College of Engineering, Design and Physical Sciences
Department of Electronic and Electrical Engineering

Brunel University London

Level 3 BEng Final Year Project Report

[Integrating Object Recognition in Swarm Robotics Management]

<Anonymous Copy>

Abstract

This project aims to create a proof-of-concept system for managing a small group of simplistic robots using a centralised approach for use in possible disaster scenarios. By integrating object detection into this system, resource taxing processes, such as detection and pathfinding, otherwise performed by the robots, can be outsourced to such a system. This will allow the system to control more basic and older robots, which will be beneficial in disaster scenarios due to their expendability and fast production time.

During this project, a python-based software package was self-produced. The application consisted of three custom programs, providing modularity and opportunity for future expandability. A webcam on a stand acted as an aerial camera, representing a drone or satellite, providing the system with image captures of the test area for analyses. Old toy robots, from an early 2000's kit, were converted to use more modern microcontrollers, to be instructed by the system for demonstration purposes.

The resulting system effectively utilised colour detection to recognise and log the positions of objects in the test area from both camera and stored images. The application was also able to perform basic pathfinding operations, including generating routes between a robot and an item of interest or creating a path to an appropriate drop-off location when carrying a payload, achievable without using any sensor inputs from the robot. However, even though digital representations of the robots worked correctly in the virtualised environment, the robots proved to be partially incapable of carrying out the tasks physically due to limitations in the accuracy of movements. Additionally, the resulting reliability of the system was not as high as expected, though this is primarily due to implementation with room for improvements; not the proposed concept.

Table of contents

Chapter 1: Introduction

Within disaster scenarios, there are many situations requiring immediate action; however, these actions commonly prove fatal for any person attempting to do so. An example used throughout this thesis is that of a nuclear disaster, where the situation required urgent intervention; nevertheless, the level of radiation is far too high for humans to even enter the region surrounding the premises.

1.1 – Project Aims

This project aims to produce a prototype system that will show how taking aerial image captures of the disaster zones can be analysed and used to create paths for a group of robots. Through using drones or satellites, the system has the advantage of taking image captures from a safe altitude whilst also covering a large amount of the surface. Additionally, this information can be used to instruct a collection of simple robots to complete crucial tasks whilst being resistant to the effects of the harsh surroundings.

It also aims to show how this system can apply to robots with limited sensing ability and computing resources. For instance, the hardware requirements of the robots can be significantly reduced by outsourcing complex processing to a vastly more powerful and upgradable computer situated in a safe location. Through this system, its compatibility with utilising older and more obsolete robots will help to maximise the availability of usable robots in times of short notice. Overall, more expendable robots with limited computing hardware can be quickly built and shielded for a lower cost.

1.2 – Project Objectives

The objective for this project are:

- To produce a means of detecting and classifying objects in a test area.
- To find a suitable way of taking aerial images for test and demonstration purposes.
- To create a method of logging and visualising detected objects into the system's memory.
- Producing a method of generating paths between robots and items.
- Building a small selection of robots controllable by the system over a network.
- To produce a completed system and combine the separate areas of the project into one software package.

Chapter 2: Literature Review

Robots have many abilities over humans that make them a valuable resource in areas of great danger, such as being designed to have a higher resistance to heat, pressure, radiation, and other environmental factors that can pose considerable risks to the health of individuals.

This research it is aimed to investigate how robots in disaster scenarios have evolved and what role they played. Additionally, it intends to discover how the introduction of swarm robotics could have helped in past scenarios.

2.1 - The use of robotics in disaster scenarios

During the response to natural or man-made disasters, the use of robotics allows operators to carry out tasks unobtainable by humans. Throughout rescue operations, firefighters and alike must both discover and extract survivors whilst also maintaining their own safety. For this reason, in situations where radiation levels, temperatures, risk of structural collapse or the thickness of smoke is too high to send responders inside, another solution is required.

This includes robots such as 'Collossus', the 2.5 feet long and 1,100-pound robot which aided in putting out the Notre Dame Cathedral fire in 2019. According to Crowe (2019), whilst also being resistant to the high temperatures of the fire, it included a water cannon used to spray approximately 660 gallons per minute to help extinguish the flames [1]. This article highlights how robots such as this allow operators to maximise the already limited time to save human life, entering areas too dangerous for human personnel, fighting fires, and bringing down temperatures so that responders can enter soon after.

Visibility is another crucial issue throughout the response to disaster situations. If personnel cannot track the progress of a problem, such as the spread of fire or flood water, it becomes challenging to plan an effective strategy. Additionally, from many perspectives, such as from the ground, this can be hard to achieve, and sending in helicopters may endanger the crews onboard.

The use of drones has many benefits in such events. The same article by Crowe (2019) mentions that firefighters used aerial drones fitted with thermal cameras to monitor the fire's spread remotely. This resource updated the firefighters with critical information to help work out how best to extinguish the blaze, including estimating where the fire may have started [2].

With the minimal setup time involved with such aerial vehicles, drones allow operators to rapidly raise cameras to appropriate heights to take valuable images of the disaster areas with little struggle. Additionally, considering the compact design of drones, it is common for many variations to be stored in specialised travelling cases when not in use. Altogether, this outlines the possibility that first responders carrying such equipment may quickly and easily deploy a series of drones to safely map the affected area rather than

waiting for helicopters to arrive whilst also covering considerable ground meantime. However, the advantages of robots and drones are not limited to solely fire and rescue situations.

2.2 - The Chernobyl accident

Nuclear incidents have devastating effects on the surrounding areas and are one of the most difficult to intervene. In many fires and floods, responders can wear protective suits to protect from harsh environments while carrying out crucial tasks. Where nuclear situations are concerned, it is common that no level of protective equipment is enough to ensure the safety of the personnel. For this reason, robotics may propose a solution where a more expendable and resistive entity may enter the premises to complete such tasks, minimising this exposure to humans.

Most famously, Chernobyl, located near the city of Pripyat in Ukraine, is home to one of the most catastrophic nuclear disasters in history. According to Cholteeva (2020), it is estimated that the area surrounding the exploded reactor, number four, will continue to be dangerously radiated for up to 20,000 years [3].

Unlike Fukushima, Chernobyl and other RBMK reactors lack a containment structure. According to an article by the International Atomic Energy Agency (accessed 2022), this led to the surrounding area being exposed to radioactive materials such as plutonium, iodine, strontium, and others thrown out by the explosion of the unenclosed reactor [4]. However, this was only one of many issues preventing responders from containing the situation.

Within RBMK reactors reside graphite blocks, which according to EDF Energy (Accessed 2022), act as both moderators to maintain a sustained reaction and a safety feature to aid in removing CO2 and house control rods used to limit activity or shut down the reactor [5]. The graphite from the reactor caught fire, releasing toxic and radioactive gasses into the area. This caused more than 30 additional fires and would burn for up to 10 days, mentioned Higginbotham (2006) [6]. The same article states that 400REM is known to be a fatal dose of radiation if exposed for 60 minutes. However, levels exceeded 20,000 roentgen per hour in areas such as the roof and would be lethal for anyone exposed for longer than 48 seconds [7]. For those who unfortunately died in the clean-up, this exposure led to their bodies also becoming highly radiated. This issue required the bodies to be buried within welded closed lead coffins to protect the environment from further contamination.

In order to carry out the clean-up of the area, remote control robots, bulldozers and other machinery were used to remove contaminated material from locations where the radiation levels exceeded too high for people to enter. This success was short-lived as soon the electronics within the machinery would eventually fail due to the extreme radiation levels, resulting in no choice but to start drafting people to complete the tasks. This army of people, otherwise known as the ‘Liquidators’, consisted of reportedly more than three

hundred thousand miners, soldiers and volunteers to clear the area of rubble, build roads and carry out other tasks to contain the disaster, as stated by chernobylgallery.com (accessed 2022) [8]. Figure 2.1, shown below, taken by Igor Kostin, shows some of the 'liquidators' tasked with clearing debris from the rooftop of reactor three [9].

Figure 2.1 - A photograph showing the people labelled "liquidators" removing hazardous debris from the rooftops at Chernobyl.

During this time, robotics was very new in disaster response, with this being one of the first instances within this application. Due to this, radiation effects on robots were an under-researched topic, and minimal quantities of viable robots existed. If the events were different, being better prepared with an abundance of suitably designed robots, additional options other than drafting might have been available.

As mentioned previously, radiation exposure is a major current issue for robots tackling these situations. It is common to find many kinds of robots with cameras, sensors, manipulators, shovels, and so on that would be suitable for the necessary physical tasks. Still, without the radiation resistance required, any success will be short-lived.

According to Abouaf (1998), the 'Pioneer' project was started to create such robots, exploring the design of specialised robots fitted with vision systems for driving and mapping, drilling equipment and an array of other critical tools and sensors. Here, it is mentioned that no robot can withstand the extreme radiation levels within the reactor itself. Outside, exposed robots are subject to effects such as malfunction of vision systems, seizing of mechanisms and total failure of onboard electronics and control systems. Additionally, the paper reveals how one robot deployed to Chernobyl in the 1990s lasted approximately 7 minutes before both electronic and mechanical elements succumbed to failure [10].

One solution discovered by the research team was to remove the electronics from the robot and tether the device to a shielded room containing the control system. The paper similarly states the difficulties of protecting against the different forms of radiation present and examples of materials known to defend each. However, unlike other forms mentioned, neutron radiation has no feasible shielding method against it and still poses a threat to the robot.

Reading this article, it was learned that shielding electronics is vital and an expensive method of ensuring the survivability of a robot in such conditions. Similarly, minimising the electronic hardware onboard reduces the thickness and size requirements of the protective enclosure, allowing for cheaper and more manoeuvrable robots to be produced.

An article by chernobylx.com (2021) emphasises how there were many unknowns related to designing the robots, and for robots that did work, interference with communication and effects of lowering battery capacity at the time of the accident were common [11]. Nevertheless, the article underlines how the robots proved essential to the operation's success even with these issues and the time constraint imposed. Some of the robots presented are still visible today from the Chernobyl Open Air Museum, as shown by Figure 2.2, taken by Zdeněk Mlnařík. This includes 'Mobot- Ч-ХВ2', the red, tracked, crane-like device specifically designed to clear rubble from the rooftops.

Figure 2.2 - Robots used at the Chernobyl incident, displayed in the Chernobyl Open Air Museum.

The article also shares a statement by Valeryi Legasov, a physicist famously involved in the operation, describing how radio-controlled bulldozers, such as "KOMATSU", were used to overcome obstacles and remove contaminants where other robots could not. This machinery was amongst drivers operating manual vehicles with lead-lined cabins, which would later replace the bulldozers from their eventual failure.

It is also important to mention some of the achievements of certain designs too. The article reveals how the 'STR-1' robot, which relates to similar technology for Soviet lunar missions, operated in areas exposed to up to 10,000 roentgen per hour. Its use helped defend up to one thousand lives. Figure 2.3, sourced by image one from the same website, reveals the unit shovelling heavy debris of a rooftop.

Figure 2.3 - Heavy debris shovelled of rooftops by the STR-1 robot.

2.3 - Shakey the robot - An early AI robot

According to sri.com (2019), 'Shakey', an autonomous robot developed by researchers at SRI International from 1966 to 1972, was the first self-deciding mobile AI robot [12]. The image below, Figure 2.4 from SRI.com [13], shows researchers Tom Garvey and Hellen Chan operating Shakey's external computer, which, as stated by history-computer.com (2021), transferred commands to the robot's body through a radio link [14].

History-computer.com also reveals examples of the commands used to control the robot. This includes 'SHAKEY = (GOTO D4)', which instructs the robot to move to a location of 'D4'. More complex commands, such as 'SHAKEY = (PUSH BOX1 = (14.1, 22.7))', utilises the robots 'SRIPS' planning system to autonomously create a route to push a specific box to a new location, avoiding other items that may pose as obstacles. The robot also uses waypoints to achieve a feasible route where no direct path is available. These articles highlight some of the critical abilities required of robots to carry out tasks independently, such as object detection, path planning, obstacle avoidance, and the ability to differentiate one type of object from another.

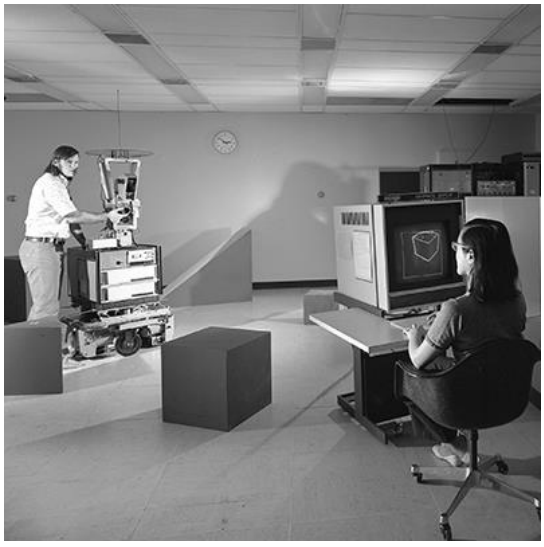


Figure 2.4 - Two researchers operating the robot named Shakey.

2.4 - The Seven Dwarfs AI robots

An archived BBC article “Happy, a Reading University robot” mentions the ‘Seven Dwarfs’ project, developed by Professor Keven Warwick from the Cybernetics Departments of Reading University in 1997. This project demonstrated the ability of one self-learning AI robot named ‘Happy’, situated in the UK, of being able to program another like it in New York through the internet [15].

The “Ultimate Real Robots” Magazine released by Eaglemoss Publications, made in partnership with the Reading University team, formed the ‘Cybot’ robot kit. Each educational magazine would supply a new part to produce the robot or expand the device with further abilities, including radio control, voice control, and eventually allowing a selection of robots and associated devices to produce competing robot football teams. This modular kit could prove a useful resource, offering a chassis and other parts as a suitable foundation for similar robotics projects.

2.5 - Swarm Robotics

The term swarm can be associated with robotics in multiple ways. Firstly, a large collection of networked robots is often referred to as a swarm. Alternatively, the term ‘Swarm Robotics’ is used to describe a decentralised approach of organising a group of robots utilising swarm intelligence.

In a paper titled “Swarm Robotic Behaviors and Current Applications” (Schranz, Umlauft, Sende and Elmenreich, 2020), a swarm of robots is described as a collection of simple autonomous robots connected together to complete complex tasks [16]. Furthermore, the behaviour of the robots relies on the interactions between robots and the surrounding environment, where without a centralised control system, swarm intelligence allows the robots to act as a collective to accomplish goals.

Examples of applications mentioned by the paper include ‘Collective transport’, where a group of robots can work together to move an object to a destination if the payload may be too big or heavy for one robot alone. Additionally, ‘Collective exploration’ uses the advantages of a group of robots to explore an area.

This research provides a great inside into the advantages of using a collection of smaller robots opposed to just one. For example, by using a large selection of robots, an area can be more easily mapped or monitored. Additionally, the use of swarm robots helps provide redundancy, where if one robot were to fail, there would be another to take its place. The individuality of each robot also helps protect against a single point of failure if communications are lost. If a robot such as Shakey were to lose connection to its centralised computer, the robot would not be able to complete its tasks.

This project also raises the question of what mutual benefits may occur when combining both technologies rather than picking one. Swarm intelligent systems prove reliable for surveillance purposes, with greater redundancy and scalability than a centralised approach. However, swarm intelligent robots are not guaranteed to be as organised as those controlled by a central system, such as during a time of crisis.

Take the scenario of using autonomous swarm robots to efficiently map and monitor a location without the need for any central control or human presence. Suppose the readings of a robot was to reveal a critical issue, such as the leaking of a harmful substance or the discovery of previously unknown radiative debris. In that case, immediate intervention may be required to prevent a higher-level catastrophe. If the robot had the ability to notify a centralised system, the commandeering of robots could be used temporarily to divert resources to achieve time-critical objectives. This may include creating optimised routes to quickly deliver robots to the location or carrying out actions listed in instructions prepared for such a scenario. Once the situation is under control, the robots can resume back to their usual mode of operation. If applicable, a robot managing system overseeing the actions and data of autonomous and non-autonomous robots could intervene when appropriate, utilising benefits from both types of the control architecture.

However, within this project, the term swarm is used to refer to a large group of robots available to a central system, not a group of swarm intelligent robots.

2.6 - The project proposal

With the proposed system, a selection of small, simplified robots can be introduced into a disaster scenario to complete clean-up operations in the form of a swarm controlled by a centralised system. If expanded, the system could map the surrounding area and utilise robots for further reconnaissance. This includes directing robots to locations to log temperature and radiation levels.

By having a large quantity of basic robots, swarms can provide a level of redundancy against robots failing that would have proved a critical advantage in scenarios such as the Chernobyl incident. Additionally, by limiting the electronics in each robot, they can be more easily shielded from a range of possible environmental factors, where the centralised system, and those operating it, can be located outside the reach of danger.

This research outlines how the production of more capable robots will prove vital in future catastrophes and the benefits a system such as that proposed can have.

Chapter 3: Methodology

The goal of this project is to achieve a working proof of concept system, consisting of a software package and appropriate accompanying apparatus, capable of managing a group of robots. The expected abilities of the system include detecting objects, planning routes between robots and items, and sending grid commands to the robots.

By producing a centralised approach to managing the robots, complicated processing tasks can be outsourced to a more powerful and upgradeable system. Additionally, more simplified control systems can be implemented in each robot, reducing the amount of shielding required and lowering production costs. In theory, this solution will also allow older robots to be used while protecting newer devices against obsolescence. For example, suppose more computing power is required for future tasks. In that case, the external system can be expanded or upgraded with minimal, or no change required of the robots themselves.

Through integrating object detection into the management system, this task will no longer need to be performed within the robots, reducing the amount of computing and sensory resources required in each robot. Additional to this, an aerial view will allow for a more favourable perspective to be utilised, otherwise unachievable by the robots directly. Altogether this will result in more optimised pathing of the robots, as all objects detected in the visible area will be considered, opposed to only those detectable by the robot at any instance.

As displayed in Figure 3.1 below, a robot relying solely on its onboard object detection system is limited in its scope, only seeing objects and pathways in its immediate area. In contrast, a system with a bird's eye view can consider all items and obstacles with the addition of a more extensive search area for calculating paths. Furthermore, whilst planning a complete route to an object, both the positions of the robot and object are known. However, when relying on the robot itself, the robot will have to avoid obstacles while attempting to find the object without knowing its location.

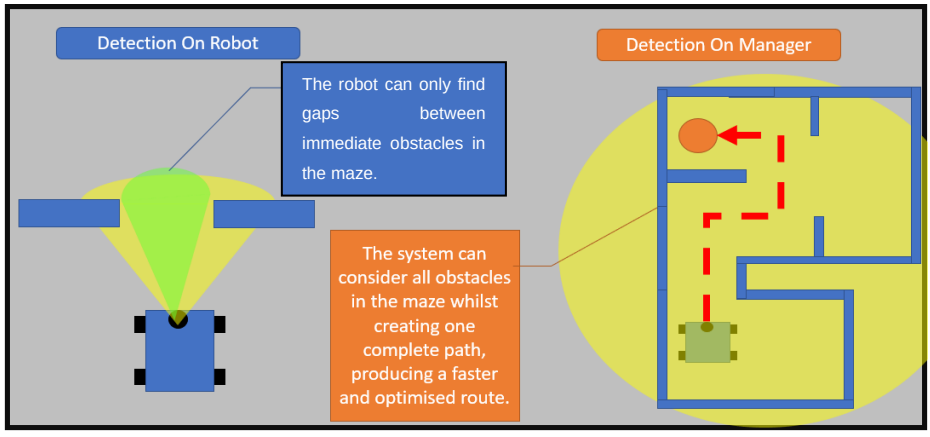


Figure 3.1 - A graphical depiction of the capabilities of robot-based object detection to the proposed system-based detection.

3.1 - The project plan overview

A Gantt chart (Appendix 1) was produced monitor the progress of the project in areas of research, implementation, and testing. By colour coding the Gantt chart by completion status, such as green for accomplished and red for behind schedule, it provides an additional indication on area that need priority as time goes on.

The software package will be produced through the combination of three smaller programs. Through this, a webcam suspended by a stand will take image captures of the below surface to be analysed by an object detection program. Detected objects will be stored into lists corresponding to the item's classification. The second program will read the lists of detected objects, tasked with plotting the items onto a virtualised environment, such as a 2D grid, where paths between robots and locations can be generated. Thirdly, the robot communications program will detect robots available on the network and allow the operator to select and configure desired robots to form the robot fleet. For each robot in the fleet, the robot communications program will send commands containing grid instructions of a location for the robot to interact with and the method of interaction. These options include going to this location, picking up an object, or placing the current payload at this destination. The following Figure, 3.2, depicts a possible solution for how these system areas interact.

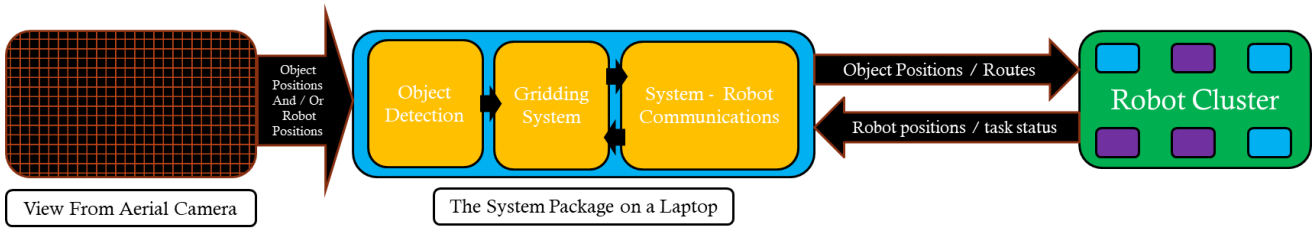


Figure 3.2 - A diagram showing a possible structure of the proposed system.

A variety of scenarios will be presented to test the system with its response monitored. Through implementing a load-from-file feature, the software can be provided with pre-produced images of situations that cannot be easily replicated in the real world. Similarly, if introduced in a production-level version of the system, an operator could test run specific disaster scenarios before the robots arrive on location to ensure they will complete the objectives and to discover any potential flaws early. Through this, the system can be examined for how well it detects objects, plans paths and communicates with the robots.

The decided steps involved to implement this system and accomplish the objectives are as follows:

- To research and create a form of object detection in python
- To find a method of mounting a webcam with an aerial view
- To develop a strategy of logging and visualising detected objects
- Producing a method of path generation
- Research and implement networking on Arduino development boards.
- Investigate a way to integrate a newer microcontroller into an older robot
- To achieve communication between the robots and the python application
- To combine the separate programs into one package

3.2 - Object detection

A form of object detection will be required to detect and classify objects in the test area. Examples of recognising items include training a neural network, implementing colour detection or scanning QR codes. Training an AI would prove to be a very effective solution if successful, but it will also be a complicated and time-consuming part of the project to produce. Through implementing an AI-based object detection, it is common to represent detections by drawing a box around the item, often colour coordinated or labelled by the classification of the object. A simplified depiction follows as figure 3.3.

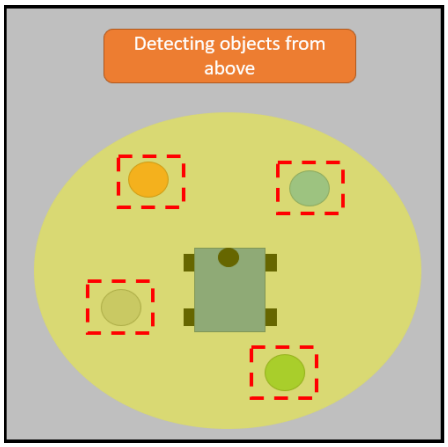


Figure 3.3 - A representation of detecting items surrounding a robot.

3.3 - Aerial image capturing

A standard USB camera can be suspended from above and controlled by a python application to achieve aerial image captures of the test area. However, a tripod may not provide enough reach to adequately cover the area, though an adapted microphone stand will as demonstrated by Figure 3.4. One drawback would be the size range of the test area, as the floor coverage will be limited by the height and reach of the microphone arm. A 3x3 grid system would likely be a suitable size for these reasons.

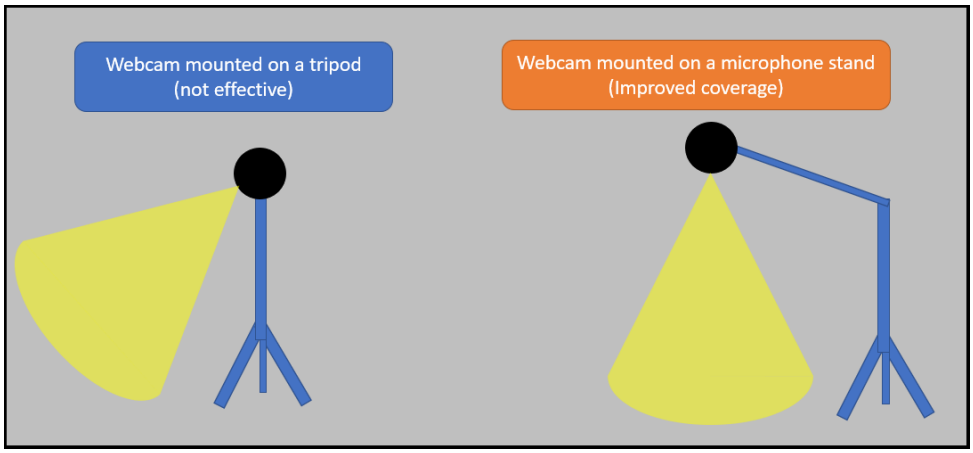


Figure 3.4 - An illustration comparing the effective floor coverage achievable by a camera mounted on a tripod to a camera fixed to a microphone stand.

3.4 - Logging items into memory

Creating a 2D grid would be an easy and effective way of visualising the locations of objects discovered by the system and stored into lists. Displaying a grid would also be useful for an operator that wants to monitor robots remotely or test how the robots will perform when given a specific scenario.

Other methods may be to render 3D representations of objects in a virtual environment. However, this will be computationally taxing and vastly more complex to implement than placing ASCII symbols onto a 2D grid. Additionally, grid spaces can be flagged as containing an item, resulting in the system knowing not to create a path through these grid locations; through this, the exact size and shape of the objects is not critical information and does not need to be calculated.

3.5 - Path generation

Path generation is an effective way of enabling a robot to reach a destination to minimise mistakes. A planned route considers objects in the surrounding area, producing a safe means of passage without the robot needing to guess where to go. There is an abundance of possible algorithms to achieve this; nevertheless, pre-made pathing algorithms may be difficult to implement. Considering the size of the project, it is intended to create a simple custom-designed method to save time.

The chosen method will create simple two-stage paths with only one major change in direction. The algorithm will start by directing the robot along the X-axis until the robot is aligned with the object. After this, the algorithm will similarly plot movements along the Y-axis until the value of that axis is also reached. In the instance where an obstacle is blocking the original path, the order of axis corrections can be swapped, producing an alternative route. A diagram depicting this is shown below in Figure 3.5.

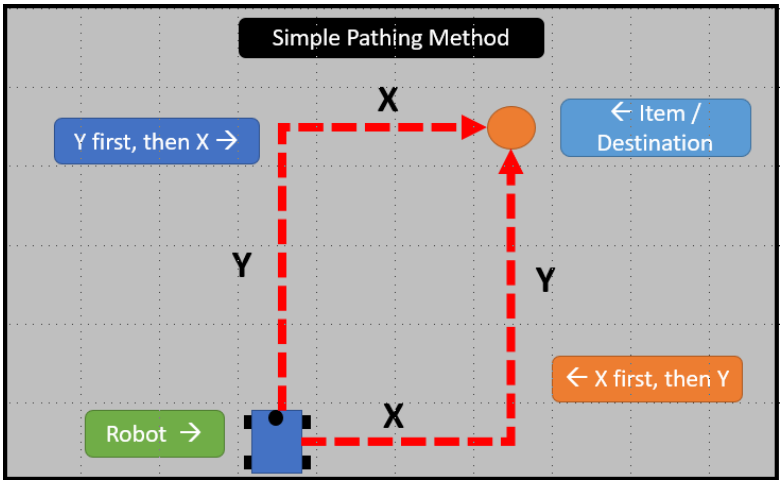


Figure 3.5 - A representation of the simple path generation method proposed.

A method of prioritising or queuing items is required to optimise the system as the number of robots and items increase. For example, selecting the nearest items and drop-off locations for a robot can help limit unnecessary travel. With robots only interacting with objects in their immediate surroundings until the number of outstanding tasks has diminished, this prevents robots spread amongst a large area from interfering with other robots' tasks.

An equation was produced to generate a score based on distance. When assigned to an object, this score will provide the system with reference to how far one object is from another. A bias on the Y-axis difference is introduced to limit the likelihood of objects being assigned the same score, weighted twice that of a distance in the X-axis. Additionally, when many objects are registered on the grid, the aim is that robots would clean up the items on their current row before progressing to other levels. This will help evenly spread their efforts instead of using a random roulette method of choosing a nearby object. The equation to generate the score is as follows:

Distance Score = $|X_{robot} - X_{item}| + 2| Y_{robot} - Y_{item}|$

3.6 - Creating the robots

Due to the modularity of the Cybot toy, released by Eaglemoss Publications in 2001, it will make a suitable foundation for the demo robots, becoming the intended chassis to use. Additionally, parts can be easily sourced online through buying second-hand bundles of unfinished robot kits, with costs typically ranging between £20 to £60 depending on the quantity and selection of the parts included.

Implementing a more modern microcontroller would be easy considering the simplicity of the motor driving circuits and no need to connect to the pre-existing sensors of the robot. A microcontroller, such as the Arduino Nano development board, can be integrated within the robot by removing the top layers of circuitry and directly interfacing with the motor driving circuits. A 3D printed adapter can be attached to the chassis using the pre-existing brass stand-offs, of which the Arduino inserted into a breakout board can

be fitted onto the other side. A breakout board will be useful as it provides a safe way to fix the microcontroller in place whilst including terminals for effortlessly connecting wires to the device, held in place with a screw, without the need of soldering. This concept is represented in Figure 3.6

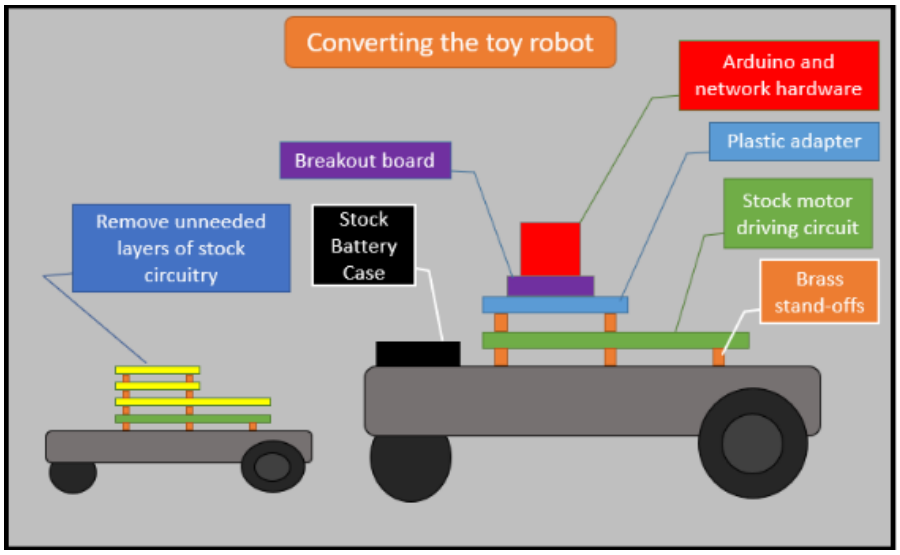


Figure 3.6 - A representation on how the conversion of the robots is planned.

The Arduino Nano development board is the proposed controller due to its low power, low cost, concise design, and its range of supported network hardware. A compatible Wi-Fi or Ethernet adapter will be chosen to allow communication between the robot and the PC application. Python has a greatly supported and documented network socket library, with a vast selection of UDP and TCP message transfer tutorials. Alternative means of connection may be through infrared or Bluetooth connections; however, it would be more challenging than connecting multiple devices through a network router or switch.

This will allow for a small selection of cheap demo robots to test the system. However, being a toy, the robot chosen does not contain any current means of motor feedback which may prove challenging to produce accurate movements without alterations. For example, a compass & accelerometer sensor package can monitor movements such as turns, allowing the robot to register and correct any under or over-shooting.

3.7 - Producing the software package

The three separate programs, object detection, grid manager, and the robot communications program, will be combined into a single software package to form the 'robot manager'. The grid manager can form the application's foundation, importing the other applications to interface with the outside world. The object detection will provide the system with inputs with the communication program allowing the system to output to the robots. By initially constructing three separate programs, one for each significant area of the project, these areas can be created and tested more easily. However, design considerations for the later integration are vital to make this combination successful.

Appendix 2, an initial overview flow chart, depicts how the various processes within the applications will collectively form the complete system.

3.8 - Producing the test area

To create a test area, a place big enough for at least a 3x3 square grid is required. On campus there are various locations open to students which use tilled floors.

Figure 3.7 shows a camera on a stand overlooking the floor at one of these locations. The cardboard test square can be used to help align the camera and set a correct angle. The cardboard square is of the minimum size required for each grid space, as visible in Figure 3.8, the 3x3 floor tile space chosen is suitable for these experiments.

This will allow the system to be presented with test scenarios, placing items on the ground with a robot and observing for whether the objectives were completed.

Figure 3.7 – A camera setup overlooking the floor of a chosen student study area.

Figure 3.8 – Floor tiles that exceed the minimum size required for the tests, being a suitable candidate for a test location.

Chapter 4: Testing the system

4.1 - Adding objects to the grid

The grid manager acted as a foundation for the system and was the first part created and tested. Python lists, each relating to a particular object category, were produced and supplied with test data. The placement of objects on the virtualised grid was verified by comparing the actual positions displayed to those given. Each item entry consists of three parameters: the item's X-axis value, Y-axis value, and a character key to show the classification. Figure 4.1 shows the four test entries manually supplied to the system.

```
# Initial Placments
robots = [[10,10,'R']]

bads = [[5,2,'B'],
        [4,5,'B']]

goods = [[2,2,'G']]
```

Figure 4.1 - Object locations of test items presented to the system.

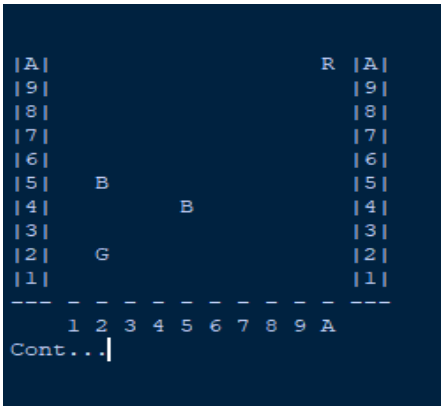


Figure 4.2 - Items successfully plotted on the grid display.

As captured in Figure 4.2, it was found that all objects were both correctly displayed and located. Repeating the test, using various combinations, ensuring that all grid corners worked effectively, proved that the current grid system worked as expected.

4.2 - Calculating the closest objects

First, the system was tested to ensure that evenly spaced objects would generate the same score when appropriate. To achieve this, the equation was presented with the locations of four items, with the robot at the centre and the score calculated by hand. Figure 4.3 shows a graphical representation along with the results table. The results table shows that all items produced the same score as expected.

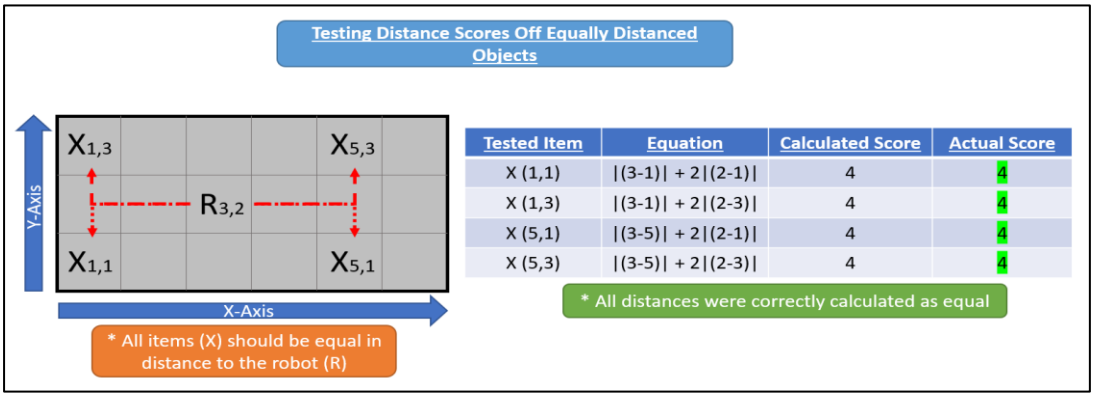
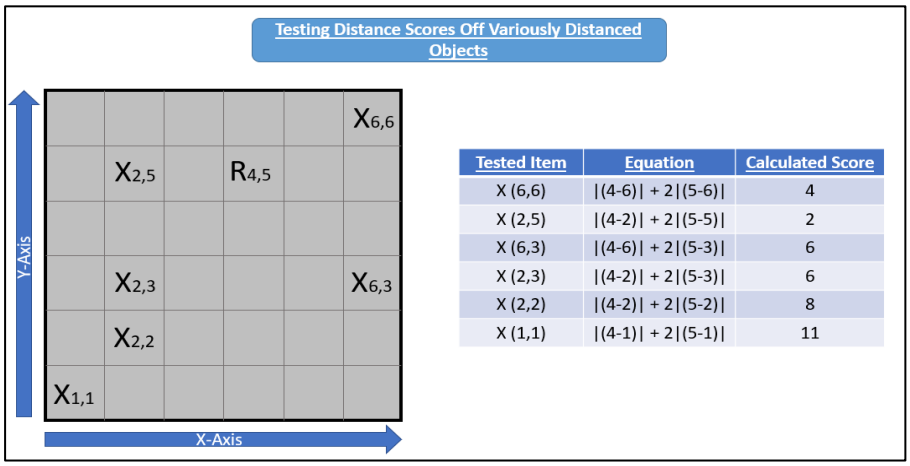


Figure 4.3 - A table showing the testing of the distance score equation.

The second test consisted of a random selection of item locations. The weighting feature was further assessed by trying various distances from the robot on both the X and Y axes. A graphical representation of this test is visible in Figure 4.4.



Tested Item	Equation	Calculated Score
X (6,6)	|(4-6)| + 2|(5-6)|	4
X (2,5)	|(4-2)| + 2|(5-5)|	2
X (6,3)	|(4-6)| + 2|(5-3)|	6
X (2,3)	|(4-2)| + 2|(5-3)|	6
X (2,2)	|(4-2)| + 2|(5-2)|	8
X (1,1)	|(4-1)| + 2|(5-1)|	11

Figure 4.4 - A table showing the testing of the distance score equation for randomly placed objects.

Each item has a score assigned in the rightest column of the above table. It can be seen that the weighting feature on the Y-axis differences is working. Objects X(6,6) is only one grid space extra of travel from X(2,5), though the score of X(6,6) is double that of X(2,5). Similarly, X(2,2) has a score two points lower than X(2,3) where again, only one grid space of difference is present. The items can be listed from lowest distance score to highest, as visible by the table in Figure 4.5.

This is evidence that the equation generates a lower value for objects in the robot's immediate proximity and higher as objects span further distances, more significantly when there is a greater difference on the Y-axis.

Tested (Closest to Furthest)	Actual (Closest to Furthest)	Pass / Fail
X (2,5)	X (2,5)	Pass
X (6,6)	X (6,6)	Pass
X (6,3) & X (2,3)	X (6,3) & X (2,3)	Pass
X (2,2)	X (2,2)	Pass
X (1,1)	X (1,1)	Pass

Figure 4.5 - The distance of the items correctly ordered from closest to furthest.

4.3 - Pathing algorithm

A pathing algorithm was produced to calculate simplistic routes for robots to arrive at a destination. The created paths only consist of one main turn, with simple corrections on the X-axis, followed by adjustments on the Y-axis. The aim was to provide a simple yet effective way of planning a path between objects for basic scenarios.

A significant test was to have two virtualised representations of robots, followed by a series of objects randomly placed on the grid. The algorithm was then observed to ensure each robot selected the closest appropriate object and created a series of movements to be stored in a list and displayed on the grid. Two categories of robots were created, collector (C) and remover (R). The remover type is to seek objects presumed to be dangerous or 'bad' (B), where alternatively, the collector type is to retrieve the safe objects, referred to as 'good' (G). Figure 4.6 displays the test scenario presented to the python program.

```
Created route 1  [[8, 9, 1], [7, 9, 1], [6, 9, 1], [5, 9, 1], [5, 8, 1], [5, 7, 1], [5, 6, 1]]
Created route 2  [[5, 2, 2], [6, 2, 2], [6, 3, 2], [6, 4, 2], [6, 5, 2]]
Steps route 1 :  7
Steps route 2 :  5


        NEXT STEP
|A|                       |A|
|9|         1 1 1 1 R     |9|
|8|         1             |8|
|7|         1         G   |7|
|6|         1 G           |6|
|5|         B 2           |5|
|4|     B     2           |4|
|3|           2           |3|
|2|       C 2 2           |2|
|1|                       |1|
--- - - - - - - - - - - ---
    1 2 3 4 5 6 7 8 9 A
```

Figure 4.6 - The system successfully creating viable paths for each robot to take.

By observation, the algorithm successfully created a path for both robots, selecting the closest object to each robot and placing numeric characters representing the ID number of the robot, indicating that the grid spaces are reserved for the path. Additionally, a list of movements was generated for each robot, with each element storing data in the form of destination X and Y and the robot's ID.

By following each step of the path, it is apparent that the sequence is in the correct order and the test was a success. The path stops one space before the destination item due to the robot needing to be facing the object to pick it up. Numerous test variations were attempted to ensure the reliability of the algorithm. However, the algorithm cannot re-route the robot if an obstruction is present, though a blocked path is detectable, as visible by Figure 4.7 showing the route of robot 'B' stopping as the obstacle 'X' blocks its way.

```
        NEXT STEP
|A|                       |A|
|9|         1 1 1 1 R     |9|
|8|         1             |8|
|7|         1         G   |7|
|6|         1 G           |6|
|5|         B             |5|
|4|     B     X           |4|
|3|           2           |3|
|2|       C 2 2           |2|
|1|                       |1|
--- - - - - - - - - - - ---
    1 2 3 4 5 6 7 8 9 A
```

Figure 4.7 - The system detecting an obstacle in robot C's path.

4.4 - Transmitting instructions to the robots

The system communicated to the robots through UDP messages. Initially, the robot communication program was designed to test the sending of simple movement messages. This was to investigate whether the system was able to transmit messages and that the robot could properly receive them. Figure 4.8 shows the messages sent, displayed by the blue python window, and then received and decoded by the robot in the overlapping white window. Figure 4.8 also shows that all messages were received by the robot in the correct order.

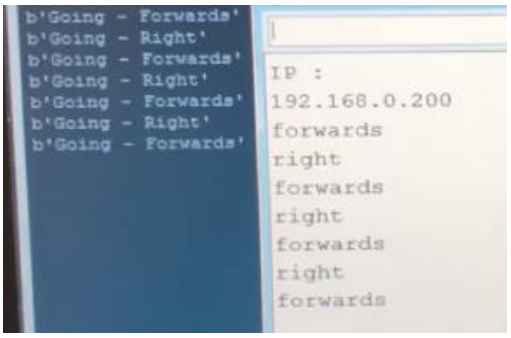


Figure 4.8 - The successful transfer of movement instructions as shown through console messages.

As the system was expanded, communicating to more robots, and sending configuration settings was required. A syntax of different categories of messages was produced, signified by the 'mode' value. Listed below are the types of messages used by the system.

- IP Verification – Probes for a response to ensure the robot is still available on the network
- Robot configuration – To initialise the robot with its type, ID value, starting position and compass direction.
- Grid instructions – Signifying a grid location and what action to take (Go-To, Pickup item, Drop-Off item).

The system was tested to verify that a robot was present on a network and then send the robot a configuration message. If a response was present for both, the system should repeat the process for the next robot.

Figure 4.9 - The system correctly discovering and configuring two robots.

Figure 4.9 shows the results of this test in the form of the python consoles output. An attempt counter stored how many times the message was sent until a reply from the robot was received. If a failure to detect a response occurred for three consecutive attempts, the robot was deemed unavailable to communicate to. In this case, all responses occurred after the first attempt, being the whole message resent back to the system. This method results in a larger payload size to a simple acknowledgement, which may be inefficient, but an easy and effective way to ensure the whole message was correctly received.

The test was successful, as evident by the system accomplishing the configuration of the first robot and progressing and succeeding the setup of the second.

4.5 - Robots receiving messages

It was essential to check that the data being received by the robot was unpacked correctly and understood. The configuration ability of the robots was tested by printing out the robot's settings to the serial monitor after a configuration message was received.

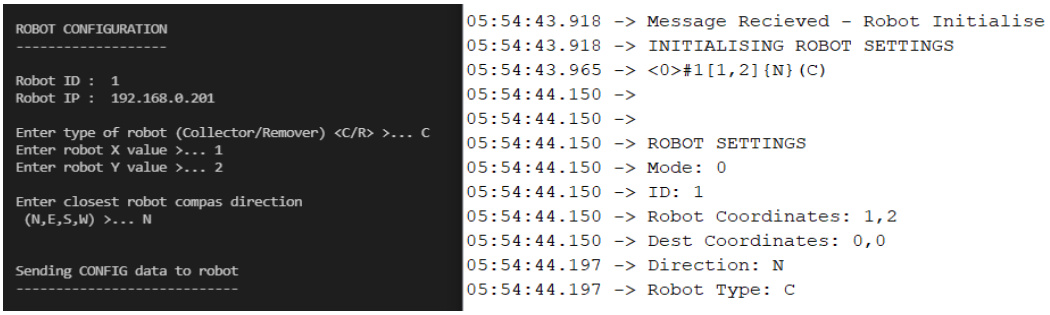


Figure 4.10 - The robot correctly being configured as evident by the data being shown successfully decoded in the white window.

Figure 4.10 displays the results of initialising the robot with custom values using the finished system. Both the black python console of the system and the white console of the robot's serial monitor show the robot is set up with the ID of '1', the starting coordinates of X=1 and Y=2, and the robot type being that of 'C' representing collector. The matching of the data displayed shows the test was a success.

4.6 - Robot moving algorithm

The robot is commanded with grid instructions, as mentioned previously. These messages consist of a location and an action, as decided by the system. The three options produced are going to the location, picking up something, or dropping off an item at the destination.

Two algorithms were produced. One algorithm allows the robot to calculate the relation of the destination's position to its own such as whether the target is to the left or the right of the robot. The second algorithm is tasked to calculate the exact movements to reach their position. For example, it would calculate how many times it would have to rotate to face the required direction. Unit tests in python were performed against a series of scenarios to test the algorithm more easily before it was implemented within the robot.

The first test was of the position relation algorithm. Figure 4.11 shows lists of several coordinate entries for the robot and an object of interest. The algorithm was assessed by iterating through the lists and comparing the produced outputs to the known answers.

```
robotPos = [[2,2],[2,2],[2,2],[2,2]]
objectPos = [[2,3],[2,1],[1,2],[3,2]]
knownAnswers = ["Above","Below","Left","Right"]
testPositions = 4
```

Figure 4.11 - Test coordinates are supplied to the algorithm.

The results in Figure 4.12 show the output of each test and the corresponding correct answer. This test was a success, with every scenario passing. The grid locations provided to the robot will always be adjacent to each side of the robot's current position; this allows testing to be more straightforward as each test will only ever have four combinations.

```
Test Number:  1  Of:  4
CALCULATED : Object to robot position: Above
ACTUAL : Object to robot position: Above

Test Number:  2  Of:  4
CALCULATED : Object to robot position: Below
ACTUAL : Object to robot position: Below

Test Number:  3  Of:  4
CALCULATED : Object to robot position: Left
ACTUAL : Object to robot position: Left

Test Number:  4  Of:  4
CALCULATED : Object to robot position: Right
ACTUAL : Object to robot position: Right


TEST COMPLETE
```

Figure 4.12 - The algorithm correctly calculating the position of items to the robot's location.

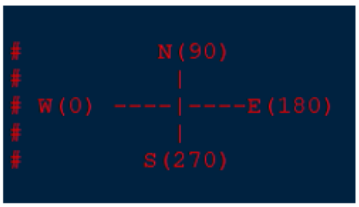


Figure 4.13 – A diagram illustrating compass direction and the corresponding angular values.

The robot's direction is logged using degrees, with 90 degrees representing facing 'North' of the test area, as depicted in Figure 4.13.

To test the turning algorithm, a list of outputs of the previous algorithm was converted into angles, where the output of the number and direction of turns was compared to known answers.

```
BEFORE: Test direction: Above, Current direction: 90, Desired direction 90

AFTER: Test direction: Above, Current direction: 90, Desired direction 90


Calculated Movements:
Press 'Enter' To Continue...

BEFORE: Test direction: Below, Current direction: 90, Desired direction 270
DURING: Current direction: 90, Desired direction 270
DURING: Current direction: 180, Desired direction 270

AFTER: Test direction: Below, Current direction: 270, Desired direction 270


Calculated Movements:
Turn-Right
Turn-Right
Press 'Enter' To Continue...
```

Figure 4.14 - The algorithm correctly calculating the number of turns required.

The results of two of these tests are shown in Figure 4.14. The algorithm can correctly calculate the number of turns required to make the robot face the direction to drive towards. In the first scenario, the robot is already facing the intended direction; no turning is required. In the second scenario, the location is 180 degrees clockwise to the robot's facing direction, resulting in two 90 degrees right-hand turns.

To test the algorithm within the robots, a robot was placed on the floor with a test script running a sequence of similar scenarios. Each movement ended with a manufactured delay, meaning each turn can be easily counted and compared to the answers written down. These tests were also successful. The intermediate step of producing the algorithm in python proved a valuable resource to create and test a prototype easily. Additionally, it was faster to make and verify small changes as the console text was printed quickly without waiting for the code to upload or the robot to move.

4.7 - Object detection

To achieve the detection of objects in the test area, a method of analysing and recognising items from image captures was required. Initially, this was attempted using the OpenCV Python Object Detection library. However, the implementation was found to be problematic at times due to false-positive results, and the changes to the blocks led to constant retraining of the AI.

Two painted blocks were presented to test the system for the system to detect, embedded with magnets for the robot to use pick up. Due to the simplicity of the blocks, each being a different colour, a colour detection method was approached. Due to the system's design, this program could later be changed to utilising a different method by importing an alternative program in the future.

The produced program uses OpenCV's colour detection library, designed to detect objects of colour within set colour value thresholds. The image below, figure 4.15, captured from a YouTube video by a content creator known as 'CreepyD', shows an HSV colour chart from the tutorial on which this code was inspired. This chart was used to find appropriate colour value ranges equivalent to the blocks, to enter the threshold values [17].

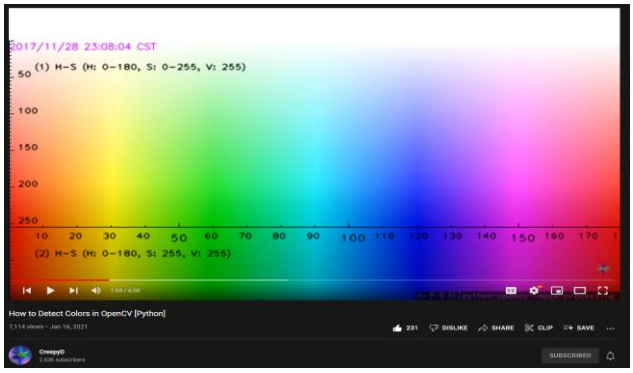


Figure 4.15 - The Open-CV HSV colour chart as found from YouTube.

Figure 4.16 shows the output of the mask layer, blocking inputs not within the specified ranges. This method proved very successful as even though the block's centre hole from the embedded magnet is clearly visible, the algorithm was still successfully able to detect the object as represented by the red detection box of Figure 4.17.

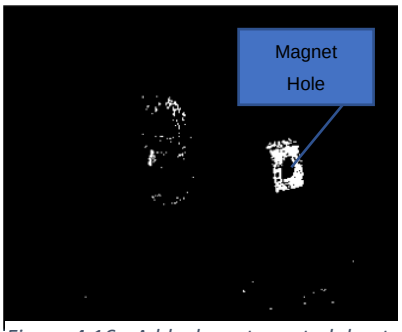


Figure 4.16 - A black spot created due to the magnet capsule.

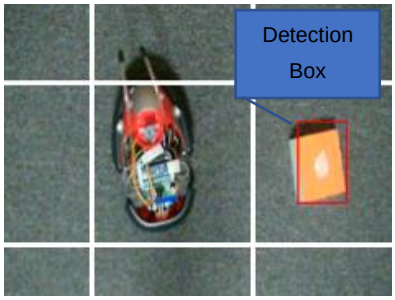


Figure 4.17 - The orange block correctly detected by the system.

Eventually, the system was expanded and tested against two objects, one of each type of item. The program was also changed to 'snap' detections into grid spaces and log them into memory by category. These objects were tied to dental floss, pulled, with detected locations monitored. An image of this test can be seen in Figure 4.18, where detections of 'good' orange objects are on the left side grid and 'bad' purple objects on the right.

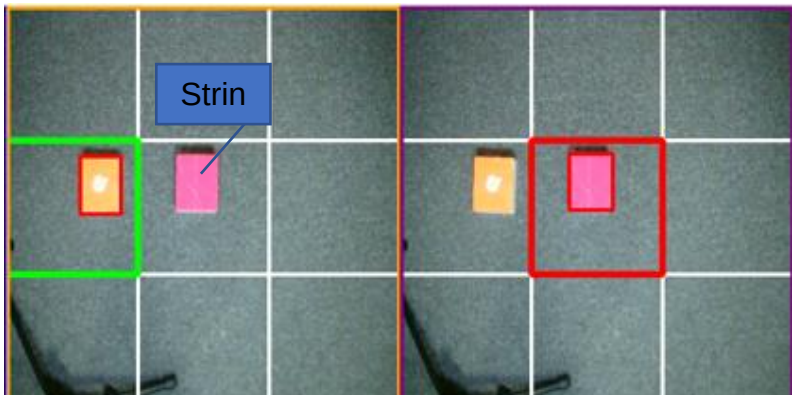


Figure 4.18 - The software correctly detecting both categories of objects.

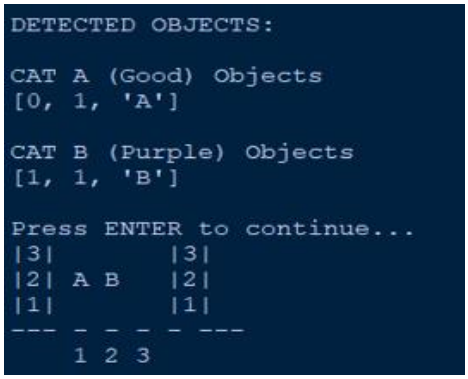


Figure 4.19 – Both detected objects correctly represented on the grid display.

The results of this test can be seen by the snapshot in Figure 4.19. Here it can be seen that the two objects have been successfully detected, located, and classified. Additionally, the results were loaded onto the grid, reading the automatically updated objects lists that were previously entered manually.

4.8 - Loading scenarios by file

The testing of the complete system is not limited to using the aerial camera and placing physical objects. In fact, no physical test area is strictly required. By powering the robots by the laptop directly, isolating power from the drive circuits, the system can communicate with the robots as usual without any real movements being made. Additionally, by supplying the system with pre-produced images of the test area, or graphical representations, the application can still be assessed on its object detection capabilities and retain the ability to log, virtualise, route for, and instruct the robots as it would in a real-world test. This test is crucial as it requires all virtual parts of the project to be functional.

In this scenario, a nuclear reactor has exploded, leaving radioactive rubble on the grounds of the surrounding area, represented by pink squares. A dotted red line represents a threshold of fatal radioactivity levels. Behind the danger threshold resides orange cubes, signifying equipment required to put out a fire. To accomplish this scenario, the system is expected to dispose of all the dangerous material into dug pits in the ground (presented as '#'), away from where firefighters will be working. Additionally, the system needs to bring all firefighting equipment to the correct locations (displayed as '$'), ready for use when the area is made safe enough for humans to enter. Image 4.20 shows an annotated view of this scenario.

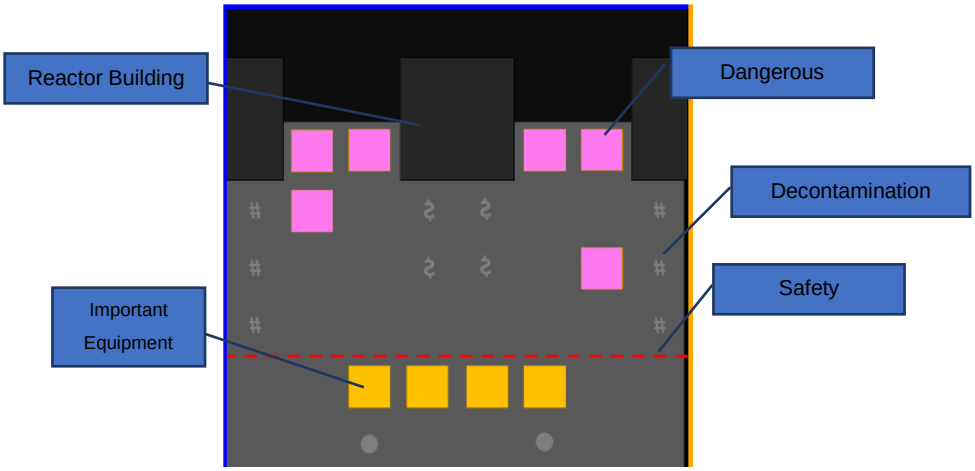


Figure 4.20 - An annotated depiction of the pre-loaded scenario.

First, the user is presented with an options menu for selecting the mode of operation. As seen in Figure 4.21, the second option was chosen to load a scenario from a file.

```
|#################################|
|      Select Mode Of Operation   |
|#################################|
|  1  | Normal Operation (Full Sys) |
|-----|-----------------------------|
|  2  | Run Pre-Loaded Scenario     |
|-----|-----------------------------|

Please Select and option >... 2
```

Figure 4.21 - The applications main menu displayed.

```
Operation Structure Tree
========================

START
 (0) : Select mode of operation
  |
 (#) : Configure grid settings
  |
 (2) : Discover avaliable robots on the network
  |
 (3) : Select and configure robot fleet
  |
  |
-> <-> : --Entering Automated Mode--
|   \
|    (4) : Image sample and object detection
|     |
|     |
^    (5) : Store objects into list
|     \
|    -> <-> : For each robot in fleet
|    |  |
^    |  (6) : Display objects on the grid
|    |  |
|    |  (7) : Select grid item for robot to interat with
|    ^  |
|    |  (8) : Create patht to item
^    |  |
|    |  (9) : Send grid intruction to robot
|    |  |
|    \ (10) : Update state of objects in memory
|     \ |
^      #
|      |
----<----- * Once all Robots are updated
```

Figure 4.22 - A displayed diagram of the systems internal processes.

From here, the program's structure is shown through a tree-like diagram, shown in Figure 4.22. This diagram is used to notify the operator of where in the program they currently are.

The program automatically loads grid settings and then begins to scan for robots on the network, as evident by Figures 4.23 and 4.24.

Figure 4.23 - Gid configuration successfully loaded from file.

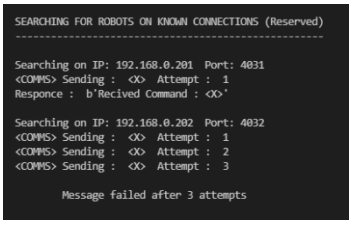


Figure 4.24 - The system scanning the network for available robots.

At this point, the setup of the system is finalised. Figure 4.25 shows the warning message notifying that the automatic management of the robots will be initiated after this point.

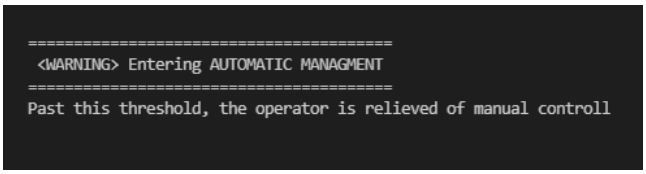


Figure 4.25 - A warning message displayed to the user.

The system successfully detects and classifies all objects in the provided scenario and is displayed on the grid, as evident in Figure 4.26.

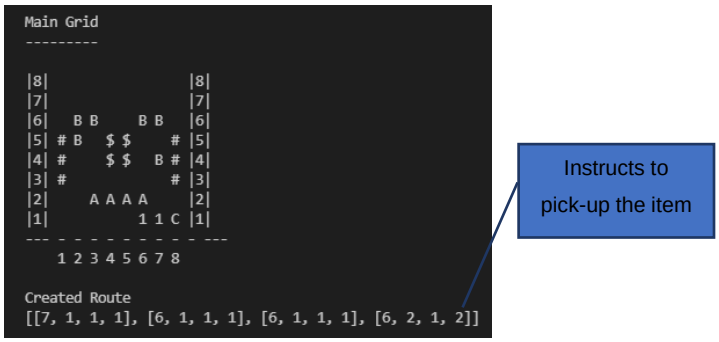


Figure 4.26 - The system successfully detecting all the objects and creates a path to one of the items.

A path is generated and sent to the sole robot in the system, situated at starting location (8,1). After following this path, the robot arrives and picks up the item, and delivers the item to a location as shown in Figure 4.27.

Figure 4.27 - The robot arriving at the drop-off location.

Unfortunately, unlike the closest object directly under the robot in Figure 4.27, it returns to the same item location, as shown by Figure 4.28. It can also be seen that the drop-off location has now been flagged as used. Additionally, the pathing algorithm mistakenly planned through objects in order to get to the next drop-off location.

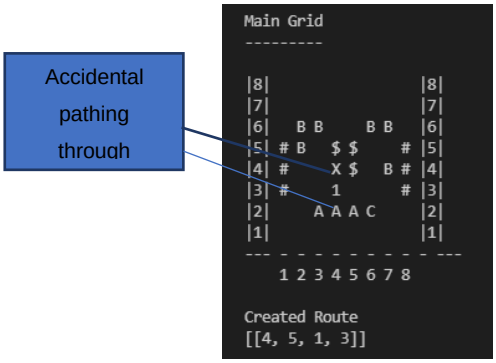


Figure 4.28 - The system accidently planning a path through obstacles.

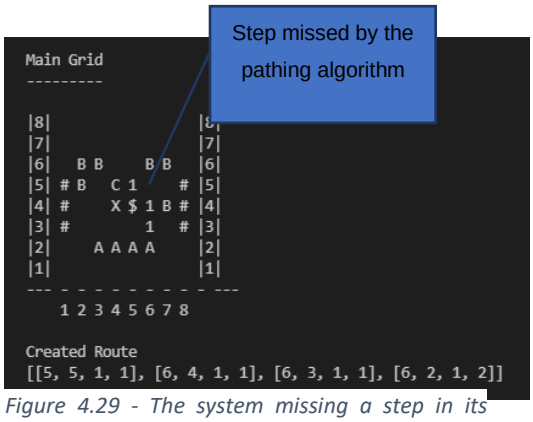


Figure 4.29 - The system missing a step in its returning path.

Another pathing error is revealed in Figure 4.29, by a missed step in the corner of the path.

In conclusion, this test was a failure, however, not entirely. Communication between the robot and system was reliable, and all objects were successfully detected. Unfortunately, issues with the pathing algorithm occurred, not allowing the system to complete the scenario objectives. As the system continues to detect objects from the file, objects otherwise physically moved by the robots were re-added to the grid, also making it impossible to move all possible items.

4.9 - Full system practical tests

The full system test was easy to construct. The camera apparatus was assembled, with the webcam facing the floor with a robot and two test blocks placed below. Set up of the system was similar to before; however, option 1 of the main menu was selected, and all information was now required to be entered manually.

In this scenario, the robot is to pick up the orange block and deliver it to a drop-off location. An image of the test area is shown below in Figure 4.30. This also includes a cardboard test square, situated at the drop-off location, used to help set up the correct camera position.

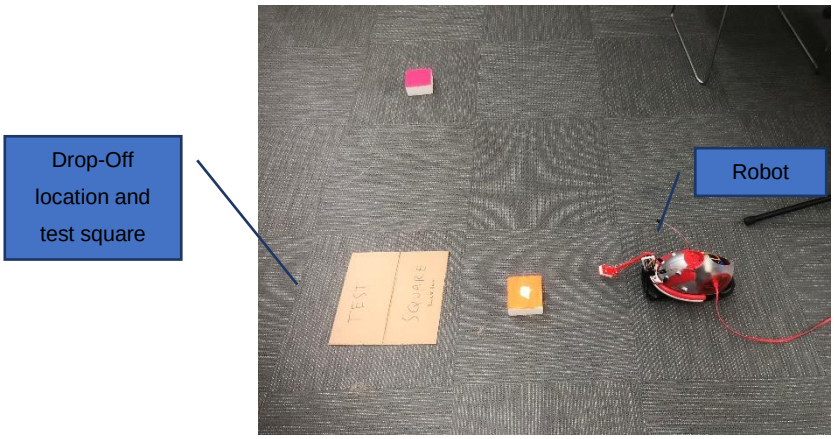


Figure 4.30 - An image showing the items in the test area.

The orange object was successfully detected; however, the pink item was not. This can be seen in Figures 4.31 and 4.32.

```
CAT A (Good) Objects
[1, 2, 'A']

CAT B (Purple) Objects


Discovered Objects
------------------

|3| $     |3|
|2| A     |2|
|1| C     |1|
--- - - - - ---
    1 2 3
```

Figure 4.31 - The orange object correctly detected.

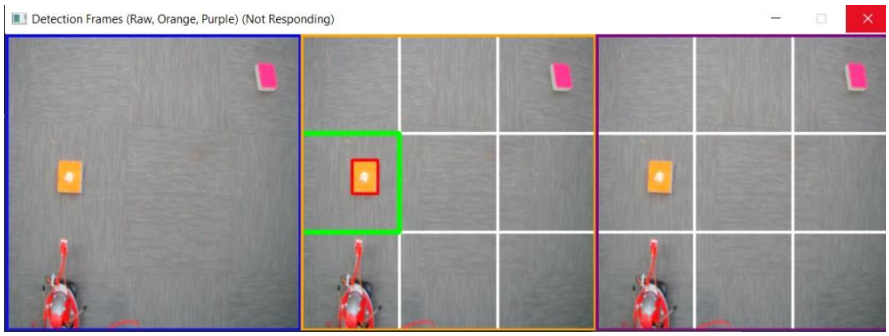


Figure 4.32 - The orange object correctly detected but the pink block not.

The robot successfully arrived at the item's location; Unfortunately, as revealed in Figure 4.33, the robot slightly turned and did not align with the magnetic pickup of the block.

Figure 4.33 - A picture showing the crane not aligned with the blocks magnetic centre.

Surprisingly, the robot still managed to place the item in the correct location, Figure 4.34, though this was accidental by the robot pushing the block rather than holding it with its magnetic crane.

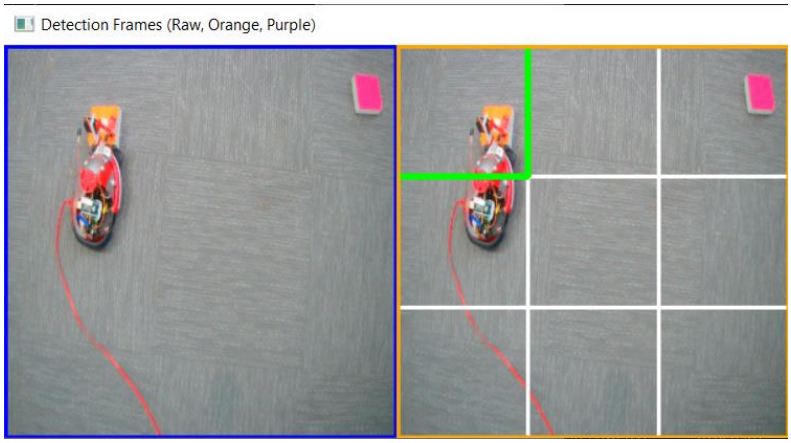


Figure 4.34 - The orange block delivered to the destination.

In conclusion, the test failed again, but not entirely. The system did not detect the pink block, and the robot could not effectively lift the block from the ground. However, the block was delivered to the correct location, and the pathing algorithm and communications worked as intended. A closer look at robot one is displayed in figure 4.35.

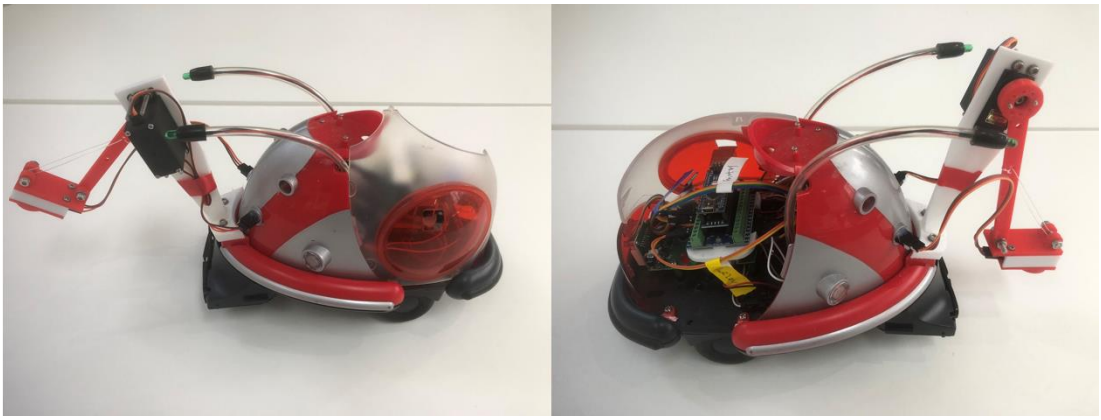


Figure 4.35 - Side views of converted robot one of two, displaying the produced 3D printed arm attachment and revealing the electronics inside.

4.10 - Testing robot movements

When performing movements, such as driving straight, the robot's distance would commonly be correct. However, Figure 4.34 from the practical scenario test shows that the robot can slightly veer in direction.

By testing the robots with various numbers of turns, it was aimed to estimate of accuracy and repeatability of the movements. As depicted in Figure 4.36, when testing single turns of 90 degrees, the robot would often result in a reasonably accurate turn with a tolerance of 10 to 15 degrees either way. Nevertheless, when performing consecutive turns, such as turning 360 degrees, the errors of each turn would accumulate, often

undershooting the target angle. This led to the conclusion that even though the robot may have decent accuracy, the robot had poor repeatability.

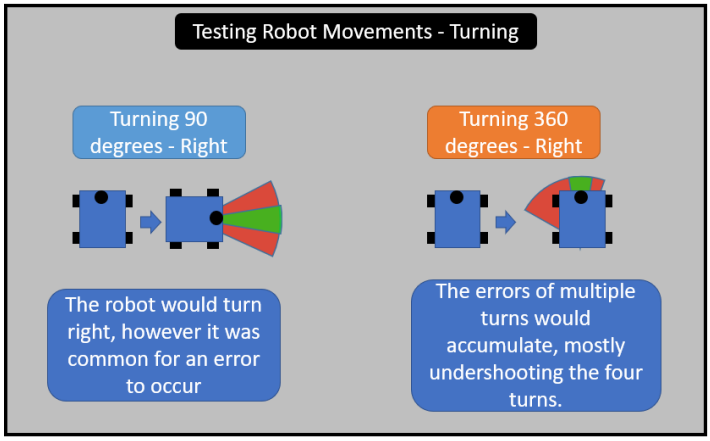


Figure 4.36 - A graphical representation of results from performing robot turn tests.

4.11 - Testing an alternative pathing algorithm

The results also show that the pathing algorithm works in simple scenarios; however, the system is more likely to plan a path through objects when more items are added. This shows how object avoidance is beneficial to making reliable paths.

In an experimental version of a new pathing algorithm, when a collision is detected, the system will alternate what axis it is correcting on, diverting the robot. Figure 4.37 shows the old algorithm being blocked by an object, with no method of overcoming the obstacle. Figure 4.38 shows the new algorithm in a similar scenario, avoiding two obstacles and reaching the destination item successfully.

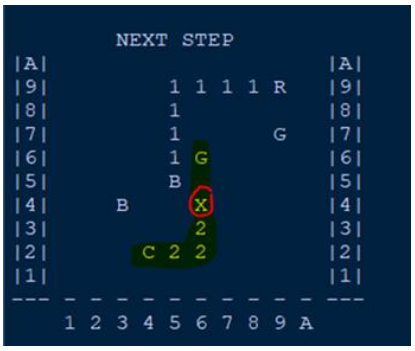


Figure 4.37 – Previous algorithm is incapable of avoiding collisions.

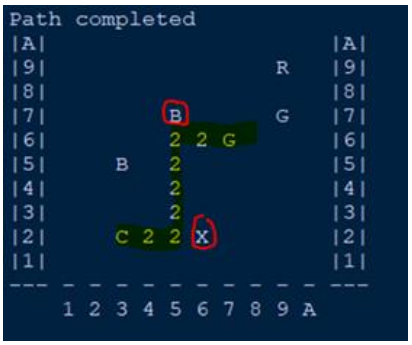


Figure 4.38 – The new algorithm can successfully avoid simple obstacles.

Chapter 5: Discussion

The research highlighted some of the benefits of using robots in certain disaster scenarios. This research also outlined some of the common issues, such as radiation damage and how limiting the electronics within a robot can limit the amount of shielding required. This proved useful to the project, as centralising complex processing away from the robots themselves can help solve issues of reducing onboard computing hardware and overcoming supply issues by manufacturing simpler robots in a shorter time frame. Similarly, using older robots can provide a level of redundancy as introducing otherwise expendable robots will create a higher availability of robots to use.

From the results, it can be seen that a system was completed. However, the system is not as reliable as intended and improvements can be made.

5.1 – Discussing the object detection and pathing issues

The results indicate that there are temperamental issues with the object detection. For example, when loading various simple test images produced in PowerPoint, the system detects all objects of all categories every time. In contrast, physical results using the camera and painted blocks can be unpredictable, and the lighting conditions in the room has a significant impact. I believe this is because the shade of the blocks appears different in a change of room location or time of day, whereas graphically produced images are the same regardless of time or place.

Two images below show the results of further testing of various colour blocks in different lighting conditions. Figure 5.1 shows three objects tested in the same location as before, illuminated primarily buy window light, including an orange, yellow and purple test block. Secondly, Figure 5.2 shows another location illuminated by artificial light, with the items showing brighter than the previous test location. These differences in how colours can be perceived can cause issues in detections and suggests colour detection in this implementation would not be suitable for real world situations.

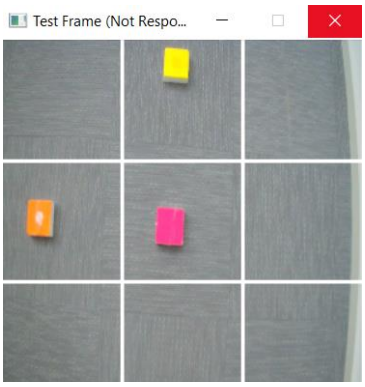


Figure 5.1 – Various object presented to the system in the original test location.

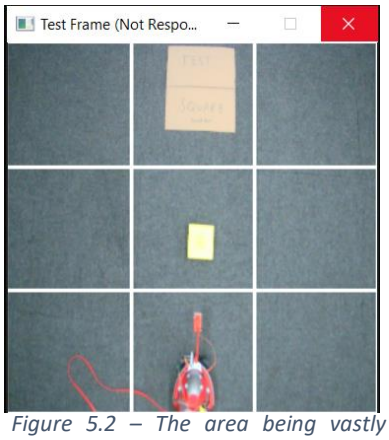


Figure 5.2 – The area being vastly higher brightened in different lighting conditions.

Further evidence includes Figure 5.3, taken at the same location and time as Figure 5.1. In this image, the results of the colour detection mask layers are displayed, showing the yellow block being passable as orange, and only the outskirts of the purple block being detected with the current threshold values and lighting conditions.

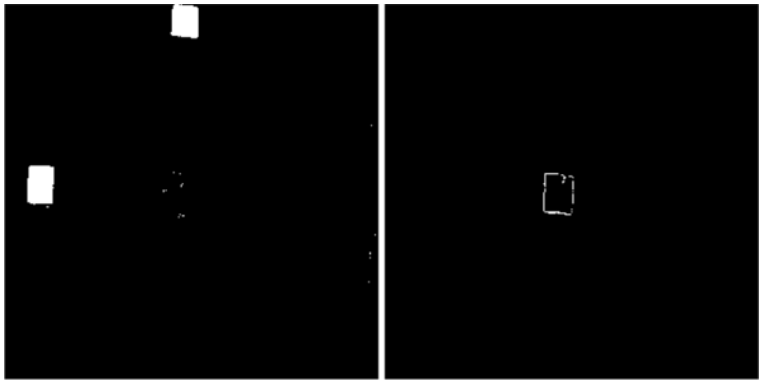


Figure 5.3 – An image showing a yellow block being detected as orange, and a purple block not being fully detected as purple.

In order to improve the system, a more sophisticated or better-tuned object detection method is required, one where changes in lighting would pose less of an effect. This is essential as the system cannot reliably fulfil tasks requirements without correctly detecting objects and issues increase the risk of the robots colliding with objects not known to be there.

Similarly, the pathfinding algorithm needs to be expanded on the matter of preventing collisions. This includes using methods of waypoints, where if an obstacle or indirect path is detected, or by combining a collection of alternative nested sub-routes. This includes the new experimental algorithm shown previously in Figure 4.38. Figure 5.4 shows an annotated representation of this method.

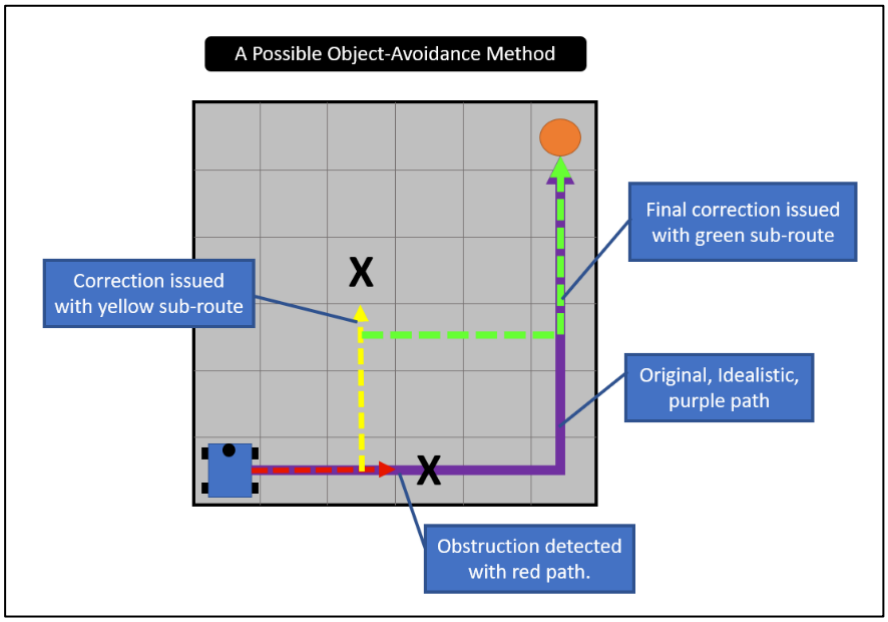


Figure 5.4 - An illustration of a possible more advanced pathing method implementing object avoidance additional to obstruction detection.

5.2 – Project limitations

There were, however, some limitations with the project. Firstly, testing adequately sized areas using the webcam proved challenging in terms of floor space, so most practical tests never surpassed 3x3 grid sizing. This would constrain the complexity of the scenarios and the results of carrying them out. For this reason, it would be hard to properly estimate the scalability of the prototype system at this time.

Secondly, the 'Cybots' proved a useful foundation for constructing the demonstration robots, where their modularity was key. However, limitation on microcontroller memory size resulted in insufficient memory to implement effective movement feedback without increasing the risk of stability issues occurring. With more time, a suitable solution may be developed. However, a similar-sized but higher memory capacity microcontroller would be ideal.

A self-proposed limitation was to restrict sensor input from the robots to the system to an absolute minimum. In many cases, the sensing equipment on a robot may be non-existent, affected by outside interference or suffer complete failure. For this reason, it was aimed to find the system's effectiveness with no object detection from the robots directly in the attempt to succeed in many provided scenarios without needing any additional input. Without this limitation, sensors such as a phased ultrasonic array could have been fitted to the robot to better align with pickup targets or detect items unregistered by the colour detection.

Chapter 6: Conclusion

The research of this project sought to find a centralised approach to manage a small group of robots to carry out disaster scenario inspired tasks. From these results, it can be concluded that the proposed concept is promising, as it helps issues referred to in the research. However, the resulting reliability of the system, apparatus and robots was lower than expected. Although considering this, the system did show potential. All project areas were shown to have worked to some extent, with failures primarily caused by project limitations and issues with implementation where reliability issues would occur with specific scenarios.

For instance, the project's first aim follows: "to produce a prototype system that will show how taking aerial image captures of the disaster zones can be analysed and used to create paths for a group of robots". A prototype system was produced, where it utilised a camera and object detection to fairly reliably detect, classify and locate objects from the test area. From this data, the system was capable of producing paths for connected robots, though at times, issues occurred.

Secondly, the project also aimed "to show how this system can apply to robots with limited sensing ability and computing resources ". Two simplistic robots were created using the parts from a 21-year-old robot kit. Unfortunately, only one was used for physical tests due to limitations of suitable grid sizes. Through this, the robot was able to partially complete tasks both virtually and physically without the need for any sensor input from the robots.

Unfortunately, it was failed to produce a system reliably enough to complete the full-system tests. From these failures, problems were recognised, and improvements can be made. Overall, the results indicated no direct issues with the proposed concept itself.

Two coloured blocks were used to test the object detection, each of a colour representing an item classification. The orange blocks were detected reliably throughout the project, both virtually and physically, regardless of location and lighting method. The purple blocks, however, were very temperamental to detect using the webcam and caused issues. By widening the acceptable thresholds this sometimes overcame the issue, but too far would lead the red parts of the robot’s body to also be detected by mistake.

Additionally, the path generation was not able to cope with complicated scenarios. In situations where many objects reside, the path generation would occasionally plan through objects or miss steps in paths. However, for simpler scenarios with limited items, the path generation worked very well. An alternative path generator was in production, with some tests conducted, but was not able to be implemented into the project with the time remaining.

The implementation approach chosen proved valuable, as the project was accomplished in smaller parts, easily tested individually and later combined.

Chapter 7: Future Work

To fulfil project expectations in the future, a selection of areas can be further explored.

Additional to implementing a better, possibly AI-based, approach to object detection, other considerations will have to be investigated for real-world scenarios. For instance, there may be natural pathways for the robots to utilise, such as roads and sidewalks. By including a means of edge detection, similar features can be detected and implemented into the pathfinding. Additionally, unlike the test scenarios presented in this project, real-world situations are not guaranteed to have an even floor. For this reason, ridges, curbs, rubble and other obstacles will also have to be recognised and avoided.

Drones were a resource mentioned in the proposal and research of this project. However, though agile, drones are not guaranteed to remain perfectly still when capturing images, unlike the webcam stand. Future research could also explore the tracking of changing drone positions to ensure the same objects won't mistakenly appear in multiple grid locations as drones stray from their place, such as from wind. Similarly, the use of multiple drones can be explored, combining object detection results for redundancy against missed items and creating a more extensive search area. A system controlling the position of drones through automation may also be a possibility.

Another key point brought up in this project was radiation shielding. Though the minimising of electronics was discussed, actual shielding itself was not attempted. Due to the selected microcontroller having restrictions of only 2 kilobytes of SRAM, alternative microcontrollers should be explored with accompanying protection enclosure designs. Similarly, a cabled connection was used to communicate to the robots, as seen with the Pioneer project. Wireless communication also has many benefits. Research on wireless communication methods and the effects on signals by shielding would be valuable when creating a production-level system.

Citations

[1] Crowe, S. (2019) "How drones & robots helped extinguish Notre Dame fire" from 'The Robot Report' website. https://www.therobotreport.com/how-drones-robots-helped-extinguish-notre-dame-fire/. Paragraphs 1 through 5. Accessed 09/03/2022.

[2] Crowe, S. (2019) "How drones & robots helped extinguish Notre Dame fire" from 'The Robot Report' website. https://www.therobotreport.com/how-drones-robots-helped-extinguish-notre-dame-fire/. Paragraphs 11 and 12. Accessed 09/03/2022.

[3] Cholteeva, Y. (2020). "Making Chernobyl safe: a timeline" from 'Power Technology' website. https://www.power-technology.com/features/making-chernobyl-safe-a-timeline/#:~:text=For%20this%20reason%2C%20the%20liquidators,from%20the%20reactor%20deep%20underground. Paragraph 6. Accessed 08/03/2022.

[4] Unknown (accessed 2022). "Frequently Asked Chernobyl Questions" from IAEA (International Atomic Energy Agency) website. https://www.iaea.org/newscenter/focus/chernobyl/faqs. Paragraph 1. Accessed 08/03/2022.

[5] Unknown (Accessed 2022). "Graphite blocks in nuclear power stations" from 'EDF' website. https://www.edfenergy.com/about/nuclear/graphite-core. Third main text box. Accessed 08/03/2022/

[6] Higginbotham, A. (2006). "Chernobyl 20 years on" from 'The Guardian' website. https://www.theguardian.com/world/2006/mar/26/nuclear.russia. Paragraph 7. Accessed 08/03/2022.

[7] Higginbotham, A. (2006). "Chernobyl 20 years on" from 'The Guardian' website. https://www.theguardian.com/world/2006/mar/26/nuclear.russia. Paragraph 27. Accessed 08/03/2022.

[8] Unknown author (Accessed 2022). "Liquidators" from 'chernobylgallery.com'. http://www.chernobylgallery.com/chernobyl-disaster/liquidators/. Paragraph 4. Accessed 08/03/2022.

[9] Unknown author (Accessed 2022). "Liquidators" from 'chernobylgallery.com'. http://www.chernobylgallery.com/chernobyl-disaster/liquidators/. The image was taken by 'Igor Kostin'. Image 1 on-page. Accessed 08/03/2022.

[10] Abouaf, J. (1998) "Trial by fire: teleoperated robot targets Chernobyl," in IEEE Computer Graphics and Applications, vol. 18, no. 4, pp. 10-14, July-Aug. 1998, doi: 10.1109/38.689654. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=689654. Page 3. Accessed 10/03/2022.

[11] Unknown. (2021) "CHERNOBYL ROBOTS" from 'chernobylx.com'. https://chernobylx.com/chernobyl-robots/. Accessed 10/03/2022.

[12] Unknown. (2019).” The man, the myth, the legend: Meet Shakey the robot, the world’s first AI-based robot” from ‘SRI International’ website. https://www.sri.com/case-studies/the-

man-the-myth-the-legend-meet-shakey-the-robot-the-worlds-first-ai-based-robot/. Accessed 2022.

[13] Unknown. (2019). “The man, the myth, the legend: Meet Shakey the robot, the world’s first AI-based robot” from ‘SRI International’ website. https://www.sri.com/case-studies/the-man-the-myth-the-legend-meet-shakey-the-robot-the-worlds-first-ai-based-robot/ . Image 4. Accessed 2022.

[14] Unknown. (2021). “Shakey the Robot Explained: Everything You Need to Know” from ‘history-computer.com’ website. https://history-computer.com/shakey-the-robot/. Accessed 2022.

[15] Unknown. (Unknown). “Happy, a Reading University robot” from the ‘BBC’ website. https://www.bbc.co.uk/ahistoryoftheworld/objects/0-6sufjRSnGw70a2l2TuvQ. Accessed 2022.

[16] Schranz, M. Umlauft, M. Sende, M. Elmenreich, W. (2020). “Swarm Robotic Behaviors and Current Applications” from ‘Frontiers’ website. https://www.frontiersin.org/articles/10.3389/frobt.2020.00036/full#:~:text=It%20can%20be%20used%20to,or%20establish%20a%20communication%20network.&text=Coordinated%20motion%20moves%20the%20swarm,be%20arbitrary%20as%20in%20flocking. Accessed 2022.

[17] Woolley, D. (2021) "How to Detect Colors in OpenCV [Python]" produced by YouTube account 'CreepyD'. https://www.youtube.com/v/cMJwqxskyek. Image from timestamp 1:00. Accessed 29/11/202

Bibliography

D. P. Stormont, "Autonomous rescue robot swarms for first responders," CIHSPS 2005. Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, 2005., 2005, pp. 151-157, doi: 10.1109/CIHSPS.2005.1500631.

Appendix

Appendix 1 – Initial Gantt chart

Initial chart

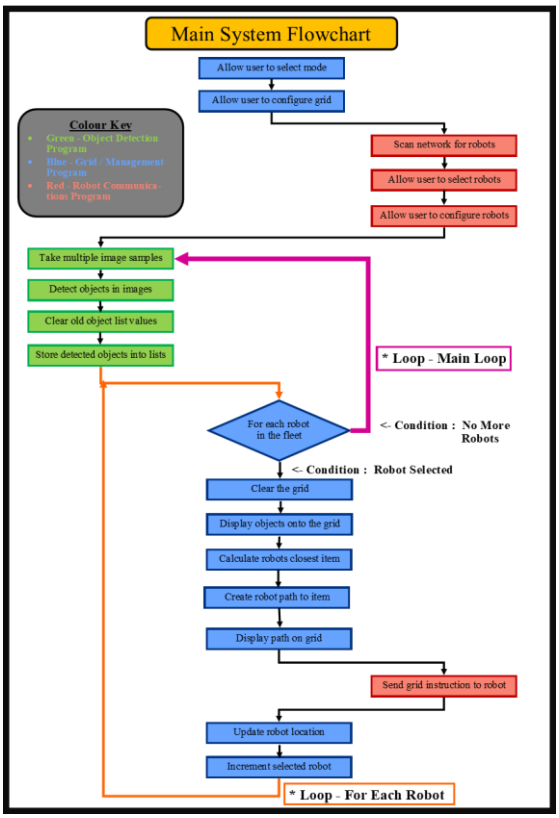
	Weeks								
Objectives	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12
Research : Object Recognition Training									
Research : Socket Programming									
Integrating Arduino into robot									
Integrating newtworking into robot									
Create method to log object positions									
Create simple path maker algorythm									
Create method to send commands to robot									
Implement object recognition									
Add another robot									
Improve networking and AI									
Prepair December presentation									

Novemeber Progress (End)

	Weeks								
Objectives	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12
Research : Object Recognition Training									
Research : Socket Programming									
Integrating Arduino into robot									
Integrating networking into robot									
Create method to log object positions									
Create simple path maker algorythm									
Create method to send commands to robot									
Implement object recognition									
Add another robot									
Improve networking and AI									
Prepair December presentation									

This chart shown the initial Gant chart made in October, compared to the first Gantt chart updated, from November.

Appendix 2 – A flow chart depicting the internal processes of the application



Appendix 3 – Initial safety Assessment form

CEDPS - EEE – 2021/22 Technical Project Initial Safety Assessment & Acquisition form

Please note: *Forms must be submitted electronically in accordance with departmental flowchart*

Hand-written forms cannot be accepted.

Incomplete forms will be returned.

Project title :	Integrating Neural Networks in Swarm Robotics Coordination	**Date :**	19/10/2021
Project initiator name :	Click or tap here to enter text.	**Student number:**	Click or tap here to enter text.
Student email address :	Click or tap here to enter text.		
Supervisor :	Click or tap here to enter text.		
Level / year *e.g. level 6, PhD, Staff Research* **:**	Level 3		

Details of any manufacture or construction work involved :

Adaptors for integrating the microcontroller into the robots will have to be 3D printed. Some soldering may be required to make connection within the robots where plugs will not suffice. 3D printed adaptors may be made to produce manipulators for the front of the robot, such as to grab or push. Basic craft materials such as polystyrene shapes will be painted using acrylic paint where the colours correspond to the classification of the object. This may include, orange painted cubes for 'good' objects and purple painted spheres for 'bad' objects.

Details of any experimental or testing work involved :

The object detection part of the software package will have to be tested for its accuracy. There will be some tests where objects of certain types will be presented and the system's ability to correctly detect and classify each object will be recorded. If the results are not satisfactory, the system will be retrained. Objects will also be placed at various places within the test area, this will be used to test the accuracy of the calculated object positions, calibrations will follow where necessary. The robots will be tested to be able to effectively manoeuvre within the test area as well as testing their ability to move the objects. To fully test the system, a series of scenarios will be tested where the quantities of each object and their positions will be varied. Each scenario will be more 'difficult' for the system than the last where quantities and other variables may be increased.

Location/s of work :	Dorm room, Worksop / Lab space, Parents house.
Manufacture / Construction *This project involves manufacture or construction work to take place the following rooms (please list all rooms to be used):*	
Some construction may require use for the labs or the workshops. For example, the soldering of wires or the cutting of material. Where available, this may also be done outside of the university at my parents' house. Programming of the robots and system can be done anywhere using my laptop.	
Experimental / Testing *This project involves experimental or test work to take place in the following rooms (please list all rooms to be used):*	
Basic testing such as the detection of objects can be done on the floor of my dorm room as a small quantity of objects at a time will be placed under the webcam's tripod. For larger tests, for example that for the robots in action or the complete system, may require larger space such some floor space of a larger room such as a lab or workshop so objects are reasonably distanced.	
*I have **read and understood** that the room risk assessment/s cover all safety requirements for this project, including physical controls, training, supervision and PPE:*	
Tick to confirm the above statement is true ☑ (click on the box to tick)	

Appendix 4A – Initial project outline

Final Year Project (EE3099), Individual Project (EE3600) Initial Project Outline

Student ID		Date	19/10/2021
Student Name		Course	
Project Title	Integrating Neural Networks in Swarm Robotics Coordination		
Supervisor's Name			

Aims and Objectives	To devise a system that will remotely manage a small group of swarm robots. This will consist of a computer with an aerial camera and some electronics to be integrated within some premade, modified robots. There will be a series of objects in a test area, where they will be classified as good objects, bad objects and robots. Where the good objects are to be collected, bad objects removed from the area and the robots being the objects that carry out the tasks. The system will be responsible for detecting the objects and their positions and updating the robots with the relevant tasks over a network. This is opposed to having a camera and computer in each robot, which may be more expensive than one computer communicating with a range of cheaper robots over a network.
Research Outline, including own academic contributions	Areas to research: • Controlling Arduino microcontrollers over a network • Object detection in python using a camera • Training object detection in python • Python network socket programming • Suitable robot manipulator designs

	Own academic contributions: • To devise a way of integrating a newer microcontroller into an older toy robot. • To devise a system to log objects and que tasks for the various robots in the network.
How to Implement (software, simulations, hardware building, user experiment)	The system will detect a range of objects within a test area using the arial camera (most likely a webcam on a stand). The system is to identify and log the objects within the area and update the robots with relevant tasks relating to the objects detected over a network. For example, telling robot #1 to remove bad object #2 from the test area. This will include training an object detection algorithm with custom images, most likely using a trainable system from a python library. Integrating microcontrollers into premade robots, where some form of communication method will be added such as Wi-Fi. This may include 3D Printing and soldering. Producing objects from cheap craft materials to train the system will also be required as well as for testing the system for the correct detection of these objects. Some form of application for a computer will have to be produced where it will relate a detected object to a task and use the network to update a robot with that task as well as the position of the object.

Needs for hardware works	**Yes** (Initial Safety Assessment Form attached)/ No
Need for an ethics review (use of human/animal tissue, user trials etc.)	Yes / **No**

Final Year Project (EE3099), Individual Project (EE3600)

November - Updated Project Outline

Student ID		Date	24/11/2021
Student Name		Course	
Project Title	Integrating Object Recognition in Swarm Robotics Management		
Supervisor's Name			

Aims and Objectives	To devise a system that will remotely manage a small group of premade but modified robots, of which make use of older or more simplistic hardware, with minimal sensing capabilities. The system will consist of a computer and an aerial camera, of which will communicate with the robots over a network, sharing relevant data such as object and robot positioning and the status of tasks. For the system to use Object Recognition to classify various objects of interest in a search area and store their positions in memory as well as assign tasks and calculate routes for the robots to take. For the robots and the system to be able to retrieve and unload the objects to their correct destinations working as a collective group. There will be a series of objects in a test area, where they will be classified as good objects, bad objects and robots. Where the good objects are to be collected, bad objects removed from the area and the robots being the objects that carry out

	the tasks. This is opposed to having a suitable computer and camera system within each robot, which would be more expensive. The additional aerial viewpoint of the system is beneficial in route planning tasks and can be used to find objects of interest/hazards as well as planning routes that may not be visible from the perspective of the robots individually. This is in an attempt to allow older and reliable robots to remain useful for longer, only requiring them to have minimal sensing and processing abilities by having the bulk of the processing carried out by a vastly more powerful and upgradeable external system.
Research Outline, including own academic contributions	Areas to research: • Controlling Arduino microcontrollers over a network • Object detection in python using a camera • Training object detection in python • Python network socket programming • Suitable robot manipulator designs Own academic contributions: • To devise a way of integrating a newer microcontroller into an older toy robot. • To devise a system to log objects and qeue tasks for the various robots in the network.
How to Implement (software, simulations, hardware building, user experiment)	The system will detect a range of objects within a test area using the aerial camera (most likely a webcam on a stand). The system is to identify and log the objects within the area and update the robots with relevant tasks relating to the objects detected over a network. For example, telling robot

	#1 to remove bad object #2 from the test area. This will include training an object detection algorithm with custom images, most likely using a trainable system from a python library. Integrating microcontrollers into premade robots, where some form of communication method will be added such as Wi-Fi. This may include 3D Printing and soldering. Producing objects from cheap craft materials to train the system will also be required as well as for testing the system for the correct detection of these objects. Some form of application for a computer will have to be produced where it will relate a detected object to a task and use the network to update a robot with that task as well as the position of the object. This system will also require the ability to produce plausible routes for the robot to follow to their destinations.

Needs for hardware works	**Yes** (Initial Safety Assessment Form attached)/ No
Need for an ethics review (use of human/animal tissue, user trials etc.)	Yes / **No**

Appendix 5A – October monthly report

Final Year Project (EE3099), Individual Project (EE3600)

Monthly Report Form

Student's Name:			
Project Title:	Integrating Neural Networks in Swarm Robotics Coordination		
Supervisor's Name:			
Month:	October		
Person-hour expenditure (elapsed):	60 hours (15 minimum x 4)	Person-hour expenditure (actual):	80 hours

Please attach current project plan Progress with respect to project plan as attached (including any problems) Self assessment of project: Student's own critical review.	**Research** I have been following online tutorials on training and using object / image recognition AI models. This includes following tutorials for detecting faces, numbers and training custom models. I have been following some tutorials on Python socket interfacing and Arduino to Python UDP communication through Ethernet. **Progress** I have made some attempts in training a model to detect the robots with varied success. An alternative model library may have to be chosen. Objects for the system to recognize have been produced using basic craft materials. Photos of the objects have been taken as preparation for the training process. An Arduino Nano has been implemented into one of the robot chassis I already have. I have attached an ethernet shield and achieved communications between the robot and my laptop using a basic python client program based on the tutorial.

	I have started producing a system in python, where I can allocate squares on a grid to a specified object. Using this, I have produced an algorithm to sort what object should be collected by the robot first, as well as another algorythm to create a simplistic path to the object. **Problems** I have experienced problems with training the object recognition using the current library I have chosen. This has resulted in many false positives. I have some ideas to explore to resolve this. I have also found an alternative library which provides a percentage value of similarity to what it is detected (e.g. object looks 50% like a bicycle), this may also be a valid option to explore as I could filter detections by a customized threshold. I am currently working on a gridding system where detected objects will be 'snapped' into grid spaces. This may help when the same object may be detected many times as well as help plan routes. I had some difficulties communicating to the robot but I believe this to be down to network settings on my PC. The laptop I intend to use communicates with the robot perfectly. **Self-Assessment** I believe I have accomplished a reasonable amount of the project so far, though there is still plenty of work left. I also need to work more in documenting my work as I go along too. *Initial project plan included below
Key objectives and milestones for next month Include estimated time to be allocated to the project for the next month.	For next month I want to be able to control the robot using my griding program, where I intend to specify what position it is currently in and command it to a new position, with the robot and the application updating through the process. I also aim to add a second robot to this system. I also want to implement Wi-Fi into the robot, as the

	Ethernet cables may restrict movements of the robot and may tangle or obstruct objects as more objects and robots are added. Another critical objective is to improve my training process and produce a trained model that can detect the relevant objects expected in the training area. I aim to allocate no less than 80 hours of the 60 minimum required.
Supervisor's comments and feedback	Very detailed report; thank you Your progress is significant and I am happy that you are such an independent learner! As milestones of the project are reached, please send photos/videos in order for me to have a glimpse of what you're doing.

Appendix 5B – November monthly report

Final Year Project (EE3099), Individual Project (EE3600)

Monthly Report Form

Student's Name:	
Project Title:	Integrating Neural Networks in Swarm Robotics Coordination
Supervisor's Name:	

Month:	November		
Person-hour expenditure (elapsed):	60 hours (15 minimum x 4)	Person-hour expenditure (actual):	60 hours

Please attach current project plan Progress with respect to project plan as attached (including any problems) Self assessment of project: Student's own critical review.	Research I have been following online tutorials on training a different object recognition package. One of these tutorials includes training the system against two classifications of objects, using images of cats and dogs, to provide a prediction of what is in the image. I found this tutorial more complex than the last but found the package to be far more powerful and easier to train. I have been following tutorials on how to use a specific Arduino compatible compass and accelerometer device, to use within my robots. Progress I have produced a better training set for object recognition, ready to create a modified version of the program in the tutorial to recognize the objects. This has been done by taking pictures of the object in various locations so that the floor is not the same in every picture, as I did using the last package. I believe that this may have previously been a cause for some of the false positives I had before, which increased when the objects were placed on the same floor as from the training.

	I have 3D printed a better mounting device for the Arduino Nano within the current robot I have been adapting. This allows the Arduino to be properly fixed into place reliably as well as allow for proper positioning, providing access to the Arduino's USB port and the RJ-45 port on the above Ethernet card. I have modified the previous virtual system on the laptop to be able to process the paths for multiple robots. I have also improved the way objects are stored in memory and placed onto the grid to allow for easier scalability. I have also done this to make it easier to distinguish what objects have already been retrieved and placed in a suitable area and those still in need of retrieval. <u>Tasks still in progress</u> I have been working on a queuing system for which order objects are to be retrieved. I have also been working on how the system is to create paths to a suitable drop off location for when the object has been picked up. My recent change to the system will make this process easier to implement. I have been working on methods for the robot to move between grid spaces rather than just being told to move forwards or turn. This includes specifying how forwards the robot is to move and how much it needs to turn. This is where I will use what I have been learning from the compass tutorials to finish this step. Now I have followed some of the tutorials on training the alternative object recognition package, and have created new images to train with, I now have to build, train and test a new system. <u>Problems</u> I found when getting the robot to turn left and right, in 90-degree increments, that the characteristics of the

	robots swivel wheel can sometimes become problematic. I found that the amount of time needed to move to the correct position changes depending on the last movement. Moving in an opposite direction to the last movement takes additional time as the swivel wheel has to correct first. For example, if turning left to right, the swivel wheel has to move to correct its direction more than if it went from forwards to right. This causes the robot to not always reliably turn the whole 90 degrees when needed. Using the compass module, the robot will be able to see how much it is turning and stop when the correct angle (within an acceptable threshold) has been achieved. This should prevent the under or overshooting of the turns as when using a fixed amount of time to turn for. <u>Self-Assessment</u> I believe I have achieved a reasonable amount of progress this month though not as much as I previously aimed. Some of the key milestones I wanted to achieve I have not been able to, though I have overcome some issues I had previously. This includes improving how the robot and laptop communicate, improving and adding to the gridding software and learning how to use a more appropriate object recognition package. I still yet have to provide the robot with a Wi-Fi connection rather than ethernet, and link the three main sections, the AI, Gridding Software and Robot Communication Software, together as one system. *New project plan included below
Key objectives and milestones for next month Include estimated time to be allocated to the project for the next month.	For next month I want to be able to control the robot using my griding program, where the robot will be notified of the next position it is to move to. This will be done by combining the software that communicates with the robot with the main gridding software.

Final Year Project (EE3099), Individual Project (EE3600)

Monthly Report Form

Student's Name:			
Project Title:	Integrating Object Recognition in Swarm Robotics Management		
Supervisor's Name:			
Month:	December		
Person-hour expenditure (elapsed):	60 hours (15 minimum x 4)	Person-hour expenditure (actual):	Approximately 60 hours

Please attach current project plan Progress with respect to project plan as attached (including any problems) Self assessment of project: Student's own critical review.	**Research** No major new research was done this month. Only a continuation of those mentioned last month. Some experimentation was done on how the system and robot will communicate. **Progress** A 3D printed mount was designed to hold the compass module, of which is currently being added. This keeps the module in a mostly, though not exact, midpoint of the robot and will be used for error checking, such as overshooting when turning. Another 3D printed Arduino Nano mount was produced where it will be used in the conversion of the second robot chassis. The new version of the gridding software now interfaces with the new robot communication program. A range of menu options allow a robot to be added or selected from a robot fleet list, where a robot can be selected for manual control. The dynamic grid making allows for the user to the

	specify how big the space is when they are controlling the robot manually. This allows the user to save and load custom grids through files, though this is a separate feature and not yet integrated into the other modes of the system. **Tasks still in progress** A new message syntax is in development, where designing and implementing the forms of the messages sent between the robot and system are in progress. Methods of initializing the robot with robot specific attributes (ID Number, Starting Coordinates, Robot type…) through the Desktop System has been established and works effectively. Currently the direct and grid control methods are in development where it is established PC side but still needs to be implemented within the robot code, modifying the manual control code produced in the previous version of the system. The grid control method sends the robot the chosen adjacent grid square for the robot to move too. The direct method sends the robot a direction and how many steps to move in that direction, similar to the language Logo. **Problems** While following some tutorials on the object detection, I found some of the tutorials to be out of date where not all of the areas work. However, I have found some more recent tutorials that I have been following. **Self-Assessment** I have been keeping more pictures of my progress through I have not achieved as much as I previously aimed.
Key objectives and milestones for next month Include estimated time to be allocated to the project for the next month.	I aim to have at least some form of trained object detection system by the end of next term, if not the fully working object detection part of the system. I aim to finish the communications ide of the system very soon, where not too much work is left to finish

	and debug as many of the features work in a very similar way. The manual control is almost done where the robot can recognize basic commands. Once finished, this can be linked to the pathing algorithm where the system will send movement commands automatically rather than the user manually.
Supervisor's comments and feedback	Happy to see some visuals of the demo so far although the student indicated that there is still communications issues to be resolved. I would have hoped that these would have been resolved by now as there is not too much time left for this to become a working prototype. Soon we will be discussing the format of the written part

Final Year Project (EE3099), Individual Project (EE3600)

Monthly Report Form

Student's Name:			
Project Title:	Integrating Object Recognition in Swarm Robotics Management		
Supervisor's Name:			
Month:	January		
Person-hour expenditure (elapsed):	60 hours (15 minimum x 4)	Person-hour expenditure (actual):	Approximately 60 hours

Please attach current project plan Progress with respect to project plan as attached (including any problems) Self assessment of project: Student's own critical review.	**Research** A tutorial on colour detection and tracking using Open CV for python was followed. This is the same library currently used to take webcam snapshots. **Progress** A colour detecting and tracking program was made, where it detects the two classifications of objects (Orange:Good, Purple:Bad) and produces a list of detection coordinates for each category. A newer version of the gridding software is in progress, providing a foundation to combine the functionalities of the separate parts of the complete system (e.g. object detection and robot communications). In essence, this means combining the project into one software package. The robot has been fitted with an arm to allow it to pick up, transport then drop objects in another grid location. The arm may not be quite at the right angle, making the robot slightly longer than it should, but this can easily be changed later.

	Capsules of various depths have been created to embed magnets into the polystyrene blocks to allow them to be picked up by the arms electromagnet. If the magnet is too close to the surface, then the magnet may still stick to the electromagnet when not energised; if the blocks are too deep, then it may be too weak to allow the block to be lifted. The quantity of magnets is also an issue. Too few magnets mean higher accuracy is needed to pick the block up (very hard for this robot to align up perfectly) though too many increase the weight and prevent it from being lifted easily. Both the communication program and robot code have been adapted for the control of the arm. When the robot is given a grid location, three categories follow. These include Go-To, Pick-Up and Drop-off, numerically represented. This allows the user to control exactly what the robot does with the given location in a remote control like fashion. This now needs to be combined with the output of the pathing algorithm, where the list of movements will be given to the robot one at a time, similar to a user entering manual commands. **Tasks still in progress** The pathing algorithm from an earlier version of the gridding system is being added to the current version. It needs to be adapted to work with the new way objects are stored in memory and the required syntax for the route. For example, when the path is produced, it needs to log whether the robot is to move to this location or to pick up or drop off an object from/into this location for each step. Changes to the robot code are being made to reduce the waste of memory recourses. Most tasks require combining, testing, and refining what the system has already.

	Problems The robot is running dangerously low on memory, specifically the 2KB of SRAM. The IDE provides a warning that stability issues may occur. As the robot is running, more of its memory will be used. This includes reading messages, producing messages, and storing data from other tasks. If the robot runs out of memory, then the robot may not act accordingly or possibly halt from working at all. I still have not produced a good AI object detection part for the system. The colour detection should work as a good alternative. As the blocks are of a basic shape and colour, there would be no benefit of using an AI approach to object detection, as this version works well enough for this project's scope. In a real-world solution where actual objects such as toxic waste may be searched for, a trainable AI would be a more suitable method. However, I have found a more recent tutorial on producing a TensorFlow object detection program. It seems well explained, so this could be a good afterthought once the main functionality of the system is working, or even an option for the operator to choose from when initialising the system. I still have not yet tried to get Wi-Fi to work on the robot. As the robot is not the final product, with it being more of a 'dumb' machine for the system to command, Wi-Fi is more likely a luxury for convenience than a necessity, as the same transport layer protocol (UDP) is used. Wi-Fi may be possible once the system is functional but not a primary concern at this point. **Self-Assessment** Things have been completed at a slow and steady rate, which is concerning when considering the time

	left. The system is mainly segregated, so there should be a working system that only needs refining when combined.
Key objectives and milestones for next month Include estimated time to be allocated to the project for the next month.	For next month I aim for a working system. As each separate part of the system works to at least a reasonable level, they just need to be combined to work together. The robots' movements are not ideal. I plan to slow the robot so its movements are not as sudden and to add the compass functionality if there is enough memory remaining. The robots power source is also not ideal after some of the recent additions. The energisation of the electromagnet causes an increased current draw and sometimes reduces the power provided to the network card under what is required. I am adding a more appropriate power source to overcome this. I aim for 60+ hours of time to be allotted for next month, or as much as needed to finish the system with consideration for the written part to be worked on and other assignments to be completed that month.
Supervisor's comments and feedback	Once again, a detailed report. Nothing for me to report here as the student has the grasp on everything. Liked the video with the robot’s electromagnetic arm!

Final Year Project (EE3099), Individual Project (EE3600)

Monthly Report Form

Student's Name:			
Project Title:	Integrating Object Recognition in Swarm Robotics Management		
Supervisor's Name:			
Month:	February		
Person-hour expenditure (elapsed):	60 hours (15 minimum x 4)	Person-hour expenditure (actual):	Approximately 60 hours

Please attach current project plan Progress with respect to project plan as attached (including any problems) Self assessment of project: Student's own critical review.	**Progress** The system (the software application) itself is near completion. Currently, the system can take image captures of the test area and detect two classifications of objects through colour recognition. The system also attempts to communicate to a list of known robots, discovering which ones are available and placing their details into an availability list. By allowing an operator to select what robots to use from an availability list, they are to be added to the current fleet. From here, the operator can add key information about the robots such as starting coordinates and the robot's role. Up to 4 robots are supported at this time, with two modified toy robots usable. The objects detected by the system are then placed on a virtual grid to log their locations. Taking multiple image captures per scan has been found to increase the chance of detection in less than ideal lighting circumstances.

	Once the system's setup reaches the pathing part of the system, the operator commands the system to go into automatic mode, where no response from the operator is needed from this point. For every robot in the fleet, the system will scan for the closest appropriate object according to the robot's type, will plan a route to such object, and send the robot a command of the next grid space to move into or interact with. If, however, the robot is carrying a payload (such as picking up the object at the destination), the system plans a route to the nearest drop-off point instead. This leads to the system being mostly complete with minor changes to be made. This is not ideal considering the time left though not terrible. **Tasks still in progress** A load-from-file feature is being added to the system to demonstrate and test the system in ways it cannot be physically, such as limitations to the current robots and testing environment. Currently, changes in the code allow loading an image file rather than live captures; however, this method is messy. The operator still needs to enter setup information that can otherwise be made automatic. A cleaner solution is currently underway and should not take long. One robot is essentially finished with a working arm produced and mostly calibrated movements. The accuracy of the movement is reasonable, though have a poor repeatability. For example, when telling the robot to turn right, a mostly 90-degree turn is achieved with some small variations each attempt. Even though one right turn may work well, the robot is poor at returning to its original location when taking many turns. A new robot from mostly new spare parts is being

	produced (as the robots are modified from toy robot kits). The drivetrain grease is almost 20 years old with dirt buildup; these have been cleaned with new grease applied for improved movability. However, the robots are still toys with no movement feedback or precision parts, which is the leading theory of their unreliability. The pathing algorithm is being adapted. The system makes simplistic paths, first correcting the X axis then the Y axis. If a solution is not possible, the system is now being made to change to Y first for that object, producing an alternative route. Additionally, on occasions, the system may mistakenly plan a path through an obstacle or miss steps; these issues are likely linked and are currently being fixed. **Problems** The robots still have issues relating to memory space. There is not enough to add the compass device for movement feedback. The caster wheel has been swapped for an adapted roller ball from a later kit; this has fixed some issues but not all and will limit testing surfaces. One method of feedback is planned to be attempted after the main system is finished. The current plan is to attach magnets to the wheel itself. By using reed switches as pickups, the rotations of the wheel can be counted. This will use far less memory than some other alternatives, with the change mainly consisting of a numeric counter for each wheel. The robot is unreliably picking up the blocks. If the robot is placed in front of the block, it performs well, but alignment is an issue with the current magnet configuration when driving to the block. Once a more effective magnet pattern is produced on the test block, this can be replicated on the fresh blocks. The aim of the project is to create a working system, to instruct robots which will mainly serve a

	demonstration and test purposes and therefore additional. For this reason, the remaining time is being prioritised to the completion of the system itself. **<u>Self-Assessment</u>** My intention was to have the system completed by the end of January – mid-February. This was so that time would be left for final touches and considering issues with my reading and writing skills, the remaining time to be used for completing the thesis itself. Though my personal deadline was not met, not a considerable amount of work remains. Time-wise, this month enough was spent on the project but more would be ideal, with this time also divided on the thesis as well as on the system. Additional to this, other module deadlines have diverted time that could have been applied to this module.
Key objectives and milestones for next month Include estimated time to be allocated to the project for the next month.	As the main deadline is coming close, the primary aim is to finish the thesis, with the coming week being used to finalise the remaining areas of the mostly finished system and for performing tests. If time is left, additional improvements will be made to the robots. The time to allocate is a minimum of 60 hours, but the aim is as much as possible.
Supervisor's comments and feedback	As usual with FYP, I can see that you had some difficulties that have set you back but you are handling those. Do send a video when you finish