRA;
data=UDR;



if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      };
   };
#asm
    pop  r26
    out  sreg,r26
    pop  r31
    pop  r30
    pop  r27
    pop  r26
#endasm


if (csel == 2)
    {
    //Identify the first byte of data stream
    if (data == toascii(35))
```

```
      {
      pos = 0;
      arR[pos] = data;
      }
         else
   //verify that the packet is meant for this station, if not, delay for packet transmission duration
            {
            pos = pos + 1;
            arR[pos] = data;
            }
         if (pos == 10)
            {
            x = 0;
            for (i = 0; i <= 5; i++)
               {
               if (callsign[i] == arR[i + 4]) x++;
               }
              if (x != 6)
               {
               delay_ms (7 * (arR[2] + arR[3]));
               }
            }

   //check for the last byte, and if so, verify the packets contents with the Checksum byte.
   if ((data == toascii(33)) && (x == 6))//'!'
      {
      cksmR = 35;
       while ((cksmR == 35) || (cksmR == 33) || (isprint (cksmR) != 1))
          {
           cksmR = 0;
           for (i = 0; i <= (pos - 2); i++)
               {
                 cksmR = fmod((arR[i] + cksmR),127);
                  }
               if ((cksmR == 35) || (cksmR == 33) || (isprint (cksmR) != 1)) arR[2]++;
           }
         //Confirm validation of packet so that the reply packet can be sent
         if (cksmR == arR[pos - 1]) Rvalid = 1;

            else
               {
                  Rvalid = 0;
                  Rx++;
               }


      }
   }
   else
      {
      if (data == toascii(64)) // '@'
      {
      Datalength = 0;
      pos = 0;
      arR[pos] = data;
      }
         else
         {
            pos++;
            Datalength++;
            arR[pos] = data;
         }
      }
}
#pragma savereg+

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];
unsigned char tx_wr_index,tx_rd_index,tx_counter;

// USART Transmitter interrupt service routine
#pragma savereg-
interrupt [USART_TXC] void uart_tx_isr(void)
{
```

```
#asm
    push r26
    push r27
    push r30
    push r31
    in   r26,sreg
    push r26
#endasm
if (tx_counter)
   {
   --tx_counter;
   UDR=tx_buffer[tx_rd_index];
   if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
#asm
    pop  r26
    out  sreg,r26
    pop  r31
    pop  r30
    pop  r27
    pop  r26
#endasm
}
#pragma savereg+

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else UDR=c;
#asm("sei")
}
#pragma used-
#endif



void main(void)
{

// Input/Output Ports initialization
// Port A initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func0=In Func1=Out Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=0 State2=T State3=T State4=T State5=T State6=T State7=T
PORTB=0x00;
DDRB=0xFF;

// Port C initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
```

```
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
GICR|=0x00;
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 1200
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0xBF;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// I2C Bus initialization
i2c_init();

// Global enable interrupts
#asm("sei")

Ttype = toascii(79);

while (1)
     {

if (csel == 2)
   {
   // Copy aRr array to rarR array and clear the arR array
   for (i = 0; i <= pos; i++)
      {
      Rarr [i] = arR[i];
      printf ("%c %c",arR[0],arR[1]);
      arR[i] = toascii(32);
      if ((Ttype == toascii(67)) && (Rarr[1] == toascii(50))) csel = 1;
      }
   }
   else
      {
      // Copy ArR array to Appdata and clear arR array and make csel = 2 when a '%' is reached
      for (i = 0; i <= pos; i++)
         {
            appdata[i] = arR[i];
            if (arR[i] == toascii(37)) csel = 2;
            arR[i] = toascii(32);
         }
      Ttype = appdata[1];
      }

//If received packet is valid and transmission has to occur, then create packet to be transmitted.
if ((((x == 6) && (Rvalid == 1)) && ((Rarr[1] != toascii(51)) || (Rarr[1] != toascii(50)))) || (txagain == 1))
   {
   //byte 0 - start byte
   Txarr[0] = toascii (35);   //'#'
   //byte 1 - type of packet
   if (Rarr[1] == toascii(48)) Txarr[1] = toascii(49);
      else if (Rarr[1] == toascii(49))
         {
         if (Ttype == toascii(79)) Txarr[1] = toascii(51);
            else if (Ttype == toascii(67)) Txarr[1] = toascii(52);
```

```
      }
      else if (Rarr[1] == toascii(52)) Txarr[1] = toascii(50);
//restart transmission if timeout occured.
if (txagain == 1) Txarr[1] = toascii(48);
// Validate the duration of packet
if ((Rarr[2] + Rarr[3] - 18) > 127)
   {
       Txarr[2] = toascii(Rarr[2] + Rarr[3] - 18 - 127);
       Txarr[3] = toascii(127);
       }
       else
           {
           Txarr[2] = toascii(32);
           Txarr[3] = toascii(Rarr[2] + Rarr[3] - 18);
           }
//if the duration fields contain a # or ! then add to the first field and subtract the second field.
while ((Txarr[2] == toascii(35)) || (Txarr[2] == toascii(33)) || (Txarr[3] == toascii(35)) || (Txarr[3] == toascii(33)))
        {
        Txarr[3] = toascii(Txarr[3] + 1);
        Txarr[2] = toascii(Txarr[2] - 1);
       }
     //if the duration field contain non-printable characters then increment that field
        while ((isprint(Txarr[2]) != 1) || (isprint(Txarr[3]) != 1))
        {
        if (isprint(Txarr[2]) != 1) Txarr[2]++;
        if (isprint(Txarr[3]) != 1) Txarr[3]++;
       }
//insert callsign of the station that transmitted to this station
for (i = 4; i <= 9; i++) Txarr[i] = othercallsign[i - 4];
//insert the callsign of this station
for (i = 10; i <= 15; i++) Txarr[i] = callsign[i - 10];
cksmT = toascii(35);
//insert data if this is a type 3 or type 4 packet
if ((Txarr[1] == toascii(51)) || (Txarr[1] == toascii(52)))
        {
        for (i = 16; i <= (Datalength - 2); i++) Txarr[i] = appdata[i-14];
        }
//generate the checksum byte
cksmT = 35;
if ((Txarr[1] == 51) || (Txarr[1] == 52))
   {
       while ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1))
          {
          cksmT = 0;
          for (i = 0; i <= 15 + Datalength - 1; i++)
            {
             cksmT = fmod((Txarr[i] + cksmT),127);
             }

          if ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1)) Txarr[2]++;
          }
       }
       else if ((Txarr[1] == 49) || (Txarr[1] == 50))
          {
          while ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1))
          {
          cksmT = 0;
          for (i = 0; i <= 15 ; i++)
            {
             cksmT = fmod((Txarr[i] + cksmT),127);
             }

          if ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1)) Txarr[2]++;
          }
          }
//checksum field
Txarr[16 + Datalength] = toascii(cksmT);
//End byte
Txarr[17 + Datalength] = toascii(33);

//verify if other stations are transmitting. if not transmit

//activate transmitter into tx mode
       PORTB.1 = 0;

if ((Txarr[1] == toascii(51)) || (Txarr[1] == toascii(52)))
        {
        PORTB.1 = 0;
        PORTB.2 = 0;
        PORTB.3 = 1;
        for (i = 0; i <=  (17 + Datalength); i++)
           {
           printf ("%c",Txarr[i]);
           }
        PORTB.1 = 1;
        Rx = Rx - 1;
        time = 0;
        x = 0;
        }
        else if ((Txarr[1] == toascii(49)) || (Txarr[1] == toascii(50)))
           {
           PORTB.1 = 0;
           PORTB.2 = 0;
```

```
                PORTB.3 = 1;
                for (i = 0; i <= 17; i++)
                   {
                   printf ("%c",Txarr[i]);
                   }
                PORTB.1 = 1;
                if (Txarr[1] == toascii(51)) csel = 1;
                Rx = Rx - 1;
                time = 0;
                x = 0;
                }
    }
    else if ((Rarr[1] == toascii(51)) || (Rarr[1] == toascii(52)))
       {
       csel = 1;
       PORTB.2 = 1;
       PORTB.3 = 0;
       //*****output to app layer
       i = 17;
       printf ("@");
       while (Rarr[i-1] != toascii(37))
       {
          printf ("%c",Rarr[i]);
          i++;
       }
       printf ("%%");
       Rx = Rx - 1;
       }

};

}
```

C.4 Robot Station RCP Code

```
/*****************************************************
Registry generation of this program was produced by the
CodeWizardAVR V1.23.8c Professional
Automatic Program Generator
© Copyright 1998-2003 HP InfoTech s.r.l.
http://www.hpinfotech.ro
e-mail:office@hpinfotech.ro

Project : Communication Protocol Layer - robot station
Version : 5
Date    : 2009/11/02
Author  : Riaan Stopforth
Company : MR2G - Search and Rescue Division - UKZN
Comments:

Chip type           : ATmega32L
Program type        : Application
Clock frequency     : 3.680000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 512
*****************************************************/

#include <mega32.h>
#include <stdio.h>
#include <string.h>
#include <delay.h>
#include <math.h>
#include <ctype.h>


const char callsign[7] = "USARR1";
short int cksmR = 0, Rx = 0, i = 0, x = 0, pos = 0;
int time = 0;
short int Rvalid = 0, txagain = 0, csel = 1, tx1 = 0;
eeprom unsigned char arR[30] = {0},Txarr[30] = {0};

const char othercallsign[7] = "USARC1";
short int cksmT = 0, Datalength = 0;
eeprom unsigned char Rarr[30] = {0}, Ttype, appdata [10] = {0};

// I2C Bus functions
#asm
   .equ __i2c_port=0x15
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
```

```
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];
unsigned char rx_wr_index,rx_rd_index,rx_counter;
// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
#pragma savereg-
interrupt [USART_RXC] void uart_rx_isr(void)
{
char status,data;


#asm
   push r26
   push r27
   push r30
   push r31
   in   r26,sreg
   push r26
#endasm
status=UCSRA;
data=UDR;



if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
  if (++rx_counter == RX_BUFFER_SIZE)
    {
    rx_counter=0;
    rx_buffer_overflow=1;
    };
  };
#asm
   pop  r26
   out  sreg,r26
   pop  r31
   pop  r30
   pop  r27
   pop  r26
#endasm


if (csel == 1)
   {
       //Identify the first byte of data stream
       if (data == toascii(35))
       {
       pos = 0;
       arR[pos] = data;
       }
           else
   //verify that the packet is meant for this station, if not, delay for packet transmission duration
               {
               pos = pos + 1;
               arR[pos] = data;
               }
           if (pos == 10)
               {
               x = 0;
               for (i = 0; i <= 5; i++)
                   {
                   if (callsign[i] == arR[i + 4]) x++;
                   }
                  if (x != 6)
                   {
                   delay_ms (7 * (arR[2] + arR[3]));
                   //printf ("x %d",x);
                   }
               }

   //check for the last byte, and if so, verify the packets contents with the Checksum byte.
   if ((data == toascii(33)) && (x == 6))//'!'
      {
      cksmR = 35;
       while ((cksmR == 35) || (cksmR == 33) || (isprint (cksmR) != 1))
          {
           cksmR = 0;
            for (i = 0; i <= (pos - 2); i++)
                {
                  cksmR = fmod((arR[i] + cksmR),94);
                    }
```

```
                cksmR = cksmR + 32;
              if ((cksmR == 35) || (cksmR == 33) || (isprint (cksmR) != 1)) arR[2]++;
            }
        //Confirm validation of packet so that the reply packet can be sent
        if (cksmR == arR[pos - 1]) Rvalid = 1;

            else
              {
                Rvalid = 0;
                Rx++;
              }
      }
      tx1 = 0;
   }
   else
      {
      if (data == toascii(64)) // '@'
      {
      Datalength = 0;
      pos = 0;
      arR[pos] = data;
      }
         else
         {
            pos++;
            Datalength++;
            arR[pos] = data;
         }
      }

}
#pragma savereg+

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];
unsigned char tx_wr_index,tx_rd_index,tx_counter;

// USART Transmitter interrupt service routine
#pragma savereg-
interrupt [USART_TXC] void uart_tx_isr(void)
{
#asm
   push r26
   push r27
   push r30
   push r31
   in   r26,sreg
   push r26
#endasm
if (tx_counter)
   {
   --tx_counter;
   UDR=tx_buffer[tx_rd_index];
   if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
#asm
   pop  r26
   out  sreg,r26
   pop  r31
   pop  r30
   pop  r27
   pop  r26
#endasm
}
#pragma savereg+

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
```

```
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else UDR=c;
#asm("sei")
}
#pragma used-
#endif



void main(void)
{
short int pause = 1, sdata = 0;
// Input/Output Ports initialization
// Port A initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
// Func0=In Func1=Out Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=0 State2=T State3=T State4=T State5=T State6=T State7=T
PORTB=0x00;
DDRB=0xFF;

// Port C initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T State7=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
GICR|=0x00;
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
```

```
// USART Mode: Asynchronous
// USART Baud rate: 1200
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0xBF;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
SFIOR=0x00;

// I2C Bus initialization
i2c_init();

// Global enable interrupts
#asm("sei")

Ttype = toascii(79);

while (1)
     {

if (csel == 1)
   {
   // Copy aRr array to rarR array and clear the arR array
   for (i = 0; i <= pos; i++)
      {
      Rarr [i] = arR[i];
      arR[i] = toascii(32);
      if (((Ttype == toascii(67)) && (Rarr[1] == toascii(50))) || (Rarr[1] == toascii(51)))
         {
         csel = Rarr[18];
         pause = 0;
         }
      }
   }
   else

      {
      // Copy ArR array to Appdata and clear arR array and make csel = 1 when a '%' is reached
      for (i = 0; i <= pos; i++)
         {
            appdata[i] = arR[i];
            if (arR[i] == toascii(37)) csel = 1;
            arR[i] = toascii(32);
         }
      Ttype = appdata[1];
      sdata = 1;
      }

//If received packet is valid and transmission has to occur, then create packet to be transmitted.
if (((((x == 6) && (Rvalid == 1)) && ((Rarr[1] != toascii(51)) || (Rarr[1] != toascii(50)))) || (txagain == 1) || (sdata == 1)) && (tx1 == 0))
   {
   //byte 0 - start byte
   Txarr[0] = toascii (35);   //'#'
   PORTB = 0xFF;
   //byte 1 - type of packet
   if (Rarr[1] == toascii(48)) Txarr[1] = toascii(49);
      else if (Rarr[1] == toascii(49))
         {
         if (Ttype == toascii(79)) Txarr[1] = toascii(51);
            else if (Ttype == toascii(67)) Txarr[1] = toascii(52);
         }
         else if (Rarr[1] == toascii(52)) Txarr[1] = toascii(50);
   //restart transmission if timeout occured.
   if ((txagain == 1) || (sdata == 1)) Txarr[1] = toascii(48);

   PORTB.5 = 1;
   // Validate the duration of packet

   if (sdata == 1)
      {
      sdata = 0;
      tx1 = 1;
      if (Ttype == toascii(79))
         {
         if ((Datalength + 54) > 127)
            {
            Txarr[2] = toascii (54 + Datalength - 127);
            Txarr[3] = toascii(127);
            }
            else
               {
               Txarr[2] = toascii(32);
               Txarr[3] = toascii(54 + Datalength);
               }
         }
         else if (Ttype == toascii(67))
            {
```

```
          if ((Datalength + 72) > 127)
             {
             Txarr[2] = toascii (72 + Datalength - 127);
             Txarr[3] = toascii (127);
             }
             else
                {
                Txarr[2] = toascii (32);
                Txarr[3] = toascii (72 + Datalength);
                }
          }
   }
   else
      {
      if ((Rarr[2] + Rarr[3] - 18) > 127)
         {
              Txarr[2] = toascii(Rarr[2] + Rarr[3] - 18 - 127);
              Txarr[3] = toascii(127);
             }
             else
                 {
                 Txarr[2] = toascii(32);
                 Txarr[3] = toascii(Rarr[2] + Rarr[3] - 18);
                }
      }
PORTB.6 = 1;
//if the duration fields contain a # or ! then add to the first field and subtract the second field.
while ((Txarr[2] == toascii(35)) || (Txarr[2] == toascii(33)) || (Txarr[3] == toascii(35)) || (Txarr[3] == toascii(33)))
        {
        Txarr[3] = toascii(Txarr[3] + 1);
        Txarr[2] = toascii(Txarr[2] - 1);
        }
     //if the duration field contain non-printable characters then increment that field
        while ((isprint(Txarr[2]) != 1) || (isprint(Txarr[3]) != 1))
        {
        if (isprint(Txarr[2]) != 1) Txarr[2]++;
        if (isprint(Txarr[3]) != 1) Txarr[3]++;
        }
PORTB.7 = 1;
//insert callsign of the station that transmitted to this station
for (i = 4; i <= 9; i++) Txarr[i] = othercallsign[i - 4];
//insert the callsign of this station
for (i = 10; i <= 15; i++) Txarr[i] = callsign[i - 10];
cksmT = toascii(35);
//insert data if this is a type 3 or type 4 packet
if ((Txarr[1] == toascii(51)) || (Txarr[1] == toascii(52)))
        {
        for (i = 16; i <= (Datalength - 2); i++) Txarr[i] = appdata[i-14];
        }
//generate the checksum byte
cksmT = 35;
if ((Txarr[1] == 51) || (Txarr[1] == 52))
   {
      while ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1))
         {
         cksmT = 0;
         for (i = 0; i <= 15 + Datalength - 1; i++)
           {
            cksmT = fmod((Txarr[i] + cksmT),94);
            }
            cksmT = cksmT + 32;

         if ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1)) Txarr[2]++;
         }
      }
      else if ((Txarr[1] == 48) || (Txarr[1] == 49) || (Txarr[1] == 50))
         {
         while ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1))
         {
         cksmT = 0;
         for (i = 0; i <= 15 ; i++)
           {
            cksmT = fmod((Txarr[i] + cksmT),94);
            }
            cksmT = cksmT + 32;

         if ((cksmT == 35) || (cksmT == 33) || (isprint (cksmT) != 1)) Txarr[2]++;
         }
         }
 //checksum field
Txarr[16 + Datalength] = toascii(cksmT);
//End byte
Txarr[17 + Datalength] = toascii(33);

//for (i = 0; i<=17; i++)
//   {
//   printf ("%c",Txarr[i]);
//   }

//verify if other stations are transmitting. if not transmit
//activate transmitter into tx mode
PORTB.1 = 0;
if ((Txarr[1] == toascii(51)) || (Txarr[1] == toascii(52)))
```

```
            {
            PORTB.1 = 0;
            PORTB.2 = 1;
            PORTB.3 = 0;
            PORTB.4 = 0;
            for (i = 0; i <=  (17 + Datalength); i++)
               {
               printf ("%c",Txarr[i]);
               }
            PORTB.1 = 1;
            Rx = Rx - 1;
            time = 0;
            x = 0;
            }
            else if ((Txarr[1] == toascii(48)) || (Txarr[1] == toascii(49)) || (Txarr[1] == toascii(50)))
               {
               PORTB.1 = 0;
               PORTB.2 = 1;
               PORTB.3 = 0;
               PORTB.4 = 0;
               for (i = 0; i <= 17; i++)
                  {
                  printf ("%c",Txarr[i]);
                  }
               PORTB.1 = 1;
               Rx = Rx - 1;
               time = 0;
               x = 0;
               }
      }
      else if (((Rarr[1] == toascii(51)) || (Rarr[1] == toascii(52))) && (pause == 0))
         {
         //*****output to app layer
         csel = Rarr[18];
         if (csel == 2)
         {
            PORTB.2 = 0;
            PORTB.3 = 1;
            PORTB.4 = 0;
         }
         else
            {
            PORTB.2 = 0;
            PORTB.3 = 0;
            PORTB.4 = 1;
            }
         i = 17;
         printf ("@");
         while (Rarr[i] != toascii(37))
         {
            printf ("%c",Rarr[i]);
            i++;
         }
         printf ("%c", toascii(37));

         while (arR[pos] != toascii(37))
            {
            }

         for (i = 0; i <= Datalength; i++)
            {
            appdata[i] =  arR[i];
            }

         Rx = Rx - 1;
         pause = 1;
         }
};

}
```

Appendix D – GUI Control Interface

D.1 GUI Interface

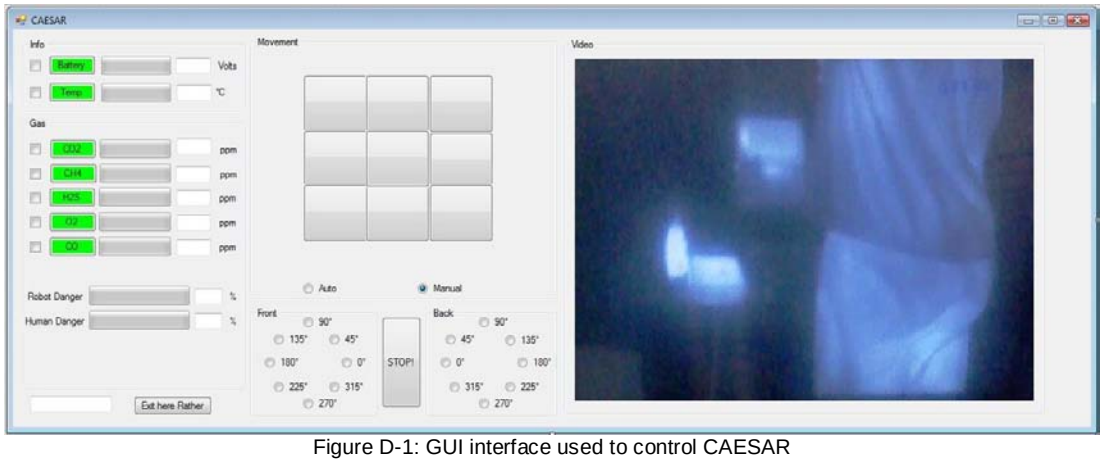


Figure D-1: GUI interface used to control CAESAR

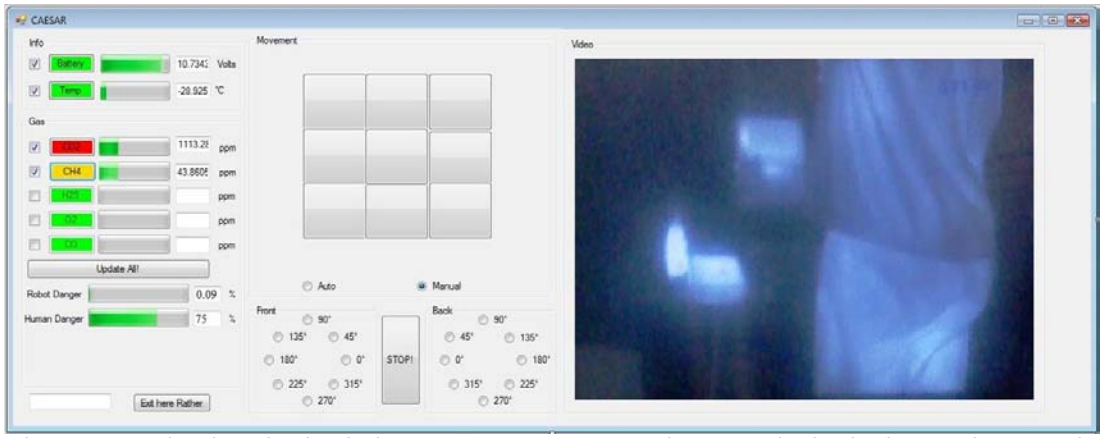


Figure D-2: GUI interface showing the battery power, temperature and concentration levels of CO_2 and CH_4. CO_2 is indicated as being dangerous, while CH_4 is considered unsafe

D.2 GUI Control Code

```
Imports System.Runtime.InteropServices
Imports System.Text
Imports System.Threading
Imports System.Windows.Forms
Imports System.IO.Ports


Public Class CAESAR

#Region "Serialport close"
    'Public Sub Serialportclose()
    'If objForm2.Visible = False Then
    '    SerialPort1.Close()
    ' End If
    ' End Sub
#End Region

#Region "System Runners"
    Dim serialportwrite As String = "" 'THE OVERALL WRITE STRING
    Dim anavalue As String = "" 'THE ANALYSIS STRING USED WHEN DATA RECEIVED

    Dim battsens As Integer 'ALL THESE FOR HANDLING THE DATA RECEIVED
    Dim tempsens As Integer
    Dim CO2sens As Integer
    Dim methsens As Integer
    Dim COsens As Integer
```

```
    Dim H2Ssens As Integer
    Dim O2sens As Integer

    Dim battvalue As Double 'ALL THESE ARE FOR DISPLAYING THE VALUES IN THE PROGRESSBARS
    Dim tempvalue As Double
    Dim CO2value As Double
    Dim methvalue As Double
    Dim COvalue As Double
    Dim H2Svalue As Double
    Dim O2value As Double

        Dim battvalueNorm As Integer 'ALL THESE ARE FOR NORMALISED VALUES, TO USE IN THE OVERALL (ROBOT AND HUMAN DANGER) PROGRESS BARS
    Dim tempvalueNorm As Integer
    Dim CO2valueNorm As Integer
    Dim methvalueNorm As Integer
    Dim COvalueNorm As Integer
    Dim H2SvalueNorm As Integer
    Dim O2valueNorm As Integer

    Dim Selectedbatt As Integer 'TO DETERMINE WHETHER THE GASES SHOULD BE USED, THEY ARE USED IN "SELECT/DESELECT SENSORS"
    Dim Selectedtemp As Integer
    Dim Selectedco2 As Integer
    Dim Selectedmeth As Integer
    Dim Selectedco As Integer
    Dim Selectedh2s As Integer
    Dim Selectedo2 As Integer

    Dim Wbatt As Integer 'WEIGHTINGS FOR AI INITIALISED
    Dim Wtemp As Integer
    Dim Wco2 As Integer
    Dim Wmeth As Integer
    Dim Wco As Integer
    Dim Wh2s As Integer
    Dim Wo2 As Integer

    Private Sub CAESAR_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        If objForm2.Visible = True Then
            SerialPort1.Open()
        ElseIf objForm2.Visible = False Then
            SerialPort1.Close()
        End If

        'THE FOLLOWING LINES WERE FOR TESTING WHAT THE BARS WOULD LOOK LIKE
        '  BatteryButton.BackColor = Color.Red
        '  BatteryProgressBar.Value = 800
        '  BattTextBox.Text = 700 / 1024 * 12

        ' TempButton.BackColor = Color.Yellow
        ' TempProgressBar.Value = 600
        ' TempTextBox.Text = 450 / 1024 * 140




    End Sub

    Private Sub ExitButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ExitButton.Click
        SerialPort1.Close()
        objForm2.Visible = False
    End Sub
#End Region

#Region "Movement"
    Private Sub ButtonFL_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonFL.MouseHover
        Dim MyTextFL As String = "#2OFL!"
        serialportwrite = MyTextFL
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonFF_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonFF.MouseHover
        Dim MyTextFF As String = "#2OFF!"
        serialportwrite = MyTextFF
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonFR_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonFR.MouseHover
        Dim MyTextFR As String = "#2OFr!"
        serialportwrite = MyTextFR
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonLL_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonLL.MouseHover
        Dim MyTextLL As String = "#2OLL!"
        serialportwrite = MyTextLL
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonRR_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonRR.MouseHover
        Dim MyTextRR As String = "#2ORR!"
        serialportwrite = MyTextRR
```

```
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonBL_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonBL.MouseHover
        Dim MyTextBL As String = "#2OBL!"
        serialportwrite = MyTextBL
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonBB_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonBB.MouseHover
        Dim MyTextBB As String = "#2OBB!"
        serialportwrite = MyTextBB
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonBR_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonBR.MouseHover
        Dim MyTextBR As String = "#2OBR!"
        serialportwrite = MyTextBR
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub ButtonSTOP_MouseHover(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonSTOP.MouseHover
        Dim MyTextSTOP As String = "#2OST!"
        serialportwrite = MyTextSTOP
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub EmergencySTOP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles EmergencySTOP.Click
        Dim MyTextSTOP As String = "#2OST!"
        serialportwrite = MyTextSTOP
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
#End Region

#Region "Auto/Man"
    Private Sub AutoRadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AutoRadioButton.Click
        Dim man_auto As String = "#2OA!"
        serialportwrite = man_auto
        TextBoxOut.Text = man_auto
        SerialPort1.Write(serialportwrite)
        FArmGroupBox.Visible = False
        BArmGroupBox.Visible = False
    End Sub

    Private Sub ManualRadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ManualRadioButton.Click
        Dim man_auto As String = "#2OM!"
        serialportwrite = man_auto
        TextBoxOut.Text = man_auto
        SerialPort1.Write(serialportwrite)
        FArmGroupBox.Visible = True
        BArmGroupBox.Visible = True

    End Sub
#End Region

#Region "Arms"

#Region "Front Arm"

    Private Sub F0RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F0RadioButton.Click
        Dim FArm As String = "#2OZF063!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F45RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F45RadioButton.Click
        Dim FArm As String = "#2OZF124!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F90RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F90RadioButton.Click
        Dim FArm As String = "#2OZF185!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F135RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F135RadioButton.Click
        Dim FArm As String = "#2OZF256!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F180RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F180RadioButton.Click
        Dim FArm As String = "#2OZF317!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F225RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F225RadioButton.Click
        Dim FArm As String = "#2OZF378!"
        serialportwrite = FArm
```

```
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F270RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F270RadioButton.Click
        Dim FArm As String = "#2OZF439!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub F315RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles F315RadioButton.Click
        Dim FArm As String = "#2OZF500!"
        serialportwrite = FArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
#End Region

#Region "Rear Arm"

    Private Sub R0RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R0RadioButton.Click
        Dim bArm As String = "#2OWF063!"
        serialportwrite = bArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R45RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R45RadioButton.Click
        Dim BArm As String = "#2OWF124!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R90RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R90RadioButton.Click
        Dim BArm As String = "#2OWF185!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R135RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R135RadioButton.Click
        Dim BArm As String = "#2OWF256!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R180RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R180RadioButton.Click
        Dim BArm As String = "#2OWF317!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R225RadioButton1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R225RadioButton1.Click
        Dim BArm As String = "#2OWF378!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R270RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R270RadioButton.Click
        Dim BArm As String = "#2OWF439!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
    Private Sub R315RadioButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles R315RadioButton.Click
        Dim BArm As String = "#2OWF500!"
        serialportwrite = BArm
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
    End Sub
#End Region
#End Region

#Region "Sensors String Send"

    '0 = methane
    '1 = co2
    '2 = temp
    '3 = batt
    '4 = co
    '5 = h2s
    '6 = all
    '7 = o2
    Private Sub BatteryButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BatteryButton.Click
        Dim InfoCheck As String = "#3C3!"
        serialportwrite = InfoCheck
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
        Dim anavalue3 As String = "3" 'FOR BATT
        anavalue = anavalue3

    End Sub
    Private Sub TempButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TempButton.Click
        Dim InfoCheck As String = "#3C2!"
        serialportwrite = InfoCheck
        TextBoxOut.Text = serialportwrite
```

```
        SerialPort1.Write(serialportwrite)
        Dim anavalue2 As String = "2" 'TEMP
        anavalue = anavalue2

    End Sub
    Private Sub CO2Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CO2Button.Click
        Dim InfoCheck As String = "#3C1!"
        serialportwrite = InfoCheck
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
        Dim anavalue1 As String = "1" 'CO2
        anavalue = anavalue1

    End Sub
    Private Sub MethaneButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MethaneButton.Click
        Dim InfoCheck As String = "#3C0!"
        serialportwrite = InfoCheck
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
        Dim anavalue0 As String = "0" 'METH
        anavalue = anavalue0

    End Sub
    Private Sub UpdateAllButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim InfoCheck As String = "#3C6!"
        serialportwrite = InfoCheck
        TextBoxOut.Text = serialportwrite
        SerialPort1.Write(serialportwrite)
        Dim anavalue6 As String = "6" 'ALL!
        anavalue = anavalue6
    End Sub
#End Region

#Region "DataReceived"
        Public Sub SerialPort1_DataReceived(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) Handles
SerialPort1.DataReceived
        'GLOBAL VARIABLES
        Dim readline As String              'OUR LINE TO READ
        Dim value As String = "!"           'THE NEWLINE CHARACTER
        Dim safetybuffchar As String = "#"  'SEE COMMENT BELOW...
        Dim analysebuffer As String = ""
        Dim ForTheSensors As String = "1"
        Dim checkboxcheck As String = "0"
        Dim AllValue As String = ""         'PASSES THROUGH THE ALLUPDATE STRING, may not be necessary
        'Dim ForTheMovement As String = "2"
        SerialPort1.NewLine = value         'STATING WHAT THE END OF LINE IS
        readline = SerialPort1.ReadLine()   'READ THE LINE!!!
        TestReadTextBox.Text = readline 'SHOW THE LINE
        analysebuffer = readline            'TRANSFER THE BUFFER
        'THE FOLLOWING TWO LINES ARE TO SEE WHETHER DATA IS BEING RECEIVED
        CheckBox3.Checked = True
        TextBoxIn.Text = analysebuffer

        If (analysebuffer(0) = safetybuffchar) Then 'STATING WHAT THE BEGINNING OF THE LINE IS AND IN DOING SO ADDING A SAFETY MEASURE
SO THAT THE WHOLE PROGRAM DOESN'T FREAK OUT WHENEVER YOU SEND TOO FEW CHAR'S AND THEN YOU ACTUALLY HAVE TO
UNPLUG EVERYTHING TO GET IT TO WORK...
            TestTextboxTwo.Text = analysebuffer(1) 'SHOW THE 2ND CHAR FOR TESTING
            TestTextBoxThree.Text = analysebuffer(2) ' SHOW THE 3RD CHAR FOR TESTING
            CheckBox1.Checked = True
            If (analysebuffer(1) = ForTheSensors) Then 'DECLARE THESE TO MAKE IF STATEMENT WORK FOR THE SENSORS
                CheckBox2.Checked = True
                Dim anavalue3 As String = "3" 'FOR BATT
                Dim anavalue0 As String = "2" 'TEMP
                Dim anavalue1 As String = "1" 'CO2
                Dim anavalue2 As String = "0" 'METH
                Dim anavalue6 As String = "6" 'UPDATE ALL
                Dim sensvalue As String = ""
                anavalue3 = TextBoxIn.Text(3)


                'BATT

                If (anavalue = "3") Then 'READING THE 4TH CHARACTER, BATT OR UPDATE ALL
                    CheckBox4.Checked = True
                    battsens = analysebuffer.Substring(3, 4) 'AFTER 3rd CHAR, READ 4 CHAR'S, make sure they are numbers only!
                    If analysebuffer(2) = checkboxcheck Then
                        CheckBox4.Checked = True
                    End If
                ElseIf (anavalue = "6") Then
                    battsens = analysebuffer.Substring(18, 4) 'AFTER 18TH CHAR, READ 4 CHAR'S, make sure they are numbers only!
                End If
                BatteryProgressBar.Value = battsens 'SHOW BATT VALUE AS WHATEVER IS INPUTTED IN % OUT OF 1024
                battvalue = battsens * 12 / 1024    'CALCULATE CORRECT VALUE IN VOLTS, ASSUMES LINEAR RELATION
                BattTextBox.Text = battvalue

                'DEFINING THE POINTS AT WHICH THE BAR TURNS RED/GREEN/GOLD, AND FOR AI
                If 8 < battvalue And battvalue < 10.5 Then
                    BatteryButton.BackColor = Color.Gold
                    '  Wbatt = 1    'THESE ARE THE WEIGHTING FOR A.I.
                ElseIf 10.5 < battvalue And battvalue < 12 Then
                    BatteryButton.BackColor = Color.Lime
                    ' Wbatt = 0
                ElseIf 0 < battvalue And battvalue < 8 Then
```

```
            BatteryButton.BackColor = Color.Red
            ' Wbatt = 2
        End If



        'TEMP
        'Or (anavalue = "6")
    ElseIf (anavalue = "2") Then
        If (anavalue = "2") Then
            tempsens = analysebuffer.Substring(3, 4) 'AFTER 3rd CHAR, READ 4 CHAR'S, make sure they are numbers only!
        ElseIf (anavalue = "6") Then
            tempsens = analysebuffer.Substring(13, 4) 'CHAR'S 14 15 16 17
        End If

        'TEMP SENSOR -40 TO 100 DEG
        TempProgressBar.Value = tempsens 'SHOW TEMP % out of 1024
         tempvalue = -40 + tempsens * 140 / 1024 * (2.2) 'CONVERTED TO DEG CELSIUS, LINEAR RELATION, and a random multiplication factor to
get 25 deg celcius
        TempTextBox.Text = tempvalue
        tempvalueNorm = tempvalue / 1.4   'NORMALISING THE TEMP


        'DEFINING THE POINTS AT WHICH THE BAR TURNS RED/GREEN/GOLD, AND FOR AI
            If 100 < tempvalue And tempvalue < 70 Then  'DEFINING THE POINT AT WHICH THE BAR TURNS RED/GREEN (585 IMPLIES 40 DEG
CELCIUS I HOPE)
            TempButton.BackColor = Color.Red
            ' Wtemp = 2
        ElseIf 45 < tempvalue And tempvalue < 70 Then
            TempButton.BackColor = Color.Gold
            ' Wtemp = 1
        ElseIf -40 < tempvalue And tempvalue < 45 Then
            TempButton.BackColor = Color.Lime
            'Wtemp = 0
        End If



        'CO2
        '
    ElseIf (anavalue = "1") Then
        If (anavalue = "1") Then
            CO2sens = analysebuffer.Substring(3, 4) 'AFTER 3rd CHAR, READ 4 CHAR'S, make sure they are numbers only!
        ElseIf (anavalue = "6") Then 'CHAR'S 9 10 11 12 then
            CO2sens = analysebuffer.Substring(8, 4)
        End If

        CO2ProgressBar.Value = CO2sens 'SHOW CO2 VALUE AS % OUT OF 1024
        CO2value = (CO2sens / 1024) '* 4000 '4000ppm IS MAX READABLE
        CO2TextBox.Text = CO2value
        CO2valueNorm = (CO2value / 400) '* Selectedco2   '/400 TO SAY THAT 100ppm IS THE NORMAL AIR VALUE

        'DEFINING THE POINTS AT WHICH THE BAR TURNS RED/GREEN/GOLD, AND FOR AI
        If (500 / 40) < CO2valueNorm And CO2valueNorm < (4000 / 40) Then      'DANGEROUS, /40 IS TO NORMALISE TO 100%
            CO2Button.BackColor = Color.Red
            '  Wco2 = 2
        ElseIf (400 / 40) < CO2valueNorm And CO2valueNorm < (500 / 40) Then   'UNSAFE
            CO2Button.BackColor = Color.Gold
            ' Wco2 = 1
        ElseIf 0 < CO2valueNorm And CO2valueNorm < (400 / 40) Then     'OK
            CO2Button.BackColor = Color.Lime
            '  Wco2 = 0
        End If



        'METH
        '
    ElseIf (anavalue = "0") Then
        If (anavalue = "0") Then
            methsens = analysebuffer.Substring(3, 4) 'AFTER 3rd CHAR, READ 4 CHAR'S, make sure they are numbers only!
        ElseIf (anavalue = "6") Then
            methsens = analysebuffer.Substring(3, 4) 'CHAR'S 4 5 6 7
        End If

        MethProgressBar.Value = methsens 'SHOW METH VALUE IN PROGRESS BAR

        Dim Rl As Integer = 5 'IN OHMS
        Dim Ro As Integer = 5 'IN OHMS
        Dim d As Double = Math.Log10(Rl / Ro * (1024 / methsens - 1))
        methvalue = 10 ^ ((d - 2.24998) / (-1.1156305)) 'ASSUME 5000PPM IS MAX VALUE
        MethaneTextBox.Text = methvalue 'SHOWN IN PPM
        methvalueNorm = (methvalue / 50) * Selectedmeth    'NORMALISE TO 100%,AND ONLY ALLOW TO BE ACTIVE IF SELECTED. THIS IS THEN
USED IN A.I. SECTION, WHERE THE VALUE IS FULL OR ZERO, SO HIGHEST ACTIVE SENSOR VALUE CHOSEN

        'DEFINING THE POINTS AT WHICH THE BAR TURNS RED/GREEN/GOLD, AND FOR AI
            If (1000 / 50) < methvalueNorm And methvalueNorm < (5000 / 50) Then      'DANGEROUS, NORMALISED TO 100% BY /50 SINCE MAX IS
5000PPM
            MethaneButton.BackColor = Color.Red
            ' Wmeth = 2
        ElseIf (400 / 500) < methvalueNorm And methvalueNorm < (1000 / 50) Then   'UNSAFE
            MethaneButton.BackColor = Color.Gold
            'Wmeth = 1
```

```
        ElseIf 0 < methvalueNorm And methvalueNorm < (400 / 50) Then     'DANGER ROBOT
            MethaneButton.BackColor = Color.Lime
            ' Wmeth = 0
        End If




        'H2S
      ElseIf (anavalue = "5") Or (anavalue = "6") Then
        If (anavalue = "5") Then
            H2Ssens = analysebuffer.Substring(3, 4) 'AFTER 3rd CHAR, READ 4 CHAR'S, make sure they are numbers only!
        ElseIf (anavalue = "6") Then
            H2Ssens = analysebuffer.Substring(123, 4) 'CHAR'S XYZ
        End If

        '   H2SProgressBar.Value = H2Ssens 'SHOW H2S VALUE IN PROGRESS BAR,OUT OF 1024

        Dim Rl As Integer = 5 'IN OHMS
        Dim Ro As Integer = 5 'IN OHMS
        Dim d As Double = Math.Log10(Rl / Ro * (1024 / methsens - 1))
        methvalue = 10 ^ ((d - 2.24998) / (-1.1156305)) 'ASSUME 5000PPM IS MAX VALUE, USE d HERE
        MethaneTextBox.Text = methvalue 'SHOWN IN PPM
        methvalueNorm = (methvalue / 50) * Selectedmeth    'NORMALISE TO 100%,AND ONLY ALLOW TO BE ACTIVE IF SELECTED. THIS IS THEN USED IN A.I. SECTION, WHERE THE VALUE IS FULL OR ZERO, SO HIGHEST ACTIVE SENSOR VALUE CHOSEN

        'DEFINING THE POINTS AT WHICH THE BAR TURNS RED/GREEN/GOLD, AND FOR AI
        If (1000 / 50) < methvalueNorm And methvalueNorm < (5000 / 50) Then     'DANGEROUS, NORMALISED TO 100% BY /50 SINCE MAX IS 5000PPM
            MethaneButton.BackColor = Color.Red
            ' Wmeth = 2
        ElseIf (400 / 500) < methvalueNorm And methvalueNorm < (1000 / 50) Then   'UNSAFE
            MethaneButton.BackColor = Color.Gold
            'Wmeth = 1
        ElseIf 0 < methvalueNorm And methvalueNorm < (400 / 50) Then     'DANGER ROBOT
            MethaneButton.BackColor = Color.Lime
            'Wmeth = 0
        End If


        'NOW FOR PROGRESSBAR AI!!!
        Dim RobotDanger As Double
        Dim DangerIlevelRob As Double
        Dim selectedGases As Integer = (Selectedco + Selectedco2 + Selectedh2s + Selectedmeth + Selectedo2)
        If selectedGases > 0 Then
            CO2valueNorm = co2NumericUpDown.Value * Selectedco2 '*SELE... TO ALLOW FOR IT TO BE USED ONLY IF IT IS SELECTED
            CO2ProgressBar.Value = CO2valueNorm
            CO2TextBox.Text = CO2valueNorm

            H2SvalueNorm = h2sNumericUpDown.Value * Selectedh2s
            H2SProgressBar.Value = H2SvalueNorm
            H2STextBox.Text = H2SvalueNorm


            methvalueNorm = methNumericUpDown.Value * Selectedmeth
            MethProgressBar.Value = methvalueNorm
            MethaneTextBox.Text = methvalueNorm

            O2valueNorm = O2NumericUpDown.Value * Selectedo2
            O2ProgressBar.Value = O2valueNorm
            O2TextBox.Text = O2valueNorm

            COvalueNorm = CONumericUpDown.Value * Selectedco
            COProgressBar.Value = COvalueNorm
            COTextBox.Text = COvalueNorm



            If methvalueNorm >= CO2valueNorm And methvalueNorm >= H2SvalueNorm And methvalueNorm >= CO2valueNorm And methvalueNorm >= COvalueNorm Then 'TO DETERMINE WHICH GAS VALUE IS HIGHEST. HERE CH4 IS SEEN AS MOST FLAMABLE, SO IT TAKES PRECEDENCE, HENCE >=. IF IT WERE ONLY >, THE PROGRESSBAR GETS STUCK
                RobotDanger = methvalueNorm + (1 / selectedGases) * (CO2valueNorm * Selectedco2 + H2SvalueNorm * Selectedh2s + O2valueNorm * Selectedo2 + COvalueNorm * Selectedco) * ((100 - methvalueNorm) / 100)  '100 IS NORMALISED FLAMMABLE MIN,0'S ARE FOR OTHER GASES.THE NORMALISED VALUES ARE WHAT THEY SHOULD BE IFF SELECTED = 1
                RobotDangerProgressBar.Value = RobotDanger
                RobotDangerTextBox.Text = RobotDanger
            ElseIf CO2valueNorm > methvalueNorm And CO2valueNorm > H2SvalueNorm And CO2valueNorm > COvalueNorm And CO2valueNorm > O2valueNorm Then
                RobotDanger = CO2valueNorm + (1 / selectedGases) * (methvalueNorm * Selectedmeth + H2SvalueNorm * Selectedh2s + O2valueNorm * Selectedo2 + COvalueNorm * Selectedco) * ((100 - CO2valueNorm) / 100)
                RobotDangerProgressBar.Value = RobotDanger
                RobotDangerTextBox.Text = RobotDanger
            ElseIf H2SvalueNorm > CO2valueNorm And H2SvalueNorm > methvalueNorm And H2SvalueNorm > O2valueNorm And H2SvalueNorm > COvalueNorm Then
                RobotDanger = H2SvalueNorm + (1 / selectedGases) * (methvalueNorm * Selectedmeth + CO2valueNorm * Selectedco2 + O2valueNorm * Selectedo2 + COvalueNorm * Selectedco) * ((100 - H2SvalueNorm) / 100)
                RobotDangerProgressBar.Value = RobotDanger
                RobotDangerTextBox.Text = RobotDanger
            ElseIf COvalueNorm > methvalueNorm And COvalueNorm > H2SvalueNorm And COvalueNorm > CO2valueNorm And COvalueNorm > O2valueNorm Then
                RobotDanger = COvalueNorm + (1 / selectedGases) * (methvalueNorm * Selectedmeth + CO2valueNorm * Selectedco2 + O2valueNorm * Selectedo2 + H2SvalueNorm * Selectedh2s) * ((100 - COvalueNorm) / 100)
```

```
                RobotDangerProgressBar.Value = RobotDanger
                RobotDangerTextBox.Text = RobotDanger
            End If

        End If

        Dim HumanDanger As Double
        Dim DangerlevelHum As Double


        'UPDATE ALL! INCLUDES EQUATIONS AGAIN   'PROBABLY NOT NECESSARY.....
    ElseIf (anavalue = "6") Then
        If BattCheckBox.Checked = True Then
            battsens = analysebuffer.Substring(18, 4) 'AFTER 18TH CHAR, READ 4 CHAR'S, make sure they are numbers only!
            BatteryProgressBar.Value = battsens
            battvalue = battsens * 12 / 1024    'CALCULATE CORRECT VALUE IN VOLTS, ASSUMES LINEAR RELATION
            BattTextBox.Text = battvalue
            If 0 < battsens < 720 Then  'DEFINING THE POINT AT WHICH THE BAR TURNS RED/GREEN (800/1024)
                BatteryButton.BackColor = Color.Red
            ElseIf 721 < battsens < 920 Then
                BatteryButton.BackColor = Color.Gold
            ElseIf 921 < battsens < 1024 Then
                BatteryButton.BackColor = Color.Lime
            End If
        End If

        If TempCheckBox.Checked = True Then
            tempsens = analysebuffer.Substring(13, 4) 'CHAR'S 14 15 16 17
            TempProgressBar.Value = tempsens
            tempvalue = tempsens * 140 / 1024 * (2.2) 'CONVERTED TO DEG CELSIUS, LINEAR RELATION, and a random multiplication factor
            TempTextBox.Text = tempvalue
            If 0 < tempsens < 585 Then  'DEFINING THE POINT AT WHICH THE BAR TURNS RED/GREEN (585 IMPLIES 40 DEG CELCIUS I HOPE)
                TempButton.BackColor = Color.Red
            ElseIf 586 < tempsens < 732 Then
                TempButton.BackColor = Color.Gold
            ElseIf 733 < tempsens < 1024 Then
                TempButton.BackColor = Color.Lime
            End If
        End If

        If CO2CheckBox.Checked = True Then
            CO2sens = analysebuffer.Substring(8, 4) 'CHAR'S 9 10 11 12
            CO2ProgressBar.Value = CO2sens
            CO2value = (CO2sens * 4 / 1024) * 1000 '4000ppm IS MAX READABLE
            CO2TextBox.Text = CO2value
            If 1000 < CO2sens < 1024 Then  'DEFINING THE POINT AT WHICH THE BAR TURNS RED/GREEN (800/1024)
                CO2Button.BackColor = Color.Red
            ElseIf 400 < CO2sens < 1000 Then
                CO2Button.BackColor = Color.Gold
            ElseIf 0 < CO2sens < 128 Then
                CO2Button.BackColor = Color.Lime
            End If
        End If

        If MethCheckBox.Checked = True Then
            methsens = analysebuffer.Substring(3, 4) 'CHAR'S 4 5 6 7
            MethProgressBar.Value = methsens
            Dim Rl As Integer = 5
            Dim Ro As Integer = 5
            Dim d As Double = Math.Log10(Rl / Ro * (1024 / methsens - 1))
            methvalue = 10 ^ ((d - 2.24998) / (-1.1156305))
            MethaneTextBox.Text = methvalue
            If 1000 < methsens < 1024 Then  'DEFINING THE POINT AT WHICH THE BAR TURNS RED/GREEN (800/1024)
                MethaneButton.BackColor = Color.Red
            ElseIf 400 < CO2sens < 1000 Then
                MethaneButton.BackColor = Color.Gold
            ElseIf 0 < CO2sens < 128 Then
                MethaneButton.BackColor = Color.Lime
            End If
        End If

        End If
        End If
    End If
  End Sub
#End Region

#Region "Select/Deselect sensors"
    Private Sub BattCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
BattCheckBox.CheckedChanged
    If BattCheckBox.Checked = True Then
        BatteryButton.Enabled = True
        Selectedbatt = 1
    ElseIf BattCheckBox.Checked = False Then
        BatteryButton.Enabled = False
        Selectedbatt = 0
    End If
  End Sub
    Private Sub TempCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
TempCheckBox.CheckedChanged
    If TempCheckBox.Checked = True Then
        TempButton.Enabled = True
        Selectedtemp = 1
```

```
        ElseIf TempCheckBox.Checked = False Then
            TempButton.Enabled = False
            Selectedtemp = 0
        End If
    End Sub
            Private Sub CO2CheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
CO2CheckBox.CheckedChanged
        If CO2CheckBox.Checked = True Then
            CO2Button.Enabled = True
            Selectedco2 = 1
        ElseIf CO2CheckBox.Checked = False Then
            CO2Button.Enabled = False
            Selectedco2 = 0
        End If
    End Sub
            Private Sub MethCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MethCheckBox.CheckedChanged
        If MethCheckBox.Checked = True Then
            MethaneButton.Enabled = True
            Selectedmeth = 1
        ElseIf MethCheckBox.Checked = False Then
            MethaneButton.Enabled = False
            Selectedmeth = 0
        End If
    End Sub
            Private Sub H2SCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
H2SCheckBox.CheckedChanged
        If H2SCheckBox.Checked = True Then
            H2SButton.Enabled = True
            Selectedh2s = 1
        ElseIf H2SCheckBox.Checked = False Then
            H2SButton.Enabled = False
            Selectedh2s = 0
        End If
    End Sub
    Private Sub O2CheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles O2CheckBox.CheckedChanged
        If O2CheckBox.Checked = True Then
            O2Button.Enabled = True
            Selectedo2 = 1
        ElseIf O2CheckBox.Checked = False Then
            O2Button.Enabled = False
            Selectedo2 = 0
        End If
    End Sub
    Private Sub COCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles COCheckBox.CheckedChanged
        If COCheckBox.Checked = True Then
            COButton.Enabled = True
            Selectedco = 1
        ElseIf COCheckBox.Checked = False Then
            COButton.Enabled = False
            Selectedco = 0
        End If
    End Sub
#End Region

#Region "Test area"
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Dim tester As String = ""
        tester = TextBoxIn.Text
        TextBoxOut.Text = tester
        serialportwrite = tester
        SerialPort1.Write(serialportwrite)
    End Sub
#End Region

#Region "Test A.I."
    Private Sub TestAIButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TestAIButton.Click
        Dim RobotDanger As Double
        Dim DangerIlevelRob As Double
        Dim selectedGases As Integer = (Selectedco + Selectedco2 + Selectedh2s + Selectedmeth + Selectedo2)
        If selectedGases > 0 Then
            CO2valueNorm = co2NumericUpDown.Value * Selectedco2 '*SELE... TO ALLOW FOR IT TO BE USED ONLY IF IT IS SELECTED
            CO2ProgressBar.Value = CO2valueNorm
            CO2TextBox.Text = CO2valueNorm

            H2SvalueNorm = h2sNumericUpDown.Value * Selectedh2s
            H2SProgressBar.Value = H2SvalueNorm
            H2STextBox.Text = H2SvalueNorm


            methvalueNorm = methNumericUpDown.Value * Selectedmeth
            MethProgressBar.Value = methvalueNorm
            MethaneTextBox.Text = methvalueNorm

            O2valueNorm = O2NumericUpDown.Value * Selectedo2
            O2ProgressBar.Value = O2valueNorm
            O2TextBox.Text = O2valueNorm

            COvalueNorm = CONumericUpDown.Value * Selectedco
            COProgressBar.Value = COvalueNorm
            COTextBox.Text = COvalueNorm
```

```
            If methvalueNorm >= CO2valueNorm And methvalueNorm >= H2SvalueNorm And methvalueNorm >= CO2valueNorm And methvalueNorm >= COvalueNorm Then 'TO DETERMINE WHICH GAS VALUE IS HIGHEST. HERE CH4 IS SEEN AS MOST FLAMABLE, SO IT TAKES PRECEDENCE, HENCE >=. IF IT WERE ONLY >, THE PROGRESSBAR GETS STUCK
                RobotDanger = methvalueNorm + (1 / selectedGases) * (CO2valueNorm * Selectedco2 + H2SvalueNorm * Selectedh2s + O2valueNorm * Selectedo2 + COvalueNorm * Selectedco) * ((100 - methvalueNorm) / 100)  '100 IS NORMALISED FLAMMABLE MIN,0'S ARE FOR OTHER GASES.THE NORMALISED VALUES ARE WHAT THEY SHOULD BE IFF SELECTED = 1
            RobotDangerProgressBar.Value = RobotDanger
            RobotDangerTextBox.Text = RobotDanger
              ElseIf CO2valueNorm > methvalueNorm And CO2valueNorm > H2SvalueNorm And CO2valueNorm > COvalueNorm And CO2valueNorm > O2valueNorm Then
                RobotDanger = CO2valueNorm + (1 / selectedGases) * (methvalueNorm * Selectedmeth + H2SvalueNorm * Selectedh2s + O2valueNorm * Selectedo2 + COvalueNorm * Selectedco) * ((100 - CO2valueNorm) / 100)
            RobotDangerProgressBar.Value = RobotDanger
            RobotDangerTextBox.Text = RobotDanger
              ElseIf H2SvalueNorm > CO2valueNorm And H2SvalueNorm > methvalueNorm And H2SvalueNorm > O2valueNorm And H2SvalueNorm > COvalueNorm Then
                RobotDanger = H2SvalueNorm + (1 / selectedGases) * (methvalueNorm * Selectedmeth + CO2valueNorm * Selectedco2 + O2valueNorm * Selectedo2 + COvalueNorm * Selectedco) * ((100 - H2SvalueNorm) / 100)
            RobotDangerProgressBar.Value = RobotDanger
            RobotDangerTextBox.Text = RobotDanger
              ElseIf COvalueNorm > methvalueNorm And COvalueNorm > H2SvalueNorm And COvalueNorm > CO2valueNorm And COvalueNorm > O2valueNorm Then
                RobotDanger = COvalueNorm + (1 / selectedGases) * (methvalueNorm * Selectedmeth + CO2valueNorm * Selectedco2 + O2valueNorm * Selectedo2 + H2SvalueNorm * Selectedh2s) * ((100 - COvalueNorm) / 100)
            RobotDangerProgressBar.Value = RobotDanger
            RobotDangerTextBox.Text = RobotDanger
          End If

        End If

    End Sub
#End Region
End Class
```

Appendix E – Publications

As of August 2010, the following publications were published:

1. R. Stopforth, G. Bright, "**Urban Search And Rescue (USAR) Robots**", 1st Mechatronics and Robotics Symposium, CSIR, Pretoria, South Africa, November 2007
2. R. Stopforth, G. Bright, "**Required Improvements and Possible Solutions for Urban Search And Rescue (USAR) Robots**", 2008 International Conference on Automation, Robotics and Control Systems (ARCS-08), © ISRST, Orlando, USA, July 2008
3. R. Stopforth, G. Bright, R. Harley, "**Robotics Communication Protocol (RCP)**", 2nd Mechatronics and Robotics Symposium, CSIR, Bloemfontein, South Africa, November 2008
4. R. Stopforth, S. Holtzhausen, G. Bright, N. S. Tlale, C. M. Kumile, "**Robots for Search and Rescue Purposes in Urban and Underwater Environments – a survey and comparison**", 15th International Conference on Mechatronics and Machine Vision Practice (M2VIP) 2008, Auckland, New Zealand, December 2008
5. R. Stopforth, G. Bright, R. Harley, "**Experienced Outcomes from the Improvements made to the USAR robot**", 2009 International Conference on Automation, Robotics and Control Systems (ARCS-09), Orlando, USA, July 2009
6. R. Stopforth, "**Contractible Arms Elevating Search And Rescue (CAESAR) Robot Communication**", Durban Amateur Radio Club (DARC), Durban, South Africa, August 2009
7. R. Stopforth, G. Bright, R. Harley, "**Fuzzy Logic Analysis of Environmental threat levels based on selected gas concentrations**", 3rd Mechatronics and Robotics Symposium, CSIR, Pretoria, South Africa, November 2009
8. R. Stopforth, C. Onunka, G. Bright, International Conference on Competitive Manufacturing (COMA) 2010, "**Robots for Search and Rescue Purposes in Urban and Water environments – A Survey and Comparison**", Stellenbosch, South Africa, February 2010
9. R. Stopforth, 25th International Conference of CAD/CAM, Robotics & Factories of the Future Conference, "**CAESAR Robot – Improvements and modifications for USAR robots**", Pretoria, July 2010
10. R. Stopforth, G. Bright, R. Harley, **"Communication and Artificial Intelligence systems used for the CAESAR robot"**, Mobile Robots Navigation, InTech, Vukovar, Croatia, March 2010
11. R. Stopforth, G. Bright, R. Harley, **"Communication Improvements and Gas Danger Analysis used for the CAESAR Robot"**, International Journal of Intelligent Systems Technologies and Applications (IJISTA), Inderscience – accepted