PROJ324 - Project O.R.C.A. - On-water Rubbish Collection robot with Automatic sensing

Luke Waller

Monday 23rd May, 2022

Abstract

Oceans make up over 70% of the world's surface. They are vital to all life on the planet as a large majority of the oxygen on the planet comes from phytoplankton that live near the surface of the water. It is estimated that over 10 million tonnes of litter end up in the ocean each year. It is estimated that by 2050 the amount of plastic in the ocean will outnumber the fish, with about 15% floating on the surface (1.5 million tonnes). This on-surface that is being targeted by the On-water Rubbish Collection robot with Automated sensing (ORCA).

Several solutions to this problem have been created; however, the ORCA based solution aims at a novel application that has previously been left unexplored. ORCA aims to collect rubbish from within in-land bodies of water, plastic chokeholds, using a conveyor-based system to remove the rubbish from the water.

Keywords

Waste Collection, Microcontroller, Conveyor belt, C++

Acknowledgements

The author would like to thank the mentor support provided by Dr Ian Howard, who agreed to to undertake the role of mentor during the start of the Autumn Term.

The author would also like to acknowledge the financial support provided by the University of Plymouth.

The author would also like to thank his family members for their continued support, and his friends for the continued inspiration to push me more and more every year.

Contents

List of Figures

List of Tables

Code

Glossary

Catamaran A catamaran is a multi-hulled watercraft featuring two parallel hulls of equal size. 33

ELEC351 Advanced Embedded Programming Final Year Module. 9

Mbed A platform and operating system for internet-connected devices based on 32-bit ARM Cortex-M microcontrollers. 10

Plastazote A nitrogen blown, closed cell polyethylene foam which is tough and flexible. The cells have regular shapes, and polyethylene walls. If the foam is compressed, each nitrogen filled cell acts as a balloon, causing the foam to spring back to its original shape. 34

Acronyms

API Application Programming Interface. 26, 29, 30

ARM Advanced RISC Machines Ltd. 9, 13, 14

BLDCM Brushless DC Motor. 10

DC Direct Current. 10, 29, 30

ESC Electronic Speed Controller. 30, 31

IR Infra-Red. 31, 32

ISR Interrupt Service Routine. 26, 31

ORCA On-water Rubbish Collection robot with Automated sensing. i

PCB Printed Circuit Board. 10

PLA Polylactic Acid. 32

PWM Pulse Width Modulation. 25, 30

RC Radio Controlled. 4, 12, 32, 33

RF Radio Frequency. 10, 12, 15, 23, 24, 27, 28

RTOS Real Time Operating System. 9

SPI Serial Parallel Interface. 15, 24–26, 28

1 Introduction

2 Design Requirements

The design requirements for this project as outlined in Appendix A and Appendix B were agreed upon by the author and their project supervisor. The table acts as a checklist for the project to ensure all the requirements were met. This was a dynamic list that was adapted and amended as the project progressed. The core requirements of the project remained the same but some of the higher level attainable aims were adapted as the project progressed. This was as a result of the critical analysis stages of the project where the feasibility of certain aims was evaluated and these were adapted accordingly.

3 Aims and Objectives

The aims and objectives, which have been listed below, are used as an indicator of the project's success. These were created using the design requirements of the project. The overall success of this project was measured against these aims and objectives:

1. Design and create a boat that is capable of floating on water and being driven around by a radio controller and receiver. The craft will need to be an appropriate size such that it can carry all the equipment on board as well as a decent payload of rubbish to be collected. I will need to create a suitable propulsion system, steering system, and make sure the craft is stable both at top speed and when cornering. It is important to make sure the craft is highly controllable and safe to operate.

2. Design and create a suitable litter collection system that can collect litter on the surface of water. The system will need to be able to be turned off, not be damaging to wildlife, and be able to collect all types of litter that collect on the surface.

3. Combine the litter collection system with the main hull of the boat. Check that the functionality of both components is not impeded by the combination of the two parts together. Get the two parts of the boat controlled separately using a radio controller as the currently available radio controller is only 3 channels.

4. Implement a custom microcontroller based remote control and receiver for the boat that would

allow broader uses for the controller. The controller would be able to display some rudimentary values such as, battery charge on the boat, the current capacity of the rubbish collection system, and the status of the litter collection system (operating/not operating).

5. Add sensors, both active and passive to test for current capacity of the litter collection basket as well as current battery charge. The status of these sensors would be outputted to the controller that would alert the user, alongside stopping the boat when the power is low or when the litter collection basket is full.

4 Project Management

The project has had a development lifecycle of approximately 36 weeks and began on the 6th September 2021.

The project was split into three phases of development. Phase one was research and development of the project and ideas. Phase two was developing prototypes and testing. The final phase was finalising the design and build.

The project was managed using the following software:

- LaTeX (TeXMaker), to write the documentation for the project.
- GitHub, for version control across the whole project.
- Microsoft Excel, to produce Gantt chart and to produce graphs.
- Microsoft OneNote, to log progress and documentation of research and experimentation.

The management style for this project followed a similar style used by Roke Manor Research Ltds. agile framework. This framework follows an agile approach to project development [1], which is evident in the adaptability of the project as it has developed.

The utilised style to manage the project was extremely effective and when utilised with software packages above, all tasks were executed in an organised manner. For general awareness of time management and deadlines, a Gantt chart was utilised. This worked well for this type of project, a solo project undertaken with the supervision of one party, but would not have scaled well to a project with a larger group and organisational supervision. Due to this utilisation method, setbacks

were minimised and able to be accounted for within the initial plan. Any unplanned setbacks were easy to coordinate within such a small working group.

The milestones laid out in the project Gantt Chart for phases 1, 2, and 3 are listed below in Appendix **??**, Appendix **??**, and Appendix **??** respectively.

4.1 Phase 1 - Autumn Term 2021/22

Phase 1 of the project was ensuring that the planning, research, and development was conducted in line with the agile methodology. The objectives from Phase 1 are outlined in Appendix **??**. At this stage, the design requirements and optioneering were considered.

From this initial planning phase, a design brief, Gantt Chart, and future steps were created and agreed upon with the project supervisor. A clear set of requirements had been established along with identification of rudimentary concept ideas. The resultant concepts and decision making processes (optioneering) are detailed in section **??** and section **??**.

4.2 Phase 2 - Winter Term 2021/22

During Phase 2, initial designs were created, tested, and evaluated. Phase 2 involved undertaking initial prototyping and light development of the work. This was to further the work conducted at the end of Phase 1 and to supplement the work being conducted within Phase 3. Phase 2 ran alongside three full-time university modules. It was used as a basis for the prototyping for the project.

The second phase helped focus the design requirements for the project, whilst evaluating what would be possible within the given time scaled budget. The use of threads, learned from ELEC351, along with the development of classes, drove the development of my project from an Arduino based solution to an Advanced RISC Machines Ltd (ARM) based solution. This enabled the use of the Real Time Operating System (RTOS) functionality.

The initial prototypes produced in Phases 1 and 2 helped me evaluate critical dimensions and functionalities of the final craft.

4.3 Phase 3 - Spring/Summer Term 2022

The final phase of the project was the largest and most time consuming. It required the most development work and planning to conduct.

The main objectives from phase three are listed in Appendix **??**. The project can be split into two main groups: the software and hardware development.

The software consisted of developing a controller and receiver board for the craft. This required a transmission protocol to be designed along with several libraries to control various devices such as, buzzers, Brushless DC Motor (BLDCM)s, and Direct Current (DC) Motors.

Most of the hardware for this project was bespoke including, the Printed Circuit Board (PCB)s, Main Body of the Craft and the Conveyor belt. The majority of the hardware to interface with the PCBs was off-the shelf as they are well-established and known to work effectively.

Not all of the software was able to be developed without the integration of some hardware. An example of this would be the development of the transmission protocol shown in Appendix C. Without the use of the nRF24L01 Radio Frequency (RF) chips being used for the project, it was ineffective to test and debug. As a result of this, the software and hardware were developed in parallel. This also allowed time for manufacturing of components, such as the PCBs and 3D printed parts.

Setbacks in both the software and hardware development were experienced. As a result of this, the time accounted for in the Gantt Chart had to be skewed further than previously estimated. Some components ordered in early April of 2022 were still expecting delivery in mid May of 2022. This caused significant setbacks in the project.

Along with massively increased shipping times, and lack of availability of some vital components, software setbacks were also encountered. The nRF24L01P API made for the Mbed board was found to be poorly documented and incomplete in parts. This required time to rectify and understand, when ideally it would have been an easy implementation.

The PCBs were another source of delay, once ordered the board arrived within the specified timeframe. Upon inspection, however, these boards, which were supposed to be fully assembled, were totally unpopulated. This required the components to be quickly ordered and then the board to populated by hand. As a result of this, everything was put on hold for a few days until these

boards were populated and fully tested.

5 Research

The majority of the coursework for the project was conducted during Phase 1 (§4.1), with some continuation into Phase 2 (§4.2). The follow three concepts were critically evaluated to find their strengths and weaknesses.

5.1 Concept Designs - Mechanical Assembly

5.1.1 Concept 1 - Dragged Net Based System

Concept 1 features a monohull craft that consists of a net that is dragged. The craft would drive through litter, collecting it with the net as it drives past.

As this design was complex and required a more unstable monohull design, it was deemed to be the contingency plan and would only be further explored if all other concepts had been deemed impractical.

5.1.2 Concept 2 - Submerged Basket System

Concept 2 uses a catamaran style hull with a gap in the middle. This gap would be filled with a removable basket that sits under the surface of the water. The craft would drive over the top of surface litter collecting it in the basket.

As this design was similar to other designs currently on the market, this plan was deemed as a backup. It has been proven to be effective but there is little novelty in this design.

5.1.3 Concept 3 - Conveyor System with Elevated Basket

Concept 3 focuses on a conveyor belt with a raised basket to hold the collected litter. This would be straddled across a catamaran style hull. The excess liquid run off would be filtered and drained back into the water. This would allow basic samples to be taken from the debris floating in the water.

5.2 Concept Designs - Controller and Receiver

5.2.1 Concept 1 - Off the Shelf Solution

Controller and Receiver concept 1 was to use an off the shelf system for controlling the craft. This system would be reliable, functional, but it would be expensive and not allow for versatility of inputs and outputs that would be required.

The lack of input/output versatility would inhibit the use of smart features on the craft. This does not address any newer technologies or concepts that would be suited to this style of custom RC craft.

Therefore it was decided that this concept would be used for initial testing and as contingency plan. Final deployment would be using this concept, if Concept 2 (§5.2.2) proved to be impractical.

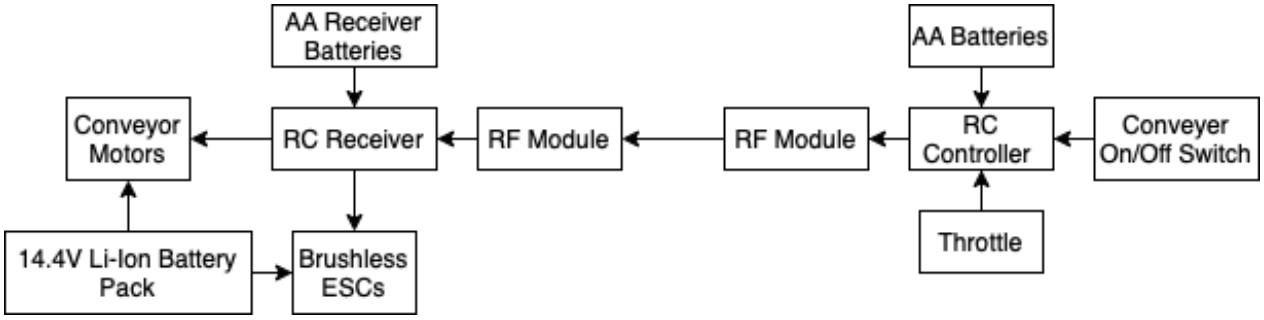

Figure 1: Concept 1 - Off the shelf RC Controller and Receiver

5.2.2 Concept 2 - Custom RF Based Solution

Concept 2 introduces "smart" features such as custom battery management circuity and litter collection basket capacity sensing. It allowed for a number of inputs and outputs on each system (up to the limit of the board used to control the systems). This would allow for a highly adaptable control system that could be easily adapted an alternative use if required. The main craft could be changed and the controller system would be easily reusable. Using a microcontroller based system allows the ability to add and remove devices as the requirements of the craft change.

One of the downsides of this system is that because it is a custom based system, all of the maintenance is required to be done by the user. This requires a lot of work and effort which would not be the case for the off the shelf system. The other issue with this solution is that the RF chips can be intermittent during their operation something that is also not present with the off the shelf

system.

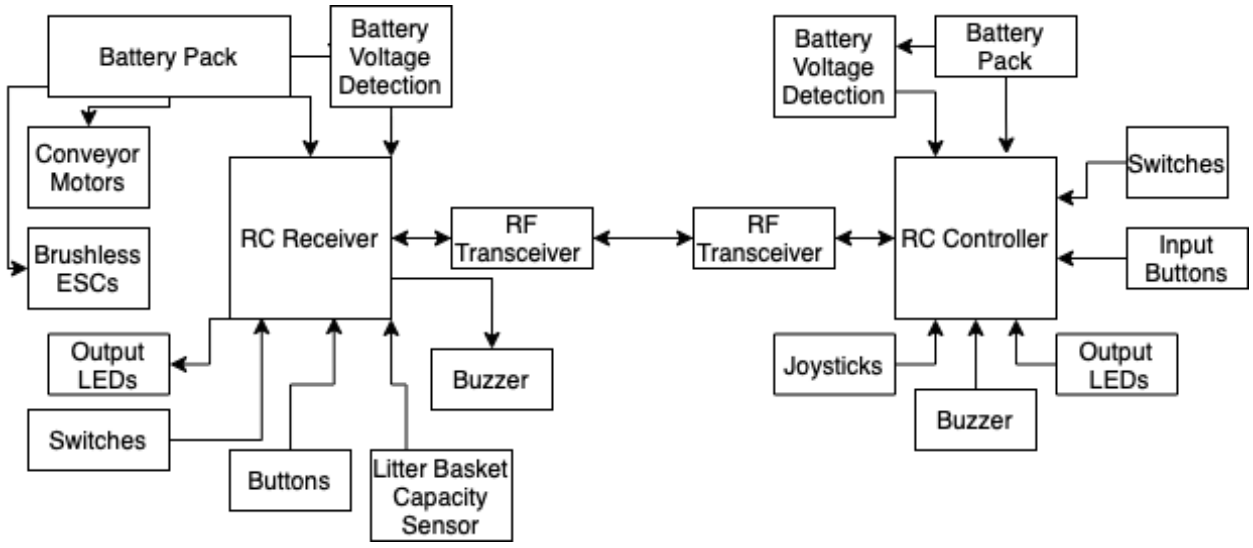

Figure 2: Concept 2 - Custom RC Controller and Receiver

Despite the downsides, this concept had been decided to be the most adaptable and most suited to this application. It is the most complicated of the two concepts; requiring a broad knowledge of embedded systems and battery management technologies. Moreover, the added complexity of the implementation is outweighed by the added functionality and expandability of this system. It was deemed better to design a system that would properly function instead of trying to adapt the correctly available solutions that may not work correctly for this purpose.

6 Optioneering

6.1 Controller

The STM32L432 was selected as the processed used for the controller device for this project. It was decided upon because it was lightweight and powerful enough for it's indented use case.

Cost was also important consideration when selecting suitable prototypes. This weighting was important to make sure that the project budget was used in the most effecitve way possible. Given that the Mbed OS IDE was free to use software the cost of development was not needed to be taken into account.

The 2 largest competing manufactures within the microcontroller industry, ARM and AVR were

included when making a choice about which proccessor to use. This allowed a full assemenet of the given options to take place. The chosen AVR processor was the ATMega328P, found in the *Arduino Nano*. Given that it was an 8-bit processor it would be expected to perform poorly against the 32-bit *STMicroelectrionics* boards. One advantage of using the the ATMega328P was its ease of use and setup given the large number of libraries avaliable and popularity within the hobbiest community.

Despite this, the STM chip was chosen, this was primarily due the lack of RTOS functionality on the AVR chip. This would not allow the use of multithreading. The Cortex-L4 chips are lightwieght and do not consumer a lot of power, this would allow the remote to be run for a long time, whilst also enabling features such as RTOS support.

Another factor in the critical desicion making process was the ongoing silicon shortage and supply chain issues. This reduced the number of avaliable options for use in the controller.

As a result of the previously mentioned features, the STM32L432 was decided to be the best choice for the controller for use within the controller for Project O.R.C.A.

6.2 Receiver

The STM32F492 was selected for use within the reciever board for this project. This decision making process was conducted in the same manner as the chip selection for the controller. This chip is powerful and includes many inputs and output required to drive the required components for the peripherals.

Power consumption for this board was not an important consideration because it was being powered by the same battery that is used by the motors and other peripherals. The power draw of these components is signicifantly higher than that of the board.

Like with the Controller Optioneering, AVR's ATMega2560 was also considered for use within the reciever. Once again the the RTOS functionality of the STMicroelection chip paired with the use of the Mbed studio IDE showed the STM32F492 to be the clear choice for chip selection. Given that it was also an ARM processor like the controller, it would be easy to develop the software for as it would only require the use of one IDE. This meant that code would be easy to write for one of the board and be ported easily for reuse on another controller without having to worry about architecture clashes.

6.3 Peripherals

6.3.1 RF Transciever

In order to deliver the functionality shown in Concept 2 [5.2.2] , an RF module would need to be sourced to allow for control of the craft at range. The nRF24L01 is used widley within embedded systems as a long range transciever device. Although a basic device it is highly capable of sending limited amounts of data quickly, such as the use case for RC receivers. The nRF24L01 comes as part of a discreet packge that contains all of the components that require it to operate correctly. The discreet package is shown in Fig 3.

Figure 3: nRF24L01 Discreet Package

The nRF24L01 package uses an 8 pin interface for communicating using Serial Parallel Interface (SPI) with an added chip enabled pin, IRQ pin, 3.3V and ground. This was the only component in which a pre-written library was used. This library was complicated and worked well. As this was not the main focus of the project it was decided that this libray would be used, as it would not detreact from the project. This library was written in Mbed OS so it did not require porting over from another architecture.

6.3.2 Brushless DC Motors

The motors for propelling the craft were decided to be brushless DC motors (BLDC motors). This type of motor was chosen primarily for its water-resistance, given that it contains no brushes

unlike a brushed DC motor [2]. BLDC motors are entirely subermergable as they do not have any exposed contacts and are wound using enamelled copper wire. For this project the A2212 1400KV BLDS motors were selected.

Figure 4: A2212 1400KV Brushless DC Motor

The A2212 motors were chosen for this project because of their small size, ease of avaliabilty, and their power output. These motors are capable of 'emptying a sink of water in under nine seconds', so they were deemed to be more than powerful enough for this project, if paired with the correct propeller.

6.3.3 Conveyor Motors

For the conveyor belt motors, there were two inital choices for motors, either stepper motors, or a brushed motor with an attached gearbox. These two types of motor were the obvious choice for driving a low speed high torque conveyor belt. The brushed DC motors were chosen to drive the conveyor belt for two main reasons, cost and easibilty of control. Brushed DC motors can be highly geared to have high torque and do not require any specialised motor controller to drive them, unlike stepper motors.

INSERT FIGURE HERE, TAKE PHOTO OF MOTOR

The motors for the conveyor were decided on being the Dunkermotoren PLG24 DC motor with an attached gearbox with a large reduction. These had an approximate torque of 40Ncm with a

gearbox reduction of 353:1. These motors are designed to run at up to 30V. These would lead to the motors being underpowered for this application but as speed was not the defining factor this was not an issue. The only issue with using brushed motors is that they are not waterproof. This would require a waterproof cowling to cover the motor to ensure that it didn't get damaged by the water.

6.3.4 Motor Drivers

Both the Brushless and Brushed DC motors for this project would require motor drivers to allow the microcontroller receiver to control the motors correctly. Each type of motor requires a different type of motor driver. To drive the BLDC motors, the included 30A 2-3S ESCs were used. These were generic no-name ESC but worked well for the motors and allowed effective control required for this project.

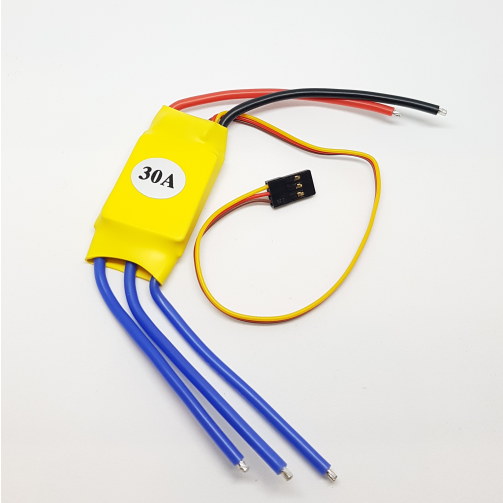

Figure 5: Generic no-name 30A ESC

Unfortunatley, these ESCs did not allow reversal of the motors, this meant an additional method of reversing the craft would be required. The easist solution would be to use two additional motors with revesrsed propellors to drive backwards.

The brushed DC motors are significantly easier to control, requiring a basic DC current to be put across them, as opposed to a complex 3-phase AC current. This mean that just connecting them across the battery would have sufficied to drive these motors forwards. As more control that just on/off was preferable, a motor driver chip would be required to control the movement of these

motors. For this, an L298N driver board was chosen.

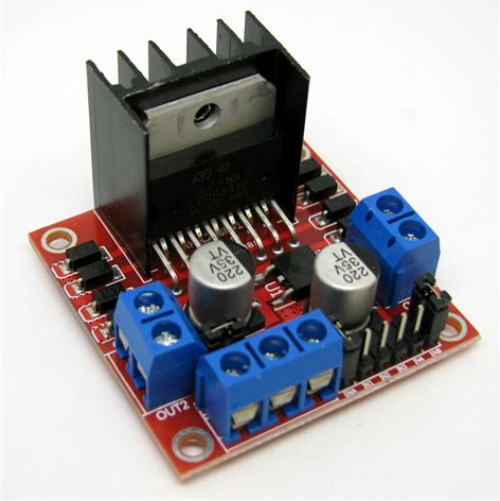

Figure 6: L298N Driver Board for DC and Stepper Motors

The L298N board is able to drive DC motors, reverse their direction, as well as using a PWM signal to control the speed of the motor. These were all prefered features to allow the best control over the conveyor possible.

6.3.5 Batteries

Batteries are one of the most important components for a remote project as they are required to power the entire craft. There also needs to be batteries for the controller, to allow for portability. For the controller Li-Ion batteries, specifcally 18650s, were decided upon as they offered good amount of energy storage within a small enough package to fit in the controller. These batteres are widley used and are readily avaliable. They have a capacity of 2200mAh at 3.7V. Two would need to be placed in series to get the required voltage for the circuit to reliably operate.

Figure 7: 18650 Li-Ion Batteries

The ideal batteries for use within the main craft would have been Li-Po batteries but unfortunatley due to restrictions within the electronincs laboritoriy in SMB303, Li-Po batteries are no allowed to be used for their unstability and fire risk. Because of this, Li-Ion batteries were chosen as a replacement, these were chosen as the next best option for their weight to capacity ratio. Being a prototype the batteries in this craft did not need to be an optimal capcaity. The batteries only needed to last just about long enough to prove the concept and be able to perform testing for the proof of concept. With these considerations in mind, a battery from a robotic vaccuum cleaner was chosen. This battery was 14.4V and 5200mAh. This would provide appromiatley 30 minutes of testing with all features for the craft enabled.

Figure 8: 14.4V 5200mAh Robotic Vaccuum Cleaner Battery Pack

Upon basic inspection the battery pack was made up of two parallel sets of four in series 18650 Li-Ion batteries, the same type of batteries used to power the controller. These batteries are easy to charge and have a higher energy density than Lead acid batteries typically used in car batteries.

6.3.6 IR Distance Sensors

Selecting a suitable IR sensor was relatively simplistic as they all function in the same way, and have similar advantages and disadvantages. As a result of this, a generic no-name IR distance sensor was used. These were inexpensive and functioned in the exact way speicified. The range was adjustable using the small potentiometer on the device. The detection distance was specified to be between 6-30cm, which allowed for ample adjustment to get the right distance.

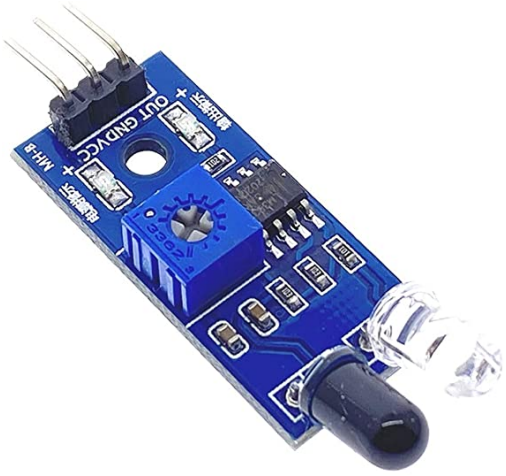

Figure 9: Generic no-name IR Distance Sensor

IR distnace sensors were selected as they were simplistic, cheap and required no further circuitry to drive them, meaning they could just be dropped into the schematic without too many operational concerns.

6.3.7 Voltage Regulators

Given that many components in this project would need to be powered by the same power input, voltage regulators were used. A 12V, 5V, and 3V were need to get all the required input volages for the various components. As the battery voltage fluctuates with varying levels, the linear voltage regulators continue to output the same voltage. This means the components will not be affected by these varying voltages and work reliability.

6.4 Finalised Solution

6.4.1 Block Diagram

The final system level block diagram shown in Figure 10 is the diagram that was decided upon after component sourcing and critial function anaylisis of the system. The diagram contains the elements proposed in Concept 2 show in §5.2.2, with the addition of battery monitoring circuitry for low voltage detection. A table summarising the block diagram is listed below in Table 1.

Component	Device
Controller Microcontroller	STM32L432KC*
Receiver Microcontroller	STM32F429ZI
Brushless DC Motors	A2212 BLDC Motors
BLDC Motor ESC	Generic no-name 30A ESC
Brushed DC Motor	Dunkermotoren PLG24
Brushed DC Motor Driver	L298N
Controller Batteries	18650 Li-Ion Batteries
Buzzer	Generic no-name buzzer
12V Regulator	12V Fixed LDO*
5V Regulator	12V Fixed LDO*
3V Regulator	12V Fixed LDO*

*selected due to constrains caused by silicon shortage.

Table 1: Component Selection

Details on how the controller microcontroller's software and hardware was developed and interfaced within the scope of the project can be found in §**??** and §**??** respectively.

Details on how the receiver microcontroller's software and hardware was developed and interfaced within the scope of the project can be found in §**??** and §**??** respectively.

Details on how the ESCs were interfaced with can be found in §**??**.

Details on how the L298N was interfaced with can be found in §**??**.

Detailed on how the 12V, 5V, and 3V Regulators were interfaced with can be found in §**??**.

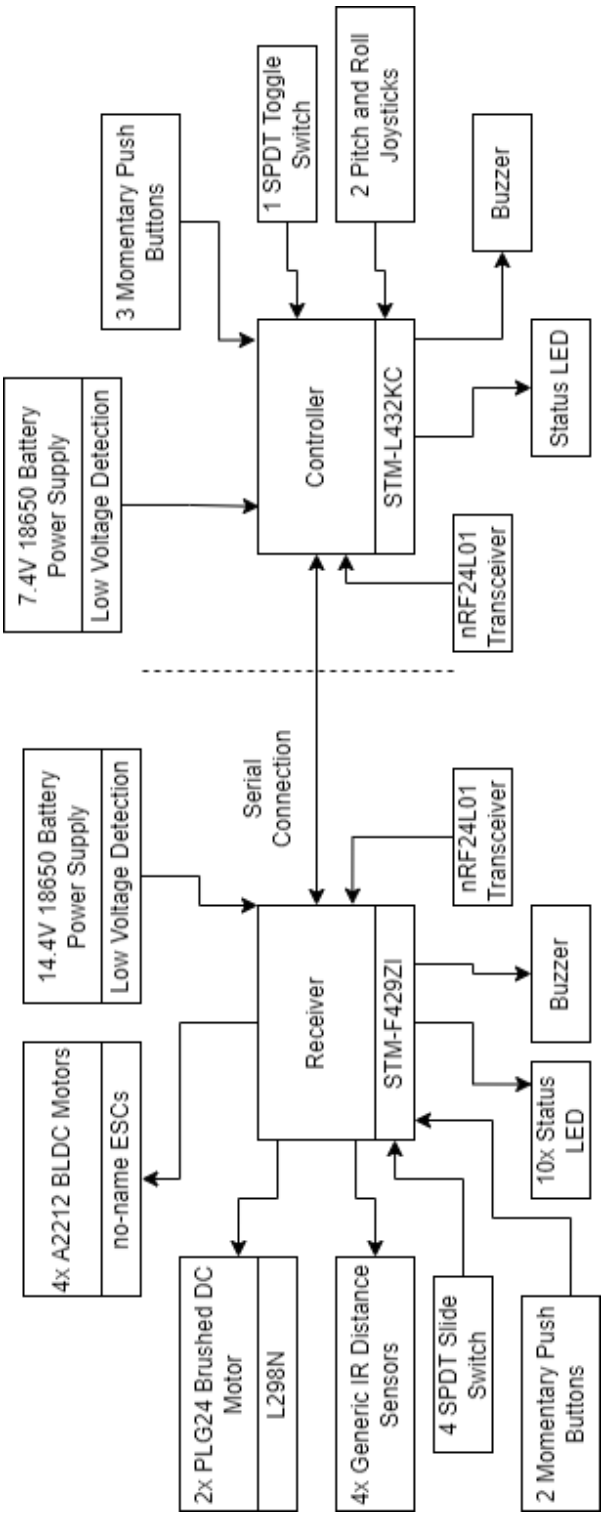

Figure 10: Final System Block Diagram of ORCA

6.4.2 Bill of Materials

The bill of materials is listed in Appendix **??**. Whilst the budget from the University was set at £100, this exceeded that amount. Given that most of the materials for this project had been recycled, the actual budget for the project is less than shown in the bill of materials.

The high cost of this project is due in part to this project being a prototype, and in part to the custom parts that needed to be manufactured for this project. Components are always expected to be more expensive at the prototyping scale of production.

Materials were, for the most part, sourced from responsible VAT registered distributors; which was important when selecting components for mass production. The majority of the components were sourced from Farnell, DigiKey US, CPC, or Ooznest. Some of the other components were sourced from Amazon and eBay. These parts could not be varified to be from a responsible distributor, but these components could have been sourced from other locations if the quantity of the components was larger.

The fabrication of custom SMD PCBs has been included within the BOM for this project as they were considered to be part of the final prototyped solution. Despite these not being a finalised PCB design for further manufacturing, if applicable, it was a cost that was considered during the prototyping phase of the project. The PCBs were important in the production of this project as they allowed the secure mounting of the RF modules. Given that all of the components needed mounting on a structure that would need to be remotely controlled and manoeuvred, the decision to make PCBs was clear. The PCBs were manufactured in Shenzhen, China but JLC-PCB.

The project was concluded within time scale set but with a slightly expanded budget. Given further time, this project could have been completed closer to the original budget. Due to the expensive nature of some of the components used however, the budget of £100 would likely have not been sufficient for this project without significant cuts to functionality and/or goals.

7 Software Design

7.1 Controller

7.1.1 Architecture

The Controller is entirely multi-threaded. The decision was taken to use multithreaded architecture due to the nature of the system being used; monitoring and controlling sensors with interrupts coming from multiple sources. A single threaded architecture was initially used during initial testing. This was found to be impractical as the statemachine used became cumbersome. When communicating using RF the controller could not respond to any other commands from any other devices. This may not have been quick enough to react to any other inputs from the user.

Because of this, the controller architecture was split into threads, listed in Table 2. Each entry in the table is represented by a thread in the program that is initialised during start-up.

Thread Name	Function
Pot Thread	Polls Values of the Joysticks and sends over RF
Input Thread	Used to handle inputs when flags raised

Table 2: Threads for Controller

7.1.2 Serial Communications

The serial communication for this project was done over SPI using nRF24L01 modules to send data. As there was already an nRF24L01 library written for Mbed [3]. This abstracted the serial interface from the user to just writing to the RF devices, and allowing the library to handle the rest.

This library was externally written and whilst it was functional, it was a bit rough around the edges and felt, in parts, to be unfinished. Whilst this functionality was not used, it would have been nice to have a complete library. Despite this, the library was operational and only required the data to be sent across using char arrays, whilst fiddly this was not too difficult to achieve.

This library could have been re-written to include its missing functionality, but this would have taken up a significant amount of the project given that the serial/RF communication was not the main objective.

Despite the class being functionality incomplete, it was easy to use for sending data using radio. Code 1 shows how to create the object for the nRF24L01 Class. It requires the use of an SPI interface on the board.

Code 1: nRF24L01 Class Constructor

```
#include <nRF24L01P.h>
nRF24L01P radio(MOSI, MISO, SCK, CSN, CE, IRQ);
```

Once the object is declared, the code in Code 2 is then used to set up the nRF24L01 so that it can be used correctly.

Code 2: nRF24L01 Set Up Transmit Mode

```
radio.powerUp();
radio.setTransferSize(TRANSFER_SIZE);
radio.setAirDataRate(NRF24L01P_DATARATE_2_MBPS);
radio.setRxAddress(NRF24L01_RX_ADDRESS);
radio.setTxAddress(NRF24L01_TX_ADDRESS);
radio.setTransmitMode();
radio.enable();
```

Code 3 shows how the data is written to the nRF24L01 module. This requires the data to be in a char array format, ideally the module would allow strings and integers to be passed to it, but the use of char arrays was acceptable.

Code 3: Sending Data Using nRF24L01

```
radio.write(char *data, int pipe, int count);
```

The serial communication was conducted using custom transmission protocols, outlined in Appendix C. These were designed to be short, 5-bits, and easy to extract the data from on the other end. This system was used to make the transfer of data to faster and efficient with hopefully minimal packet loss.

7.1.3 Buzzer API

The buzzer used for this project is a standard piezo buzzer that required a Pulse Width Modulation (PWM) pin to drive at a given frequency. To make the buzzer audible and fit within the

western standard of music notation, a class was written to to encapsulate the buzzer functionality, allowing the buzzer to be easily used.

This buzzer class was inspired by the University of Plymouth's buzzer class written for the Module Support Board.

This class has been written such that it is thread safe. The object-oriented nature of the class has allowed the class and code to be utilised on both the controller and receiver, both of which utilise buzzers as a method of alerting the user.

Code 4: Buzzer Class Constructor

```
1 #include "Buzzer.h"
2 Buzzer buzzer(BUZZER);
```

Using this Application Programming Interface (API), the buzzer could be turned on or off, the note can be changed, and the octave the note is in can also be changed. This allowed multiple different alerts to the user to be utilised. This increased the versatility of the buzzer.

This class includes operator overloads such that using the buzzer is easy and intuitive.

7.1.4 Main Code

The main code on the controller board runs in two separate threads, one for monitoring the button and switch states, and one for monitoring the joysticks and extracting their relative positional data.

They three buttons and switch are all driven using interrupts. When the interrupts are triggered, the Interrupt Service Routine (ISR) raises a thread flag which wakes up the thread. The thread then handles this flag then going back into the waiting state where the thread is put to sleep and not scheduled until another event flag is triggered.

Being an analogue inputs, the joysticks cannot be run using the same interrupt based methodology. The joysticks were run using a ticker. The ticker is set to create an interrupt every 200ms and runs the joystick ticker ISR. This wakes up the joystick thread, polls the joystick values, if required it sends the values of SPI to the nRF24L01 then puts the thread back to sleep until the next ticker interrupt.

If the joystick values have not changed, the new joystick values get stored and then the thread

goes back to sleep. This helps reduce power usage and RF usage when unnecessary.

Code 5: Joystick Value to Serial Data to be Sent

```
1 newLeftPitchVal = abs(newLeftPitchVal);
2 sprintf(tempThrottleChar, "%d", newLeftPitchVal);
3 fwdLeftPitch[3] = tempThrottleChar[0];
4 radio.write(fwdLeftPitch, DEFAULT_PIPE, TRANSFER_SIZE);
```

Code 5 shows the method of tansfering the data from integer form into the char array that is required to send data over the nRF24L01 device. This is an example of how this library could have been improved for this project. If the nRF24L01 library was able to accept integers to be sent, the code on the controller could have been simpler and easier to understand.

Despite its complexity, the code for extracting the joystick values worked repeatable and did not cause any detectable false readings.

7.2 Receiver

7.2.1 Architecture

The Receiver is entirely multi-threaded. The decision was taken to use a multithreaded architecture due to the nature of the system being used; controlling multiple real time devices that, many of which do not need rapid periodic updating. A single threaded architecture was initially utilised during testing. This was found to be impractical as the number of inputs and outputs became unmanageable using rapid polling. Reading from the RF module is a blocking operation. This would cause the program to halt when fetching and reading the data from the RF module. This could cause updates to the motor speeds and other sensors to be missed, potentially causing delays to the real time system.

Because of this, the architecture was split into threads, listed in Table 3. Each entry in the table is represented by a thread in the program that is initialised during start-up.

Thread Name	Function
Left Motor Thread	Controls throttle Values of Left Motor
Right Motor Thread	Controls throttle Values of RIght Motor
Radio Thread	Used to handle incoming RF Data
LED Thread	Used to change the LED status'
IR Thread	Used to handle interrupt from when litter collection basket is full
Input Thread	Used to handle inputs when flags raised
Battery 30 Percent Thread	Used to raise warning when battery is at 30 percent
Battery 15 Percent Thread	Used to raise warning when battery is at 15 percent

Table 3: Threads for Receiver

7.2.2 Serial Communications

The serial communication for this project was done over SPI using nRF24L01 modules to send data. An overview of the serial communication can be seen in §7.1.2.

The main difference in serial communication on the receiver, is that it primarily uses receive mode. Therefore the setup for the RF module is almost identical, except that it is initialised into receive mode.

Code 6: nRF24L01 Set Up Receive Mode

```
1 radio.powerUp();
2 radio.setTransferSize(TRANSFER_SIZE);
3 radio.setAirDataRate(NRF24L01P_DATARATE_2_MBPS);
4 radio.setRxAddress(NRF24L01_RX_ADDRESS);
5 radio.setTxAddress(NRF24L01_TX_ADDRESS);
6 radio.setReceiveMode();
7 radio.enable();
```

The serial communication on the receiver board also used the transmission protocols, outlined in Appendix C. This allows the boards to communicate effectively.

To handle the incoming transmission codes, a separate radio thread is used, as shown in Table 3. This thread does not cause any external blocking behaviour to any other threads. This allows the system to remain real time.

7.2.3 Buzzer API

The buzzer API used for the receiver is the same as used within the controller. This can be found at §7.1.3.

7.2.4 L298N API

The DC motors were driven using an off the shelf L298N motor controller. This required a simple API to be written. This was to make the motors easier to control by abstracting functionality away from the user.

This class has been written such that it is thread safe. The object-orientated nature of the class would allow the class to be used on any Mbed project that utilises an L298N to drive DC motors.

This API uses four arguments, a motor A, a motor B, an enable, and a motor enabled. The motor enabled argument has a default value so it does not need to be used. If unused, the value will default to 0. This can be seen in Code 7.

Code 7: L298N Class Constructor

```
#include "L298N.h"
L298N DCMotor1(PinName motorA, PinName motorB, PinName enable, int
    motorEnabled = 0);
```

Using this API, the motor can be enabled, disabled, and the direction can be set and read. These methods can be seen in Code 8.

Code 8: L298N Class Methods

```
    void setDirection(int direction);
    void enableMotor();
    void disableMotor();
    int direction();
```

Given the nature of the motors being controller, basic DC motors, the control was trivial. Despite this, abstracting this control made the motors easier to control within the main body of the code.

7.2.5 ESC API

Unlike the brushed DC motors, brushless motors require an Electronic Speed Controller (ESC) to operate. This requires more complex control to make the motor move. The ESCs require a PWM signal that at 50Hz that varies between 1ms-2ms. This is similar to a servo control signal using a PWM output [4]. This class was inspired by the control system for servos.

This class has been written such that it is thread safe. The object-orientated nature of the class allows the class to be used on any Mbed project that utilises an ESC that requires a 50Hz signal with a duty cycle varying between 1ms-2ms.

The ESC API uses two arguments, the PWM pin, and calibrate. The calibrate argument has a default value so this does not need to be used. If unused, the value will default to 0. This can be seen in Code 9.

Code 9: ESC Class Constructor

```
#include "ESC.h"
ESC brushlessMotor(PinName pin, int calibrate = 0);
```

Using this API the motors can be set a certain speed, have the PWM value altered, and the value of speed can be read. This can be seen in Code 10.

Code 10: ESC Class Public Methods

```
void write(float speed);
void setPWM(int PWMValue = DEFAULT_PWM_VALUE);
float speed();
```

This API also contains private members, these are used to connect the ESC, calibrate the ESC, and normalise the speed values. These can be seen in Code 11.

Code 11: ESC Class Private Methods

```
float normalise(float speed)
void connect();
void calibrate();
```

The normalise method is used to ensure that a value between 0-1 is written the ESC. If the value was not between these values, the ESC may not behave as expected.

To connect the ESC the throttle value is set to its minimum value then held there until the ESC beeps. This can be a varying amount of time therefore 5s was chosen as this would be long enough for the ESC to connect. The calibration of the ESC is more complex. It requires the throttle value to be at its maximum value for 4s, then at it's minimum value for 4s. This allows the ESC to 'learn' the range of values that it is expecting as an input. This calibration should only be required for the first time the ESC and board are connected together, after this, the board remembers the previous values.

7.2.6 Main Code

The main code on the receiver board runs in eight threads. These have been outlined in Table 3. The left and right motor threads, are for controlling the speed of the motors, the radio thread is for reading the serial data and either raising the appropriate flag for that action of performing an action dictated by the transmission codes. The LED thread is used for turning the LEDs on and off in a thread safe way that can be handled when the scheduler is available to, as these LEDs are less important than the functionality of the motors. The tother remaining four threads are all controlled by flags controlled by interrupts. This allows the inputs to be responsive whilst allowing the system to remain real time.

When the Infra-Red (IR) sensors get triggered, the ISR is triggered raising a flag to alert the board that the collection basket is full. This activates the IR thread which disables the IR ISRs and triggers the buzzer periodically.

When the battery level is running low, the battery low input pins are pulled high. This triggers an ISR which periodically sounds the buzzer,. This alerts the user that the battery is running low.

Table 4 shows the functionality of the buttons and switches on the receiver board.

Input Name	Function
Button 1	Disable Alarms and deactivate IR sensors
Button 2	Litter basket emptied, Resets Sensors and Enables Buzzer
Switch 1	Disable Motors
Switch 2	Enable Motors
Switch 3	Not in use
Switch 4	Not in use
Switch 5	Not in use
Switch 6	Not in use

Table 4: Functionality of Buttons and Switches in the Main Receiver Code

8 Hardware Design

8.1 Prototyping

The initial prototyping started during Phase 2 (§4.2) of the project. The initial prototyping goals were discussed and clearly laid out before any work began. The primary object of the prototyping stage was to create a small scale model craft that fits into testing tank eg. a bath. After that, it would need to be controlled remotely using an RC controller. The third aim of prototyping was to investigate how to make a conveyor system working and what the advantages and disadvantages of this system may be. The final aim was to create an approximately 1:1 scale cardboard prototype to do a variety of tests on.

8.1.1 Small Scale Design

The small scale prototype design was made using milkshake bottles and 3D printed Polylactic Acid (PLA) parts. This design was christened the Milkshake Catamaran. The design for this can be seen in Figure 11.

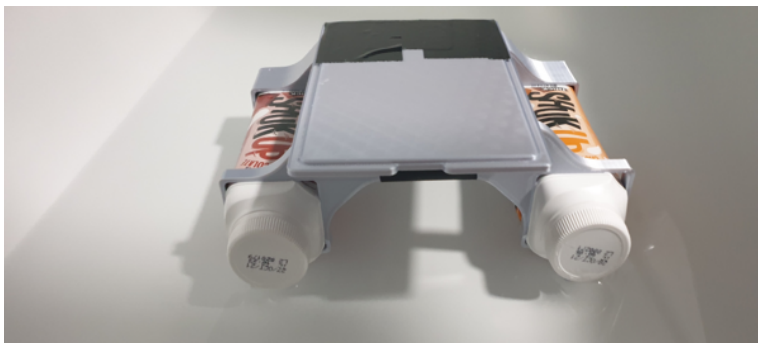

Figure 11: Milkshake Catamaran Prototype

This design was tested for buoyancy and stability in the bath. Being a Catamaran design, the craft was extremely stable and well balanced. This was encouraging for the future developments of this project.

Once the base catamaran had been tested, RC electronics were added. This enabled the craft to be controlled. This was used to test propulsion systems. The fully loaded Catamaran can be seen in Figure 12.

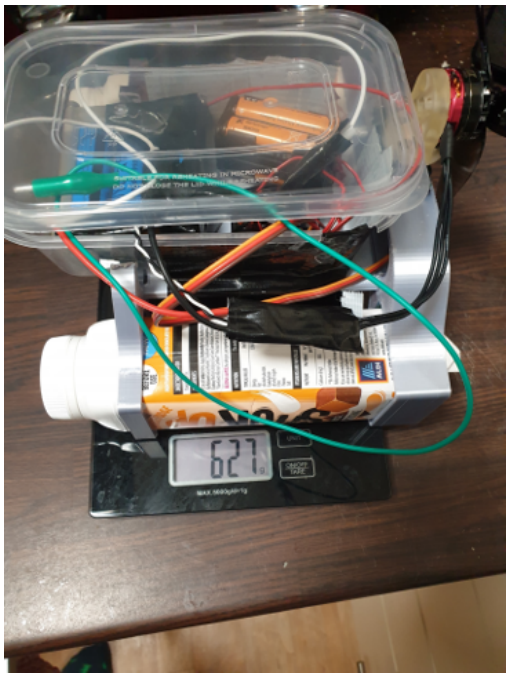

Figure 12: Milkshake Catamaran with RC electronics

From this testing it was discovered that the maximum capacity of this craft was about 670g. This was not surprising as each of the milkshake bottles was 335ml, so the total mass of displaced

water was 670g. Given that the weight of the craft including the basic electronics system was 627g, there was very additional capacity for the craft to carry any litter.

From this discovery, it was decided that the scale of this craft would need be larger to accommodate a heavier electronics system, batteries, and the additional overhead to carry any litter that it may collect.

8.1.2 Conveyor Research

Early in the project it was decided that a conveyor would be the preferred method of collecting litter. A conveyor belt would provide a suitable level of challenge whilst adding some additional novelty to the design. A conveyor belt would also allow the collected litter to be removed from the water sooner and removing the litter from the craft will not require the whole craft to be removed from the body of water it is operating in.

One of the biggest issues when looking at the conveyor was the material of the belt. This was vital to get correct in order to create an effective conveyor belt. Table 5 shows the pros and cons of each of the considered materials.

Material	Pros	Cons
Linoleum	• Waterproof • Cheap • Moderately Flexible • Non-reactive	• Heavy • Coefficient of friction decreases when wet • Expensive
Cargo Netting	• Strong • Waterproof • Light • Cheap • Non-reactive	• Litter may get caught • Hard to get into a strip • Easy to get caught in mechanism
Plastazote	• Waterproof • Cheap • Lightweight • Easy to source • Non-reactive • Very Flexible • Coefficient of friction increases when wet	• Stretches and Deforms

Table 5: Material Considerations for Conveyor Belt

It was ultimately decided that Plastazote would be used for the conveyor belt. It was overall the best material for the job. it had all the best aspects of the other two evaluated materials but without their negatives.

Plastazote is easy to machine. Adding parts to the conveyor belt was as easy as poking a hole in the material with a round file and pushing through a bolt. It was a mixture of this along with the points highlighted in Table 5 that made the decision to use this material such an obvious one.

8.1.3 1:1 Scale Prototype

Figure 13: 1:1 Scale Cardboard Prototype Hull

Figure 14: 1:1 Scale Cardboard Prototype Hull in Water with Additional Load

References

[1] S. Sharma, D. Sarkar, and D. Gupta, "Agile processes and methodologies: A conceptual study," *International Journal on Computer Science and Engineering*, vol. 4, 05 2012.

[2] P. Millett. (2021) Brushless vs brushed dc motors: When and why to choose one over the other. [Online]. Available: https://www.monolithicpower.com/en/brushless-vs-brushed-dc-motors

[3] O. Edwards. (2011, Jan) Brushless vs brushed dc motors: When and why to choose one over the other. [Online]. Available: https://os.mbed.com/users/Owen/code/nRF24L01P/

[4] W. Zhang, W. Wang, N. Ding, and C. Li, "The main-loop and servo-loop structure control strategy of esc," *Chongqing Daxue Xuebao/Journal of Chongqing University*, vol. 35, pp. 19–24, 07 2012.

A Design Requirements Table

Ref	Requirement	Required?
1	Main body of the craft	Yes
2	Suitable Propulsion Method	Yes
3	Collection System for the litter	Yes
4	Custom remote controller and receiver	No - highly desirable
5	Battery powered	Yes
6	Suitably Waterproofed	Yes

B Software Design Requirements Table

Ref	Requirement	Required?
1	Suitable control system for movement of the craft	Yes
2	Suitable control system for collecting litter	Yes
3	Sense when rubbish collection basket is full and alert the user	Yes
4	Sense when the batteries (on either craft or remote) is running low and alert the user	Yes
5	Suitable communication system between the remote and receiver	Yes
6	Relay information from the receiver to the controller about battery voltage and time of current usage, etc.	No - nice to have
7	System to track amount of rubbish collected	No - nice to have
8	Calculations of carbon nuetrality of the collected litter, how much litter would need to be collected to make the craft environmentally viable	No - nice to have

C Transmission Codes

The 1 bit denotes the type of device.

The 2 bit denotes the component in that device type.

The 3 and 4 bits denote the specific device instruction.

- 1xxx - POTs
  - x1xx - Left Pitch
  - x2xx - Left Roll

  - x3xx - Right Pitch
  - x4xx - Right Roll
    - xx1x - Forwards
    - xx0x - Reverse
      - xxx0 - Value of throttle

- 2xxx - Buttons
  - x1xx - Button 1
  - x2xx - Button 2
  - x3xx - Button 3

- 3xxx - Switches
  - xxx1 - Switch On
  - xxx0 - Switch Off