

Robust Ultrasonic Skittle Hunter

(RUSH)

Technical Documentation File

Samuel Kliskey 202028443

Ross Inglis 202032175

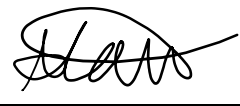
Andrew Law 202009334

Signature: 

Date: 08/05/24

Signature: 

Date: 08/05/24

Signature: 

Date: 08/05/24

We hereby declare that this work has not been submitted for any other degree/course at this University or any other institution and that, except where reference is made to the work of other authors, the material presented is original and entirely the result of our own work at the University of Strathclyde



Times Higher Education University of the Year 2012 & 2019
Times Higher Education Widening Participation Initiative of the Year 2019
The University of Strathclyde is rated a QS 5-star institution

WINNER
UK UNIVERSITY
OF THE YEAR
FOR A SECOND TIME



THE QUEEN'S
ANNIVERSARY PRIZES
FOR HUMAN AND FREEDOM EDUCATION
2019

Contents

1	Introduction.....	1
2	User Guide	2
3	Hardware Overview	5
3.1	Electronic Peripherals	5
3.1.1	Ultrasonic Sensor (Distance).....	5
3.1.2	Infrared Sensor (Colour)	7
3.1.3	Microcontroller	7
3.1.4	Motor Driver	8
3.1.5	Servo	8
3.1.6	Power Source	8
3.1.7	Control Board.....	10
3.1.8	Low Battery Indicator	10
3.2	Structure and Mounting.....	11
3.2.1	Disassembly Instructions	11
3.2.2	Wiring Harness Assembly	15
4	Software Overview	17
4.1	General Functionality	17
4.2	Individual State Complete Functionality	19
4.2.1	Can Calibration	19
4.2.2	Ready.....	19
4.2.3	Search.....	20
4.2.4	Can Align	21
4.2.5	SONAR Scan	21
4.2.6	Can Approach.....	22
4.2.7	Colour Detect	23
4.2.8	Reverse Car	23
4.2.9	Can Hit	23
4.2.10	Wall Realign.....	23
4.2.11	Circumnavigate	23
4.3	Scheduler and Event Flags	24
4.4	Microcontroller Utilisation.....	25
4.5	Peripheral Control	26
4.5.1	Ultrasonic	26
4.5.2	Infrared	27
4.5.3	User Interface	27
4.5.4	Servo Motor	27
4.5.5	DC Motor Control	28
5	Tuning Parameters for Performance	29
6	Troubleshooting	30
6.1	Visual Inspection.....	31
6.2	Low Battery Check	31
6.3	Hardware Tests.....	31
6.4	Ultrasonic Error Diagnosis.....	34
6.5	Infrared (IR) Error Diagnosis.....	36
6.6	Motor Error Diagnosis	37
6.7	Servo Error Diagnosis	38

6.8	Indicator LED Error Diagnosis	39
6.9	Button/Switch Error Diagnosis	39
6.10	General Error Diagnosis.....	39
7	Design Validation.....	40
8	Summary.....	40
9	Future Improvements	41
9.1	Improved Scanning Sweep.....	41
9.2	Realignment	42
9.3	Improved steering.....	42
9.4	Pre-game Search Mode	42
9.5	Improved Power source.....	43
10	References	44
11	Appendix	45
11.1	PCB Schematic	45
11.2	Bill of Materials (BoM)	46

Nomenclature

BoM	Bill of Materials
GPIO	General Purpose Input / Output
IC	Integrate Circuit
IR	Infrared
LED	Light Emitting Diode
MSP	Mixed Signal Processor
NiMH	Nickel Metal Hydride
PCB	Printed Circuit Board
PLA	Polylactic Acid
RC	Remote Control
RUSH	Robust Ultrasonic Skittle Hunter
SONAR	Sound Navigation and Ranging
ToF	Time of Flight
uC	Microcontroller

List of Figures

Figure 1: RUSH viewed from front (left) and rear (right)	1
Figure 2: Base RC car before modification [1]	1
Figure 3: Setup Diagram	2
Figure 4: RUSH on start line.....	2
Figure 5: Calibration Walls setup to target black skittle on right.....	3
Figure 6: Power Switch.....	3
Figure 7: Right wall alignment chosen	4
Figure 8: Car calibrated and ready to start	4
Figure 9: Diagram of HC-SR04 Ultrasonic sensor	5
Figure 10: Ultrasonic Detection Scope	6
Figure 11: Diagram of IR sensor.....	7
Figure 12: MSP430G2553 IC (left) [2] , MSP430G2ET Development board (right) [3]	7
Figure 13: Diagram DRV8833 Dual H-Bridge Motor Drive of [4]	8
Figure 14: SG90 servo motor [5]	8
Figure 15: Base RC car steering adjustment.....	9
Figure 16: Single cell Alkaline vs NiMH Voltage discharge graph [6].....	9
Figure 17: AMS1117 3.3 V Regulator Module	10
Figure 18: Control Board PCB top (left) and test points (rear).....	10
Figure 19: Low Battery Indicator Circuit on Control Board under motor driver (left) and circuit diagram (right)	11
Figure 20: Top SONAR Ultrasonic structure Disassembly Diagram	12
Figure 21: Top SONAR servo structure Disassembly Diagram	12
Figure 22: Upper Main Structure Disassembly Diagram.....	13
Figure 23: Ultrasonic Sensor in Upper Main Structure Disassembly Diagram	13
Figure 24: Control Panel structure Disassembly Diagram.....	14
Figure 25: Stages of wiring harness production: strip insulation (left), first stage crimp (middle) and last stage crimp (right)	15
Figure 26: Functional State Diagram	17
Figure 27: Can Calibration State Illustration	19
Figure 28: Align to Wall Decision Making	20
Figure 29: Left Can Confirmation & Alignment	21
Figure 30: SONAR Anomaly Detection	22
Figure 31: Can Approach with Locking on to Can	22
Figure 32: Circumnavigate Movements.....	24
Figure 33: Ultrasonic Reading Control	26
Figure 34: IR Control.....	27
Figure 36: Output of Timer Linked Pin in Mode 7 [9]	28
Figure 39: Expected flow of testing	30
Figure 40: Battery Voltage Measurement Points.....	31
Figure 41: Wiring harness (left), Programmer PCB connection (middle) and dev. board connection (right).....	32
Figure 42: Ultrasonic connections on control board.....	35
Figure 43: Expected Ultrasonic Trigger waveform.....	35
Figure 44: Expected ultrasonic echo waveform at sensor (left) and on control board (right).....	36
Figure 45: Motor driver control panel test points	37
Figure 46: Motor PWM Control signal waveform.....	37

Figure 47: Servo test points on control panel	38
Figure 48: Servo PWM control signal at maximum and minimum angle	38
Figure 49: Skittle detection with Ultrasonic and Time of Flight (TOF)	41

List of Tables

Table 1: HC-SR04 Ultrasonic Sensor Specifications.....	6
Table 2: Wiring Harness Connection Specifications.....	16
Table 3: MSP430 Functionality and Purpose.....	25
Table 4: Ultrasonic Testing expected resistances	34

1 Introduction

This technical specification document details all information regarding **RUSH (Robust Ultrasonic Skittle Hunter)** depicted in Figure 1. RUSH aims to autonomously play a game of skittles in a corridor without user input (after setup) against an opposing car. RUSH is based on a small, simple remote controlled (RC) car that was retrofitted to play the game and therefore RUSH will commonly be referred to as the “car” in this document. The aim of the game is to navigate to and knock over one or two black skittles whilst leaving any white skittles untouched. The winner of the game is determined by the quickest car to knock a black skittle over. However, if a white skittle is knocked over at any point during the two minute round by the car, it is disqualified from the round. The skittles were constructed of two stacked and painted soup cans and are therefore often referred to as “cans” throughout.

This document covers all aspects of RUSH including a quick start user guide and hardware / software descriptions. Also included is detailed guidance of how to perform troubleshooting and repair should the car break or behave unexpectedly. This technical specification is intended for anyone with basic technical knowledge that intends to use the car to play autonomous skittles, understand how it works, aims to fix any issues that occur or make further improvements.

The base RC car (see Figure 2) that RUSH is based on was to be retrofitted with an array of sensors to accomplish the goal while adhering to the guidelines stating no alterations to the chassis or drive train.

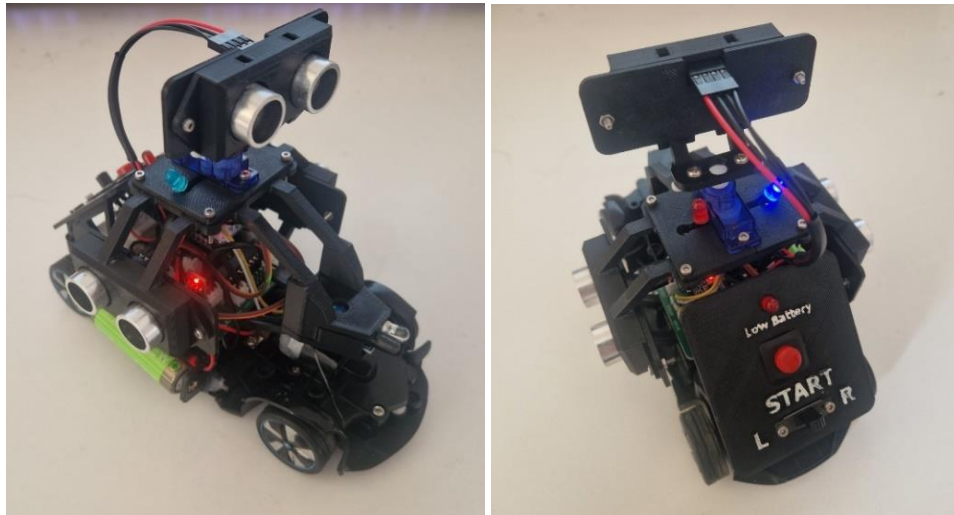


Figure 1: RUSH viewed from front (left) and rear (right)



Figure 2: Base RC car before modification [1]

2 User Guide

This user guide details instructions on how to quickly setup RUSH for a game of skittles. For the car to function, the game must be setup in a long straight corridor. The corridor should be at least 10 m long with an expected distance from the start line to the first skittle of at least 6 m. The walls of the corridor should be unbroken and not have any alcoves or adjoining corridors in the game space as this will disrupt the cars' function. Some tolerance of wall geometry changing is accounted for in software but it performs best where the wall is flat and performs worse if the wall has multiple changes close together or passes an adjoining corridor. The setup of the car in the corridor should be roughly as seen in Figure 3.

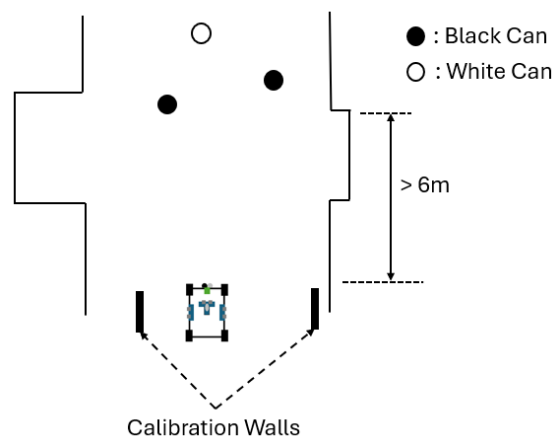


Figure 3: Setup Diagram

To begin the game, follow the steps below:

1. **Place RUSH on start line:** The car should be facing towards the game area with the skittles, roughly in line with the black skittle that would be aimed for. In the example in Figure 4, it has been aimed at the black skittle on the far right.

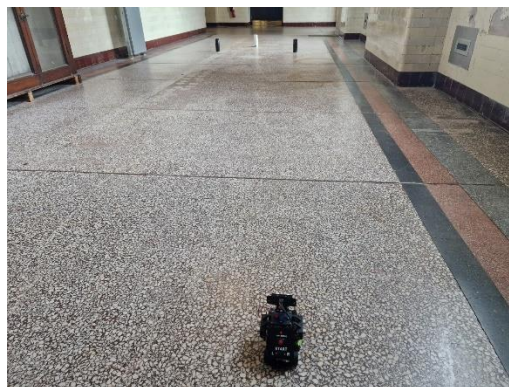


Figure 4: RUSH on start line

2. **Setup calibration walls:** The provided calibration walls should be setup either side of the car in such a way that they are just beyond the skittle(s) that are aimed for. Construct the walls by slotting the cardboard into the provided feet for stability. These walls are used in a calibration reading to define the maximum distance at which the car will detect a skittle either side of the car. Anything too far from the car at the side will not be detected as a skittle. These are also set to limit the cars side view to not detect walls as skittle. Figure 5 shows the calibration walls setup to ensure only the far-right black skittle is within limits and will therefore be targeted.

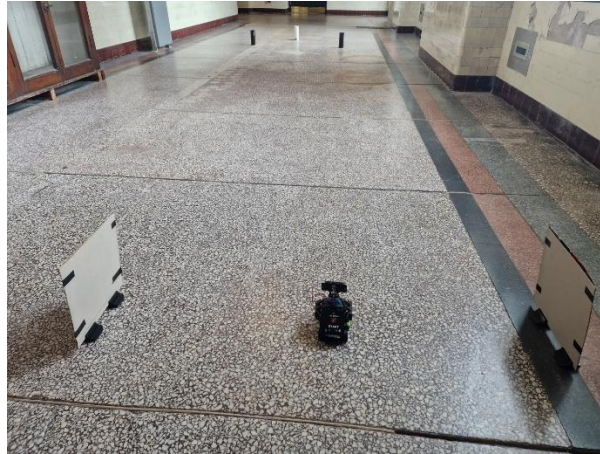


Figure 5: Calibration Walls setup to target black skittle on right

3. **Turn on car:** Using the switch on the underside (see Figure 6), turn on the car. One LED should light with this indicating whether the left or right wall is currently selected for alignment. Ensure at this stage that the top ultrasonic sensor is pointing forwards. If not, this will need to be reattached pointing in the correct direction (as detailed in Section 3.2.1). If the low battery indicator is illuminated at this point, the batteries should be removed and recharged before continuing. Note that AA Alkaline batteries cannot be used as an alternative.



Figure 6: Power Switch

4. **Select wall for alignment:** Using the switch on the control panel, positioned between ‘L’ and ‘R’, select which wall the car will use to align to as it attempts to travel down the corridor towards the skittles. The wall should be chosen depending on distance to car and how varying and ‘flat’ it is. The car will tend to align better with a closer wall that does not vary much. The wall chosen should also ideally not have the opposition car in between. In the example in Figure 7, the right wall is chosen for alignment as it is slightly more consistent than the left wall.

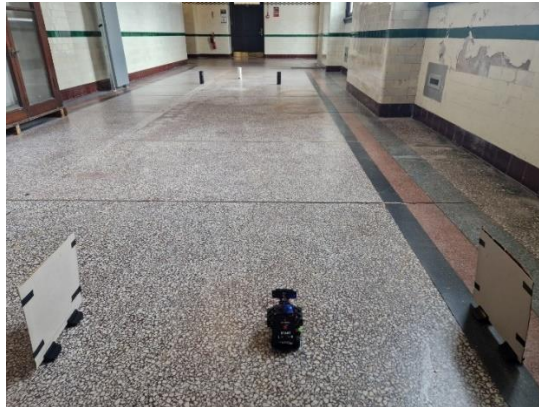


Figure 7: Right wall alignment chosen

5. **Press Start button once:** This stage performs the calibration with the walls placed previously. After the button is pressed, both LEDs should illuminate (as seen in Figure 8), indicating the car is ready to race. The calibration walls do not need to be removed.

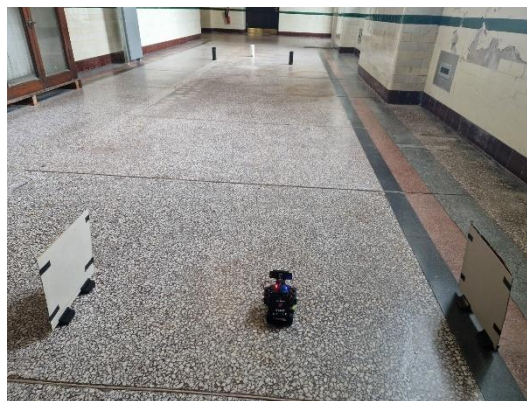


Figure 8: Car calibrated and ready to start

6. **Race!** When ready to race press the start button once more and watch the car begin the game.

Expected Operation: The car will travel forwards at a fast speed making small alignments to stay straight in reference to the selected side wall. In this stage no skittles can be detected. When both red and blue indicator LEDs illuminate, the car is in search mode and will slow down. In this mode it will detect any skittles (provided it has been calibrated correctly). At this stage the car will stop upon detecting a skittle and then slowly manoeuvre towards it. Once reaching it RUSH will determine its colour, driving into it and knock it over if black or move around it if white.

3 Hardware Overview

The base RC car contained the mechanisms and motors for both steering and driving. For the car to be able to manoeuvre down the corridor, detect the skittles, determine the colour, and knock it over if black, it needed to be modified with an array of sensors and control systems.

For control of all functionalities of the car, an MSP430G2553 microcontroller is used in conjunction with the bespoke PCB control board. All sensors and electrical systems externally connected to this control board will be referred to as “peripherals” throughout this document.

To detect the cans and align with the wall, three ultrasonic distance sensors were used (two side facing, one front) as seen in Figure 1. To determine the colour of the cans, a simple infrared sensor was used.

This section of the document contains detailed descriptions of how all the peripherals operates as well as the structure and wiring used to connect them all together.

3.1 Electronic Peripherals

3.1.1 Ultrasonic Sensor (Distance)

The HC-SR04 ultrasonic distance sensor was chosen due to its low cost, ease of use and wide availability. It emits ultrasonic waves and measures the time taken for the waves to bounce back from the object, therefore distances can be measured all the while without interference from ambient light like infrared (IR) based distance sensors. A pinout diagram of the HC-SR04 ultrasonic sensor is shown below in Figure 9.

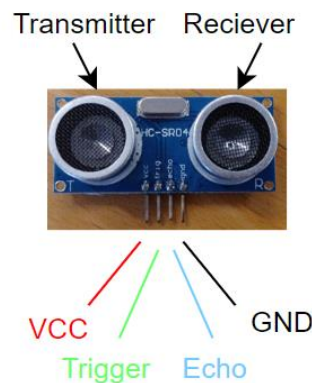


Figure 9: Diagram of HC-SR04 Ultrasonic sensor

The specifications of the sensor can be found in Table 1. Note that although the ultrasonic sensor is rated to operate at 5 V, it was tested to consistently operate down to approximately 4.3 V.

Table 1: HC-SR04 Ultrasonic Sensor Specifications

Specification	Value
Operating Voltage	5 V
Silent Current draw	2 mA
Operating Current draw	15 mA
Measurement Range	2 cm to 4 m
Detection Scope	30°

Due to the nature of ultrasonic sound waves and the sensor specifications, the detection scope of the sensor is relatively large at up to 30° as depicted in Figure 10. This means that the sensor can detect objects which are not directly ahead of it.

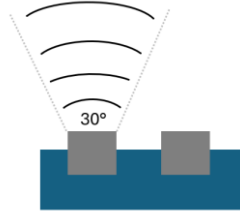


Figure 10: Ultrasonic Detection Scope

For the sensor to take a distance measurement, a short pulse of at least 10 μ S must be applied to the trigger pin. The sensor will then output a pulse from the echo pin for a time proportional to the time taken for the emitted pulse to return. The distance (in metres) can therefore be calculated using Equation (1) which also accounts sound waves travelling both to and from the object.

$$Distance (m) = \frac{speed\ of\ sound \times pulse\ time}{2} \quad (1)$$

$$Distance (m) = \frac{343ms^{-1} \times pulse\ time}{2}$$

Since the ultrasonic sensor is powered at ~ 5 V but the MSP430G2553 microcontroller's GPIO pins are only 3.3 V tolerant, the echo pin output was stepped down using a voltage divider on the control board. The output voltage of the voltage dividers with the set resistances for a 5 V input echo signal is calculated in equation (2). This value of 3.4 V is within the microcontrollers GPIO voltage tolerance.

$$V_{echo_out} = 5\ V \times \frac{1\ k\Omega}{1\ k\Omega + 470\ \Omega} = 3.4\ V \quad (2)$$

3.1.2 Infrared Sensor (Colour)

With the car only needing to determine either black or white skittles, an Infrared (IR) sensor is used for simplicity as a black skittle will reflect less IR light than a white skittle. The FC 51 IR sensor (see Figure 11) was chosen for its low cost and simplicity. It uses an LED to emit IR radiation alongside an IR photodiode to detect any reflected IR radiation. The photodiodes receiver is wrapped with electrical tape such that it is less prone to visible light and direct interference from the emitter. It provides skittle colour detection from approximately 2 – 30 cm but for consistent results less than 11 cm.

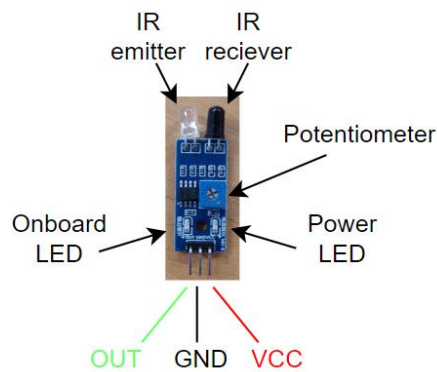


Figure 11: Diagram of IR sensor

This sensor can be powered via 5 V or 3.3 V with the 5 V providing better range due to brighter illumination of the IR emitter LED. However, the 5 V supply in RUSH is less consistent than 3.3 V, therefore, 3.3V is used as the supply voltage to maintain distance calibration of the sensor. The output of the sensor is digital with a “LOW” output indicating a great enough reflected light received and a “HIGH” indicated for when too little reflected light was received. This is slightly counterintuitive as an on-board LED uses the opposite logic to illuminate. The potentiometer on the sensor was used for adjusting the sensitivity of the sensor.

3.1.3 Microcontroller

The MSP430G2553 microcontroller was selected for complete control of the operation of the car due to its familiarity and low cost. This microcontroller operates at 3.3 V. Figure 12 shows the MSP430G2553 IC and the associated development board that was used for programming the microcontroller. Throughout the rest of this document this will often be referred to as “MSP430”.

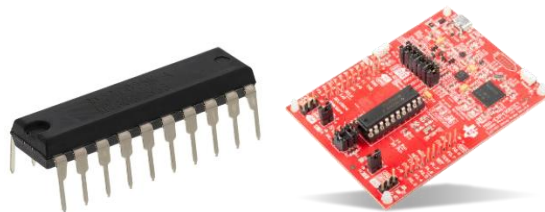


Figure 12: MSP430G2553 IC (left) [2] , MSP430G2ET Development board (right) [3]

3.1.4 Motor Driver

The DRV8833 Dual H-Bridge Motor Driver module was used to allow the microcontroller to control both steer and drive motors while drawing current from the battery (at nominal 4.8 V) and not the GPIO pins (at 3.3 V). A diagram of the DRV8833 Dual H-Bridge Motor Driver module is shown in Figure 13.

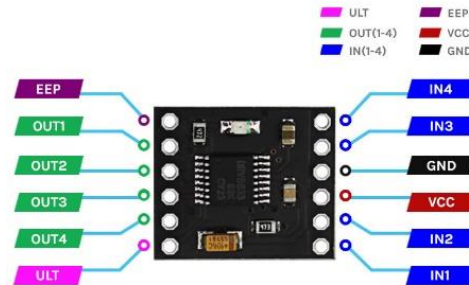


Figure 13: Diagram DRV8833 Dual H-Bridge Motor Drive of [4]

3.1.5 Servo

For the car to function, it was required to be able to rotate an ultrasonic sensor to a known angle to find skittles. To accomplish this an SG90 servo motor was chosen for low cost and wide availability. The servo requires 5 V to operate. A diagram of the SG90 servo motor is shown in Figure 14.

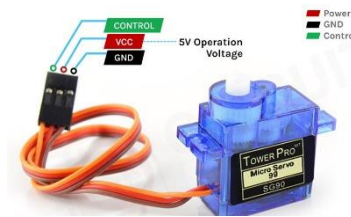


Figure 14: SG90 servo motor [5]

3.1.6 Power Source

Four AA sized nickel-metal hydride (NiMH) batteries were used as the power source to allow integration into the base RC car with minimal alterations. The NiMH batteries are rechargeable and keep a very consistent nominal voltage of 4.8 V total (see Figure 16) which can power all systems rated for 5 V. Note that no reverse polarity protection is installed, and so care should be taken when installing the batteries. The fourth battery can be seen to be mounted on one side of the car which leads to RUSH having a slight uneven weight distribution and pulls to the right whilst attempting to drive straight. This was compensated for by adjusting the dial on the underside of the base RC car chassis to the left or “L” side as seen in Figure 15.



Figure 15: Base RC car steering adjustment

The car cannot be powered off four AA Alkaline batteries as the starting voltage would be too high and the inconsistent nature of the voltage levels would significantly alter the speed of the motors and therefore the functionality of the car itself would be very inconsistent. It would be possible to power the car off three AA alkaline batteries, however they would last a very short period before dropping below minimum operating voltage of 4.2 – 4.3 V.

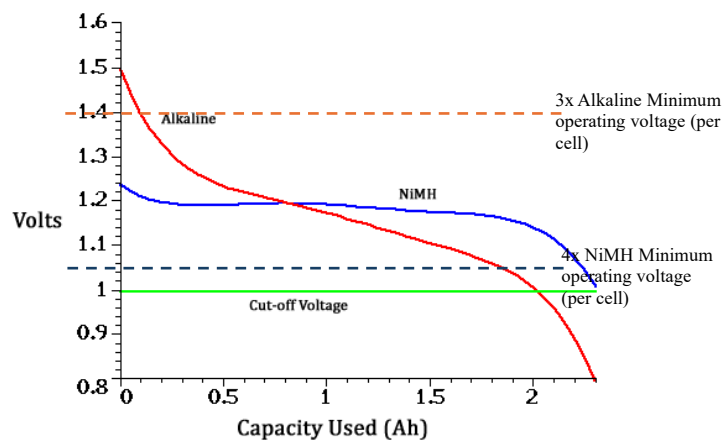


Figure 16: Single cell Alkaline vs NiMH Voltage discharge graph [6]

When fully charged, the battery was measured to have a starting voltage of 5.329 V which quickly dropped to the nominal 4.8 V.

To obtain the steady 3.3 V required for the microcontroller and IR sensor, a low-dropout DC linear voltage regulator was used, specifically a through-hole module containing the AMS1117 and all required passive components seen in Figure 17. A through-hole module was used to allow quick and easy replacement should it break.

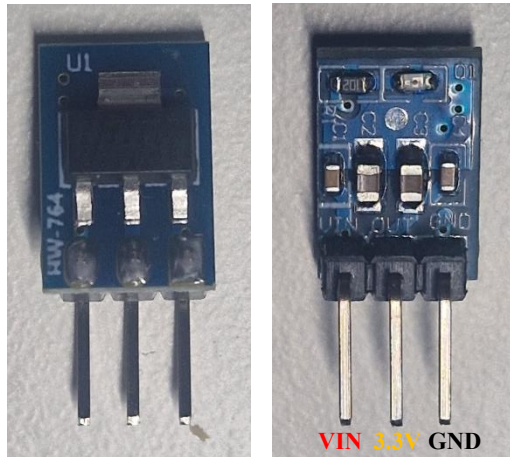


Figure 17: AMS1117 3.3 V Regulator Module

3.1.7 Control Board

All systems are connected and managed through the control board seen in Figure 18. Highlighted in the diagram are the colours of the wires to be connected to the Dupont headers to ensure correct orientation of connection for the servo and programming header. The remaining connectors cannot be orientated incorrectly due to the header design. The labelled test points on the rear of the board can also be seen in the figure with label “Tr-T” being corrected as PCB label was placed at incorrect position. Note the dotted green circles as these highlight indicators for the orientation of the MSP430, motor driver and 3.3 V regulator. For full schematic of Control Board, see Appendix 11.1.

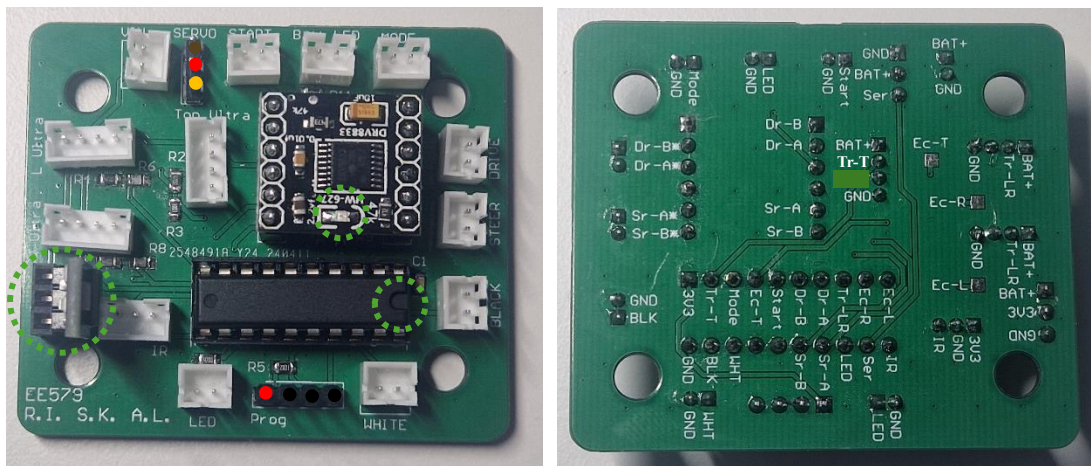


Figure 18: Control Board PCB top (left) and test points (rear).

3.1.8 Low Battery Indicator

Contained on the control board is a low battery indicator circuit seen in Figure 19. It uses a Zener diode in reverse bias as a set voltage reference for comparison to the current battery voltage. The potentiometer shown is used to vary the proportion of the battery is compared against the set Zener voltage and therefore controls the battery voltage at which the low battery indicator illuminates. It is by default set to illuminate at approximately 4.3 V as this was the lowest measured voltage that the whole car would consistently correctly operate at.

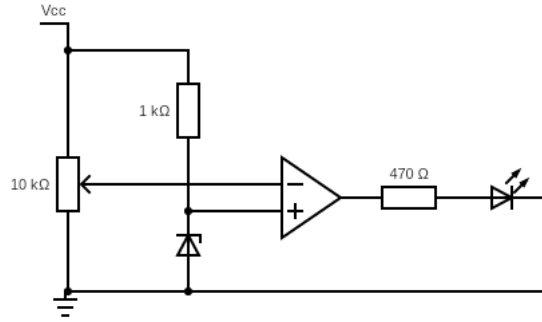
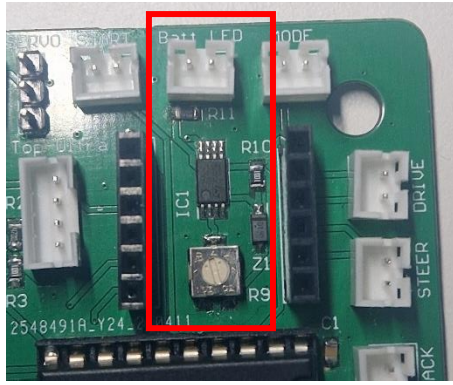


Figure 19: Low Battery Indicator Circuit on Control Board under motor driver (left) and circuit diagram (right)

3.2 Structure and Mounting

To house all the required electronics a mounting structure was designed to replace the base RC car's cover. This section of the report details the methods of (dis)assembly of the structure and wiring harness construction. All bolts used in the structure are various length M2 bolts with a 1.5 mm Allen key bolt head except for the screw used to mount the top ultrasonic structure to the servo which uses a Phillips head.

All electronic peripherals and structure components are recorded in the Bill of Materials (BoM) listed in Appendix 11.2.

3.2.1 Disassembly Instructions

To fully disassemble the car (as necessary for testing and upgrades), there are four main stages listed in order of disassembly as: Top SONAR Ultrasonic structure, Top SONAR servo structure, Upper Main Structure and additionally the Control Panel structure which can be removed at any stage as it is not directly connected to the other structures. Each structure consists of multiple 3D printed parts that are glued and/or bolted together. The structures are used to mount the sensors and other electrical peripherals.

The exploded view diagram of the disassembly of the Top SONAR Ultrasonic structure can be seen in Figure 20 with the BoM listing of each of the disassembled parts. This stage of disassembly requires the removal of the single Servo Screw with a Phillips head and also two M2 nuts and bolts if removal of the ultrasonic sensor is required. When the car is first turned on, if the Top SONAR ultrasonic sensor is not pointing forwards, the full structure should be taken off by removal of the servo screw (M-19) only, realigned and then reattached by replacing the screw.

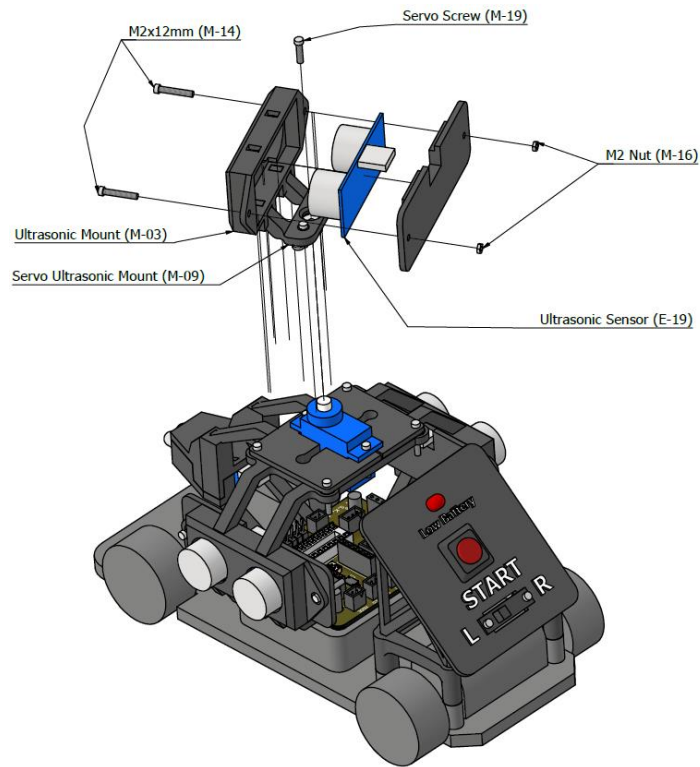


Figure 20: Top SONAR Ultrasonic structure Disassembly Diagram

Figure 21 shows the next stage: Top SONAR servo structure disassembly. This stage could occur after the Top SONAR Ultrasonic disassembly. It requires the removal of six M2 nuts and bolts along with four washers.

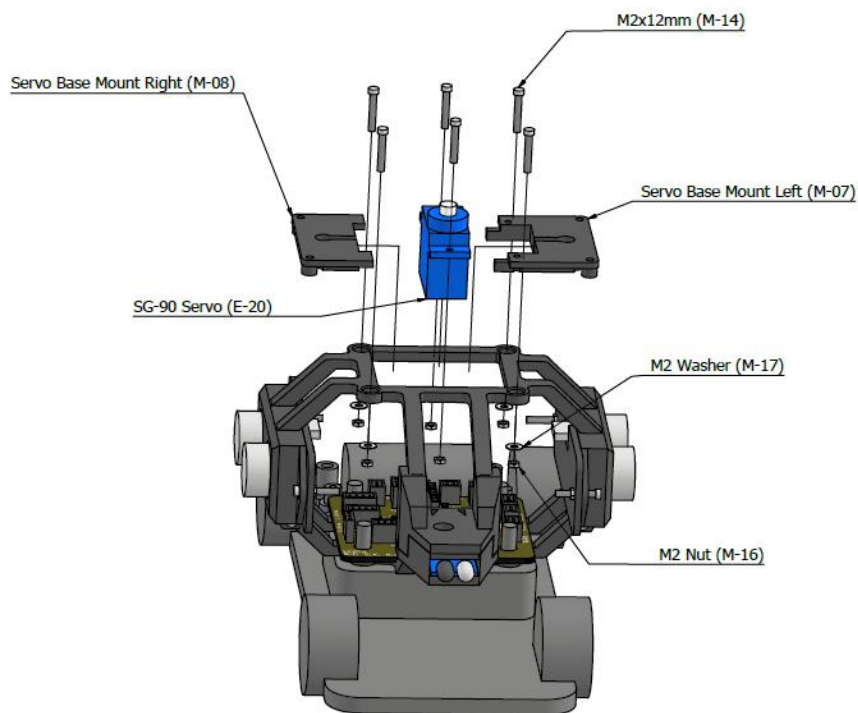


Figure 21: Top SONAR servo structure Disassembly Diagram

Disassembly of the Upper Main Structure is depicted in Figure 22. This stage requires four M2 bolts to be removed.

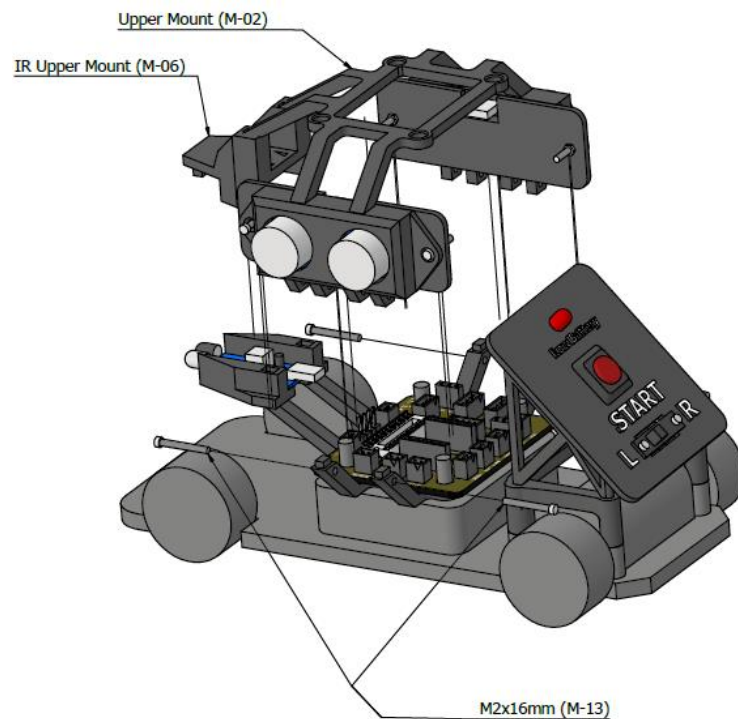


Figure 22: Upper Main Structure Disassembly Diagram

Removal of the ultrasonic sensors from within the Upper Main Structure can be seen in Figure 23. This stage requires removal of two M2 nuts and bolts.

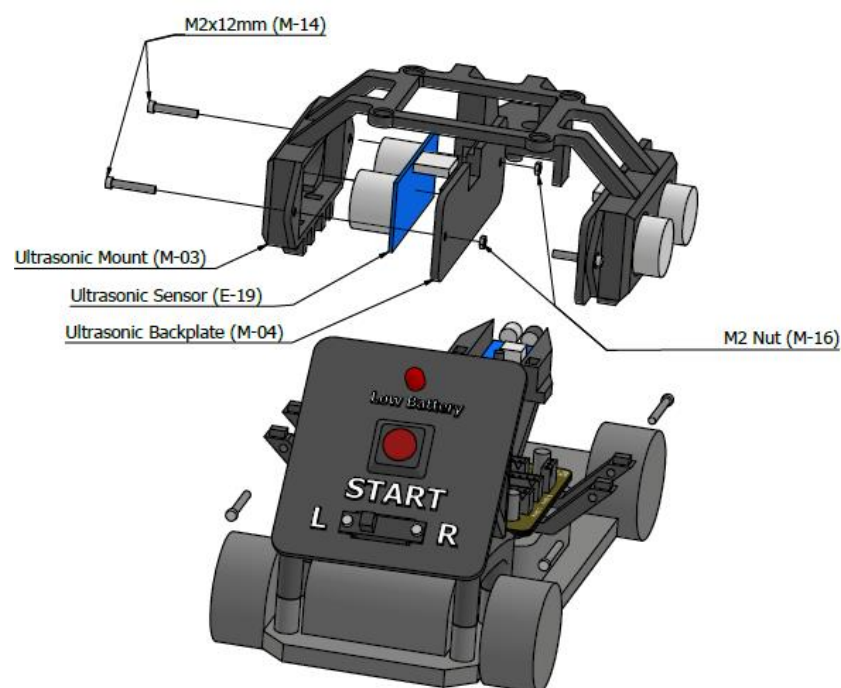


Figure 23: Ultrasonic Sensor in Upper Main Structure Disassembly Diagram

The full disassembly diagram of the Control Panel structure is seen in Figure 24. To remove the control panel structure from the chassis, only two M2 bolts need to be removed and the panel lifted up as one. The LED within the panel is friction mounted and can be removed from pulling from the reverse side of the panel. The Start button can be removed by pushing from the underside of the panel, ensuring it is disconnected from the Control Board. The slide switch requires two M2 nuts and bolts to be removed.

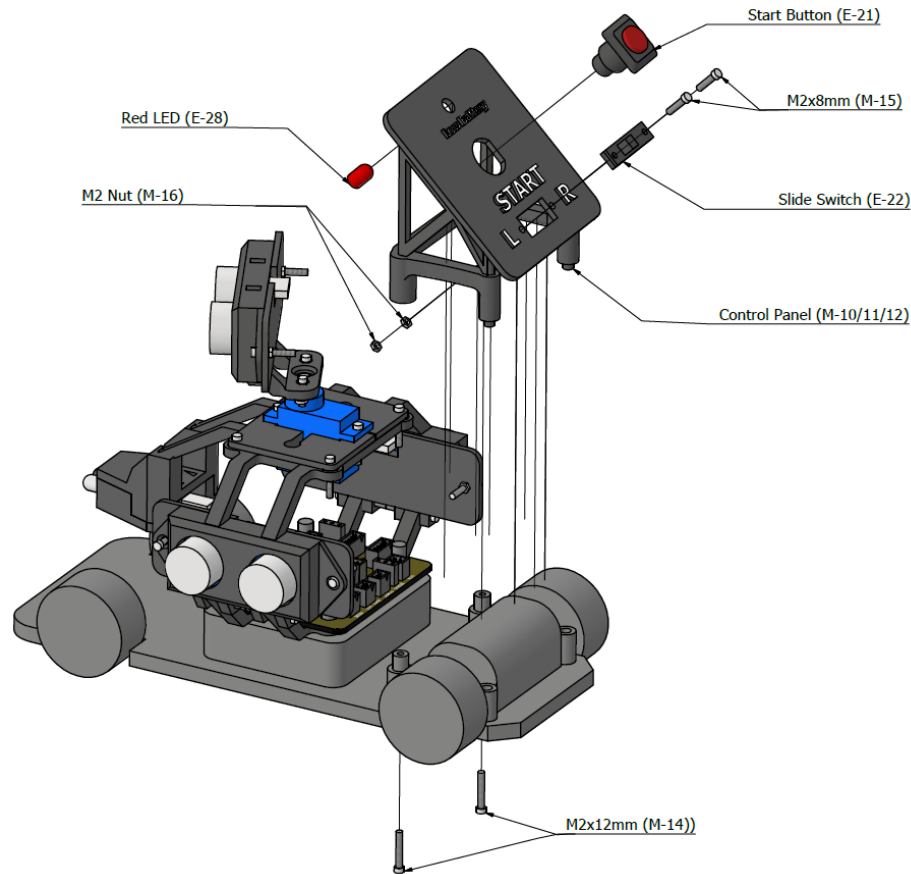


Figure 24: Control Panel structure Disassembly Diagram

All sections of the mount are 3D printed with PLA (polylactic acid) material. The BoM in Appendix 11.2 lists each individual part, which were glued and bolted together to make combined structures. All files required for printing are found in the projects GitHub repository.

3.2.2 Wiring Harness Assembly

To electrically connect the components to the circuitry of the control board, wiring harnesses are used. All harnesses were tested to stay firmly connected after three consecutive drop tests from 20 cm onto a hard surface. This section describes the methods for producing them in case of breakages. To determine if a wiring harness requires replacement, refer to Section 6.

The harnesses are made using two colours of wires: red and black. In every connection, the red wire defines the VCC connection to ensure correct orientation of the Dupont connectors. On the PCB side, most of the connections use JST-PH2.0 which cannot be inserted in the wrong orientation. However, all sensors use Dupont 2.54mm pitch headers which could be connected in the wrong orientation if care is not taken.

Connections to both motors and the battery are 2-pin JST-PH2.0 and were a part of the base RC car and so are not covered in this section however the same procedures would apply. The servo also contains its own connection and therefore if it were to break, a whole new servo would be required.

The LEDs, buttons and switches use 2-pin JST-PH2.0 connectors on the control board side and on the component side they are directly soldered to the pins with heat shrink tubing encasing the connections to insulate them. Note that the blue indicator LED (connected to BLACK header on the control board) requires a 470 Ω resistor to be soldered in line.

To produce the wiring harnesses the following steps should be followed:

1. Cut all wires to length listed in Table 2.
2. Strip both ends of each wire
3. Crimp each end with either a JST or Dupont crimp (see Figure 25) depending on listing in Table 2 or solder connection if listed. For further guidance on crimping, refer to [7].

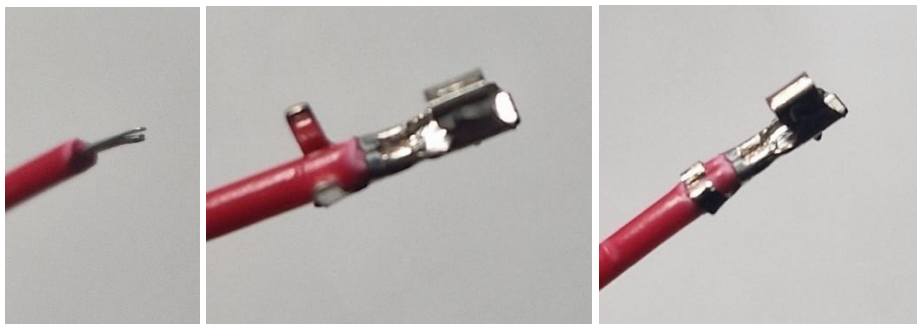


Figure 25: Stages of wiring harness production: strip insulation (left), first stage crimp (middle) and last stage crimp (right)

4. Insert crimped wire ends into connector specified in Table 2. For the order of the wires in the connector refer to the pinout of the peripheral and match with the label on the reverse of the Control Board. At this stage the full bundle of wires can be heat shrunk or taped together.

Table 2: Wiring Harness Connection Specifications

Connection (PCB Label)	Number of Wires	Length (cm)	PCB Connector	Component Connector
Right Ultrasonic (R Ultra)	4	7	4-pin JST-PH2.0	4-pin Dupont
Left Ultrasonic (L Ultra)	4	8	4-pin JST-PH2.0	4-pin Dupont
Top Ultrasonic (Top Ultra)	4	14	4-pin JST-PH2.0	4-pin Dupont
IR Sensor (IR)	3	6	3-pin JST-PH2.0	3-pin Dupont
Start Button (START)	2	5	2-pin JST-PH2.0	Soldered
Red Indicator LED (LED)	2	6	2-pin JST-PH2.0	Soldered
Blue Indicator LED (BLACK)	2	6	2-pin JST-PH2.0	Soldered with inline 470 Ω resistor
Side Select Switch (MODE)	2	5	2-pin JST-PH2.0	Soldered
Low Battery LED (Batt LED)	2	4	2-pin JST-PH2.0	Soldered

4 Software Overview

Detailed in this section is the functionality of RUSH and the software design used to achieve it. The software is organised as a state machine with a scheduler used to control the transitions between states. Some sub-sections of the software overview require some basic knowledge of programming and how a scheduler operates. All code is written in C and can be found on the project's GitHub repository (EE579-Skittles/CarReorganised/).

4.1 General Functionality

The program software operates using states that control the cars behaviour based off its current state. Figure 26Error! Reference source not found. depicts the states and transitions between states implemented. The use of states aids modularity as if extra functionality is to be added, it is easy to insert a new state. Having separate states also aids in debugging and improvements as issues can be narrowed down to a particular state. A set of MACROS at the start of *main.c* define all the states used.

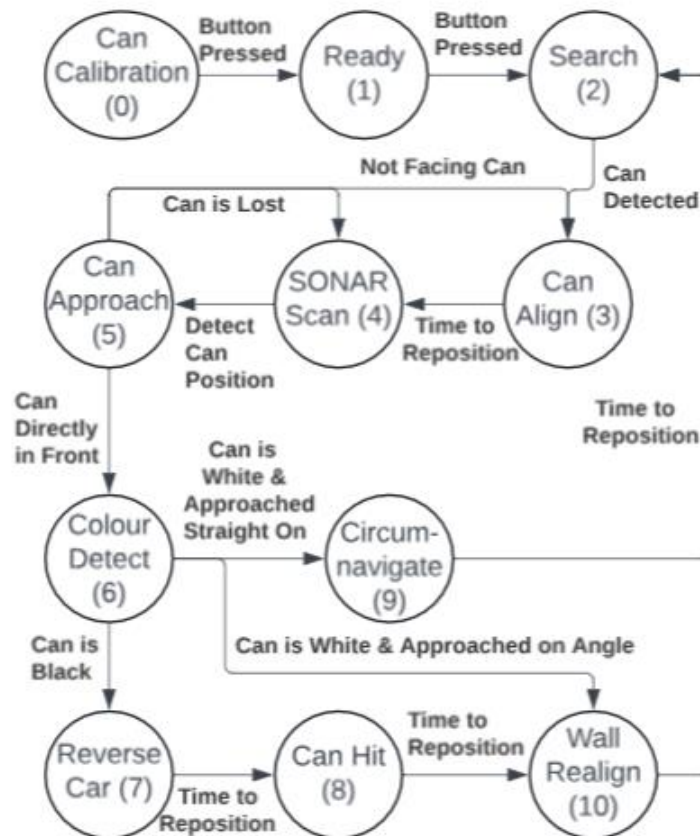


Figure 26: Functional State Diagram

The initial state on power up, *Can Calibration*, does nothing until the start button is pressed. When the button is pressed the ultrasonic sensors on each side of the car take an average of several readings. The average is used to determine the furthest distances either side at which cans will be detected as such. Any distances readings beyond this calibration can be attributed to the wall at the boundary of the play

area. One second after the first button press, the *Ready* state is then entered. This state waits for a further start button press before starting the autonomous phase of the car.

In the *Search* state the car drives forward whilst constantly aligning straight to the wall ensuring that the car does not veer off course. This alignment performs best when there are no changes in the wall geometry. When an ultrasonic sensor detects a can closer than the set threshold, *Can Align* state is entered to confirm the can reading and realign the car to be able to approach the can. *SONAR Scan* is then used to sweep through the range of the servo which has an ultrasonic sensor attached. The ultrasonic sensor combined with the servo determines where the can is relative to the car. After finding the can, the state *Can Approach* is entered. During this state the top ultrasonic sensor, referred to as SONAR, alters to always point at the can and the car drives on the heading of the SONAR to the can. If the SONAR has lost sight of the can, then the car returns to *SONAR Scan* state to refocus on to it.

Having reached the can, the angle of the SONAR to the can is checked. If the angle is too large in either direction from straight ahead then the main car body, where the IR sensor is mounted, is not considered straight enough to the can. The car then reverses to readjust and reapproach. This is required for the mounted IR sensor to determine whether the can is black or white which needs to be close to the can and facing the can directly to function correctly. It also aligns the car correctly should a black can be detected and therefore should be knocked over. When the can has been approached at a good enough angle the state is changed to *Colour Detect*. In this state the IR sensor is read to determine the colour of the can.

If the colour is black, then the car enters *Reverse Car* followed by *Can Hit* to knock over the can. If the colour is white, then the car enters one of two states. If the can was originally detected in front of the car, then the car enters *Circumnavigate* which drives the car around the white can to get past it. If the can was originally detected by a side ultrasonic sensor and approached, then the *Wall Realign* state is used to get the car back away from the can and start to straighten the car parallel with the wall. The *Wall Realign* state is also used after a black can has been hit. After realigning to the wall, the *Search* state is re-entered to start searching for more cans.

4.2 Individual State Complete Functionality

4.2.1 Can Calibration

The first car state is *Can Calibration*. In this state the car does nothing until a user presses the button. At this time, each side ultrasonics takes five readings to determine an average distance. The user should setup the calibration walls as in Section 2 to calibrate the furthest distance that a can will be detected at. This is illustrated by Figure 27 This distance would ideally be slightly closer than the closest wall of the full length of corridor play area. If a single can is to be aimed for, the calibration walls can be setup to do so by bringing them closer to the car such that they are only just either side of the can being aimed for.

The car will slightly drift left or right during operation even with the align to wall functionality. Because of this nature it is better to use the provided makeshift calibration walls to set a distance between the can and the closest wall to allow for greater tolerance as the car ends up closer to or further from the wall whilst also being able to detect cans that appear closer than the makeshift calibration wall. This average is used throughout the program to determine if a can has been detected by reading a distance closer than the threshold set.

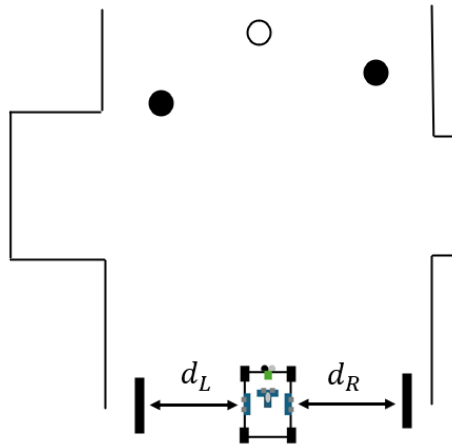


Figure 27: Can Calibration State Illustration

In this state the side select switch can be used to set the left or right ultrasonic to control the wall side that is aligned to. This was implemented to account for starting on either the left or right of the opposition car in the space. The left (red) or right (blue) LED's are lit when the switch is pressed. The state will change to *Ready* one second after the initial button press.

4.2.2 Ready

Upon changing to the *Ready* state both indicator LEDs are lit to indicate readiness for the user input and to start autonomy. The *Ready* state awaits the user pressing the button. When the button has been pressed and debounced successfully the car enters the *Search* state. This state on the button press also schedules the enabling of *search* flag which is discussed in Section 4.2.3.

4.2.3 Search

The purpose of the *Search* state is to drive through the space using the three ultrasonic sensors to detect the position of cans. A can is detected when the SONAR ultrasonic, which is pointed forwards, detects a can in front; or when one of the side ultrasonics reads a distance shorter than the *Can Calibration* initial readings. When a can is detected, the source of direction (Straight, Right, Left) is noted for future use. The cans on the side are only detected when the *search* flag is enabled. The reason to have a *search* flag is that no cans are expected to be seen for the first 6 metres (as per the game rules) so disabling searching for an amount of time before then allows for the car to get to the region the cans are expected to be faster. It can be faster whilst not searching as the ultrasonic sensors are read in a round robin and at high speeds a can could be missed. The ultrasonics are triggered in a round robin to mitigate interference. After the left is read the right is triggered, when right is read top SONAR is triggered and then back to the left.

Simultaneously to the can detection, the *alignToWall()* function ensures that the car is driving straight. The function is called when the relevant ultrasonic to the wall being aligned to is read. The relevant ultrasonics readings are analysed to determine the trend in the data and corrects the car heading with a short 50 ms turn to re-adjust the car. Figure 28 depicts the decision-making process of the wall alignment.

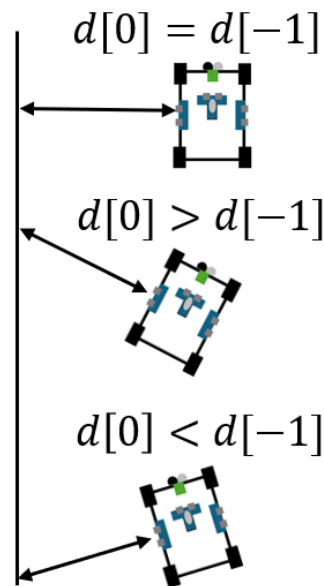


Figure 28: Align to Wall Decision Making

If the readings are trending closer, then the car is travelling toward the wall, so the car needs to readjust by turning away from the wall. The opposite is true as readings increase. When, withing tolerance, the distance is the same as previous readings then continue driving straight.

4.2.4 Can Align

To account for a false reading or detection of the other car a two second delay is implemented before rechecking that the can is still detected where it is thought to be. The car, in the case of a left or right can detection, may have driven past the can by the time it comes to a complete stop and so a reverse is also implemented to make car become parallel to the can. When a left or right can is verified to be there, the car reverses in the opposite direction. This is so that the car should then be roughly facing the can. At this point the state is changed to *SONAR SCAN*. In the case of a front detection the car immediately enters *SONAR SCAN* upon confirmation of can presence. If a can is not confirmed, then the car re-enters the *Search* state. Figure 29 depicts the car behaviour on entering this state, confirming can location and realigning to face can.

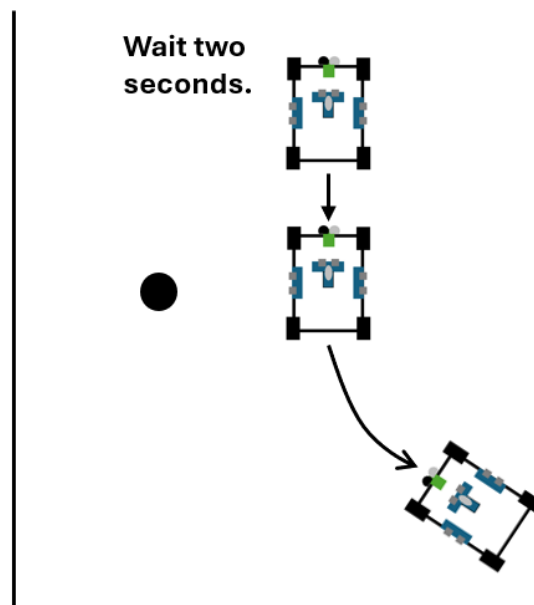


Figure 29: Left Can Confirmation & Alignment

4.2.5 SONAR Scan

The *SONAR Scan* purpose is to find the cans position in the space at a relative angle to the car and the distance it is away. The Servo is initially turned clockwise and then takes five readings at each angle moving anti-clockwise. The average reading at each angle is stored. By default, the scan takes the averaged readings at fifteen angles of the full sweep. Upon completion of the sweep the distance readings are analysed. An 'anomaly' is detected when the distance at the checked angle is significantly shorter than the previous angle distance. The anomaly is registered at every angle whilst the shorter distance is read. This process is depicted in Figure 30.

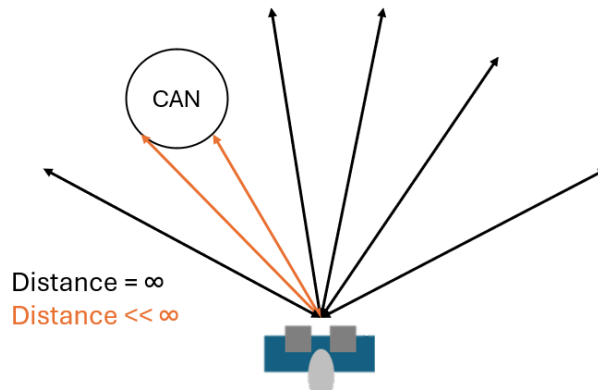


Figure 30: SONAR Anomaly Detection

The SONAR can detect multiple anomalies in a sweep and can also differentiate between two anomalies if one can is in front of another. During the sweep if an anomaly is detected the red LED will light and if a second anomaly is detected in front of a current anomaly, then the blue LED will also light. To find the closest anomaly all the distance readings on each angle are averaged and the closest anomaly is chosen to be locked on to. Where an anomaly is present for multiple angles the SONAR will lock on to the median angle of where the anomaly is present. If no anomaly is found *Search* mode is returned to but practically this never occurs as walls also cause anomaly readings. Once the anomaly is locked on to the *Can Approach* state is entered.

4.2.6 Can Approach

In the *Can Approach* state the car drives to the heading of the SONAR. For example, if the servo is pointing left of the centre position (pointing straight) the car drives to the left. Whilst the car is moving the SONAR stays in position. As the car is changing angle the SONAR will lose sight of the can. At this point the car will stop, and the SONAR will attempt to re-find the can in a short number of angle rotations in the expected direction of the can. The direction the can should be is predictable relative to the car when it has been lost. If the car was turning left and the SONAR is no longer looking at can then the can will likely be to its right, this reasoning is depicted by Figure 31.

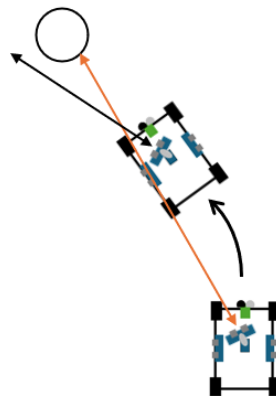


Figure 31: Can Approach with Locking on to Can

The SONAR does a couple of checks on angles in the expected direction of the can when can is out of view. If the can is not found within checking four angles, then the *SONAR SCAN* state will be re-entered to get a lock on the can again.

When the car reaches within IR reading distance of the can the angle of the SONAR is checked to ensure that the car is facing the can head on. If the angle is not straight enough the car will reverse and turn to get a better angle before using *SONAR SCAN* to lock on to the can and approach again.

When the angle and distance to the can is appropriate the *Colour Detect* state is entered.

4.2.7 Colour Detect

In *Colour Detect* the IR sensor is read to determine if the can is black or white. If black is detected, then *Reverse Car* is entered. If white is detected, then either *Wall Realign* is entered or *Circumnavigate*. *Wall Realign* is entered to get the car back parallel to the wall if the can was found on the left or right and therefore approached at an angle. *Circumnavigate* is entered if the can was detected straight on and therefore approached from the front.

4.2.8 Reverse Car

The car is reversed for a second before entering *Can Hit*. This allows the car space for acceleration so that the car can hit the can with enough force to knock it over.

4.2.9 Can Hit

In *Can Hit* the car drives forward at full speed to knock the can over and then enters *Wall Realign*.

4.2.10 Wall Realign

In the *Wall Realign* state the car reverses straight and then reverses in the direction which makes it become parallel with the wall. The direction to realign is known from the cans original detect position. This is a rough estimate to get parallel which is followed up by the align to wall functionality until the car determines that it is straight with the wall as it has not had to readjust for multiple readings. When straight the car re-enters search mode to continue searching for cans. In this time of realignment, no cans will be detected.

4.2.11 Circumnavigate

The *Circumnavigate* state performs several movements to manoeuvre around the white can that has been approached and detected straight on. However, the first step is to check which wall is further away. This allows the car to navigate around the side of the can with more space if the can is close to the wall. The order of steps taken by the car is depicted in Figure 32 before re-entering the *Search* state to continue searching for cans.

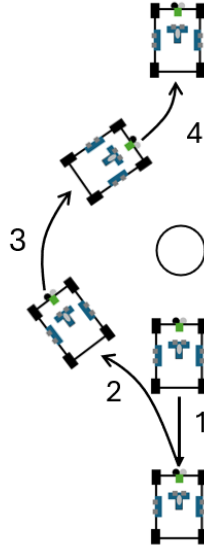


Figure 32: Circumnavigate Movements

4.3 Scheduler and Event Flags

To aid the state structure used, a scheduler and set of flags are employed. These are implemented as their own type structures to group all these types of variables together.

The scheduler is kept on time by Timer A0 which is used to trigger an interrupt every 2 milliseconds and add this to the running track of time. The running track of time counts to 5 minutes before resetting to zero. A structure for time type variables was created that contains integer second and integer millisecond variables. To set a time for an event to occur the *timeIncrement()* function is called with arguments of the event that should occur and when it should occur. The function accepts a pointer to a time variable and an integer value for a second and a millisecond respectively. It will then set the input time variable to the current time plus the input second and millisecond values. Each time the Timer A0 interrupt occurs, a flag is set high to check all the schedule variables and determine whether it is time to act on an event or not. Upon the time of the event, depending on the nature of the event either a sensor is triggered; a flag is set high to be dealt with elsewhere; a parameter is changed; or an action is immediately dealt with.

The event flags occur when certain conditions are met. This includes the debouncing of buttons; the completion of ultrasonic readings; the changing of DC motor behaviour and the changing of states. Each flag is attended and alters car behaviour or sets other flags to be attended within the control of each state.

The scheduler and flags assist with peripheral control as well as general timings of delays for code functionality. For example, the *Circumnavigate* state has the car drive in each heading for set durations which is controlled by the scheduler. Within the main code, states can be entered by setting the *nextState* variable to the desired state to transition to and setting either the *flags.stateChange* high or setting a time with *timeIncrement(&Schedule.stateChange, sec, ms)* for the state to change at.

4.4 Microcontroller Utilisation

Multiple internal MSP430 functionality was used for internal timing operations as well as peripheral control. Table 3 contains a list of the MSP430 functionality used and its purpose. For how further details on the MSP430G2553 components used, refer to its datasheet [8].

Table 3: MSP430 Functionality and Purpose

<u>Component Used</u>	<u>Purpose</u>
Timer A0 (Counts to 2 ms) <ul style="list-style-type: none"> • ISR 0 • ISR 1 	Scheduler Clock & SONAR Reading <ul style="list-style-type: none"> • Increment and flag scheduler timer. • Capture SONAR Ultrasonic Pulse (P1.2)
Timer A1 (Counts to 20 ms) <ul style="list-style-type: none"> • Outmode 7 with TA1CCR2 • ISR 1 	Sonar Servo Timings & Side Ultrasonics <ul style="list-style-type: none"> • Servo PWM signal (P2.4) • Capture Side Ultrasonic Pulses (P2.1 & P2.2)
Port 1 <ul style="list-style-type: none"> • P1.0 • P1.1 / Pull-Up • P1.2 • P1.3 / Port Interrupt / Pull-Up • P1.4 • P1.5 • P1.6 • P1.7 	<ul style="list-style-type: none"> • SONAR Ultrasonic Trigger (Output) • Side Switch (Input) • SONAR Ultrasonic Echo (TA0 Capture ISR 1) • User Button (Input with Interrupt) • Steer Motor Left (Output) • Steer Motor Right (Output) • Drive Motor Forward (Output) • Drive Motor Back (Output)
Port 2 <ul style="list-style-type: none"> • P2.0 • P2.1 • P2.2 • P2.3 • P2.4 • P2.5 • P2.6 • P2.7 	<ul style="list-style-type: none"> • Side Ultrasonics' Trigger (Output) • Right Ultrasonic Echo (TA1 Capture ISR 1) • Left Ultrasonic Echo (TA1 Capture ISR 1) • Infrared Sensor (Input) • Servo PWM control (TA1 Output) • Red LED (Output) • Blue LED (Output) • Unused in code

4.5 Peripheral Control

Each peripheral type has an associated header and c file to define an associated component type structure variable that holds certain values the component will require. The files also contain any associated functions to setup and control the peripheral. This makes the code modular and easy to use so that the component variable with its relevant information can be instantiated in the main file and only the pointer of the component variable is passed to external functions to setup, read and control the peripherals. All peripheral types and how they are controlled are covered in this section.

4.5.1 Ultrasonic

The HC-SR04 uses two pins to start and return readings. The ultrasonic variable type contains the port with the trigger and echo pins used as well as the time at which the returned pulse changes and therefore the length of the returned pulse. The setup of the ultrasonic uses a function that accepts the pointer to the ultrasonic data. This function uses the pins input to setup the pins and the relevant timer capture compare functions to determine whenever the pulse polarity changes. Integral to the ultrasonic setup is that each timer is setup in the main code for the capture compare interrupts to capture the time values when the edges are detected.

The HC-SR04 is triggered using a 20 micro-second pulse from the setup trigger pins. When the ultrasonic is triggered the timer capture compare interrupt is reconfigured to be activated by the echo pin of the desired ultrasonic. The interrupt, contained in the main file, occurs when the pulse changes polarity. When both the up edge and down edge have triggered the interrupt the difference in time between the two edges is calculated. This functionality is visualised in Figure 33. The length of pulse is relative to the distance that the ultrasonic sensor has measured. In the main file when a new distance reading has been returned a flag is set high that indicates for the rest of the code that the new distance reading is ready to be utilised.

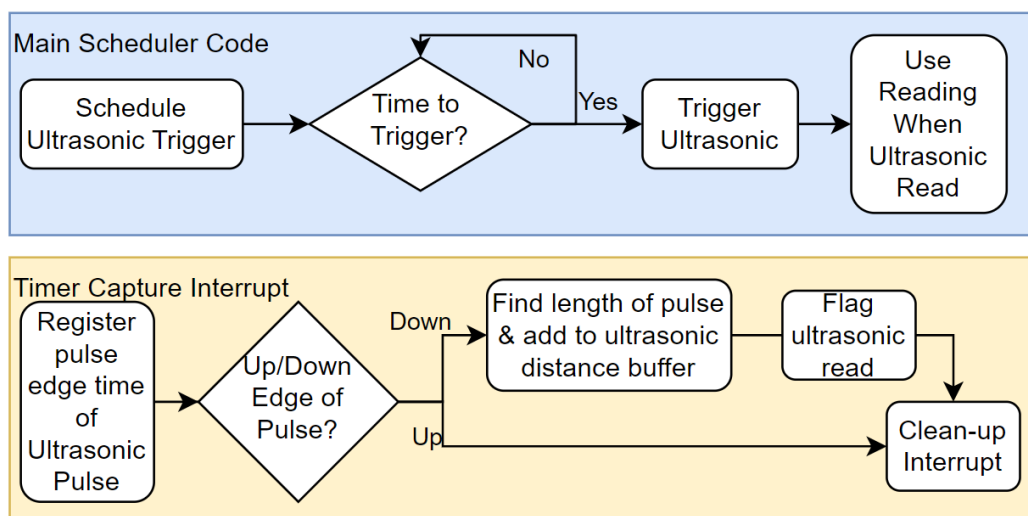


Figure 33: Ultrasonic Reading Control

4.5.2 Infrared

The infrared (IR) sensor variable structure contains the port and pin it is connected to as well as the colour that has been read. To setup, the pin is set to be an input. A function, “IRRead()” is used which sets the current feedback from the IR sensor to the colour variable where a white skittle is found when the value is low and black skittle found when the value is high. The “IRRead()” functionality is shown in Figure 34.

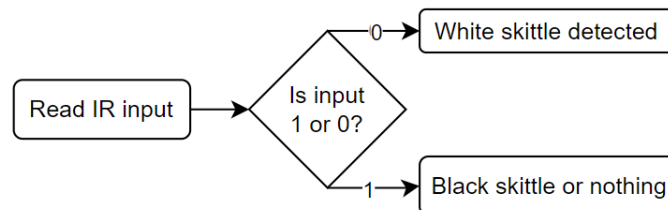


Figure 34: IR Control

4.5.3 User Interface

The user interface consists of a button, a switch and two indicator LEDs. The LEDs are setup with the port and pin it is connected to. The switch and button classes are the same as the LED with the additional variable of *val* which is the value populated when the component is read to be high or low. To setup the switch and buttons, the selected pin is set to have an internal pull-up resistor. Meaning that when the switch or button is used to pull the input to ground, the voltage is changed from high to low. The switches nature is to be steady high or low whereas the button is pressed and released. To confirm the button has been pressed an interrupt has been enabled to flag the event occurred. Also, due to the mechanical “bouncing” of the button, when pressed and the interrupt triggered, a time of 20 milliseconds in the future is set to check the button again for debouncing and ensure the button was intentionally used. There are functions to setup each of the user interface components; read the buttons and switches; as well as control the LED outputs.

4.5.4 Servo Motor

The servo motor is setup with the variables ‘direction of rotation’, ‘angular displacement of a rotation’ and the ‘PWM pin’ it is connected to. To setup the servo, Timer A1 is used to create a PWM (Pulse Width Modulation) signal that defines the angle the servo should rotate to. This involves setting the maximum value of the timer and the value at which the connected timer output should switch from high to low. The period of the PWM signal used is 20 milliseconds and an appropriate TA1CCR0 value is selected of 20,000 when the SMCK of 1 MHz is used. This is depicted in Figure 35. The output can be connected to pins 2.4 or 2.5 [8]. For this project 2.4 was selected.

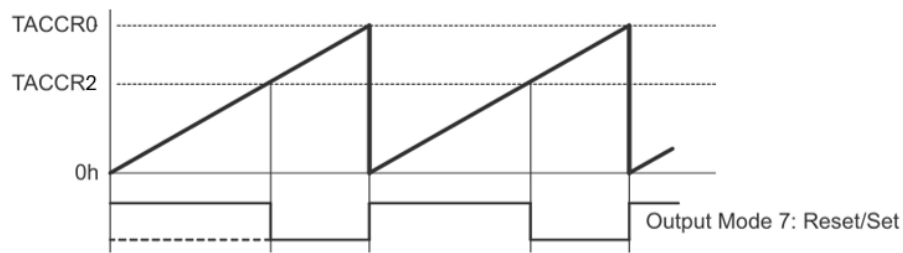


Figure 35: Output of Timer Linked Pin in Mode 7 [9]

To rotate the servo a function is used to add or subtract the time that the PWM period should be high for (the duty cycle) using the direction and the angular displacement values instantiated. The ideal servo range of motion is from 0 to 180 degrees using pulse widths from 1 to 2 milliseconds. These timings, with SMCK, relate to TA1CCR2 values of 1000 to 2000. Through testing it was found that the full 180 degrees cannot be rotated and that the maximum range that is achieved occurs between TA1CCR2 values of 700 to 2300. The servo can be turned in code using the function `servoTurn()` which accepts the pointer to the servo information and will turn in the direction and amount specified in the variable.

4.5.5 DC Motor Control

A DC motor is setup with the pins it is connected to, the direction the motors should spin and a variable of PWM type. The PWM type is defined to allow the DC motor to be controlled at differing speeds. The PWM structure contains the total PWM period, the duty cycle time and whether the PWM signal is currently high or low. In the main file, the PWM times are used to regularly schedule the times that the PWM state changes at. The functionality provided allows setup of the pins to control the DC motors and the function `motorOutput()` sets the relevant pins high or low based on the direction the motor is set to move. The PWM signal to control the motor speeds is generated using the scheduler rather than using a dedicated timer as all timer functionality is already used for other functions.

5 Tuning Parameters for Performance

This section details some of the tuneable parameters that can be altered within the main code as seen in Code Extract 1 with some commonly adjusted parameters highlighted. “TIME_BEFORE_SEARCHING” should be tuned in association with “SPRINT_SPEED” as both will alter the distance before entering search mode and skittles can be detected. Increasing “SPRINT_SPEED” too much could cause the car to lose wall alignment while increasing “FIND_SPEED” too much could cause skittles to be undetected. “MAX_FRONT_DETECT” defines the distance where a skittle will be detected in front. Increasing this too much could cause irregularities in the ground (such as large cracks) to be picked up as skittles.

```
//=====
// CHANGEABLE SETTINGS MACROS
//-----
//Scheduler control
#define TIMER_INC_MS      2           //Scheduler interrupts period (2 ms)

//Drive Speeds
#define MOTOR_PWM_PERIOD  100         //PWM period of motors.

//Wall readings control
#define WALL_READINGS     5           //Number of wall readings remembered
                                        // (MAKE 1+2^x)
#define AVERAGE_SHIFT    2           //log2(WALL_READINGS-1), used for
                                        // computationally efficient find average
#define TIME_BEFORE_SEARCHING 6       //Time in seconds on start before
                                        // searching
#define FIND_SPEED        35          //Speed once in search time
#define SPRINT_SPEED      60          //Sprint speed to get to search area

//SONAR Reading control
#define SONAR_READINGS    5           //Make equal to wall readings
#define MAX_FRONT_DETECT  2500        //Distance front can is detected in
#define ANOMALY_DISTANCE  500        //Distance drop off for start of
                                        // anomaly in SONAR()

//SONAR SCANNING
#define NUMBER_OF_ANGLES_CHECKED 15   //Number of angles in SONAR sweep

//CAN ALIGN
#define INITIAL_CAN_ALIGN_SEC 0        //Time to re-adjust from initial
                                        // detection and parallel
#define INITIAL_CAN_ALIGN_MS  900
#define SECONDARY_CAN_ALIGN_SEC 1      //Time to re-adjust after approached
                                        // can but at a bad angle
#define SECONDARY_CAN_ALIGN_MS 100

//CAN APPROACH
#define FACE_CAN_ANGLE_TOLERANCE 100   //Directly facing tolerance (In servo
                                        // PWM period ~15 degrees)
#define FACE_CAN_DIST_TOLERANCE  600   //Distance away tolerance (Ultrasonic
                                        // pulse length ~6 cm)
#define CAN_APPROACH_SPEED      40     //Speed when approaching

//REVERSE TIME FOR RUN UP TO CAN
#define REVERSE_TIME           750     //750 ms reverse

//HIT CAN
#define HIT_TIME               900     //Time to drive into can for
```

Code Extract 1: Changeable Parameters in main code

6 Troubleshooting

This section of the document gives a step-by-step guide of how to find a broken element and repair it should anything fail, or the cars behaviour becomes unexpected. This includes running individual hardware test procedures on the MSP430, electrical continuity tests and examining expected signals at different points of the system. The expected order and flow of the testing performed to find the fault can be seen in Figure 36. All tests beyond running the hardware tests will require a multimeter and an oscilloscope as well as some disassembly of the car.

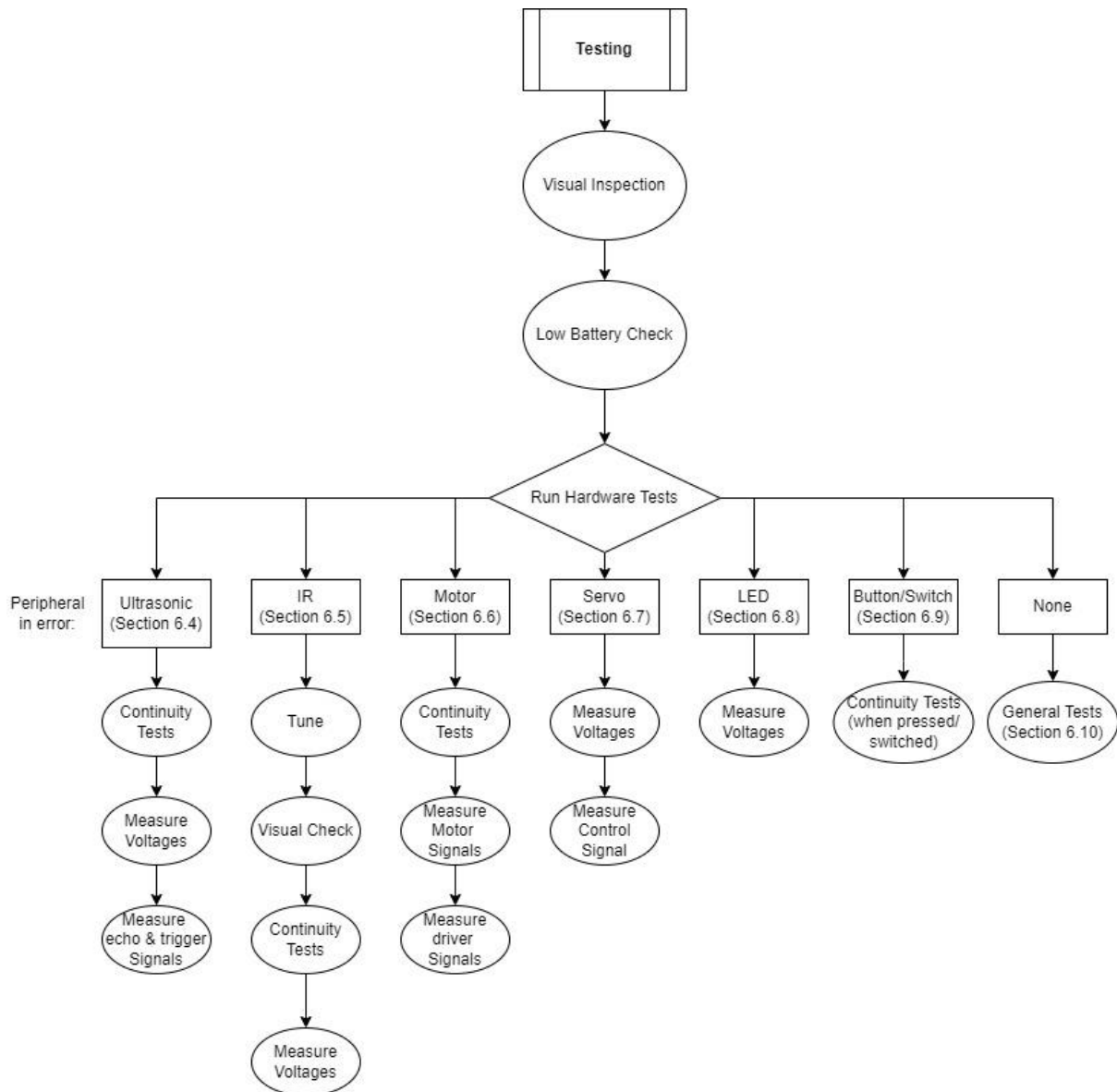


Figure 36: Expected flow of testing

6.1 Visual Inspection

If the car has stopped functioning as expected, the first check is to visually inspect the full car to see any wiring harnesses have become disconnected or any wires clearly broken. If a wire is broken, follow the instructions in Section 3.2.2 to construct a replacement wiring harness. Another visual check is the power LEDs on the 3.3 V regulator and motor driver modules. Both should be illuminated whenever the car is switched on. If not, check the connections between the battery and control board. On power up, one of the indicator LEDs on top should be illuminated. If this is not the case, continue to the hardware tests to find the issue.

6.2 Low Battery Check

If the low battery indicator LED on the control panel is lit when car is first turned on then the batteries are low. If this is the case, then the batteries should be removed and recharged. Note that due to large power draw and transients created from moving motors and servos the low battery indicator will flash when car is running. This is not an indicator of low battery. The low battery indicator should only be checked when car is first turned on. In case of failure of the low battery indicator circuit, the voltage across the battery itself can be measured with a multimeter between the metal contacts highlighted in Figure 37. Remember to remove the hot glue (added to ensure robustness). The nominal voltage is roughly 4.8 V with a minimum operating voltage of 4.3 V and a fully charged maximum voltage of 5.33 V.

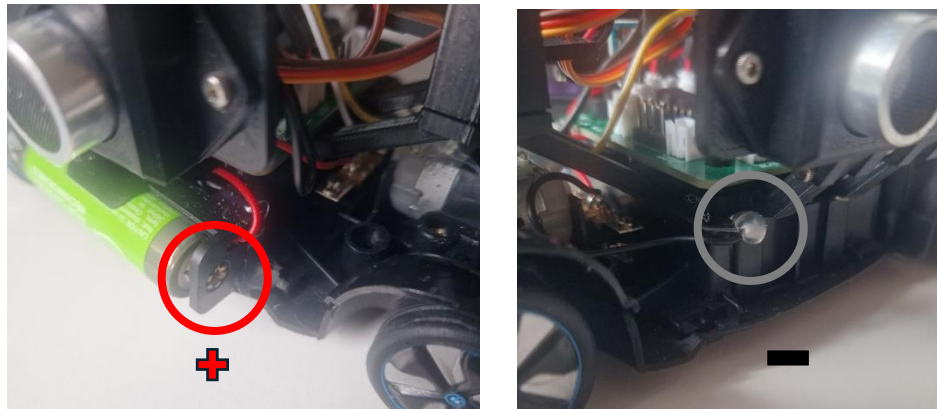


Figure 37: Battery Voltage Measurement Points

6.3 Hardware Tests

If the visual and battery checks do not reveal any issues, then the next stage is the hardware tests which should be programmed on to the control board. These test every one of the peripherals connected to the control board individually. The hardware tests (along with the main code) can be downloaded from the GitHub (EE579-Skittles/HardwareTests/). Note that the main code (EE579-Skittles/CarReorganised/) will also be required after testing so that it can be reprogrammed onto the car again once any issues are resolved. The tests work by programming the onboard MSP430 with one at a time in a specific order. The test that will be run is determined by the setup of the `'HardwareTests/main.c'` file.

The functionality of each individual peripheral is stored within header and source files. Each file is usually titled with the component name, such as ‘*Servo.h*’ and ‘*Servo.c*’ for the servo motor. The ‘*Common.h*’ and ‘*Common.c*’ files lay out everything required for an operating system and sets up common elements such as timers and the structures such as *flags*, such that the *main.c* source file has the highest level interface for user ease.

For the onboard MSP430 to be programmed with the hardware tests, a programmer is required to be connected to the Control Board. A simple wiring harness is provided to connect the MSP43G2ET development board to the Control Board as the programmer. Before connecting, ensure that the car is switched off and that all LEDs are off. The car should not be disassembled for these tests. The wiring harness with the required connections to the PCB and development board can be seen in Figure 38. Ensure correct orientation regarding position of red wire. As previously stated, the red wire is connected to VCC and in the case of the development board this is 3.3 V.

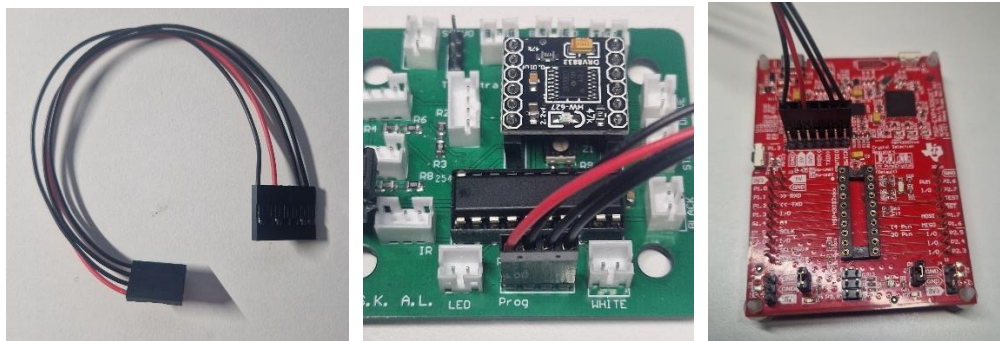


Figure 38: Wiring harness (left), Programmer PCB connection (middle) and dev. board connection (right)

Once the development board is connected to the car’s control board, and the development board connected to a computer, the hardware tests can be uploaded. Using Code Composer Studio (CCS) or some other appropriate IDE, the main function within the ‘*main.c*’ file can be edited to select the test to be run as seen in Code Extract 2. These tests should be run in numerical order as many of the hardware tests are dependent on the correct operation of other peripherals and so these need to be tested first.

```
// Enter 1 for indicator LEDs test
// Enter 2 for start button test
// Enter 3 for slide switch
// Enter 4 for drive test
// Enter 5 for steer test
// Enter 6 for servo test
// Enter 7 for SONAR ultrasonic test
// Enter 8 for LEFT ultrasonic test
// Enter 9 for RIGHT ultrasonic test
// Enter 10 for Infrared test

selectComponent(1);
```

Code Extract 2: Hardware Tests *main.c* file

Changing the value within the `'selectComponent()'` function selects the current test. Once selected it can be uploaded to the car. The following section details the expected operation of each test if component is fully working. Note that tests 4 – 9 require the car to be switched on **after** the car is programmed due to the components under test requiring battery voltage level of nominal 4.8 V.

If the control board cannot be programmed, first check that all connections are correct, that the car battery is switched off (note that power from the programmer will cause the control board to power up). If still errors are being thrown by CCS, remove the MSP430 IC from the control board (noting orientation) and place directly into the development board socket. Remove programming header, reattach jumpers and attempt to program. If still errors are thrown, it can be confidently concluded that the MSP430 is broken and will require replacement. If no errors, perform continuity tests of the programming wiring harness. If this is found to be broken, replace according to 3.2.2.

A short description and the expected results of each test are listed below in running order:

1. **LED Indicators:** This test ensures both red and blue indicator LEDs on top of the car are operating correctly. These LEDs are used for most other tests. When the test is run both LEDs should illuminate.
2. **Start Button:** This test ensures that the start button on the control panel works. Pressing the start button will toggle the red and blue indicator LEDs.
3. **Slide Switch:** This test ensures that the side select switch operates, where sliding it left will turn the red LED on and sliding it right will illuminate the blue LED.
4. **Drive:** This test ensures that rear drive motor can turn forwards, backwards and stop. There are two speeds, with the faster speed indicated by the blue LED and the slower speed indicated by the red LED. The wheels should switch between running forwards and backwards at two different speeds. For this test remember to lift the rear wheels from the ground to ensure the car does not drive off. Ensure to switch on the car using the underside switch after programming the test to the car.
5. **Steer:** This test ensures that front motor controlling steering works and does this by switching between turning left, right, and straight continuously. This test also requires the car to be switched on.
6. **Servo:** This test ensures that the servo motor for the top ultrasonic sensor works, where the servo will continually move from one rotational limit to the other. This test requires car to be powered.
7. **SONAR (top ultrasonic):** All ultrasonic tests from 7 to 8 test use the same methods to test different ultrasonic sensors. The LED indicator(s) will illuminate when the ultrasonic trigger pulse is transmitted and is turned off when the ultrasonic sensor returns an echo pulse. For the top ultrasonic, both the red and blue indicator LEDs are flashed to show functionality. The ultrasonic is only triggered again after a pulse has been returned. This test requires car to be switched on. If the LED illuminates but never switches off, this could be an indicator that either

the ultrasonic is not receiving the trigger, the sensor is broken or that it is working and the echo pulse is not received.

8. **Left ultrasonic:** See 7 for expected operation. For the left ultrasonic, the red indicator LED is used to show functionality.
9. **Right ultrasonic:** See 7 for expected operation. For the right ultrasonic, the blue indicator LED is used to show functionality.
10. **Infrared:** This checks the operability of the IR sensor. It illuminates both indicator LEDs if it detects a white skittle and turns them both off if a black skittle or nothing is present. This can be tested by placing any white or black object in front of the IR sensor. Note that one of the LEDs on the IR sensor itself (see Figure 11) should match the indicator LED (ignoring the IR sensor power LED). If they match yet are not switching as expected when shown a black or white object then this could indicate the IR sensor requires tuning (see Section 6.5). If the on chip and indicator LEDs do not light under the same conditions this could indicate a broken wiring harness.

If the hardware tests reveal an inoperable peripheral, then continue on to the appropriate section of the document for further testing. If no tests reveal an issue, attempt to reupload the main running code as the car peripherals should be entirely functional.

6.4 Ultrasonic Error Diagnosis

The procedure for testing is the same for all three ultrasonic sensors. The first test is to measure the supply voltage at the sensor itself. This will require removal of the sensor from the mounting. Refer to Section 3.2.1 for how to do this. The voltage can be measured between the Vcc and GND pins previously shown in Figure 9. This should give a value ranging from ~ 4.3 V to 5.3 V. If not, then the ultrasonic is not powered correctly. This leads onto the continuity/resistance tests which check that the wiring harness has not broken. This test will require full disassembly of the car and removal of the control board. Once control board is removed, reattach the ultrasonic wiring harness and battery power source. The continuity/resistance should be measured from the pins on the reverse of the control board to the pins of the ultrasonic sensor itself, matching up the equivalent pins and expected resistance as in Table 4. Note that the control board has a mis-labelled pin which has been corrected in Figure 39. The schematic for the control board can be found in Appendix 11.1 to assist in understanding.

Table 4: Ultrasonic Testing expected resistances

PCB	Ultrasonic	Expected Resistance
BAT+	Vcc	0
Tr-X	Trigger	0
Ec-X	Echo	$\sim 470 \Omega$
GND	GND	0

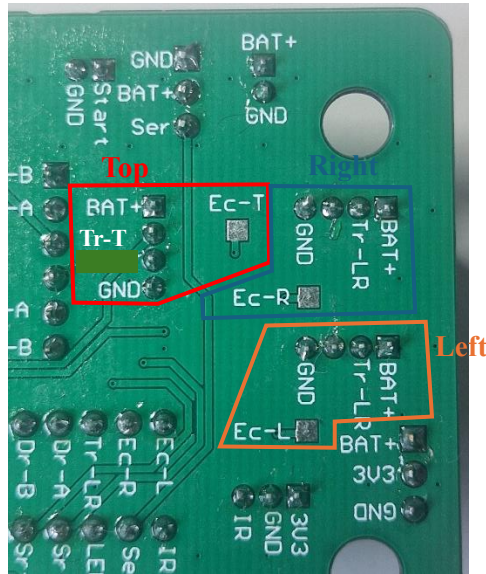


Figure 39: Ultrasonic connections on control board

Should any of these test result in unexpected values, then the wiring harness should be replaced following the steps in Section 3.2.2.

If no issues are found at this stage, testing is progressed to measuring the signals with an oscilloscope. First the control board must be loaded with the hardware test for the specific ultrasonic.

To check the ultrasonic is being triggered correctly an oscilloscope should be connected to the trigger pin directly on the ultrasonic sensor. The appropriate ultrasonic Hardware Test should be loaded onto the MSP for that ultrasonic sensor. The oscilloscope should be connected before powering the car as each consecutive trigger pulse is dependent on receiving the echo of the previous; therefore, if it is not working only one pulse seen shortly after power up will be visible. A square pulse of approximately 20 μ S should be viewed with an amplitude of about 3.3 V as seen in Figure 40. This means that the ultrasonic is correctly triggered. If not witnessed, measure at the trigger pin on control board to determine if wiring harness is at fault.

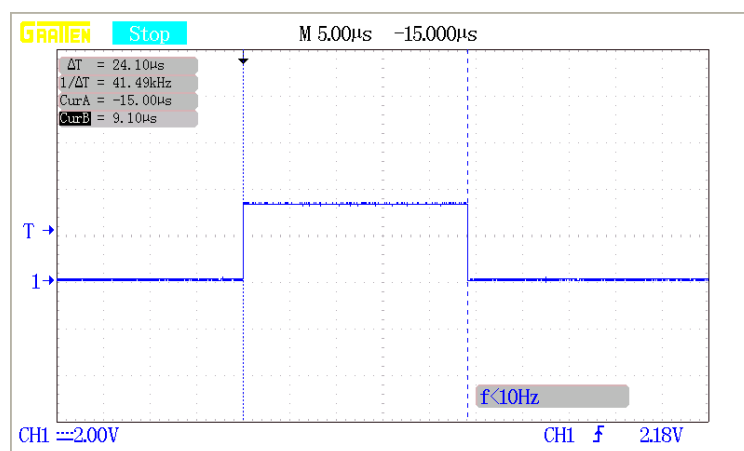


Figure 40: Expected Ultrasonic Trigger waveform

Next the echo signal should be verified by connecting the oscilloscope directly to the sensor. This test should be performed with the sensor under test pointed towards a flat faced object roughly 30 cm away relating to a pulse length of approximately 1.85 ms. The echo signal measured at the sensor should be approximately as seen in Figure 41. If this is not observed, then the sensor itself is likely at fault and should be replaced. The echo pulse should also be viewed to be the same when measured at test point “Ec-X” (where X is T:top, L:left or R:right) on control board except of a lower amplitude of approximately 3 to 3.3 V.

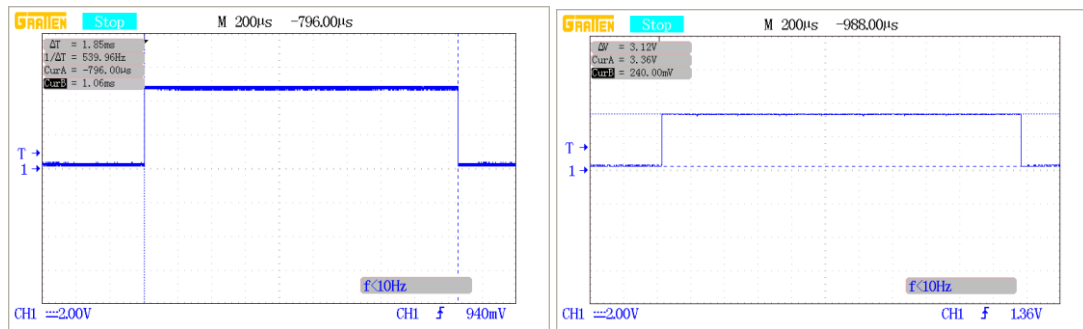


Figure 41: Expected ultrasonic echo waveform at sensor (left) and on control board (right)

6.5 Infrared (IR) Error Diagnosis

If the IR sensor is found to be at fault the first step is to check that the power LED is illuminated on the sensor (see Section 3.1.2) when the car is powered. If not the voltage between Vcc and GND pins on the sensor should be measured. If this is measured to be ~3.3 V then the sensor is broken and should be replaced.

If the sensor is correctly powered, then the calibration of the sensor should be checked. Powering the car up and placing white and matt black objects in front of the sensor, the LED onboard the sensor (see Section 3.1.2) should light when presented with a white object and turn off when a black object is placed in front. If this is not the case, the potentiometer on the sensor can be adjusted. Turning clockwise increases sensitivity and anti-clockwise decreases sensitivity.

Ambient light will reduce the effective range of the sensor and can stop it functioning altogether. If the sensor cannot differentiate between white and black objects, there is likely too much ambient light. Electrical tape has been wrapped around the receiver to reduce these effects but further shrouds to block the light could be constructed if ambient light is causing complications.

If the sensor's onboard LED is seen to illuminate correctly yet the car's indicator LEDs do not match it then the wiring harness connection from the sensor to the control board is likely at fault so continuity between the sensor and reverse of control board should be measured to verify this with a similar procedure to the ultrasonic sensor. If this is the fault then the wiring harness can be replaced following the procedure in Section 3.2.2.

6.6 Motor Error Diagnosis

If the one of the motors are found not to turn in hardware tests the likely issues are a broken connection or a faulty motor driver. To tell if it is a fault wire, continuity test between the motor metal tabs (after removing hot glue support) and the output connections on the PCB seen in Figure 42 should be completed. If this is the issue, either the harness will need to be replaced and/or resoldered to the motor.

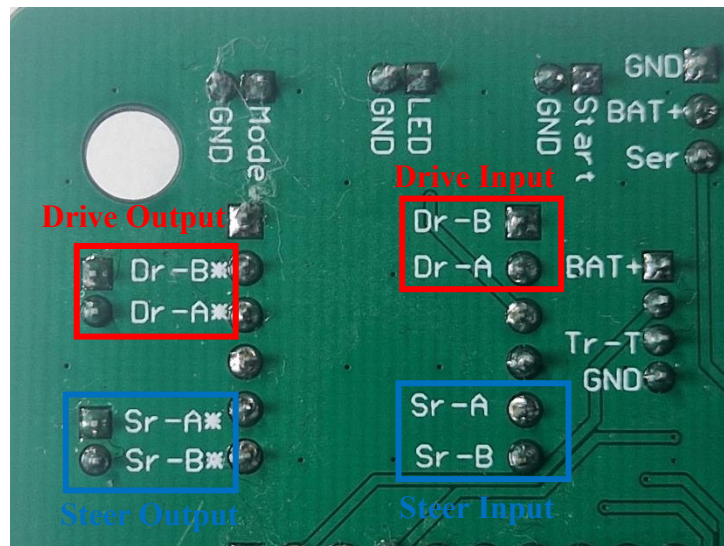


Figure 42: Motor driver control panel test points

If this is not the issue, it is likely the motor driver at fault. This can be verified by using an oscilloscope to measure the PWM (Pulse Width Modulation) signal input to the motor driver and at the output. After loading the MSP430 with the motor hardware test, lifting the wheels from the ground and powering the car, the signals can be measured. The signal can be measured at the test points input to the motor driver as highlighted in Figure 42. The expected waveforms at the two different speeds with measured pulse widths can be viewed in Figure 43. The pulse period is approximately 50 ms for this test.

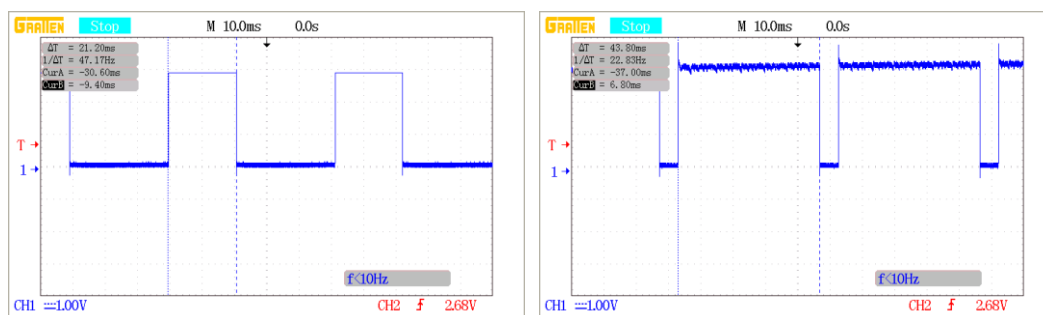


Figure 43: Motor PWM Control signal waveform

A signal like this should also be measured on the control panel at the motor driver output test points except likely with more noise, transients and at about 4.8 V rather than 3.3 V. If the output signal cannot be seen as expected, then the motor driver is at fault and should be replaced. Due to the nature of the design the motor driver can simply be pulled out to be replaced.

6.7 Servo Error Diagnosis

If the servo is not turning as expected, the first check is to measure the supply voltage between BAT+ and GND on the PCB as highlighted in Figure 44. The voltage measured is expected to be between approximately 4.3 V and 5.33 V. If this is not as expected, then the battery or connection to battery is likely the issue.



Figure 44: Servo test points on control panel

The next stage is to measure the PWM servo control signal if the supply voltage is acceptable. The servo hardware test should be uploaded which should make the servo turn from maximum to minimum limit continuously. The expected pulse widths to be measured on the control board test point (Ser) at maximum and minimum angles can be seen in Figure 45.

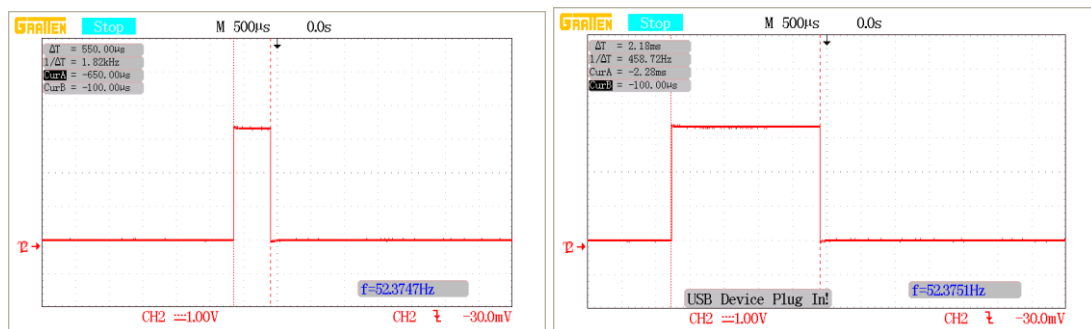
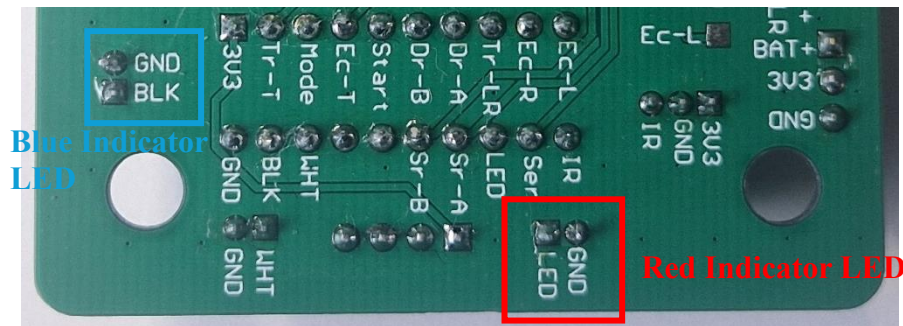


Figure 45: Servo PWM control signal at maximum and minimum angle

If these pulses are as expected and yet the servo still is not turning then the servo is likely at fault and should be replaced.

6.8 Indicator LED Error Diagnosis

If the indicator hardware tests reveal an issue with the LEDs, then the voltage should be measured to be non-zero between the two pins of each LED set of pins when the hardware test is loaded. This indicates that the LED or wires are broken, due to the nature of its construction it would be more efficient to remake the full LED harness with a new LED.



6.9 Button/Switch Error Diagnosis

If the hardware tests reveal an issue with the start button or side select switch, then they should be quickly continuity tested. To do this unplug from the control panel and continuity test between the two points on the button/switch harness connector while pressing/switching. Continuity should be achieved for the start button while it is pressed and for the side select switch when it is switched to only one side not the other. If continuity check reveals that the button or switch is non-functional, they should be fully replaced with a new button/switch and harness. If they are functional, reconnect to the board as it could have been an unsecure connection.

6.10 General Error Diagnosis

If the control board can be programmed but does not function when powered from the car, then the 3.3 V voltage regulator is likely at fault. It is very simple to remove and replace with a new regulator.

Another major fault to check for is a short between VCC and GND. This will cause the car to not function and not be able to be reprogrammed. This should be checked at both BAT+ and 3V3 to GND as these are separated by the voltage regulator. GND is common to everywhere however.

7 Design Validation

RUSH's complete functionality was validated with a demo in competition with nine other competing cars. Each team competed in two heats with the teams with the lowest four combined times to hit black skittles to progress to the semi-finals. In the first heat the car won by hitting a black skittle first. The second heat was lost as the opposing team hit a black can first however RUSH still hit the black skittle. RUSH reached the black skittle first but due to the slow SONAR sweep it was last to knock it over. This highlighted how much improving the SONAR sweep would improve the speed of the car. With the lowest combined time from the heats, RUSH proceed to the semi-finals where it won by hitting a black can directly in front in 8 s. It was this quick due to the car not entering search mode in time for the black to be detected. For the final, no skittles were knocked over by either team, with RUSH winning due to larger displacement from the start line. This was due to no skittle being detected as during calibration the users' setup the calibration walls too close for the skittle to be detected. Due to the anti-climactic nature of the final a second final was held in which RUSH detected and dealt with all three skittles. It encountered the white first and avoided it and then went on to knock down the two black skittles. However, some assistance was required to remove an already knocked black can as it was detecting this in the SONAR sweep and could not therefore get to the next can.

Throughout each game, RUSH was proven to be very consistent and robust, able to get to a black skittle almost every time (given that it was correctly calibrated by the users). The wall alignment performed very well and was able to keep the car very straight down the corridor. Details on possible future improvements that could solve some of the issues encountered in the demo are covered in Section 9.

8 Summary

This technical documentation file has covered all aspects of the hardware (Section 3) and software functionality (Section 4) of RUSH. Disassembly instructions have been presented in Section 3.2.1 should anything break or future improvement listed in Section 9 be implemented. The primary variables that can be altered to tune RUSH's performance was detailed in Section 5. Full instructions on the steps of testing that should be taken if RUSH behave expectedly (conflicting with the expect functionality described in Section 4.1) or break entirely was shown in Section 6. RUSH's design was validated to be robust and very consistent with the results of testing in Section 7. For any more details including full code listing, CAD designs, hardware tests, BoM and videos of operations refer to the GitHub repository (<https://github.com/Ross1416/EE579-Skittles>).

9 Future Improvements

Although RUSH operates well, there are a number of improvements that could be made to the car that would make it faster and more reliable. This section details the primary improvements that would likely make the most impact and were not explored for RUSH's development due to time constraints and the aim to keep costs low.

9.1 Improved Scanning Sweep

From extensive testing, the primary improvement would be to replace the top ultrasonic sensor with an alternative with a smaller detection scope. This would improve RUSH's ability to locate skittles in the SONAR scan state and its reliability in staying locked on to a skittle whilst navigating towards it. It would allow determination of the exact centre of the skittle resulting in a higher accuracy in positioning the IR colour sensor face on with the skittle. It would also increase its ability to differentiate between skittle and walls. A possible alternative that would meet this criterion would be a Time-of-Flight sensor (ToF) which uses pulses of laser light to determine range. This beam would be very narrow and therefore have a very small detection scope as seen in Figure 46.

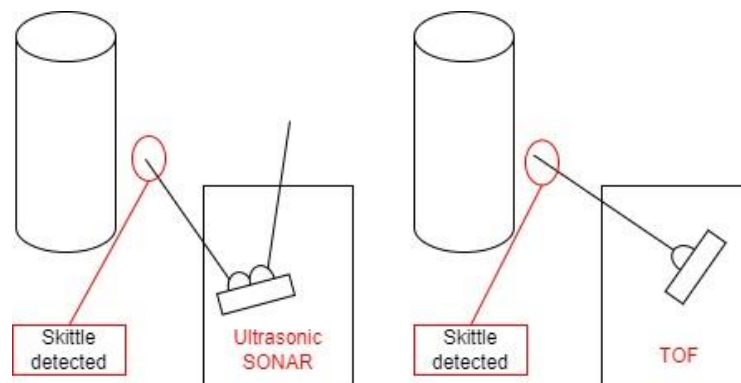


Figure 46: Skittle detection with Ultrasonic and Time of Flight (ToF)

The main sensor used by competing cars is the ultrasonic sensor and therefore use of the ToF sensor would also negate any possible interference caused by competitor's cars. Interference wasn't found to be a large issue, however, it was tested to be possible. ToF sensors commonly take advantage of the I2C communication protocol. This would increase the complexity of the main code for the MSP430 but it does contain functionality for I2C communication. A large issue however would be that this functionality operates on specific pins which may require a different MSP package and/or reworking of the pins already used which would have further implications on the designed PCB. With the original top ultrasonic sensor also performing the job of detecting objects in front of the car whilst searching, the ToF sensor would not perform as well. It would not detect skittles slightly off centre of the car and allow the car to strike skittles without realising it. Therefore, a stationary mounted ultrasonic or equivalent with a wide detection scope would likely still be required.

Another similar issue noticed with the ultrasonic sensors was that skittles that had been knocked over would still be detected and therefore cause the car to attempt to manoeuvre towards an already knocked skittle. A simple fix for this would be to either tilt the sensors up slightly or redesign the structure to raise them up to a point knocked cans will no longer be detected.

9.2 Realignment

Another issue noted in testing was the way the car attempted to reposition after dealing with a skittle. The realignment procedure uses set times and directions depending on original detection of the skittle. If the car ends up facing an unexpected direction, then this realignment has failed. This is not too important should the car have reached and dealt with a black can. Should the black skittle be positioned behind a white can however this could pose an issue. A possible solution to know where the car is in the space and in what direction it is facing could be an IMU (Inertia Measurement Unit) which contain accelerometer, gyroscope and magnetometer. The IMU could also assist in other operations such as driving straight and aid the wall alignment with side ultrasonic sensors. The drawback of this method, similar to ToF, is that I2C would have to be implemented for the IMU's and likely cause large redesign implications.

9.3 Improved steering

RUSH's steering mechanism is unaltered from the base RC car. It relies on a DC motor and only allows turning left, right (by about 60°) or straight. The mechanism itself is very loose and allows a lot of play in the steering; this translates to the car struggling to drive straight. Since the game rules allow modification to the steering, this whole mechanism could be redesigned to reduce the play and use a servo or stepper motor instead. This would allow accurate control of the angle of the wheels and not just left or right. The steering angle could also possibly be increased therefore reducing the turning radius of the car. This improved steering could be paired with the IMU and possibly a form of PID (Proportional Integral Differential) controller to allow car to drive straight. This would allow the speed of the car to be increased at least in the initial sprint mode.

9.4 Pre-game Search Mode

The rules stipulate that in the minute before the game of skittles start, both teams can view the position of the skittles in the play area. RUSH does not exploit this, however many possibilities of doing this were proposed. One possibility would be for the car, in the minute preparation time before the start, race down the centre of the track and note the positions of all skittles before returning to the start line. The user could then input the colour of cans in the order of detection via buttons on the control panel. On start up the car could then move directly towards the first black skittle since it already has knowledge of its position. This would cause issues in ensuring that the car returns to the start line before the minute has elapsed. The functionality of this would be simple to integrate with the software structure and PCB design as it is. In software the needed states to achieve this can be easily added and will work independently of other states just need to add the entry and exit points at the right time for functionality.

In hardware there is a port already setup for an additional button which could operate as the black / white input and that GPIO is currently unused in code.

9.5 Improved Power source

The power source of RUSH was chosen due to simple integration into the base RC car and that it only required a single step-down linear voltage regulator to be paired with it for the car to function. This source had a number of drawbacks however. Since the battery voltage is not entirely consistent and varies from about 5.33 V to 4.3 V, where the car will begin to fail, the speed of the car varies. This is seen to be significant when not operating in the relatively consistent 4.8 V nominal region of discharge. This causes some issues as a large amount of the manoeuvring depends on timings and if the motors run at different speeds this results in a difference in movements. This could, alternatively, be solved by a sensor for motor wheel speeds and update timings accordingly. However, a combination of alternative sources and regulators/convertors could be used to achieve a stable 5V VCC to maintain consistent motor speed. Lithium Ion (Li-Ion) batteries could be exploited for their higher energy density; however these batteries pose safety concerns and would likely require additional protection systems. It would allow, with some additional circuitry, simple charging of the battery within the car, improving overall useability. Any alterations made to the battery/power source should ensure that the weight is evenly distributed as the batteries for RUSH are unevenly distributed and causes the car to drift right slightly with the weight of the externally mounted NiMH battery.

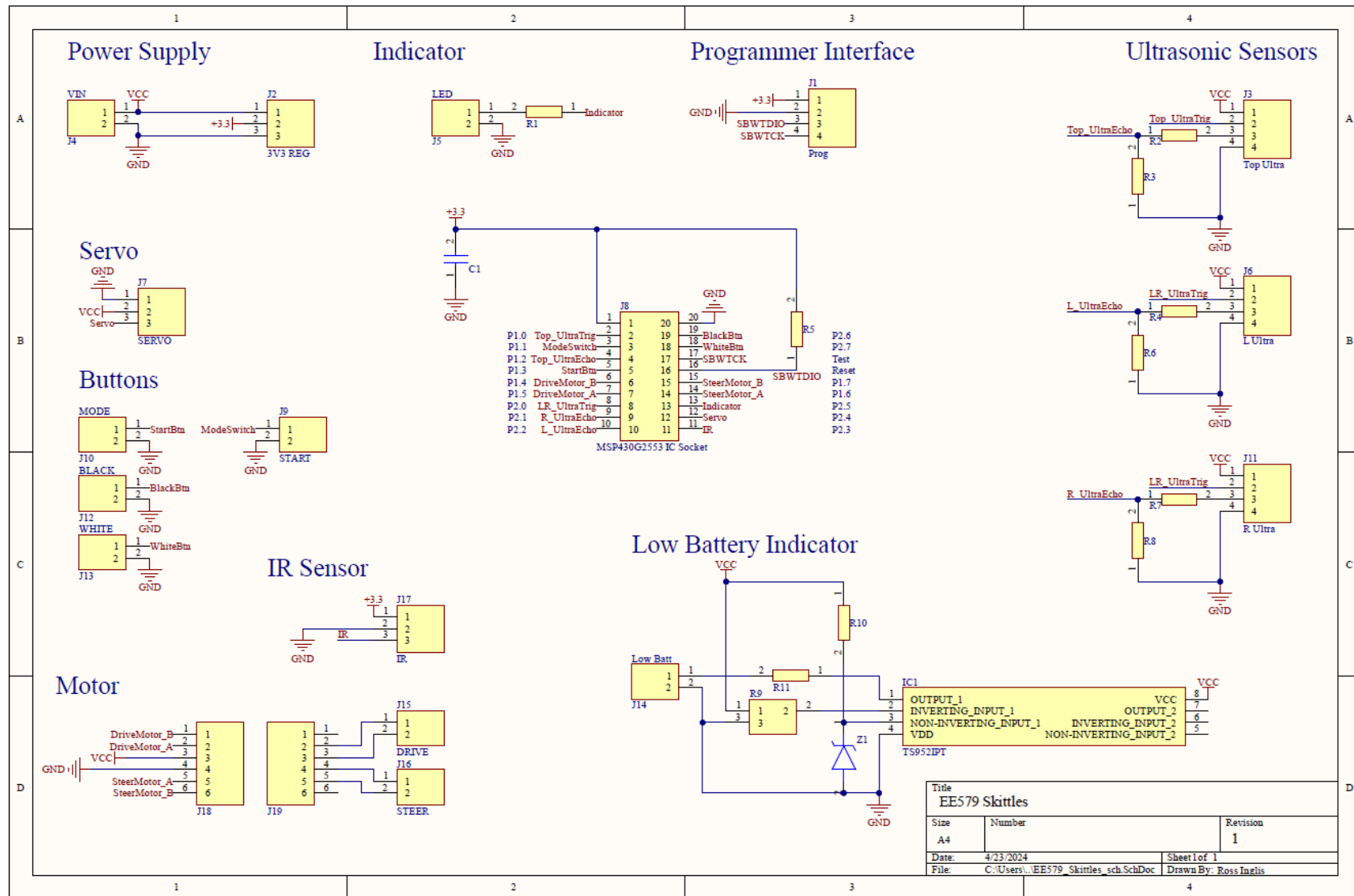
The low battery indicator could also be improved to stop the flashing of the LED when the car is moving. A simple solution could possibly be to add some capacitors for some small energy storage and smooth out any transients.

10 References

- [1] Argos, “BMW i8 1:24 Radio Controlled Sports Car,” Argos, [Online]. Available: <https://www.argos.co.uk/product/7655915?clickSR=slp:term:rc%20car:1:39:1>. [Accessed 02 05 2024].
- [2] “Texas Instruments MSP430G2553IN20, 16bit MSP430 Microcontroller, MSP430, 16MHz, 16 kB Flash, 20-Pin PDIP,” RS Components, [Online]. Available: <https://uk.rs-online.com/web/p/microcontrollers/8176411?gb=s> . [Accessed 05 05 24].
- [3] “Texas Instruments MSP-EXP430G2ET Development Kit,” Mouser, [Online]. Available: <https://www.mouser.co.uk/new/texas-instruments/ti-msp-exp430g2et-dev-kit/>. [Accessed 05 05 2024].
- [4] “DRV8833 Dual H-Bridge Motor Driver Module,” Components101, [Online]. Available: <https://components101.com/modules/drv8833-dual-h-bridge-motor-driver-module>. [Accessed 05 05 24].
- [5] “How does a Servo Motor Work and How to Interface it with ESP32?,” CircuitDigest, [Online]. Available: <https://circuitdigest.com/microcontroller-projects/interfacing-sg90-servo-motor-with-esp32> . [Accessed 05 05 25].
- [6] S. Vorkoetter, “Choosing and Using Nickel-Metal-Hydride (NiMH) Rechargeable Batteries,” [Online]. Available: http://www.stefanv.com/electronics/using_nimh.html. [Accessed 04 05 24].
- [7] “Make a Good Dupont Pin-Crimp EVERY TIME!,” [Online]. Available: <https://www.instructables.com/Make-a-Good-Dupont-Pin-Crimp-EVERY-TIME/>. [Accessed 05 05 24].
- [8] Texas Instruments, “MSP430G2x53 Datasheet,” Texas Instruments Incorporated, Dallas, 2013.
- [9] Texas Instruments, “MSP430F2xx, MSP430G2xx Family: User’s Guide,” Texas Instruments Incorporated, Dallas, 2022.

11 Appendix

11.1 PCB Schematic



*BLACK button was unused, instead blue indicator LED; WHITE was unused entirely.

11.2 Bill of Materials (BoM)

	Component ID	Name	Type	Quantity	PCB Designator(s)	Package	Manufacturer	Manufacturer Part Number	Distributor	Distributor Stock Number	Individual Cost (inc VAT) (£)	Total Cost (£)	Note
Electronics	E-01	MSP430G2553	Microcontroller	1	-	DIP-20	Texas Instruments	MSP430G2553IN20	RS	817-6411	£2.802	£2.802	Free from Lab
	E-02	20-pin IC socket	IC Socket	1	J8	DIP-20	Winslow	W3120TRC	RS	402-793	£0.143	£0.143	
	E-03	2-pin JST PH2.0 Header	Header	9	J4,J5,J9,J10,J12,J13,J14,J15,J16	2mm THT	JST	B2B-PH-K-S(LF)(SN)	RS	166-7311	£0.068	£0.612	
	E-04	3-pin JST PH2.0 Header	Header	1	J17	2mm THT	JST	B3B-PH-K-S(LF)(SN)	RS	820-1431	£0.046	£0.046	
	E-05	4-pin JST PH2.0 Header	Header	3	J3,J6,J11	2mm THT	JST	B4B-PH-K-S(LF)(SN)	RS	172-4893	£0.118	£0.354	
	E-06	3-pin Dupont Male Header	Header	1	J7	2.54mm THT	Würth Elektronik	61300311121	RS	172-5145	£0.070	£0.070	
	E-07	3-pin Dupont Female Header	Header	1	J2	2.54mm THT	Würth Elektronik	61300311821	RS	233-9349	£0.296	£0.296	
	E-08	4-pin Dupnt Male Header	Header	1	J17	2.54mm THT	Würth Elektronik	61300411121	RS	172-5155	£0.095	£0.095	
	E-09	6-pin Dupont Female Header	Header	2	J18,J19	2.54mm THT	Würth Elektronik	61300611821	RS	205-3156	£0.221	£0.442	
	E-10	AMS1117 Module	3.3V Regulator Module	1	-	2.54mm THT	-	-	Amazon	B0CV371GGF	£0.370	£0.370	
	E-11	DRV8833 Module	Motor Driver Module	1	-	2.54mm THT	-	-	Amazon	B0922JV3LY	£1.198	£1.198	
	E-12	470R Resistor	Resistor	5	R1,R2,R4,R7,R11	0805	Bourns	CR0805-FX-4700ELF	RS	161-6929	£0.005	£0.025	
	E-13	1k Resistor	Resistor	4	R10	0805	Bourns	CR0805-JW-102ELF	RS	167-4709	£0.004	£0.016	
	E-14	10k Resistor	Resistor	1	R5	0805	Bourns	CR0805-FX-1002ELF	RS	167-5121	£0.005	£0.005	
	E-15	Zener Diode	Zener	1	Z1	SOD-123	onsemi	MMSZ5226BT1G	RS	545-2989	£0.115	£0.115	
	E-16	10k Potentiometer	Potentiometer	1	R9	-	Bourns	3314G-1-103E	RS	100-1177	£1.564	£1.564	
	E-17	TS952IPT OP-AMP	Operational Amplifier	1	IC1	TSSOP-8	STMicroelectronics	TS952IPT	RS	795-8694	£0.850	£0.850	
	E-18	IR Sensor	Infrared Sensor	1	-	-	-	-	Amazon	B08BK45B5W	£1.632	£1.632	
	E-19	HC-SR04 Ultrasonic Sensor Module	Ultrasonic Sensor	3	-	-	-	-	Amazon	B07XF4815H	£1.498	£4.494	
	E-20	SG-90 Servo	Servo	1	-	-	-	-	Amazon	B09FL1W3KM	£2.000	£2.000	
	E-21	Start Button	Push Button	1	-	-	Shin Chin	R13-510A-05-BR	Mouser	112-R13-510A-R	£0.963	£0.963	Free from Lab
	E-22	Side Select Switch	Switch	1	-	-	E-Switch	EG1201A	Mouser	612-EG1201A	£0.610	£0.610	Free from Lab
	E-23	NIMH Batteries (2400 mA)	Batteries	4	-	AA	Amazon	B08BK45B5W	Amazon	B08BK45B5W	£1.365	£5.460	
	E-24	2-pin JST Connector Housing	Connector Housing	5	-	2mm THT	JST	PHR-2	RS	166-7504	£0.034	£0.170	
	E-25	3-pin JST Connector Housing	Connector Housing	1	-	2mm THT	JST	PHR-3	RS	820-1475	£0.065	£0.065	
	E-26	4-pin JST Connector Housing	Connector Housing	3	-	2mm THT	JST	PHR-4	RS	820-1478	£0.085	£0.255	
	E-27	JST PH Female Crimp	Female Crimp	25	-	-	JST	SPH-002T-P0.55	RS	166-7577	£0.018	£0.450	
	E-28	Red LED	LED	2	-	5mm THT	Lite-On	LTL-307E	RS	168-9439	£0.040	£0.080	Free from Lab
	E-29	Blue LED	LED	1	-	5mm THT	Würth Elektronik	151051B504000	RS	146-0102	£0.137	£0.137	Free from Lab
	E-30	470R Resistor	Resistor	1	-	THT	RS PRO	739-7430	RS	739-7430	£0.010	£0.010	Free from Lab
	E-31	Single AA Battery Holder	Battery Holder	1	-	-	RS PRO	512-3574	RS	512-3574	£1.490	£1.490	
	E-32	3-pin 2.54mm Connector Housing	Connector Housing	1	-	2.54mm	HARWIN	M20-1060300	RS	173-0802	£0.078	£0.078	
	E-33	4-pin 2.54mm Connector Housing	Connector Housing	2	-	2.54mm	HARWIN	M20-1060400	RS	173-0807	£0.113	£0.226	
	E-34	2.54mm Female Crimp	Female Crimp	15	-	-	HARWIN	M20-1180046	RS	681-2887	£0.058	£0.870	
Mounts	M-01	Lower Mount	Mount	1	-	-	-	-	-	-		£0.000	
	M-02	Upper Mount	Mount	1	-	-	-	-	-	-		£0.000	
	M-03	Ultrasonic Mount	Mount	3	-	-	-	-	-	-		£0.000	
	M-04	Ultrasonic Backplate	Mount	3	-	-	-	-	-	-		£0.000	
	M-05	IR Lower Mount	Mount	1	-	-	-	-	-	-		£0.000	
	M-06	IR Upper Mount	Mount	1	-	-	-	-	-	-		£0.000	
	M-07	Servo Base Mount Left	Mount	1	-	-	-	-	-	-		£0.000	
	M-08	Servo Base Mount Right	Mount	1	-	-	-	-	-	-		£0.000	
	M-09	Servo Ultrasonic Mount	Mount	1	-	-	-	-	-	-		£0.000	
	M-10	Control Panel Bottom	Mount	1	-	-	-	-	-	-		£0.000	
	M-11	Control Panel Middle	Mount	2	-	-	-	-	-	-		£0.000	
	M-12	Control Panel Top	Mount	1	-	-	-	-	-	-		£0.000	
	M-13	M2x16 Bolt	Bolt	6	-	M2x16	-	-	-	-		£0.000	
	M-14	M2x12 Bolt	Bolt	6	-	M2x12	-	-	-	-		£0.000	
	M-15	M2x8 Bolt	Bolt	3	-	M2x8	-	-	-	-		£0.000	
	M-16	M2 Nut	Nut	8	-	M2	-	-	-	-		£0.000	
	M-17	M2 Washer	Washer	5	-	M2	-	-	-	-		£0.000	
	M-18	M2 Threaded Inserts	Threaded Inserts	2	-	M2	-	-	-	-		£0.000	
	M-19	Servo Screw	Screw	1	-	-	-	-	-	-		£0.000	Only Phillips head
	Total											£27.993	
Extras	E-01	Base RC Car	Remote Controlled Car	1	-	-	Raster Toys	-	Argos	-	£12.000	£12.000	
	E-02	MSP-EXP430G2ET Dev Board	Development Board	1	-	-	Texas Instruments	MSP-EXP430G2ET	DigiKey	296-50264-ND	£9.510	£9.510	