



Building An Autonomous Robot

Fourth Year Honours Project

Scott McIntosh Gibb

Computer and Electronic Systems

Computer Science and Electronic and Electrical Engineering

University of Strathclyde, Glasgow

January 28, 2021

I 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 my own work at the University of Strathclyde under the supervision of Dr John Levine.

Student:_____

Date:_____

Acknowledgements

Firstly, I would like to thank both my Mother and Father for their continued support throughout my degree. Especially my father Colin Gibb, for the continued advice and vital assistance with the construction aspect of the project.

I would like to also thank Dr John Levine for being a very supportive Mentor, allowing me to follow my dreams with the project. I would like to thank Dr James Irvine for the emotional support as well as the advice he has given me in my honour's year.

Abstract

A skills gap is emerging in which not enough young people are pursuing STEM-related fields. The Rampaging Chariots is an organisation which aims to close this gap by creating advanced robots and presenting the different types of engineering to young people.

This report proposes an Autonomous Upgrade Kit for the Radio Controlled Rampaging Chariots Robot, which consists of a variety of sensors and multiple hardware designs that can be built by young people. Along with hardware, the report consists of software, firmware and of course mechanical solutions to building an autonomous robot for the Rampaging Chariots. It is shown that achieving a fully autonomous robot from the ground up requires a significant amount of work and understanding out with the time constraints of a typical honours project.

Contents

Acknowledgements	i
Abstract	i
List of Figures	x
List of Tables	xvi
1 Introduction	2
1.1 Brief	2
1.2 Motivation	3
1.3 Report Overview	4
2 Related Work	6
2.1 Rampaging Chariots Base Model	6
2.2 Rampaging Chariots Existing Autonomous Platform	7
2.3 Autonomous Robots	9
2.3.1 Sojourner Mars Rover	9
2.3.2 HUSKY	10
2.3.3 Pioneer	10
2.4 Rampaging Chariots Assault Course	11
3 Specification and Methodology	13
3.1 System Specification	13
3.2 Requirements	13

Contents

3.2.1	Rampaging Chariots Guild Requirements	14
3.2.2	Functional Requirements	14
3.2.3	Non Functional Requirements	15
3.3	Methodology	15
3.3.1	Power Electronics	15
3.3.2	Control Theory	22
3.3.3	Microcontrollers	24
3.3.4	Single Board Computers	26
3.3.5	Sensors	26
3.3.6	Kinematics	31
3.3.7	localisation	33
3.3.8	Communication Networks	34
3.3.9	Design Patterns	38
4	Mechanical Design and Implementation	44
4.1	Requirements	44
4.2	Design Process	44
4.3	Implementation	45
4.3.1	Battery Holder	45
4.3.2	Camera Module	45
4.3.3	Control Panel	45
4.3.4	Encoder Mounts	46
4.3.5	Inertial Measurement Unit Mount	46
4.3.6	Infrared Sensor Mounts	46
4.3.7	Ultra Sonic Sensor Mounts	47
4.3.8	PCB and Veroboard Stack Panels	49
4.3.9	Cable Management and Dividers	52
4.3.10	Top Panel Assembly	52
4.4	Full System Assembly	52

Contents

5 Software Design and Implementation	54
5.1 System Overview	54
5.2 High Level Communications	54
5.3 Control Interface	55
5.3.1 Requirements	55
5.3.2 System Structure	56
5.3.3 Model Structure	56
5.3.4 Observer Pattern Implementation	61
5.3.5 View and Presenter Structure	64
5.4 Robot Controller	67
5.4.1 Requirements	68
5.4.2 Python	68
5.4.3 Observable Pattern	68
5.4.4 System Architecture	68
5.4.5 Miscellaneous	71
5.4.6 Movement	71
5.4.7 Proportional Integral Derivative Controller	72
6 Electronic Design and Implementation	74
6.1 Requirements	74
6.2 Design Process	74
6.2.1 Low Level Communication Network	74
6.3 Hardware	75
6.3.1 Power Converter Board	76
6.3.2 Primary Power Distribution Board	80
6.3.3 Secondary Power Distribution Board	82
6.3.4 Sensor Controller Board	84
6.3.5 Raspberry Pi Hardware Attached on Top	90
6.3.6 Inertial Measurement Unit External Hardware	93
6.4 Firmware	96
6.4.1 Implementation	96

Contents

7 Discussion	100
7.1 Unforeseen circumstances and Project Delays	100
7.1.1 Software	100
7.1.2 Firmware	101
7.1.3 Hardware	101
7.1.4 Mechanical	102
7.2 Improvements	102
7.2.1 Software	102
7.2.2 Firmware	103
7.2.3 Hardware	104
7.2.4 Mechanical	106
7.3 Further Work	107
7.3.1 Rampaging Chariots Challenges	108
7.3.2 Exploded View Video Rendering	109
7.3.3 Computer Vision	109
7.3.4 Swarm Robotics	109
7.3.5 Robot Arm Add-on	110
7.3.6 Suspension System	110
7.3.7 Internal Robotic System Add-ons	110
7.3.8 Transition to Fusion 360	111
8 Conclusion	112
Bibliography	112
A System Testing	122
A.1 Mechanical Testing	122
A.2 Software Testing	122
A.3 Firmware Testing	123
A.4 Hardware Testing	123

Contents

B User Guide	125
B.1 System Setup	125
B.2 Control Interface Instructions	125
C Maintenance Guide	127
C.1 Software	127
C.1.1 Python Robot Controller	127
C.1.2 Java Control Interface	130
C.2 Firmware	130
C.2.1 Adding Ultrasonic Sensors	130
C.2.2 Adding Infrared or Analogue Sensors	131
C.2.3 Adding Encoders	131
C.2.4 Adding Digital Sensors	131
C.2.5 Maintaining Communications	131
C.2.6 Changing Main Application Control	132
C.3 Hardware	132
C.3.1 Power Converter Board	132
C.3.2 Primary Power Distribution Board	133
C.3.3 Secondary Power Distribution Board	133
C.3.4 Sensor Controller	133
C.3.5 Adding Digital Sensors	136
C.3.6 Raspberry Pi Hardware Attached on Top	136
C.3.7 Ensuring correct button input voltage	137
C.3.8 Inertial Measurement Unit External Hardware	137
C.4 Mechanical	138
C.5 Program Versions and Libraries	138
C.5.1 Programming Software Libraries	138
C.5.2 Software Tools Versions	141
C.5.3 Raspberry Pi Start Up Script	141

Contents

D Communication Ports and I2C Addresses	143
D.1 I2C Addresses	143
D.2 Internet Protocol Ports	143
E Wiring Colouring Scheme	145
F Hardware Schematics, Layouts and Renders	146
F.1 Hardware Schematics	146
F.1.1 Raspberry Pi Hardware on Top Schematic	146
F.1.2 Sensor Controller Schematic	149
F.1.3 Power Convertor Schematic	152
F.1.4 Primary Power Distribution Schematic	152
F.1.5 Secondary Power Distribution Schematic	153
F.1.6 Inertial Measurement Unit External Hardware Schematic	154
F.2 PCB Layouts	155
F.2.1 Raspberry Pi Hardware on Top Layout	156
F.2.2 Sensor Controller Layout	157
F.2.3 Power Converter Layout	158
F.2.4 Primary Power Distribution Layout	159
F.2.5 Secondary Power Distribution Layout	160
F.2.6 Inertial Measurement Unit External Hardware Layout	161
F.3 PCB Eagle Renders	162
F.3.1 Raspberry Pi Hardware Attached on Top Renders	163
F.3.2 Sensor Controller Renders	163
F.3.3 Power Converter Renders	164
F.3.4 Primary Power Distribution Board Renders	165
F.3.5 Secondary Power Distribution Board Renders	166
F.3.6 Inertial Measurement Unit External Hardware Renders	167
F.4 Required PCB Components	169
F.4.1 Raspberry Pi Hardware Attached on Top PCB Components . . .	169
F.4.2 Sensor Controller PCB Components	169

Contents

F.4.3	Power Converter PCB Components	170
F.4.4	Primary Power Distribution PCB Components	171
F.4.5	Secondary Power Distribution Components	171
F.4.6	Inertial Measurement Unit External Hardware PCB Components	172
G	Required Parts Overview	173
G.1	Full System Assembly	173
G.2	Battery Holder Assembly	177
G.3	Camera Module Assembly	177
G.4	Control Panel Assembly	178
G.5	Encoder Modules Assembly	178
G.6	Top Panel Assembly	179
G.7	Inertial Measurement Unit Assembly	179
G.8	Infrared Sensor Modules	179
G.9	Ultrasonic Sensor Modules	180
G.10	Cable Management and Mounting Modules	180
H	Mechanical Model Renders	183
H.1	Base Rampaging Chariots CAD Models	183
H.1.1	Strengthening Brackets	185
H.1.2	Chassis Panels	185
H.1.3	Wheel Assembly Parts	186
H.1.4	Electronic Specific Components	187
H.2	Battery Holder	188
H.3	Camera Module	189
H.4	Control Panel	191
H.5	Encoder Mounts	195
H.6	Top Panel	196
H.7	Inertial Measurement Unit Mount	197
H.8	Infrared Sensor Mounts	198
H.9	Ultrasonic Sensor Mounts	202

Contents

H.10 Cable Management Parts	206
H.11 PCB Footprints	207
H.12 Autonomous Robot Assembly	208
I Robot Construction	209
I.1 Battery Holder	209
I.2 Camere Module Construction	210
I.3 Control Panel	210
I.4 Inerial Measurement Unit	210
I.5 Cable Management Dividers	211
J Printed Circuit Board Manufacturing Guide	212
J.1 Introduction	212
J.2 University of Strathclyde PCB Workshop	212
J.3 JLCPCB PCB Prototype and Fabrication	213
K 3D Printing and Laser Cutting Guide	214
K.1 Introduction	214
K.1.1 Prusa I3 Mk3S	214
K.1.2 Wanhao I3 Duplicator V2	216
K.1.3 University of Strathclyde Design Manufacture and Engineering Management Workshop	218
K.1.4 Filament Requirements	218
L COVID-19 Setbacks	221
L.1 Cancellation of Face-To-Face Activities	221
L.2 Steps Undertaken to Mitigate Technical Risks to the Project	222
L.3 Impacts of Regular Project Management Activities	222

List of Figures

2.1	Rampaging Chariot Radio Controlled Robot [1]	7
2.2	Rampaging Chariot CAD Drawing [1]	7
2.3	Autonomous Rampaging Chariots Kit Extract [1]	8
2.4	Autonomous Rampaging Chariot	8
2.5	Sornjour Mars Rover [2]	9
2.6	ClearPath Robotics HUSKY Robot [3]	10
2.7	Pioneer Robots [4]	11
2.8	Rampaging Chariots Assault Course [1]	12
3.1	Boost Converter Circuit Diagrams [5]	16
3.2	Buck Converter Circuit Diagrams [5]	18
3.3	Motor Control Theory Diagram	19
3.4	Brushed DC Motor Example Diagrams [6]	20
3.5	Servo Motor Examples	21
3.6	Proportional Integral Derivative Controller Diagram [7]	22
3.7	PID Controller Unstable States	23
3.8	Moving Average Filter Example [8]	24
3.9	STM32F103CT8 Functional Diagram [9]	25
3.10	STM32 BluePill [10]	25
3.12	AS5600 Functional Block Diagram [11]	28
3.13	HC-SR04 Diagrams	29
3.14	Ultrasonic Sensor Common Detection Problems	30
3.15	Reflected Light Sensor Operational Diagrams	30

List of Figures

3.16 GP2Y0A21YK0F SHARP IR Sensor Diagrams	31
3.17 Differential Drive Robot Reference Frames [12]	31
3.18 Kinematic Matrices [12]	32
3.19 Robot Kinematics Example [12]	33
3.20 Robot Localisation [13]	34
3.21 Inter-Integrated Circuit Example Network [14]	36
3.22 Serial Peripheral Interface Example Diagram [15]	37
3.23 Universal Asynchronous Receiver Transmitter [16]	38
3.24 Factory Method Class Diagram [17]	39
3.25 Observer Generic Class Diagram [18]	40
3.26 Adapter Design Pattern [19]	41
3.27 Private Class Data Design Pattern [20]	42
3.28 Model View Controller Use Case Diagram [21]	43
3.29 Model View Presenter Design Pattern [22]	43
4.1 SHARP IR Sensors Pictures	47
4.2 Infrared Sensor Coverage	47
4.3 Ultrasonic Sensor Coverage	48
4.4 Ultrasonic Sensor Mounts Pictures	48
4.5 Autonomous Robot Raspberry Pi Stack	49
4.6 Autonomous Robot Sensor Controller Stack	50
4.7 Autonomous Robot Front Stack	51
4.8 Autonomous Robot Full Assembly	53
5.1 High Level Communications Diagram	55
5.2 Control Interface Model Package Structure	56
5.3 AutoChariot Package Class Diagram	57
5.4 Movement Package Class Diagram	58
5.5 Sensor Controller Class Diagram	59
5.6 Camera Module Class Diagram	60
5.7 Camera Module Java Class Diagram	60

List of Figures

5.8 Java Interface Observer Pattern High Level Implementation	61
5.9 Java Control Interface Communication Package Class Diagram	62
5.10 Primary Packager-Decoder Class Architecture	63
5.11 CameraModule Presenter Class Structure	64
5.12 Camera Module View	65
5.13 Motor Control View	65
5.14 MotionDynamics View	66
5.15 SensoryData View	66
5.16 Telemetric FX View	67
5.17 WheelDynamics View	67
5.18 Python Module Structure	69
5.19 Manual Control Interface Package Class Diagram	71
6.1 I2C Communication Network	75
6.2 Power Converter Functional Diagram	77
6.3 Power Converter Sub Section Circuit Diagram	78
6.4 Power Converter Through Hole Components	79
6.5 Populated Power Converter Board	80
6.6 Primary Power Distribution Board Functional Diagram	81
6.7 Populated Primary Power Distribution Board	82
6.8 Secondary Power Distribution Functional Diagram	83
6.9 Secondary Power Distribution Circuit Diagram	83
6.10 Populated Secondary Power Distribution Board	84
6.11 Sensor Controller Functional Diagram	85
6.12 Sensor Controller Pin Utilisation	86
6.13 Sensor Controller: Ultrasonic Circuitry	87
6.14 Sensor Controller: Infrared Input Circuitry	88
6.15 Sensor Controller: Encoder Input Circuitry	89
6.16 Sensor Controller: Power Regulation Circuitry	90
6.17 Populated Sensor Controller	90
6.18 Raspberry Pi HAT Functional Diagram	91

List of Figures

6.19	SparkFun Logic Converter [23]	92
6.20	Raspberry Pi HAT Voltage Divider Circuitry	92
6.21	Populated Raspberry Pi HAT	93
6.22	IMU External Hardware Functional Diagram	94
6.23	Populated IMU External Hardware	95
6.24	Sensor Controller High Level Firmware Diagram	97
6.25	Sensor Controller Drive Control Firmware Diagram	98
6.26	Sensor Controller Ultrasonic Sensors Handler Firmware Diagram	98
6.27	Sensor Controller Infrared Sensors Handler Firmware Diagram	99
C.1	Power Converter Test Probe Points	132
C.2	Adding More Ultrasonic Sensors Connection Points	134
C.3	Adding Analogue/SHARP IR Sensors to the Sensor Controller	135
C.4	Adding More encoders	135
C.5	Adding more digital Sensors	136
C.6	Raspberry Pi HAT Maintenance Diagram	137
C.7	Raspberry Pi Boot Script	142
F.1	Raspberry Pi Hat Schematic Sheet 1	147
F.2	Raspberry Pi Hat Schematic Sheet 2	148
F.3	Sensor Controller Schematic Sheet 1	149
F.4	Sensor Controller Schematic Sheet 2	150
F.5	Sensor Controller Schematic Sheet 3	151
F.6	Power Convertor Schematic	152
F.7	Primary Power Distribution Schematic	153
F.8	Secondary Power Distribution Schematic	154
F.9	Inertial Measurement Unit External Hardware Schematic	155
F.10	Raspberry Pi Hardware on Top PCB Layout	157
F.11	Sensor Controller PCB Layout	158
F.12	Power Converter PCB Layout	159
F.13	Primary Power Distribution PCB Layout	160

List of Figures

F.14 Secondary Power Distribution PCB Layout	161
F.15 Inertial Measurement Unit External Hardware Layout	162
F.16 Raspberry Pi HAT PCB	163
F.17 Sensor Controller PCB	164
F.18 Power Converter PCB	165
F.19 Primary Power Distribution PCB	166
F.20 Secondary Power Distribution PCB	167
F.21 Inertial Measurement Unit External Hardware PCB	168
H.1 Rampaging Chariots Base Kit Assembled CAD Model	184
H.2 Rampaging Chariots Strengthening Brackets CAD Models	185
H.3 Rampaging Chariot Chassis Panels	186
H.4 Rampaging Chariots Wheel Assembly Parts	187
H.5 Rampaging Chariots Electronic Specific Components	187
H.6 Battery CAD Models Renders	188
H.7 Battery Holder CAD Model	188
H.8 Battery Holder CAD Model Assemblies	189
H.9 Camera Module Predesigned Parts	189
H.10 Camera Module CAD Models	190
H.11 Camera Module CAD Assemblies	191
H.12 Control Panel Button, Switch and Screen CAD Models	192
H.13 Control Panel CAD Model	193
H.14 Control Panel CAD Model Assembly Views	194
H.15 AS5600 and Magnet CAD Models	195
H.16 Drive Wheel Encoder Mount CAD Models and Assemblies	195
H.17 Fly Wheel Encoder Mount CAD Models and Assemblies	196
H.18 Top Panel CAD Models and Assemblies	197
H.19 Inertial Measurement Unit CAD Models and Assemblies	198
H.20 Infrared Sensor and Required Casing CAD Models	199
H.21 Infrared Sensor Front Left Mount CAD Model and Assemblies	200
H.22 Infrared Sensor Corner Mount and Assemblies	201

List of Figures

H.23 HC-SR04 Ultrasonic Sensor CAD Model	202
H.24 Single Mount Ultrasonic Sensor CAD Model and Assemblies	203
H.25 Dual Mount Ultrasonic Sensor CAD Model and Assemblies	204
H.26 Ultrasonic Sensor Corner Mount CAD Model and Assemblies	205
H.27 Mounting Tack CAD Models	206
H.28 Cable Divider CAD Models	207
H.29 PCB and Veroboard Holder CAD Models	207
H.30 PCB and Veroboard Holder CAD Models	208
I.1 Autonomous Robot Battery Holder Pictures	209
I.2 Camera Module Picture	210
I.3 Control Panel	210
I.4 IMU External Hardware Top View	211
I.5 Robot Implemented Dividers	211

List of Tables

1	Acronyms and Definitions	xix
3.1	DC Brushed Motor Characteristics Summary [24]	20
3.2	Servo Motor Characteristics Summary [24]	21
3.3	Stepper Motor Characteristics [24]	22
3.4	Raspberry Pi 3 Model B+ Specifications [25]	26
6.1	Electronic Hardware: PCB Design Summary	76
C.1	Required Java Control Interface Software Versions	138
C.2	Required Python Software and libraries part 1	139
C.3	Required Python Software and libraries part 2	140
C.4	Required Raspberry Pi Software	141
C.5	Software Tools and Versions	141
D.1	I2C Currently Utilised Addresses	143
D.2	High Level Internet Protocol Utilised Ports	144
E.1	Autonomous Robot Wiring Colour Scheme	145
F.1	Raspberry Pi Hardware On Top PCB Component List	169
F.2	Sensor Controller PCB Component List	170
F.3	Power Converter Component PCB List	171
F.4	Caption	171
F.5	Secondary Power Distribution PCB Component List	172
F.6	Inertial Measurement Unit External Hardware PCB Component List	172

List of Tables

G.1	3D Printed components List	174
G.2	Wooden And Laser Cutting Parts	174
G.3	Printed Circuit Board Components	175
G.4	Total Actuator Parts	175
G.5	Required Screws, Bolts and Nuts	176
G.6	Miscellaneous Parts	176
G.7	Battery Holder Full Component List	177
G.8	Camera Module Full Component List	177
G.9	Control Panel Full Component List	178
G.10	Encoder Modules Full Component List	178
G.11	Top Panel Assembly Full Component List	179
G.12	Inertial Measurement Unit Full Component List	179
G.13	Infrared Sensor Modules Full Component List	180
G.14	Ultrasonic Sensor Modules Full Component List	180
G.15	Cable Management and Mounting Modules Component List	181
J.1	University Of Strathclyde PCB Tolerances	213
J.2	JLCPCB PCB Tolerances	213
K.1	Prusa I3 Mk3S Print Settings	215
K.2	Wanhao I3 Duplicator V2 Print Settings	217
K.3	DMEM Laser Cutter	218
K.4	Autonomous Robot CAD Assembly Filament Information	219

Acronyms and Definitions

Since the project is multidisciplinary, acronyms are needed to understand the different parts of the system, a list of acronyms and definitions are shown in table 1

Table 1: Acronyms and Definitions

Acronym	Definition
HAT	Hardware Attached on Top
I2C	Inter Integrated Circuit
GPIO	General Purpose Input Output
ADC	Analogue to Digital Converter
HAL	Hardware Abstraction Library
LCD	Liquid Crystal Display
UART	Universal Asynchronous Receiver/Transmitter
LED	Light Emitting Diode
SPI	Serial Peripheral Interface
STL	Standard Tessellation Language
DXF	Drawing Exchange Format
FXML	Java FX Markup Language
SMBUS	System Management Bus
CAD	Computer Aided Design
FRQ	Functional Requirement
NFRQ	Non Functional Requirement
SMPS	Switch Mode Power Supply
MOSFET	Metal Oxide Silicon Field Effect Transistor
PWM	Pulse Width Modulation
DC	Direct Current
JST	Japanese Solderless Terminal
SSH	Secure Socket Shell
IC	Integrated Circuit

List of Tables

Chapter 1

Introduction

This report details a project by a fourth-year Computer and Electronic Systems (CES) Honours Undergraduate. The project involves developing an autonomous upgrade kit for the Rampaging Chariots Remote controlled robot.

1.1 Brief

The key objectives of the project were first to research current autonomous robotic solutions and understand the problems with these solutions and what trade-offs different robots have. Once this research was undertaken, an informed decision regarding what the best solutions are for robotic systems and how this can be adapted and implemented on the Rampaging Chariot radio-controlled platform was made. The robot should do all movement and observation calculations onboard the robot and should do it on the go. The robot should be able to cope with uncertainties within sensor data and false readings from its environments, as well as this the robot should be able to move with some amount of reliability such that it stays on course with its programmed path. Additional features include expanding the visual interface to be more informative and user-friendly.

1.2 Motivation

In this current day, there is a growing gap in the stem industry in which we need more and more qualified engineers. According to the telegraph, there is a significant UK engineering shortage in which 1.8 million engineers are needed by 2025. [26] To remedy this problem, projects are targeted at young people to engage their interest with technology and how they can contribute to this field. It also shows young people how things work and how they can build things themselves.

There are multiple studies which support the use of STEM-based projects targeted at young people, showing that engagement at a young age, increases the likeliness of them pursuing a STEM-based career.

The Rampaging Chariots are a group of unpaid volunteers that focus on building STEM(Science, Technology, Engineering and Mathematics) based projects. These projects are targeted at young people to engage their interest in engineering and science-based careers. As of 2020, the Rampaging Chariot Guild has only one successful platform, which is the base radio-controlled Rampaging Chariot. Throughout the years the guild has received numerous comments about making the Robot Autonomous and programmable, they launched there own Autonomous version in 2015. However, the robot had some issues with the differential drive control, in which it would skid and as such would be incredibly sensitive to drift errors. The low-level abstraction between the control of the robot and the implementation of the different components, created confusion with the young people, leading to the system not being adopted. [1]

This leaves a massive gap within the guild in that they need an Autonomous platform that can be built by young people between the ages of 16-20. Not only does Robotics interest many people. Studying, building and engaging with robots helps educate people in multiple discipline's which are incredibly useful in the technology industry. Robotics as a discipline is a combination of three primary disciplines, Computer Science, Mechanical Engineering and finally Electronic Engineering. Thus by actively engaging young people with Robotics, we can solve the skill gap by getting young people to study one of the three major engineering disciplines.

1.3 Report Overview

The report consists of eight chapters, each detailing a particular aspect of the project.

Chapter 2 discusses the related work that was studied—explicitly focusing on existing Autonomous robot solutions and the radio-controlled Rampaging Chariots platform. It also goes on to give a high-level general overview of existing autonomous robots looking at their shortfalls and the advantages to each platform.

Chapter 3 looks at the requirements of the autonomous robot, precisely what level of autonomy and what rules apply to the robot regarding the Rampaging Chariots guild. The chapter then goes on to look at the background theory for all the individual components of the robot.

Chapter 4 analyses the mechanical aspect of the robot, looking at the physical requirements and the design process and how the parts were assembled and constructed.

Chapter 5 Opens with a system overview looking at the software elements of the robot, how the system is split up into packages and finally how each module interacts internally and externally. It also looks at the requirements of the software.

Chapter 6 is split into subsections, firmware and hardware. This chapter focuses on the low-level software and how both the firmware and hardware interact.

Chapter 7 looks at the entire system and discusses its drawbacks and how the system can be improved.

Chapter 8 concludes all aspects of the report.

appendix A contains all the different methods used to test the different aspects of the system.

Appendix B gives a high-level guide of how to operate and use the system.

Appendix C explains in further detail how the system can be modified and maintained for future improvement and possible areas for bugs to appear.

Appendix D details the system ports used for communication along with I2C addresses used by the onboard components of the robot.

Appendix E gives a brief description of the wiring colour scheme of the entire robot,

Chapter 1. Introduction

a dedicated scheme to simplify the addition of modules to the robot.

Appendix F gives specific details on the different PCB's required such as the full schematics, layouts and renders.

Appendix G gives a complete overview of the essential parts for the Autonomous upgrade kit. Detailing a full system parts list as well as sub-assembly parts list.

Appendix H provides renders of each CAD model and the individual assemblies for each mechanical component of the system.

appendix I contains pictures related to the physical construction of the prototyped autonomous robot.

Appendix J Opens with the choices of PCB manufacturing available to the project and gives details on the different limitations of PCB manufacturing choices.

Appendix K starts with an overview of the additive manufacturing methods available to the project and the settings required for the models shown in Appendix H.

appendix L discusses the effects of COVID-19 on the developer and the project setbacks associated with the Coronavirus.

Chapter 2

Related Work

The projects first point of research was to look at the Rampaging Chariots Base model and understand precisely how it operates and what drawbacks there is to the design and how this might affect an autonomous solution. After this, other autonomous robots were looked at to see how they operated and what was the benefits to there approach and how they can be improved or what was the significant drawbacks with their approach.

2.1 Rampaging Chariots Base Model

The Base Rampaging Chariots Robot in laymen's terms is a simple radio controlled car(2.1). The design consists of a differential drive steering system powered by 18V batteries. The motors used are Direct Current(DC) Brushed Drill Motors. The CAD Drawing of the layout of the robot is shown in figure 2.2. The drawbacks with this sort of design is that the steerability of the robot relies on the wheels skidding. This results in many measurements based on encoders are likely to be wrong due to wheel slippage. The reason for this is that the robot only has standard wheels which have only two degrees of freedom, the rotation around the axle and the contact point. The benefits of this design is that there are four wheels and thus is far more stable than a caster robot. The design becomes hyperstatic. However, it does not have a suspension system. The design shown in Figure 2.2 also requires that two of the four wheels are not touching the ground. Specifically the flywheels, this, in turn, means that encoders couldn't be

Chapter 2. Related Work

mounted on the flywheels with that setup as the values will be very error-prone and not give viable results.

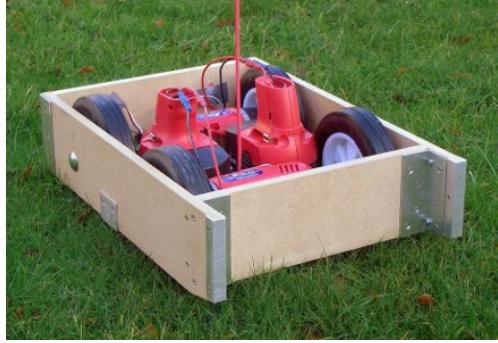


Figure 2.1: Rampaging Chariot Radio Controlled Robot [1]

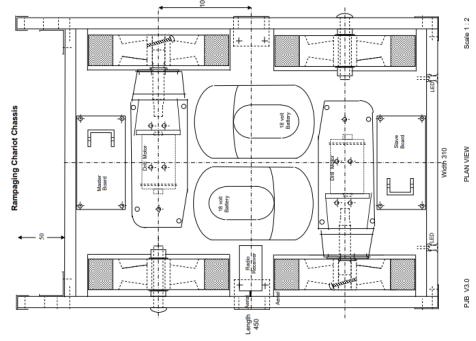


Figure 2.2: Rampaging Chariot CAD Drawing [1]

2.2 Rampaging Chariots Existing Autonomous Platform

The existing Rampaging Chariots Autonomous Platform underwent secondary school testing. However, there were many issues with the design. The school's supervisors for the teams had to be proficient in the autonomous upgrade and the technology used in it as well. This meant that schools couldn't grasp the robot enough to build it. The robot also suffered from numerous problems in that the device would miss calculate its heading and was prone to cumulative errors in the speed and distance calculations. In turn, this meant that the robot would regularly go off course and not follow the defined path. The robot consisted of a variety of proprioceptive and exteroceptive sensors to gauge where it was in its environment. A sample extract of the kits components is shown in Figure 2.3. [1]

Chapter 2. Related Work

The Autonomous Upgrade Kit contains:

- Two Raspberry Pi 3 Model B computers
- SD Cards - Preloaded with the operating system and baseline autonomous software, written in Python and fully documented. This provides a starting point for personal modification and extension of the code. It includes supporting software development tools for Simulation, Visualisation, Telemetry and Datalogging.
- Two Odometers to sense the distance travelled by each drive wheel and allow robot heading to be determined.
- Infra-Red and Ultra Sonic Distance Measurers.
- Stepper Motor and Servo motor for Scanning.
- Interface Module - Protects the R-Pi from damage and facilitates connection to sensors of all descriptions.
- Power Converter (18V to 5V) and interconnecting Switches and Cables.

Figure 2.3: Autonomous Rampaging Chariots Kit Extract [1]

An implemented Autonomous Rampaging Chariot can be seen in Figure 2.4 which was constructed by Peter Bennet(The Head of the Rampaging Chariots Guild). This design shows the implemented features listed in Figure 2.3.



Figure 2.4: Autonomous Rampaging Chariot

The autonomous robot shown in Figure 2.4 uses both Ultrasonic and Infrared Distance sensors. However, the control mechanism for these sensors rely solely on the I2C connected PIC microcontroller. The PIC acts a slave in this network and is controlled from the Raspberry Pi through various pre-coded commands. The Ultrasonic Sensor is mounted on top of an SG90 servo motor which is facing behind the robot. The SHARP IR Sensor is facing forward attached to a stepper motor, the stepper motor uses an end stop switch to reset its position back to zero. The position of the sensor is entirely dependent on the number of steps the motor has taken since it was last reset.

Chapter 2. Related Work

All electronic components are mounted on top of the robot except from the encoders. The encoders are only mounted to the inside of the body and are only sensing the drivewheels.

2.3 Autonomous Robots

There have been many different autonomous robots over the years, and thus a variety of different platforms researched and developed. In order to create an Autonomous Robot, research had to be conducted into existing systems.

2.3.1 Sojourner Mars Rover

The first mars rover was the Sojourner Rover, shown in Figure 2.5. It consisted of six wheels, rotational servo steering and a fully functioning suspension system. Its mission was to explore Mars in the summer of 1997. It was mostly teleoperated from Earth. However, it did have some onboard sensors that were used for obstacle detection and avoidance. [2]

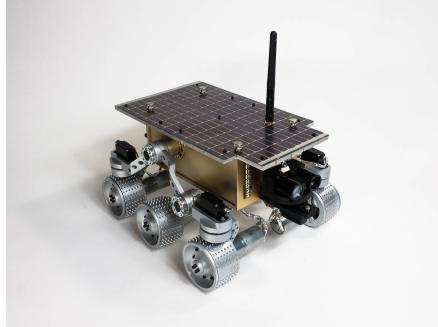


Figure 2.5: Sornjour Mars Rover [2]

One of the onboard sensors was a Laser Imaging Detection and Ranging(LIDAR) system. It had two of these systems(front and back) allowing a full 360 degrees Field of View(FOV). It was also used for autonomous navigation as well as obstacle avoidance. [2]. It also contained a camera so that somewhat live video could be captured and sent back to the home station.

2.3.2 HUSKY

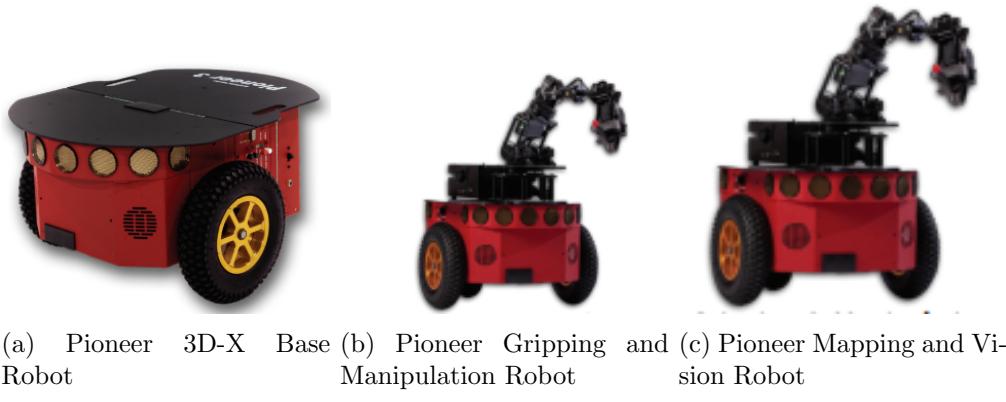
Another Four-wheel drive robot is the HUSKY unmanned ground vehicle made by ClearPath Robotics. [3]. It is built with modularity in mind allowing different payloads to be selected dependent on research needs. The robot itself is shown in Figure 2.6. It is a four-wheel-drive robot, which is the same as the Rampaging Chariots Robot. It's designed for a rugged environment and has precise control over its movement based on high-resolution encoders. [3]



Figure 2.6: ClearPath Robotics HUSKY Robot [3]

2.3.3 Pioneer

The Pioneer 3-DX is a two-wheel differential robot designed for indoor laboratory and classroom use. The robot is shown in Figure 2.7, it comes equipped with SONAR, wheel encoders and an onboard microcontroller running the Pioneer advanced robotics software package. It comes with a modular design allowing different experiments to be undertaken. The options range from a robotic arm to more advanced mapping components. The different add-ons are shown in Figure 2.7. [4]



(a) Pioneer 3D-X Base (b) Pioneer Gripping and Manipulation Robot (c) Pioneer Mapping and Vision Robot

Figure 2.7: Pioneer Robots [4]

2.4 Rampaging Chariots Assault Course

The Rampaging Chariots hosts a set of Robotic games annually at Leonardo [27] Headquarters, the games consist of three categories, two a side football, assault course and finally tug of war. The autonomous robots are automatically entered in the assault course. During there time in the assault course, no human interaction can be done to aid the robot, the only control allowed is a stop and start button for this part of the competition. [1]

The assault course layout is shown in Figure 2.8. With the course there is a set of requirements the robot has to achieve to be classed as an Autonomous Rampaging Chariot, they are listed below: [1]

- must drive through the chicane without knocking any cones down
- push the football through the goal
- pass through the paddle
- go over the see-saw
- push the barrel so that it knocks over the cone
- reverse through the chicane back to the start

All of these requirements have to be achieved while another robot is also doing the

Chapter 2. Related Work

same obstacles—the guild reserve to change the map whenever they deem suitable as well.

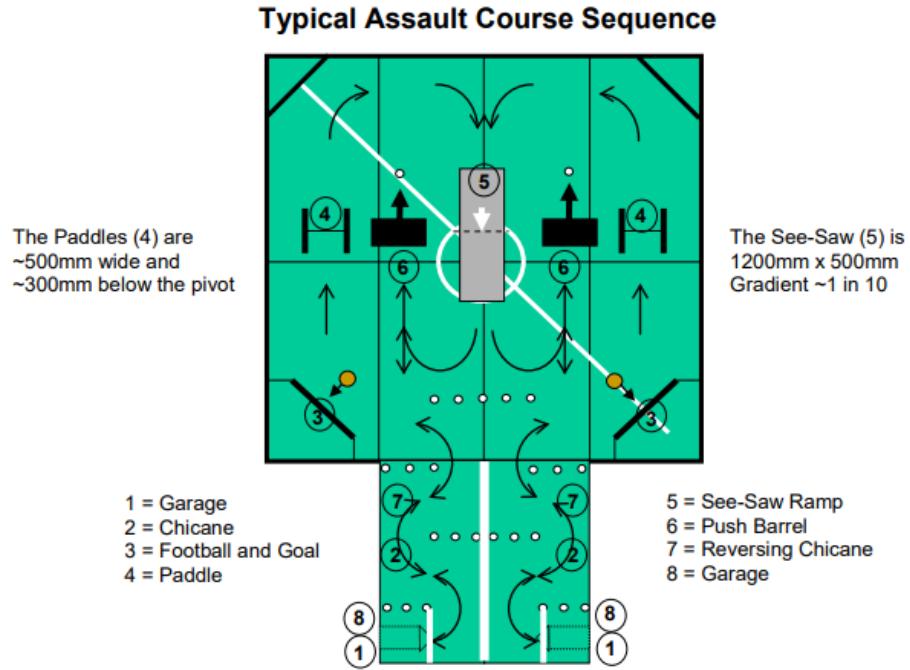


Figure 2.8: Rampaging Chariots Assault Course [1]

Taking a more in-depth look at the assault course, we can see that the movements required for the assault course are such that the robot must be able to do arc turns around the cones after which it must be able to locate and hit the ball into the net with enough force to move the ball but not too much as to cause errors with navigation. Once the ball has been hit, it then needs to go through the paddle which again has a very similar problem to the ball but more likely to damage navigation when travelling through the door due to it having to collide with it. Possibly the biggest challenge is going over the see-saw in which many navigational errors will occur due to the movement of the ramp and possible collisions with other robots.

Chapter 3

Specification and Methodology

This part of the report outlines the specifications of the Autonomous Robot and the various requirements of the robot. It also describes in detail the relevant methodology of how the system aims to solve the problem and the relevant theory needed to understand the operation of the robot. The chapter also includes the various design patterns used for programming the various aspects of the robot.

3.1 System Specification

Using the brief described in chapter 1 and the research into existing solutions in chapter 2, specifications were made for the robot.

All calculations for system movement will be done on-board the robot; the robot will consist of a Raspberry Pi Main Controller tethered to a control Interface allowing Manual interaction with the robot itself. The robot will have a variety of exteroceptive and proprioceptive sensors. The data from these sensors will be processed on the robot itself and relayed to the control interface. The entire system will be powered off a battery and not tethered to the user in any physical way.

3.2 Requirements

To correctly specify the requirements of the project, there must be a clear indication of where the requirements are coming from, and whether or not they are functional or

Chapter 3. Specification and Methodology

non-functional requirements. The requirements are used to clarify what is expected of the robot and what must be done in order for the goals to be achieved

3.2.1 Rampaging Chariots Guild Requirements

The first set of requirements are specified by the Rampaging Guild and are functional requirements in that they have to be abided by in order for the robot to be allowed entry into the Rampaging Chariots Competition. The requirements from the guild are: [1]

- FRQ-1 The robot must have four wheels(Tracks may be fitted)
- FRQ-2 Differential steering must be used
- FRQ-3 The movement of the robot within its environment must be powered by two cordless drill motors with epiclyptic gearboxes
- FRQ-4 Two Rampaging Chariot Motor Control Boards with preloaded Rampaging Chariots firmware must be used
- FRQ-5 The robot must have a remote kill function
- FRQ-6 All navigation and control functions must be undertaken on board the robot

3.2.2 Functional Requirements

The remaining functional requirements of the robot are defined as follows:

- FRQ-7 A hardware and software interface for the pre-existing motor controllers for the Raspberry Pi must be built
- FRQ-8 A variety of different sensors allowing the robot to perceive its environment must be added to the existing body
- FRQ-9 A control application allowing the robot to be controlled through an external device not on the robot itself must be developed

Chapter 3. Specification and Methodology

- FRQ-10 Develop power regulation and distribution circuitry for the robot
- FRQ-11 Develop a solution to the Rampaging Chariots Autonomous Assault Course

3.2.3 Non Functional Requirements

The non-functional requirements are classed as the "wants" of the system and are thus not essential to the project. They are listed below:

- NFRQ-1 Develop appropriate abstraction layers to allow system functionality to be used by school children
- NFRQ-2 Pursue other Autonomous activities such as researching Simultaneous Localisation and Mapping(SLAM)
- NFRQ-3 Use Java [28] and Python [29] as the main programming languages of the system

3.3 Methodology

Within the realm of robotics, there are multiple technologies in which the reader and the developer must become acquainted with—these range from all three major disciplines, Computer Science, Electronic Engineering and finally Mechanical Engineering.

3.3.1 Power Electronics

The first aspect of Electronics that must be researched was Power electronics. Unlike Digital and Analogue electronics in which signals representing some sort of value such as binary data in the case of digital electronics or audio signals in the case of analogue electronics. Power Electronics specifically looks at processing energy and changing current and voltage. [30]

3.3.1.1 Buck and Boost Converters

There are two types of DC-DC converters, Buck and Boost each with different purposes. They do share one characteristic which is creating an output voltage which is different from its input. In Robotics this is incredibly helpful because there is usually only one battery and thus only one voltage. They are also both a form of Switched Mode Power Supplies(SMPS) which uses a high-frequency switching mechanism to change the input voltage to a different output voltage.

3.3.1.1.1 Boost Converters Boost Converters are responsible for increasing an input voltage to a predefined output voltage higher or equal to the input voltage. An example of a boost converter circuit is shown in Figure 3.1. [5]

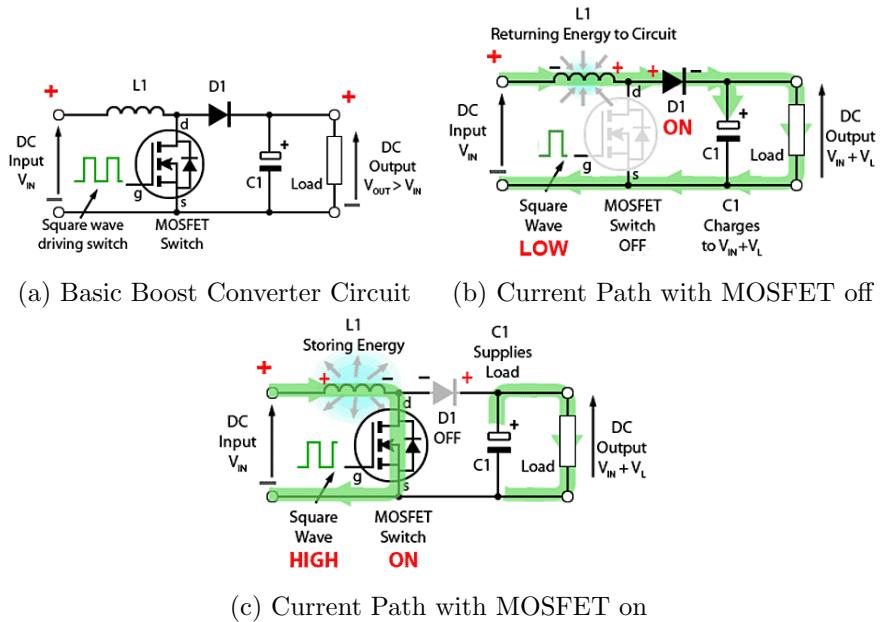


Figure 3.1: Boost Converter Circuit Diagrams [5]

An alternating DC square wave is applied to the MOSFET with a very high frequency to get the best results. During the on-stage 3.1c the MOSFET essentially shorts the circuit and current flows through the MOSFET. The inductor in this instance begins to charge and creates a back EMF-based on the change in current. Diode (D_1) is used to prevent current coming in the opposite direction in the output circuitry

Chapter 3. Specification and Methodology

and potentially damaging the DC load. During this time the load circuitry is being supplied by the capacitor in parallel with the load. During this time the output voltage begins to drop. [5]

During the MOSFET off period 3.1b, the current is flowing into the load circuitry. Since the inductor sees a massive change in current, the inductor creates a back emf in relation to the previous voltage drop. At this point, the load and capacitor are seeing a voltage equal to the voltage over the inductor(back emf) and the voltage at Vin. In this point in the circuit, the inductor is returning its stored energy to the circuit and as such the voltage across the inductor is lowering. [5]

The combination of these two cycles allow a steady DC signal to be created, however, due to the equation shown in equation 3.1 (Power Equation), the consequence with increasing the voltage with respect to the input voltage is that the output current must lower to be the same power. The theoretical output of the boost converter is based on the duty cycle of the switching waveform and can be estimated using the equation shown in equation 3.2. [5] Where V_{OUT} is the output voltage, V_{IN} is the input voltage, and finally, $DutyCycle$ is the corresponding duty cycle of the switching waveform.

$$P = IV \quad (3.1)$$

$$V_{OUT} = \frac{V_{IN}}{1 - DutyCycle} \quad (3.2)$$

3.3.1.1.2 Buck Converters Buck Converters work very similar to boost converters except doing the reverse operation, in that they create a reduced voltage but higher current. An example of a Buck converter circuit is shown in Figure 3.2a. [31]

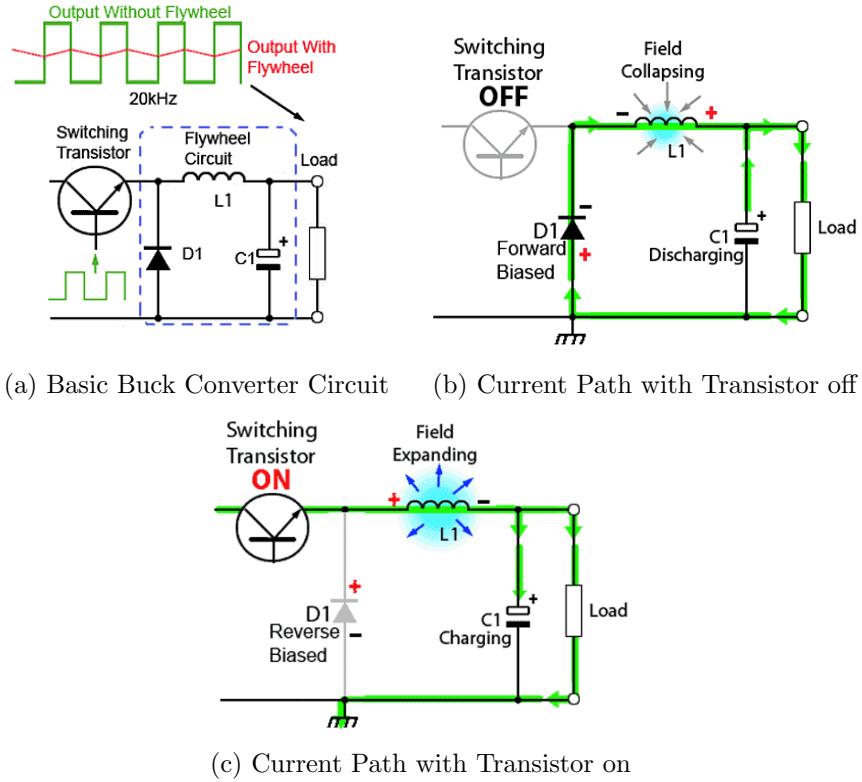


Figure 3.2: Buck Converter Circuit Diagrams [5]

During the on phase(Figure 3.2c, the supply is supplying the load with the current through the inductor, during this time the inductor is charging with respect to the change in current, as well as this the capacitor C1 is also charging. The diode in this configuration will be reverse biased and thus block all current through it. [31]

When the transistor enters the off phase(Figure 3.2b, the inductors magnetic field decreases and thus releases energy into the circuit. The voltage across the inductor is now in reverse polarity compared to the on state(Figure 3.2c). During this time the inductor is forcing the current through the load and back round to the inductor due to the diode being forward biased. Once the inductors magnetic field has weakened significantly, the capacitor (C1) becomes the primary source of current, allowing the load to be powered until the next transistor on period. [31]

This combination of on and off periods results in a triangular ripple waveform at the

load end with smaller amplitude compared to that of the input. The theoretical output voltage of the converter can be calculated using equation 3.3. [31] Where V_{OUT} is the output voltage, V_{IN} , is the input voltage, T is the period of the switching waveform, and t_{ON} is the on period.

$$V_{OUT} = V_{IN} \frac{t_{ON}}{T} \quad (3.3)$$

3.3.1.2 Motor Control

Another important aspect of robotic systems is motor control. Every robotic system has a form of propulsion. Most use a form of motor, the three types of motors that will be discussed in this report is that of brushed DC motors, servo motors and finally a high-level overview of stepper motors.

At an elementary level to drive a motor, the current is flown through a coil which is in a magnetic field. When this happens, a force is produced orthogonal to both magnets. This effect can be shown in Figure 3.3. This force then causes a rotation of the motor rotor. Change of current direction changes the rotation of the motor. The moving coil in a magnetic field causes an induced current along the coil, which in turn creates back emf proportional to the motor speed. [24]

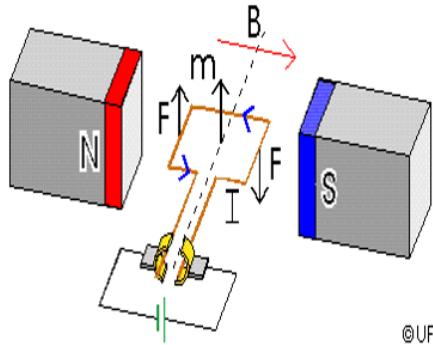


Figure 3.3: Motor Control Theory Diagram

3.3.1.2.1 Brushed DC Motor Brushed DC motors are often the simplest forms of motors. They are composed of a rotor with electromagnets and a stator with permanent magnets. This setup can be seen in Figure 3.4a. The motor starts when the

coil is powered, thus creates a magnetic field around the armature. The left side of the armature is pushed away from the left magnet(blue) and drawn toward the right magnet. During the movement shown in figure 3.4b, the armature becomes vertically aligned and continues to rotate. When the armature reaches the position shown in Figure 3.4c, the torque in the motor becomes zero. At this point, the commutator reverses the direction of current through the coil, which then reverses the magnetic field. This entire process repeats causing continues rotating, the more coils the motor has, the smoother the rotation of the motor and the more constant the torque is. A summary of the DC Brushed Motor is shown in table 3.1. [24]

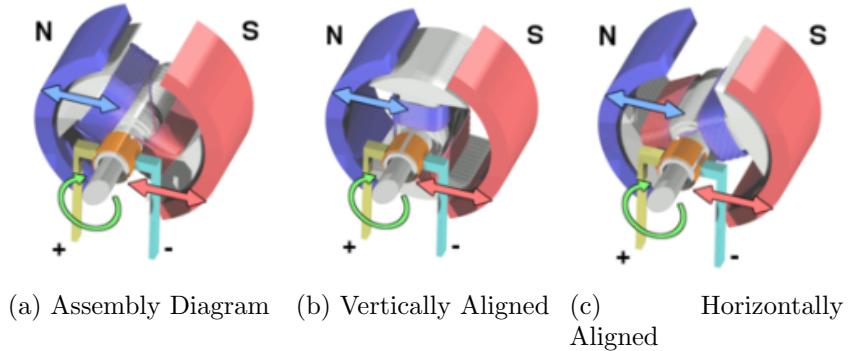


Figure 3.4: Brushed DC Motor Example Diagrams [6]

Table 3.1: DC Brushed Motor Characteristics Summary [24]

DC Brushed Motor Characteristics Summary	
Power Source	DC
Gearbox	Optional
Control Mechanism	Speed dependent on voltage
Torque	Variable
Applications	Drive wheels

3.3.1.2.2 Servo Motor A servo motor is essentially a DC motor but with three additional elements. It contains a gearbox between the motor shaft and the output shaft. Providing a low-speed output but increased torque. Pulse Width Modulation (PWM) is used to drive the servo to the commanded shaft angle, within the servo motor there are two sensors, one for sensing the shaft angle using a potentiometer and

Chapter 3. Specification and Methodology

one for current sensing which is responsible for torque sensing. The final element of a servo motor is the limit stops on the output shaft; these mechanical mechanisms allow the servo to have a minimum and maximum position. For this reason, they are ideal for sensor applications in which a sensor must be rotated around a specific axis. An example of a servo as well as disassembly of one is shown in Figure 3.5 and a summary table of the servo characteristics are shown in 3.2. [32] [24]

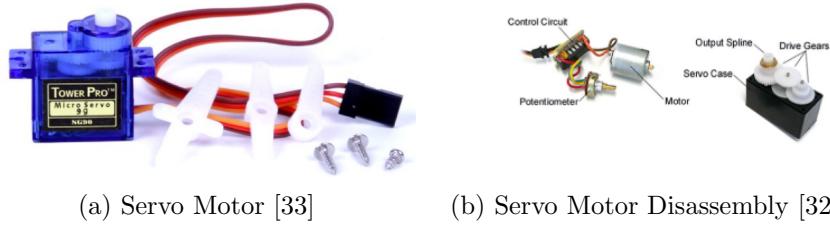


Figure 3.5: Servo Motor Examples

Table 3.2: Servo Motor Characteristics Summary [24]

Servo Motor Characteristics Summary	
Power Source	DC
Gearbox	Built In
Control Mechanism	Feedback based position using PWM
Torque	High Torque
Applications	Scanning Sensors

3.3.1.2.3 Stepper Motor The final motor to be discussed in this report is that of a stepper motor. At its heart, a stepper motor is a DC motor that moves in discrete steps. Inside the stepper motor is a set of coils which are organised into groups called "phases". When each phase is energised in a particular sequence, the motor will move one step at a time. With a microcontroller controlling the motor, they can achieve exact positioning and speed control. A characteristics summary of a stepper motor is shown in table 3.3. [34] [35] [24]

Table 3.3: Stepper Motor Characteristics [24]

Stepper Motor Characteristics Summary	
Power Source	DC
Gearbox	N/A
Torque	High torque
Applications	Robot Arms and precise movement control

3.3.2 Control Theory

One of the most critical aspects of Robotics is control. Within each Autonomous system, there is some level of control, either hardware control or software control, or even in some cases mechanical control. One of the best controllers ever conceived is that of a PID controller it's designed for general applications. However, it is not perfect for every situation, but it tends to be a good starting point.

3.3.2.1 PID Control

PID stands for Proportional, Integral and Derivative and each of these parts contribute to its function. [36] The complete structure of a PID controller can be seen mathematically in equation 3.4 and visually in Figure 3.4.

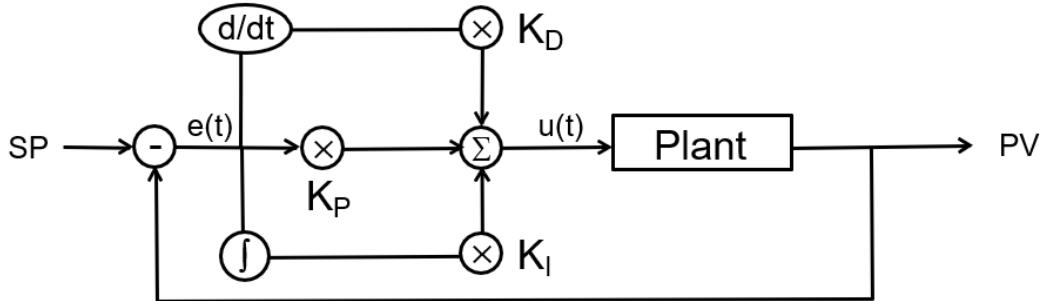


Figure 3.6: Proportional Integral Derivative Controller Diagram [7]

$$[h!]u(t) = [K_P \times e(t)] + [K_I \times \int(e(t))] + [K_D \times \frac{d[e(t)]}{dt}] \quad (3.4)$$

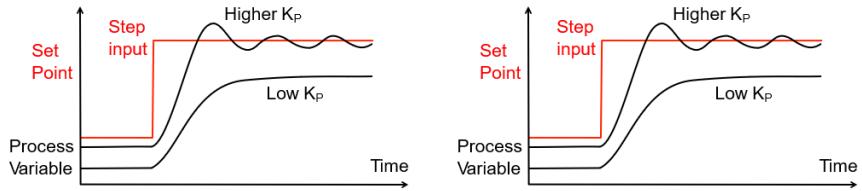
The individual terms are assigned a multiplication factor K , which allow each part of the controller to be tuned. The Proportional value is responsible for assigning a

Chapter 3. Specification and Methodology

multiplication factor such that the individual terms are a multiple of the system error value $e(t)$. The goal of the proportional value is to reduce the magnitude e . A higher P_k value, the faster the PID controller response time. However, by increasing this gain, the controller itself will become unstable due to overshoot and oscillation will occur. This effect is shown in Figure 3.7a. [7]

To combat the overshoot created by the proportional term. A Derivative term is used; this derivative term is based on the error value and effectively allows the controller to dampen the proportional impulse response. It is mainly predicting where the overall process is going and subtracting it from the process to prevent overshoot. However, if the derivative gain is too high, the system will be too damped and won't respond quick enough. This effect is shown in Figure 3.7b. [7]

The final term of the PID controller is the integral term; it's responsible for minimising the error state $e(t)$. It's similar to the proportional term. However, it's not for immediate response and its effect on the system is accumulated based on the PID controller calculation rate. [37]



(a) Proportional Instability Example (b) Derivative Instability Example [7]
[7]

Figure 3.7: PID Controller Unstable States

3.3.2.2 Filtering

In any electronic system, noise is always present in the signals, which causes many errors within the processing part of the system. One way to filter signals on a software level is to introduce a moving average filter. It allows the signal to be averaged out without causing too many discrepancies in the distance. It does this by creating a rectangular window which is moved through the results allowing the filter to average

the signal based on most recent results from the source. An example of this is shown in Figure 3.8. Finally, the filter is expressed as an equation, shown in equation 3.5. [8]

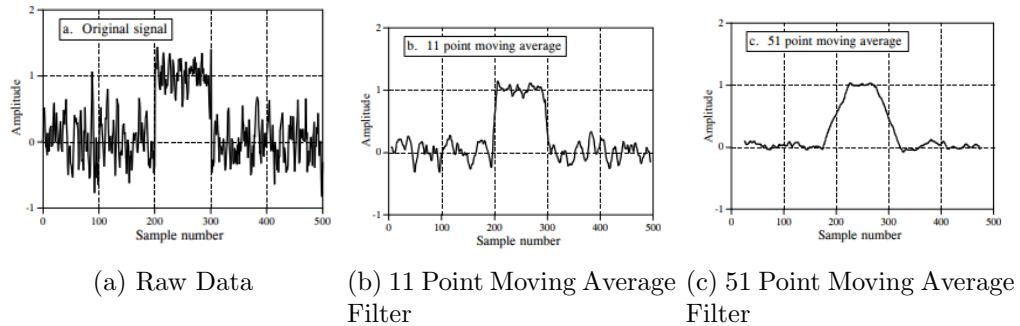


Figure 3.8: Moving Average Filter Example [8]

$$[h!]y[i] = \frac{1}{M} \times \sum_{j=1}^{M-1} X[i+j] \quad (3.5)$$

3.3.3 Microcontrollers

When interacting with sensors, a microcontroller is required. The purpose of a microcontroller is to run a single program and nothing more. The program should be designed to do one task and one task only. They are also heavily engraved in the embedded community in which their purpose is to interact with and control hardware systems. [38]

3.3.3.1 STM32F103CT8

A very powerful yet lightweight microcontroller is the STM32F103CT8 by STMicroelectronics, the functional diagram of which is shown in Figure 3.9.

Chapter 3. Specification and Methodology

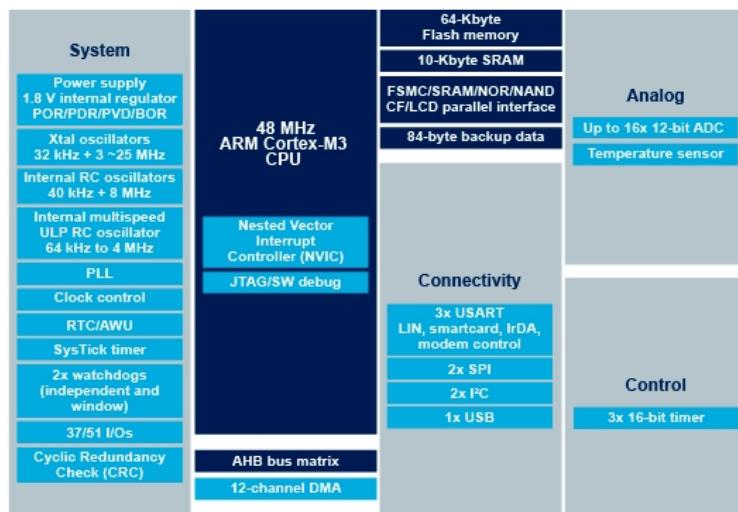


Figure 3.9: STM32F103CT8 Functional Diagram [9]

This type of microcontroller uses the Hardware Abstraction Library (HAL) driver and has extensive documentation explaining how to use each of the peripherals. It also comes with an Arduino Nano board package called the STM32 BluePill. This is shown in Figure 3.10.

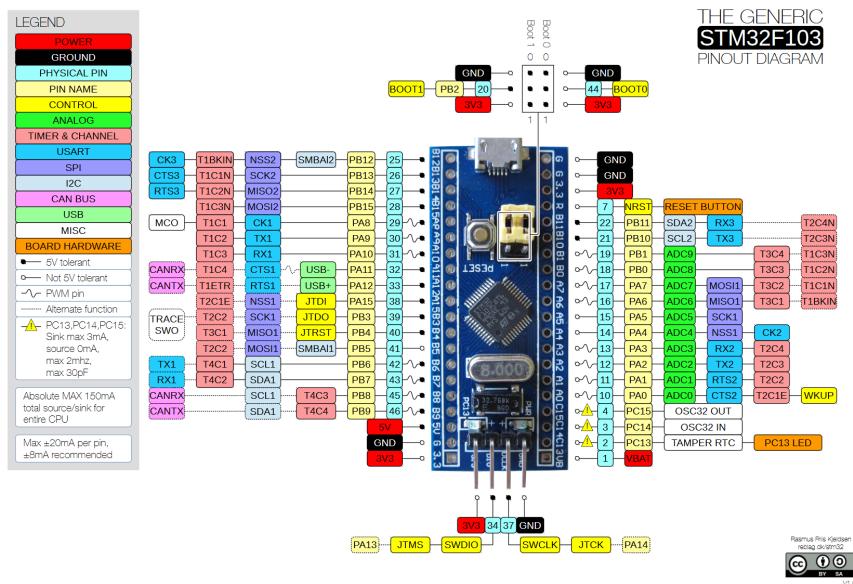


Figure 3.10: STM32 BluePill [10]

3.3.4 Single Board Computers

Some robots use microcontrollers as their primary computational platform; although this is an acceptable method, the Rampaging Chariots Guild would like a Raspberry Pi as the central computational platform. Doing this creates a variety of advantages and allows the target audience to be given a bigger perspective of what is possible in robotics.

3.3.4.1 Raspberry Pi

A Raspberry Pi is a credit-card-sized computer. It can run a variety of different operating systems; in this project Raspbian [39] was used, which is a Linux based operating system specifically designed for Raspberry Pis. The Raspberry Pi also allows for hardware interfacing through its forty General Purpose Input Output(GPIO) pins. Making it the ideal computer for the robots primary computational platform. A summary of its key features are given in table 3.4. [40]

Table 3.4: Raspberry Pi 3 Model B+ Specifications [25]

Raspberry Pi 3 Model B+ Technical Specifications	
Processor	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
Random Access Memory	1GB LPDDR2 SDRAM
Connectivity	Ethernet, WiFi and Bluetooth
Ports	4 X USB2.0, Full Size HDMI
GPIO	40 GPIO Pin Header

3.3.5 Sensors

3.3.5.0.1 Introduction There is a variety of Sensors in robotics. The reason for this is that robots, in general, are unreliable, they are prone to many mistakes such as wheel slippage, differences in motors. Leading to errors building up and causing more and more problems with the robot. Sensors provide a way in which the robot can perceive its environment and correct these positional errors and come in many different types. [41]

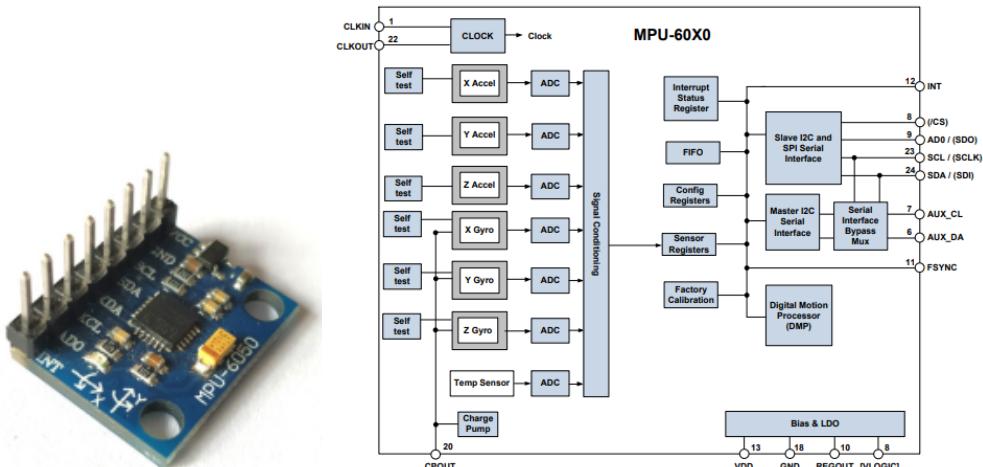
3.3.5.0.2 Proprioceptive Versus Exteroceptive Sensors are usually categorised into two categories, Proprioceptive and Exteroceptive. The meaning of these types are listed below:

- Proprioceptive "sensing internal state - stimuli arising from within the agent (e.g., muscle tension, limb position, motor state)" [41]
- Exteroceptive " sensing external state – external stimuli (e.g., vision, audio, smell, etc.)" [41]

The combination of both these types of sensors allows a robot to have a perceptual system which it can then use for navigation and interacting with its environment.

3.3.5.1 Proprioceptive

3.3.5.1.1 Inertial Measurement Unit One way of measuring a robots orientation is the use of an Inertial measurement unit(IMU) device. An IMU is a combination of multi-axis combinations of precise gyroscopes and accelerometers. In some cases, other sensors are also used such as magnetometers. [42]The IMU used for this project is the MPU6050; the breakout board is shown in Figure 3.11b.



(a) IMU BreakoutBoard [43] (b) MPU6050 Functional Block Diagram Board [44]

3.3.5.1.2 Encoders Encoders fall in to two different categories, incremental and absolute. Absolute encoders give an exact position of the object, whereas an incremen-

tal encoder provides signals which are pulsed every time the object moves a certain amount, from this we can calculate how many times the object has moved a certain amount. The encoder chosen for this project was the AS5600 Absolute Magnetic Encoder (Figure 3.12).

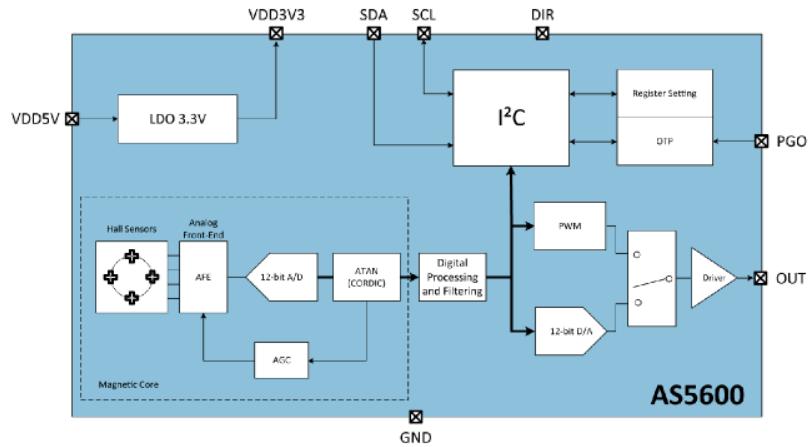


Figure 3.12: AS5600 Functional Block Diagram [11]

The breakout board supplies a PWM 5V signal, all that is required is a magnet attached to the rotating object. The PWM signal represents the angle of the object.

3.3.5.2 Exteroceptive

In terms of exteroceptive sensors, infrared and ultrasonic sensors are used. Both have a different set of problems associated with them.

3.3.5.2.1 Ultrasonic Distance Sensors One way of measuring distance to a target is by using Ultrasonic Distance Sensors. Specifically the HC-SR04 is a digital sensor outputting PWM signal which is directly proportional to the distance to the detected object. The measurement cone along with a picture of the device is shown in Figure 3.13. [45] [46]

The Ultrasnic Sensor works sending out four or more frequencies when the sensor is activated. The reason for this action is that it prevents errors caused b the distance

Chapter 3. Specification and Methodology

being an exact multiple of a particular frequency. The HC-SR04 uses Pulse Mode, which evaluates the time taken for the sound wave to reach the target and reflect. Therefore the distance of the detected object can be calculated using 3.6. Where V_{Air} is the speed of sound in air, δt is the total time for transmission and reception and finally d is the distance.

$$d = \frac{1}{2} \times V_{Air} \times \delta t \quad (3.6)$$

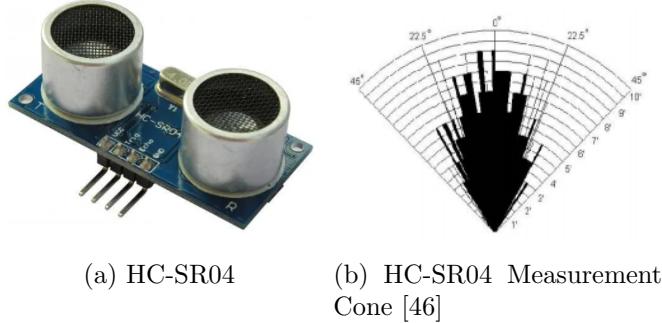


Figure 3.13: HC-SR04 Diagrams

With Ultrasonic Sensors there is a variety of problems associated with them, the most crucial problems are listed below:

- Sound waves bounce off multiple objects before returning to the detector. This is known as specular reflection.
- Some form of objects absorb the sound waves causing the distance calculation to be wrong.
- Cylinders or curved shapes cannot be perceived due to the sound wave reflecting with the wrong angle.
- Objects too close to the sensor can interfere with the results of the sensor.

Representations of these types of problems are shown in Figure 3.14.

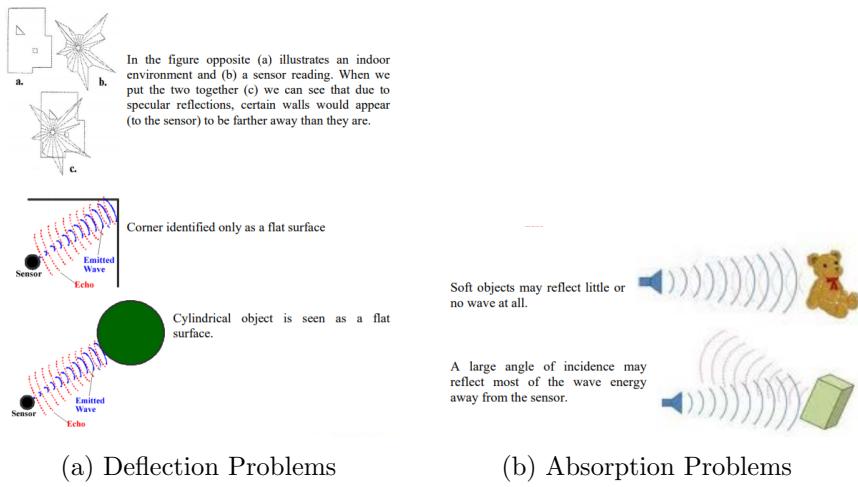


Figure 3.14: Ultrasonic Sensor Common Detection Problems

3.3.5.2.2 Infrared Distance Sensors Another way to detect and calculate distance to objects is to use Infrared light waves rather than sound waves. This type of sensor is known as a Reflected Light Sensor. This type of sensor still has its own set of problems which are listed below:

- Transparent objects are "invisible" to the sensor
- Dark objects absorb the light and thus are harder to detect

The concept of how this particular sensor works is shown in Figure 3.15.

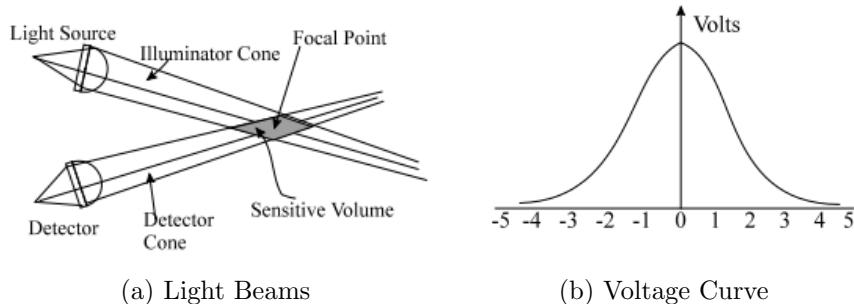


Figure 3.15: Reflected Light Sensor Operational Diagrams

The infrared Distance sensor chosen for this project was the GP2Y0A21YK0F SHARP IR Sensor which is shown in figure 3.16 along with its functional block diagram. [47]

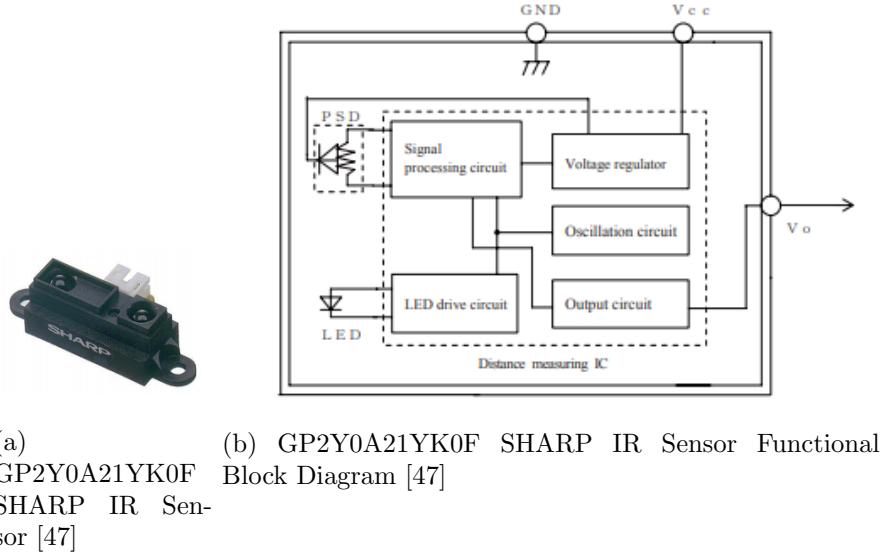


Figure 3.16: GP2Y0A21YK0F SHARP IR Sensor Diagrams

3.3.6 Kinematics

Kinematics is the study of the mechanical behaviour of a system. In this report its the study of a differential drive robot. World frames are used to represent robots in two-dimensional and three-dimensional space. These represent the orientation of the robot in all axes Pitch Yaw and Roll. The initial frame is known as the world frame. It describes the orientation of the world. The robot then uses its frame of reference, which is based on the world frame. A Visual example is shown in Figure 3.17. [48]

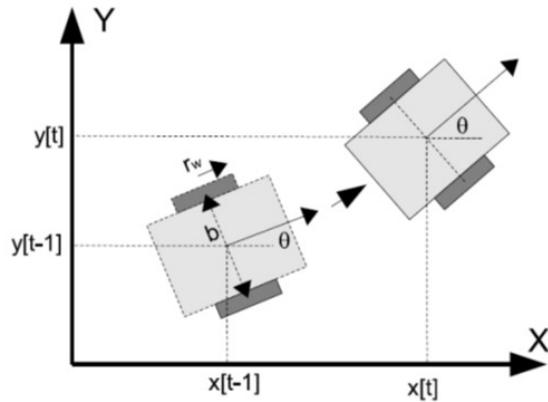


Figure 3.17: Differential Drive Robot Reference Frames [12]

3.3.6.0.1 Kinematic Model Finally, to represent the robot in the world frame, a kinematic model must be of the robot. The process involves looking at each of the wheels and determining their kinematics constraints and forming a rotational and speed matrix. These different matrices can be shown in Figure 3.18. Mappings are made between frames to evaluate the speed and rotation of the robot.

$$\begin{aligned} \dot{\zeta}_R &= R(\theta) \dot{\zeta}_I = R(\theta) \cdot [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T \\ R(\theta) &= \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \dot{\zeta} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\dot{\phi}_1, \dots, \dot{\phi}_n, \beta_1, \dots, \beta_m, \dot{\beta}_1, \dots, \dot{\beta}_m) \end{aligned}$$

(a) Rotational Matrix [12]

(b) Speed Matrix

Figure 3.18: Kinematic Matrices [12]

In Figure 3.18a, it shows a rotational matrix $R(\theta)$, which arguments are the θ which represent the robots turn angle. This is used to evaluate the rotational matrix, which can then be applied to the speed matrix to determine the robots speed in a particular direction. As well as this, the robots rotational matrix can be applied to the robots position matrix, which will evaluate the robots rotational speed $\dot{\zeta}_R$. The next matrix shown in Figure 3.18b, evaluates the robots speed $\dot{\zeta}$, which is a function of the wheel speeds $\dot{\phi}_i$, steering speeds $\dot{\beta}_i$, steering angles β_i and the geometry of the robot($\beta_m, \dot{\beta}_m$).

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

AMR: Fig 3.2

$$\dot{\xi}_R = R\left(\frac{\pi}{2}\right)\dot{\xi}_I = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ -\dot{x} \\ \dot{\theta} \end{bmatrix}$$

Figure 3.19: Robot Kinematics Example [12]

3.3.6.0.2 Odometry Odometry involves creating kinematic models for each of the wheels by looking at how they turn and how many degrees of freedom they have and evaluates a model for this. In the case of differential, the wheels are known as fixed standard wheels.

3.3.7 localisation

Localisation involves knowing where the robot is in its environment. For true localisation, the values from encoders can not be used due to the fact that the encoders produce accumulative errors and thus are not reliable. This is where fusion and localisation come in. Fusion involves using all the sensors available to determine an environment map via an array of algorithms. This effect is shown in Figure 3.20a. Localisation involves using the map and re-evaluating the position of the robot this is graphically shown in figure 3.20b. [49]

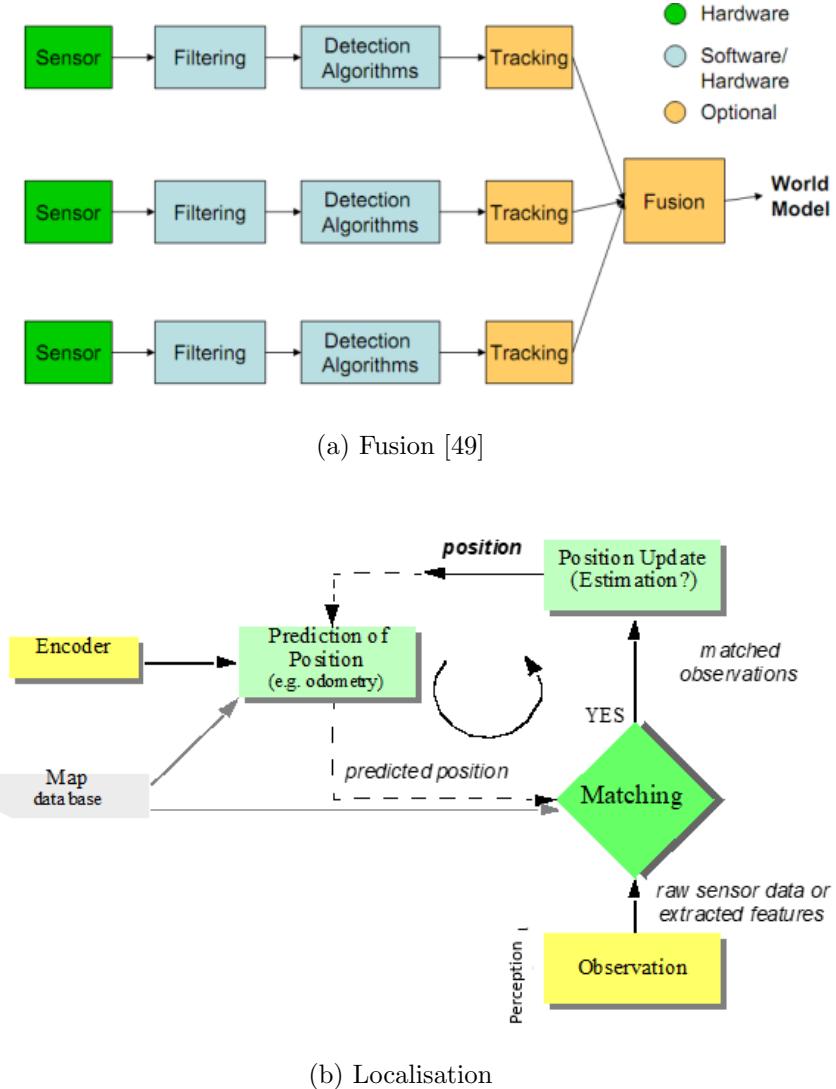


Figure 3.20: Robot Localisation [13]

3.3.8 Communication Networks

Robots contain several areas of functionality and thus require some form of communication medium between them all. Within this section, the different forms of communications are discussed both high level and low level to give an overview of the different communications used in robotics and embedded systems.

3.3.8.1 High Level Communications

For high level communications a multiple of different communication protocols are used. Each one focuses on transferring data prioritising different characteristics such as speed and reliability.

3.3.8.1.1 TCP: Transmission Control Protocol In terms of internet communication Transmission Control Protocol is one of the fundamental protocols. It provides a reliable, connection orientated means of communication. The benefits of TCP is that the message always reaches the receiving end due to the way it functions. When TCP sends the data, the data is split into segments and sent down the communication medium and reassembled in the receiver's buffer. Each segment is received, and an acknowledgement is sent back to the sender. Although this method at first glance seems effective, packet loss is a severe detriment to this design as it slows the whole system down due to timeouts along with retransmission times.

3.3.8.1.2 UDP: User Datagram Protocol Unlike the Transmission Control Protocol(TCP), User Datagram Protocol(UDP) is not the most reliable but is designed for high throughput, ideally streaming services where high throughput but no accuracy is required. This means that the messages are not always consistent and thus can lose information over the communication line. [50] The benefit for this method is that it allows for high-frequency transmissions if one message is lost but a high amount got through then that particular application might still operate within acceptable limits.

3.3.8.1.3 HTTP: Hyper Text Transfer Protocol The final type of protocol to be discussed is the Hyper Text Transfer Protocol(HTTP), it's an application protocol used for communication for websites. It defines a set of standards allowing the exchange of data between webpages. [51]

3.3.8.2 Low Level Communications

As for lower level communications, these are concerned with the hardware level, specifically looking at how circuit boards communicate with each other. The three methods

that are discussed is Inter-Integrated Circuit, Serial Peripheral Interface and finally Universal Asynchronous Receiver Transmitter

3.3.8.2.1 Inter-Integrated-Circuit Inter-Integrated-Circuit also known as I2C is a multi master slave network specifically designed for embedded systems. It was designed with the purpose in mind of allowing multiple slave devices to communicate all through a two-wire bus. It's designed for short-distance applications and requires a common ground between all components. It requires a clock line and a data line, the clock line is controlled by the master, and the data line is either written to by the slave or the master, depending on what the master wanted. An example I2C network is shown in Figure 3.21. [14]

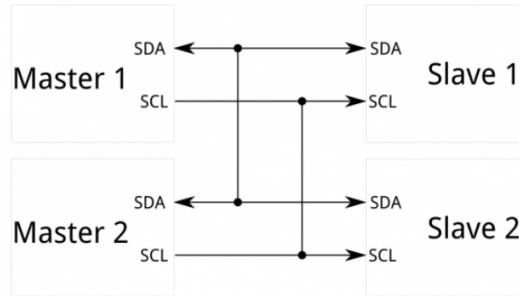


Figure 3.21: Inter-Integrated Circuit Example Network [14]

The benefits of I2C is that devices do not have to agree on a clock speed and thus reduced coupling between individual circuit boards. Unlike Serial Peripheral Interface and Universal Asynchronous Receiver Transmitter, there is not a restrictive amount of devices that need can be connected to the network. The network works with an addressing scheme that can be either 10 or 7 bit allowing 1024 or 128 devices to be connected at once, provided the devices don't share the same address. The downfalls with I2C is that there can be addressing issues in which devices share the same address and thus cant work, the speed of the network can be quite slow compared to Serial Peripheral Interface(SPI) and Universal Asynchronous Receiver Transmitter(UART). The maximum speed that can be obtained with I2C reliably is 400000bit/s, whereas UART and SPI can get speeds well over this. [14]

3.3.8.2.2 Serial-Peripheral-Interface Another Inter Board Communication network is Serial Peripheral Interface(SPI), unlike I2C with a multi-master network and addresses. SPI consists of a single master and multiple slave devices. An example network diagram of this type of communication is shown in Figure 3.22. [15]

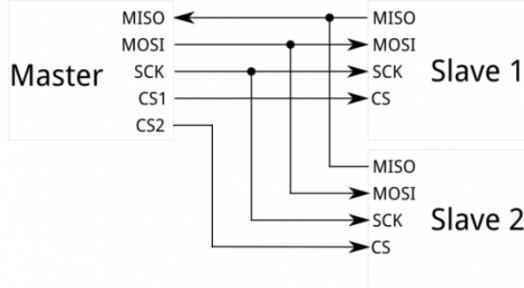


Figure 3.22: Serial Peripheral Interface Example Diagram [15]

SPI networks use a synchronous method in which the clock between the master and the slaves is controlled by the master, and the transmission is done through the SCK, MOSI and MISO lines. The bus operates using slave select pins to pick which slave is allowed to be in communication with the master. Using the MISO(Master In Slave Out) and MOSI(Master Out Slave In) wires communication is allowed to happen between the selected slave and the master. The advantage of this solution is that there is high throughput and that it is full-duplex, the disadvantage, however, is that there are many pins required for this solution. [15]

3.3.8.2.3 Universal Asynchronous Receiver Transmitter The final communications network to be discussed is the Universal Asynchronous Receiver Transmitter(UART), unlike the previously discussed networks, UART can only have two devices connected and is full-duplex. It has a very high throughput and is usually implemented entirely in hardware. The diagram for this network is shown in Figure 3.23. It's an asynchronous solution both boards need to agree on a set baud rate, there is no need for a clock line, and only two data lines are used for transmission, and common ground is needed. [16]

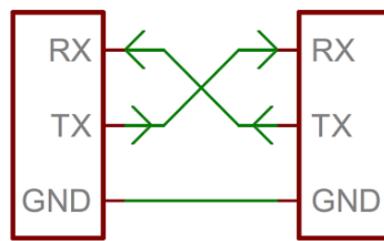


Figure 3.23: Universal Asynchronous Receiver Transmitter [16]

3.3.9 Design Patterns

Within Object Orientated programming there is a set of predefined structures known as design patterns, which provide the most effective way of implementing certain aspects of functionality and creating a design that is both modular and long-lasting. In Summary a design pattern is defined as: [52]

”a general repeatable solution to a commonly occurring problem in software design. A design pattern isn’t a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations.”

3.3.9.1 Creational design Patterns

Creational patterns are used for object creation along with class structures and creation of new classes. The difference between the two different types is, object creation patterns use delegation, and class creation uses inheritance in their structures. [52]

3.3.9.1.1 Factory Method The Factory Method is concerned with object creation. It defines an interface for object creation and lets subclasses decide on implementation. It also separates object creation from the class that contains the objects. Allowing full decoupling of the object creation. This design approach is shown in the class diagram in Figure 3.24. [17]

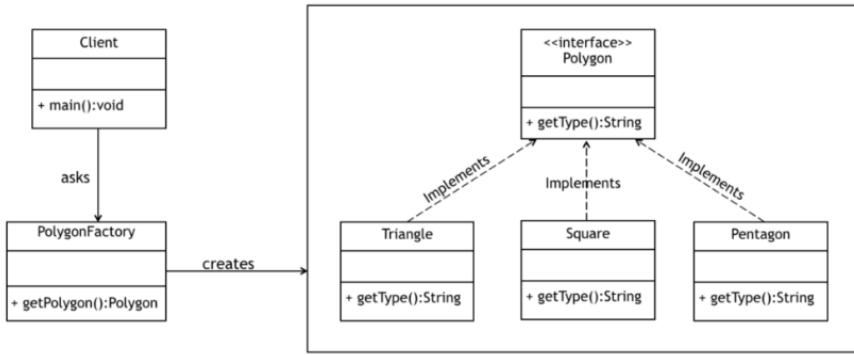


Figure 3.24: Factory Method Class Diagram [17]

3.3.9.2 Behavioural Design Patterns

Patterns focusing at communication between classes are known as behavioural patterns. They are designed with multiple intentions in mind, the main ones being class decoupling and removing concrete functionality from classes that are likely to change. [52]

3.3.9.2.1 Observer The Observer design pattern is one of the most effective patterns ever conceived. The problem it tries to solve is with object relations. It defines an object which is the keeper of all the data and logic. It has many connections to other classes which use the information from the subject periodically. The pattern reduces coupling by making the subject not aware of all objects watching the subject. The watching objects are grouped into a single interface known as an observer, and the model notifies the observers whenever data is changed. Each observer implements an update method and takes from the model what they need to update their respective functionality. The basic class diagram for this pattern is shown in Figure 3.25. [18]

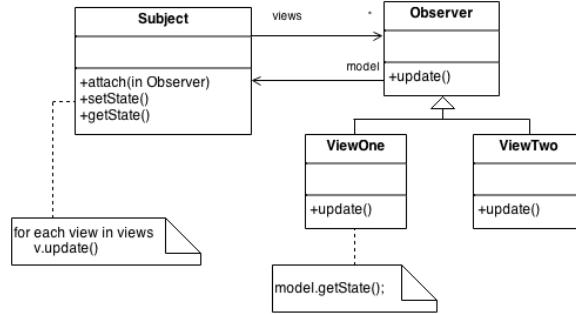


Figure 3.25: Observer Generic Class Diagram [18]

3.3.9.3 Structural Design Patterns

Patterns that are concerned with how classes and objects are composed are known as Structural Design Patterns. These types of design patterns aim to simplify the different structures by identifying relationships between the different elements in a structure.

3.3.9.3.1 Adapter The Adapter pattern is used for converting one interface of a class into another interface that is used in the design. The Adapter pattern is also known as a wrapper. An example of the adapter pattern is shown in Figure 3.26. In example one, the rectangle acts as the adapter for the LegacyRectangle Class. In example two, the wrapper/adapter structure is shown in a more generic sense. In which the Wrapper class is acting as the adapter and the LegacyComponent is the adaptee. The method that is being adapted is the `doThat()` method. [19]

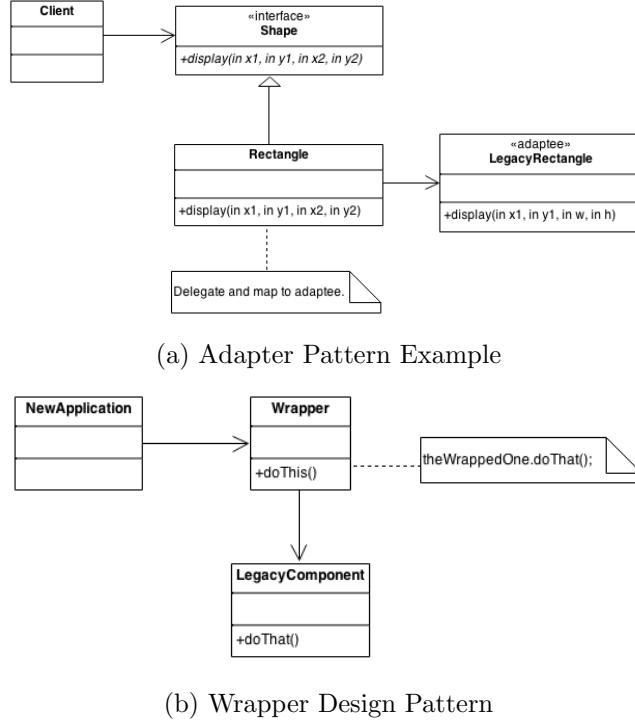


Figure 3.26: Adapter Design Pattern [19]

3.3.9.3.2 Private Class Data The Private Class Data Design Pattern aims to solve the encapsulation problem and control write access to objects. The pattern revolves around class attributes and when manipulation is no longer required. The structure of this pattern involves creating a data class of the target class and moving fields that are unlikely to change in the data class rather than in the target class. This type of patterns structure is shown in Figure 3.27. [20]

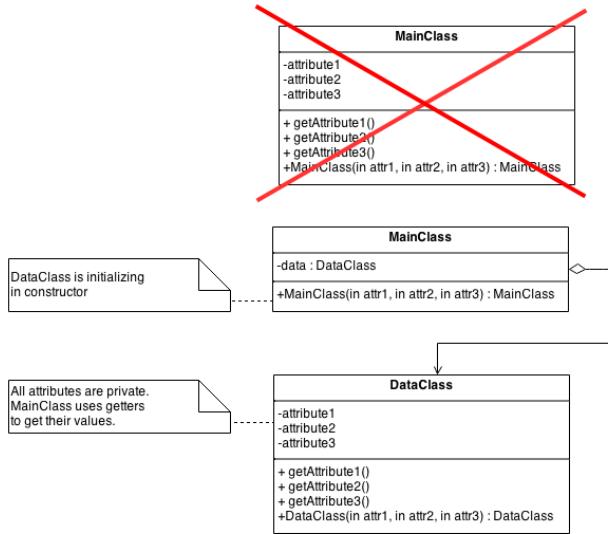


Figure 3.27: Private Class Data Design Pattern [20]

3.3.9.4 Architectural Design Patterns

Architectural design patterns aim to function as a blue print for the software system. They aim to address various issues with software development, such as scalability and modularity. For User Interface applications, the two main patterns used are the Model View Controller and Model View Presenter Patterns. They aim to solve the same issues; however, their structures differ slightly. The designs are listed below:

3.3.9.4.1 Model View Controller The Model View Controller pattern relies heavily on the observer pattern discussed earlier to implement model view interactions. The idea behind the MVC pattern is that each section has a particular purpose, some of the code holds the functionality of the application(model), the view part is responsible for rendering the information and displaying it to the user, and finally, the controller is responsible for controlling the model. A use case diagram of the MVC pattern is shown in Figure 3.28. [53] [21]

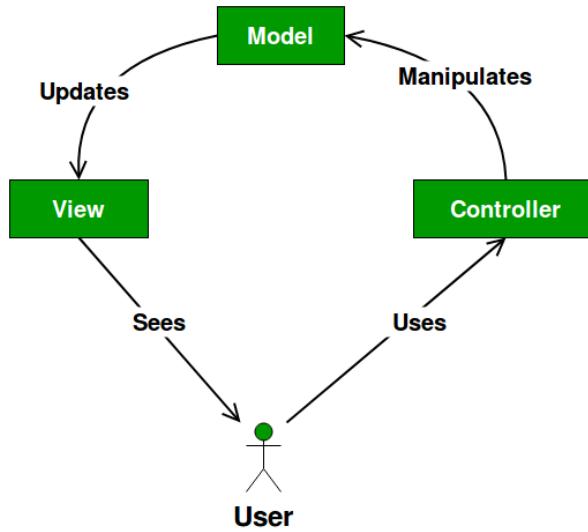


Figure 3.28: Model View Controller Use Case Diagram [21]

3.3.9.4.2 Model View Presenter A very similar pattern to the MVC pattern is the Model View Presenter(MVP), instead of isolating the view controller aspect of the design and having the view directly referencing the model. This interaction is removed, and all communication goes through the presenter. An overview of this design pattern is shown in Figure 3.29. [22]

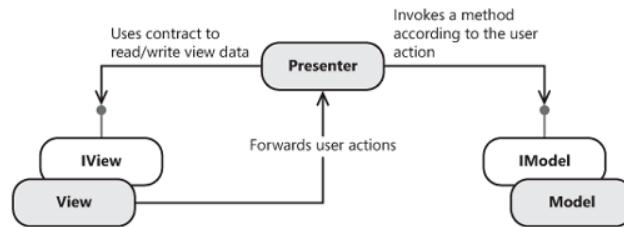


Figure 3.29: Model View Presenter Design Pattern [22]

The interactions in this particular pattern shows the view is completely decoupled from the model. All interactions with the model are done through the presenter.

Chapter 4

Mechanical Design and Implementation

The project contains many mechanical aspects; these are more focused on mounting the different systems rather than introducing entire mechanical systems into the robot.

4.1 Requirements

The requirements for the mechanical aspect of the robot is to fit all the different Printed Circuit Board components and new sensors into the robot while keeping it reasonably easy to build and easy to dismantle to an extent.

4.2 Design Process

To correctly tackle this problem, a Computer-Aided Design(CAD) package was required as this allows the creation of different models in a simulation. For this task, Autodesk Inventor 2020 [54] was chosen as this particular program was easy to use and was already familiar with the developer. It is also used widely across schools, which is a significant bonus for the Rampaging Chariots Guild. As parts could be distributed in the schools instead of on relying on the guild to produce the parts and send them with kits. [55]

4.3 Implementation

In order to turn the default Rampaging Chariot robot into an Autonomous Version, many different types of mechanical designs had to be created. The process was split into multiple systems and are listed below:

4.3.1 Battery Holder

The first design that had to be created was the Battery Holder; its purpose was not only to store batteries but also to provide cable management and Primary Power Distribution Board Mounts. For information regarding Power Distribution Boards, see chapter 6. The renders for this particular model is shown in H.2. The full component list is shown in table G.7. The design incorporates six channels at the side of the holders allowing the cables to feed through. Below the PCB mounts are two more cable channels which allow direct feed into the cable dividers. The implemented model can be seen in Figure I.1.

4.3.2 Camera Module

The Camera module assembly consists of five parts(seven including servo motors). The renders associated with these modules are shown in H.3. The design has two mounting holes for the top panel to secure the module and consists of bolts to mount the servo motors. The top half of the camera assembly is held by a pressure fit to the rest of the system. The full component list is shown in table G.8. The implemented design is also shown in Figure I.2.

4.3.3 Control Panel

The control panel is shown in H.4 it consists of one 3D printed part, five buttons, one switch and Keyestudio LCD Screen [56]. It has three mounting holes for the panel and a clip on the front to hook it on to the top panel. The component list for this assembly is shown in table G.9. The implemented assembly is also shown in Figure I.3.

4.3.4 Encoder Mounts

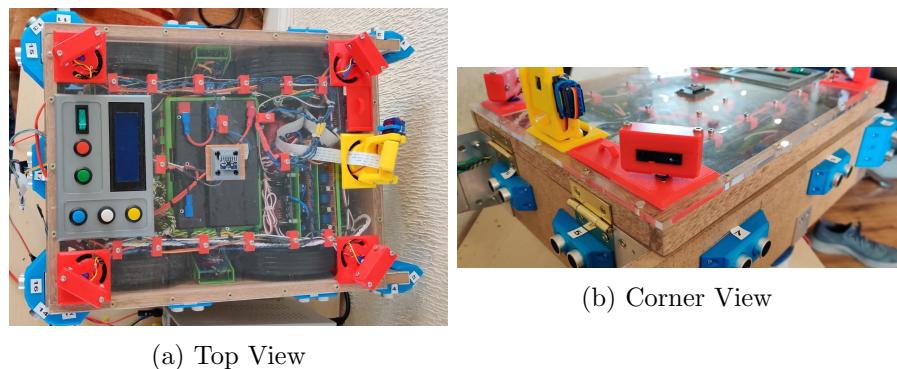
AS5600 Encoder PCB's require two different mounting mechanisms, one for the drive wheels and one for the flywheels. The different assemblies are shown in H.5 the flywheel assemblies work by mounting a cap with the magnet on to the wheel over a bolt and this is then measured by the AS5600 PCB using a specialised mount. The complete component list is shown in table G.10.

4.3.5 Inertial Measurement Unit Mount

The inertial measurement unit system required a specific holder in order to ensure the placement of the MPU6050 module [44] in the middle of the robot. The renders for this module is shown in H.7. The component list for this assembly is shown in table G.12. The module can be seen implemented on the robot in the picture shown in Figure I.4.

4.3.6 Infrared Sensor Mounts

In order to mount the SHARP IR Sensor(GPY2Y0A21YK0F) [47], an entire assembly had to be constructed to allow the sensors to allow movement in the yaw axis. This was achieved by using a SG90 servo [33]. The two assemblies are shown in H.8. The mounts that don't contain the Rasberry Pi [57] consist of four mounting holes allowing the assembly to be secured in the corner. The full component list for the assemblies is shown in table G.13. The assemblies are designed to reduce dead zone areas by allowing the sensors to rotate 180 degrees. The overall coverage of the infrared sensors alone are shown in Figure 4.2. Along with the coverage renders, the implemented infrared sensors are shown in Figure 4.1. The rotation range of each sensor overlap but provide complete coverage for the robots perception system. The overlapping areas can be negated by moving each sensor differently so that they do not create destructive interference.



(a) Top View

(b) Corner View

Figure 4.1: SHARP IR Sensors Pictures

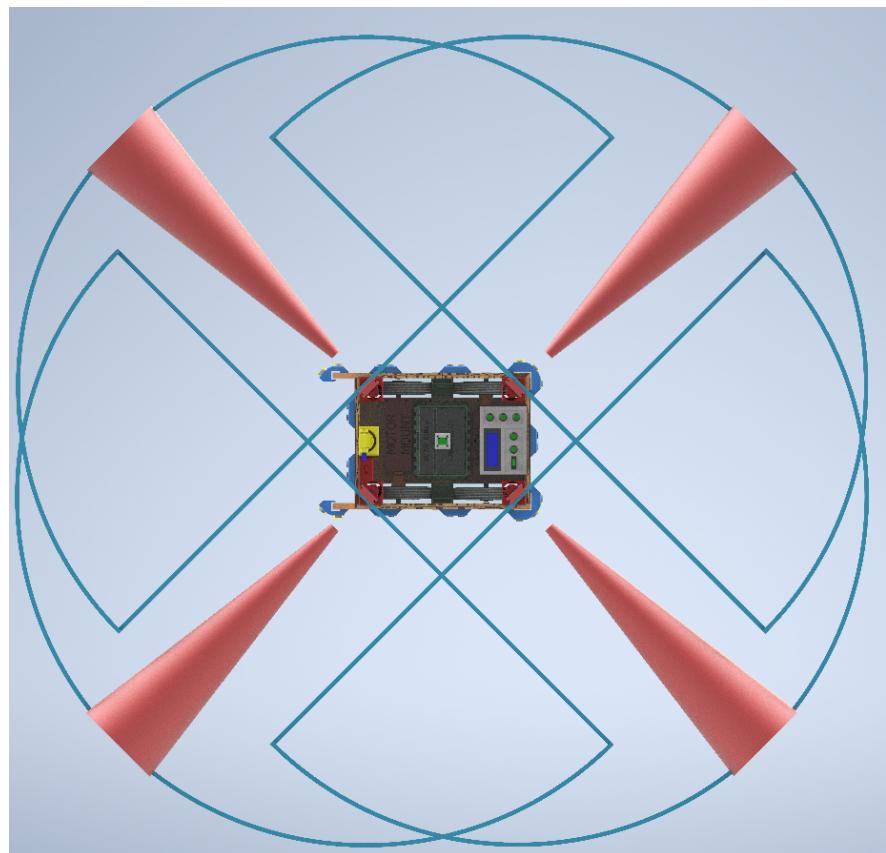


Figure 4.2: Infrared Sensor Coverage

4.3.7 Ultra Sonic Sensor Mounts

The ultrasonic sensors were mounted with the purpose of increasing the overall perception of the robot's environment; a total of seventeen sensors were mounted with the aim

Chapter 4. Mechanical Design and Implementation

of increasing total perception at any one time. Due to this, many different mounts had to be designed to fit in different places in the robot. The different models are shown in appendix H.9. Due to the different models, a different set of components are required, the complete list of components is shown in table G.14. The field of view of this particular type of sensors is shown in Figure 4.3. Due to the overlapping areas, a system was devised for dealing with destructive interference and pulsing each of the sensors. For more information, refer to chapter 6. The different sensors are also highlighted in the pictures shown in Figure 4.4.

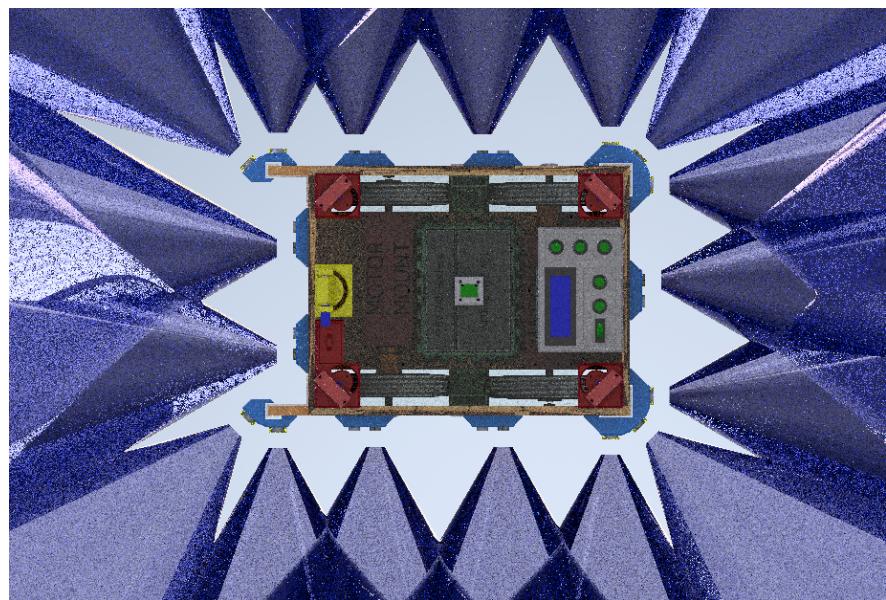


Figure 4.3: Ultrasonic Sensor Coverage

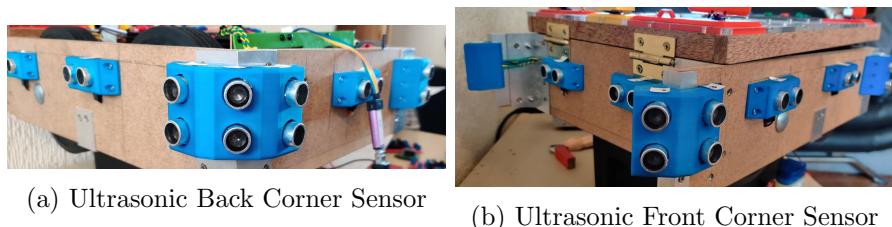


Figure 4.4: Ultrasonic Sensor Mounts Pictures

4.3.8 PCB and Veroboard Stack Panels

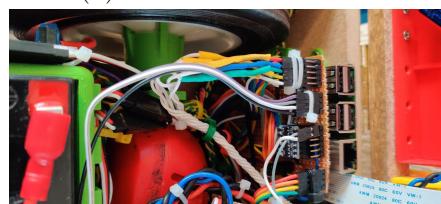
In terms of electronics, there are many different PCBs which are required for the robot to function; this causes issues with placement as there is limited space inside the robot. To solve this problem, a stacking system was developed. There are three stacks which contain multiple PCBs. The different stacks can be seen in Figures 4.6, 4.7, 4.5. Due to testing and time restraints, the PCB stacks were not installed inside the robot. The two main stacks use the motor controller boards as a basis which in turn hold the entire stack up. The last stack is for the Raspberry Pi, which is a single stack that consists of the Raspberry Pi and the Raspberry Pi HAT. For more details about the electronic design, refer to chapter 6.



(a) PCB Stack Top View



(b) PCB Stack Side View



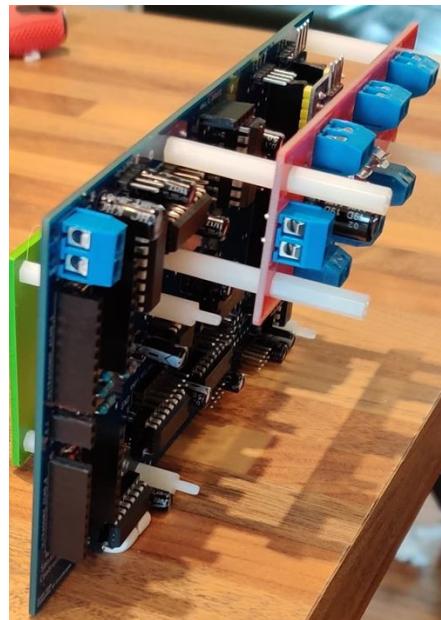
(c) Veroboard Stack Top View

Figure 4.5: Autonomous Robot Raspberry Pi Stack

Chapter 4. Mechanical Design and Implementation



(a) PCB Stack Top View



(b) PCB Stack Side View



(c) Veroboard Stack Top View



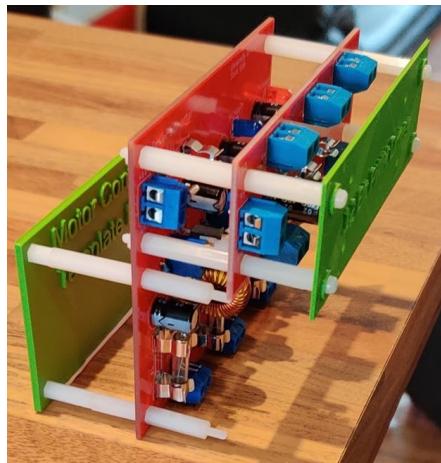
(d) Veroboard Stack Side View

Figure 4.6: Autonomous Robot Sensor Controller Stack

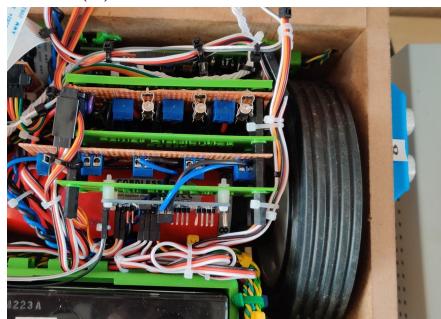
Chapter 4. Mechanical Design and Implementation



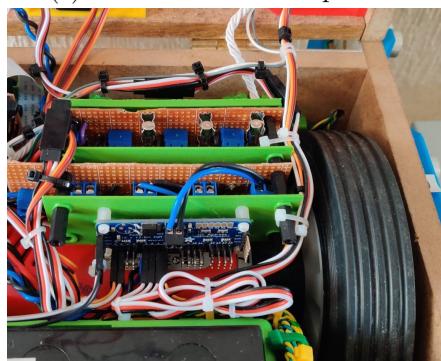
(a) PCB Stack Top View



(b) PCB Stack Side View



(c) Veroboard Stack Top View



(d) Veroboard Stack Side View

Figure 4.7: Autonomous Robot Front Stack

4.3.9 Cable Management and Dividers

Due to the number of cables in the system, some form of cable management is required. For this task, two cable dividers were designed to feed the sensors power lines directly into the Primary Power Distribution boards mounted on the Battery Holder. These designs are shown in Figure H.28. The entire component list for cable management is shown in table G.15. The dividers inside the robot can be seen in the implemented version shown in Figure I.5.

4.3.10 Top Panel Assembly

The last mechanical part of the project was the development of the top panel of the robot which houses the Infrared sensors, Control Panel and the Camera Module discussed earlier. This part could not be manufactured by 3D printing and was laser cut using the University Design Manufacture and Engineering Management Laser Cutter using the settings shown in table K.3. The entire component list for this assembly is shown in tableG.11. The renders for this assembly are shown in appendix H.6.

4.4 Full System Assembly

Using all the different assemblies discussed earlier, the total robot was assembled and is shown in Inventor [54] in appendix H.12. Using a variety of different 3D printers(Wanhao I3 V2 [58] and Prusa I3 MK3s [59] the different assemblies were printed and assembled. The full assembly is shown in Figure 4.8.

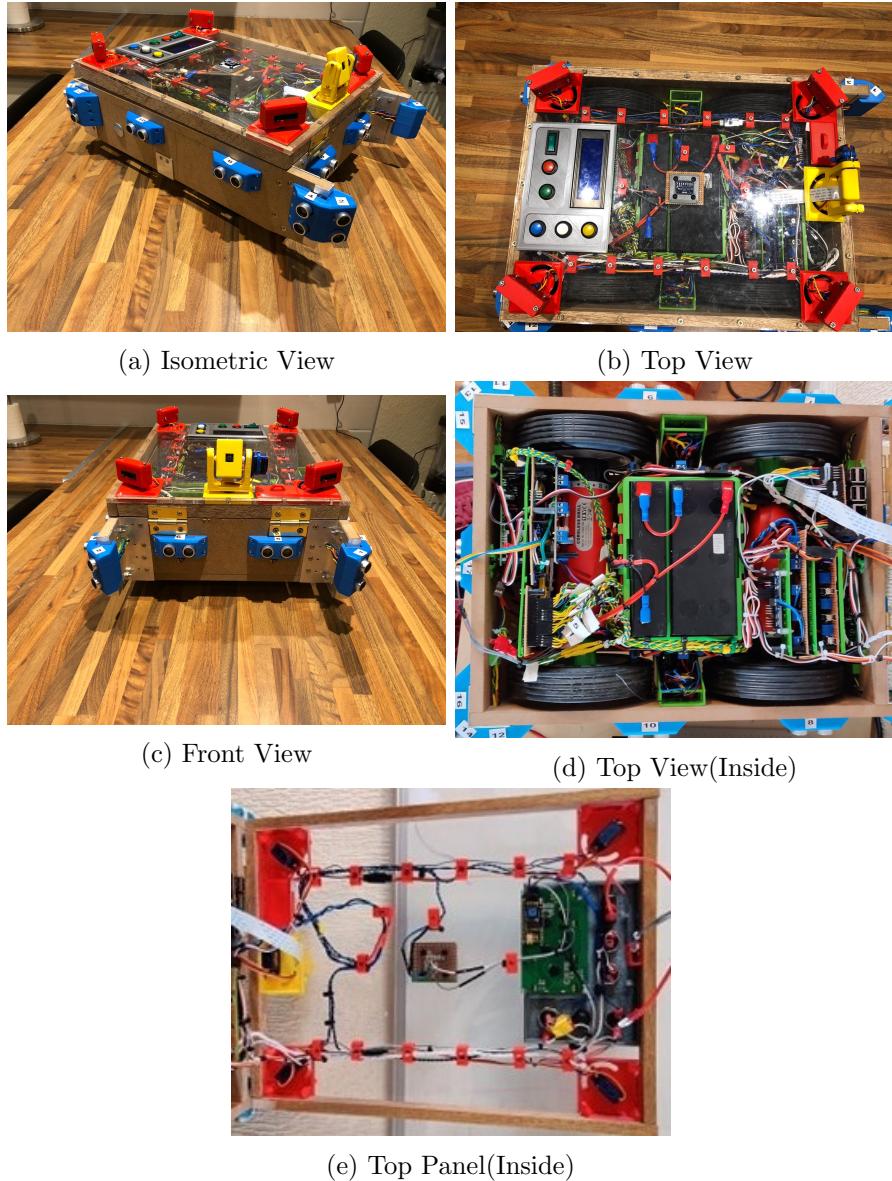


Figure 4.8: Autonomous Robot Full Assembly

Chapter 5

Software Design and Implementation

5.1 System Overview

In terms of Software, there are two applications running, both responsible for different aspects of the system. The Control Interface was code in Java and uses a variety of communication protocols to communicate with the Raspberry Pi on-board the Robot.

5.2 High Level Communications

The communications used consisted of Transmission Control Protocols(TCP); this was split into four communication sockets. One was used for controlling the camera, the other for controlling the movement and navigation of the robot. These were explicitly one-way sockets to give the maximum response time in the system. Each socket runs on its own thread, allowing immediate response time. Along with the control sockets, there were sensor and vision sockets. These are responsible for relaying all the information from the robot back to the control interface. The high-level communications architecture is shown in Figure 5.1. From this diagram we can see that the sockets with the highest throughput is the sensory data and camera data.

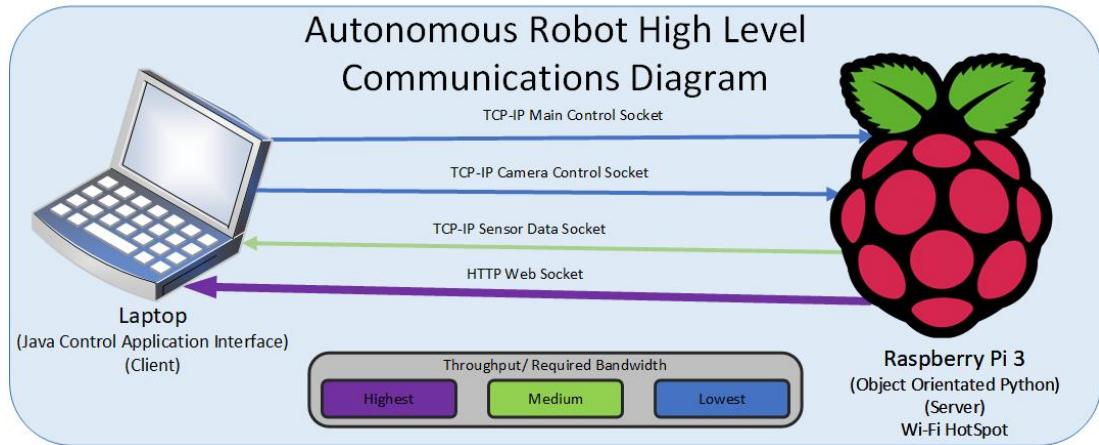


Figure 5.1: High Level Communications Diagram

5.3 Control Interface

The control interface is responsible for sending commands to the robot over the different sockets discussed in section 5.2. The language chosen for this application was Java [28], the reason for this is that it has much familiarity with the developer as well as it is a good starting language for beginners. [60]

5.3.1 Requirements

The requirements for the interface is that it should be able to control the robot much like the radio-controlled version as such, it should have some manual control functionality. It should also be able to access the logs on the robot and display sensory information coming from the robot.

5.3.1.1 JavaFX

The modern way of creating Graphical User Interfaces(GUI) in Java is to use JavaFX [61]; this allows the view to be completely separated from the model and also allows the View part of the application to be designed with a markup language. It makes for far cleaner code and more aesthetically pleasing GUIs. [62]

5.3.2 System Structure

In order to allow for code modularity and re-use, the system employed decoupling methods, using interfaces when possible and abiding to design patterns to create a maintainable design. This involved using a famous and well renowned architectural the Model View Presenter(MVP) Pattern, for more information, refer to chapter 3.

5.3.3 Model Structure

Inside the model, there is a hierarchy of packages each dedicated to its own functionality. There is a direct mirror to this structure and the structure of the Robot Controller. The high-level structure of the model is shown in Figure 5.2.

The structure by having an AutoChariot Object which is essentially a data and control handler which is then observed using the Observer pattern by both the packagers and the presenters in the view. The model is fully decoupled from communications any changes in the client-side model are notified to the corresponding Communications class and then sent over to the robot through the appropriate socket.

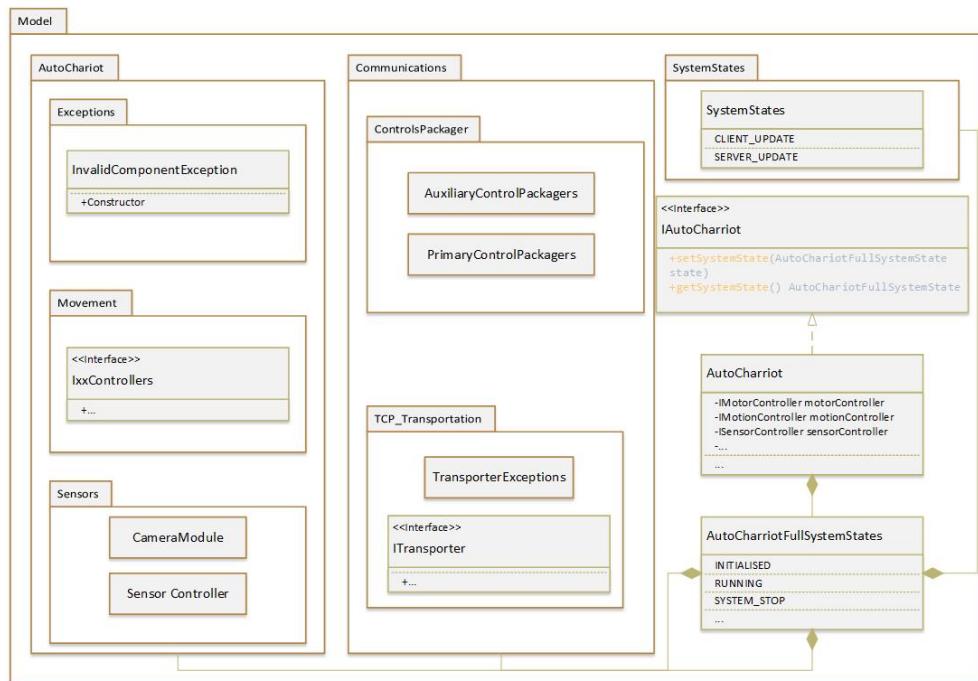


Figure 5.2: Control Interface Model Package Structure

5.3.3.1 AutoChariot

Within the AutoChariot package there is many sub packages all responsible for particular aspects of the design. The critical aspect of the design is to decouple as many components as possible; the reason for this is to increase designs flexibility. The Structure of the design is represented in the class diagram shown in Figure 5.3.

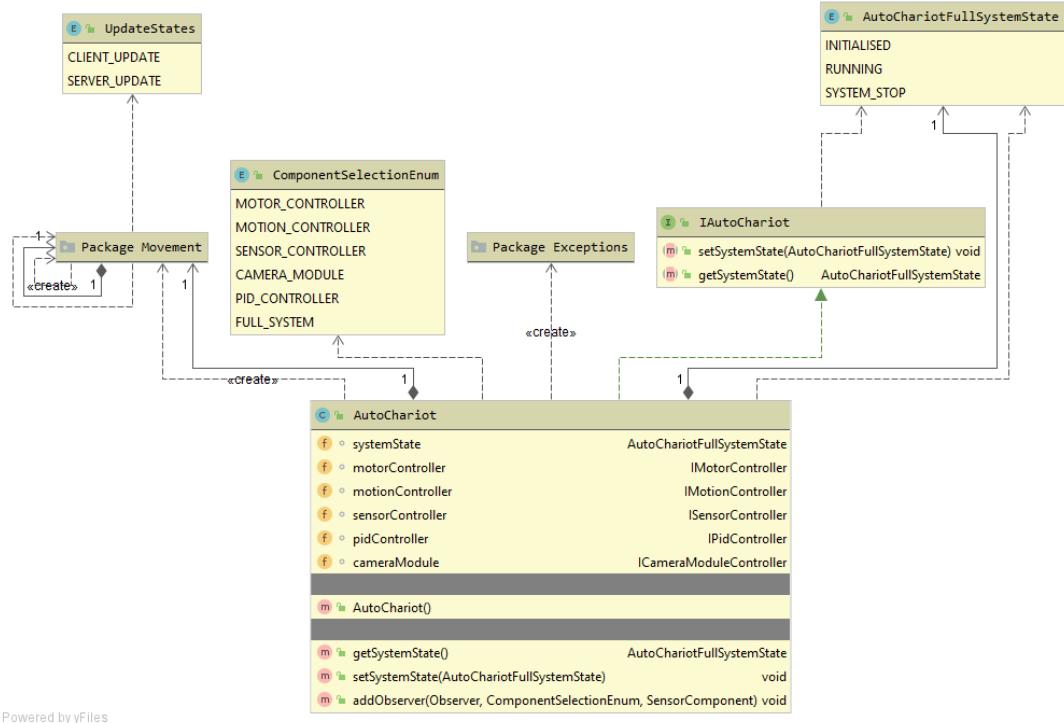


Figure 5.3: AutoChariot Package Class Diagram

The AutoChariot class is an Observable object and holds all components that are required for the robot. Each sub-component is selected and placed using a ComponentSelectionEnum. This enum specifies what specific component it is. The AutoChariot is also decoupled from the rest of the program via the IAutoChariot Interface.

5.3.3.1.1 Movement The first and most crucial package to be discussed is the Movement package, this covers the high-level functionality and the class diagram for this side of the application is shown in figure 5.4.

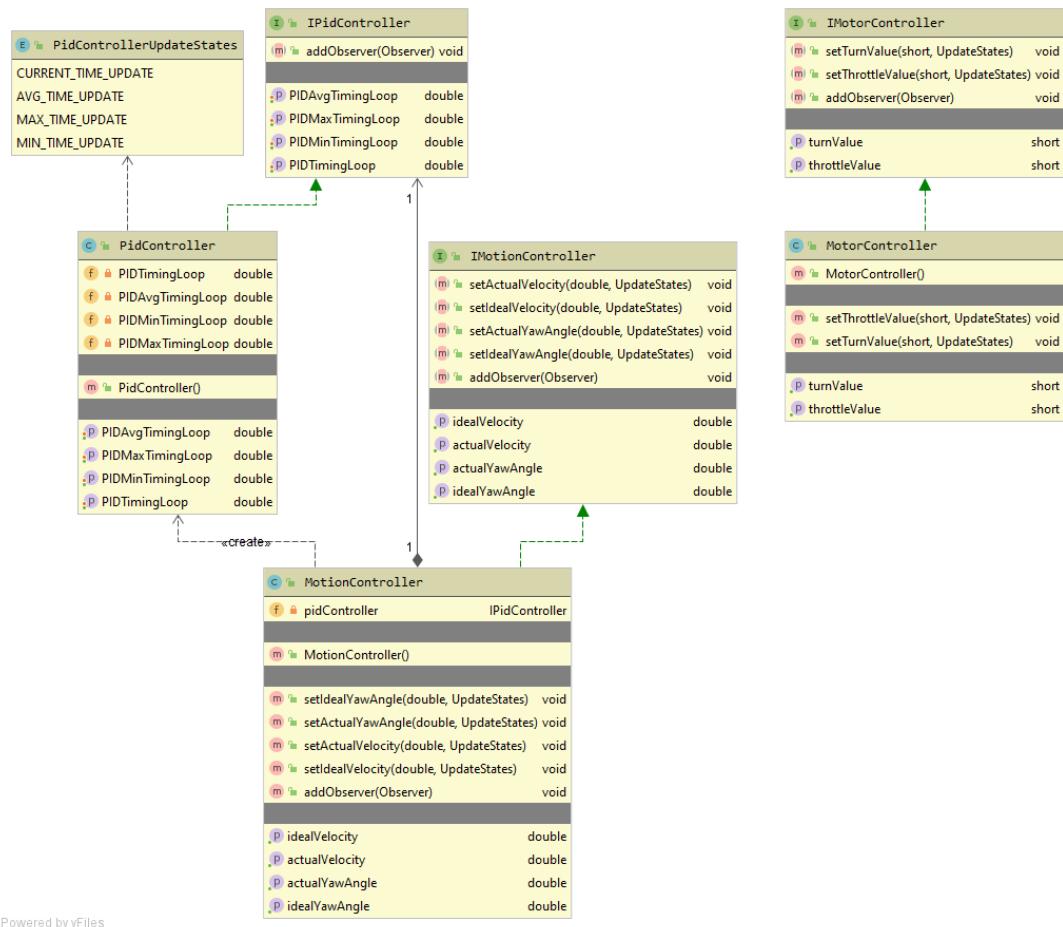


Figure 5.4: Movement Package Class Diagram

The Movement classes are strictly designed for holding the current data of the robot and if changed on the client-side, are kept in sync with the server-side.

5.3.3.1.2 Sensors As for the sensors, the application incorporates a SensorController and holds different types of sensors in Maps and ArrayLists. The system is designed such that each of these are updated via the SensoryDataPackager. The incorporated design architecture for this is shown below in Figure 5.5.

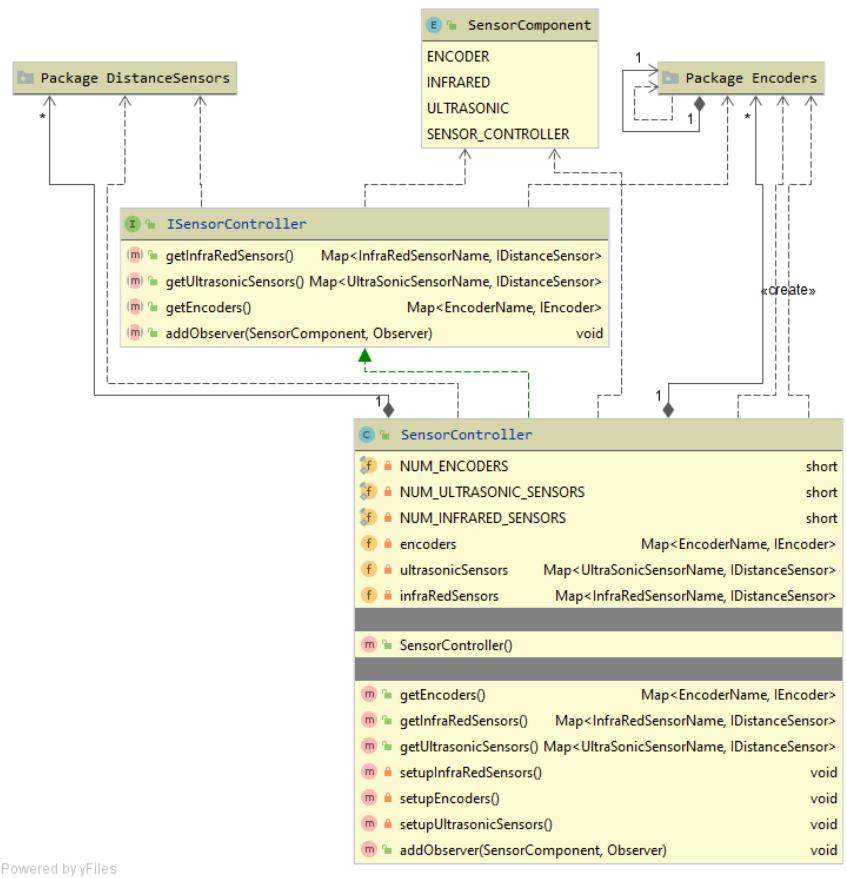


Figure 5.5: Sensor Controller Class Diagram

5.3.3.1.3 CameraModule Finally, the last aspect to be covered in the Java Control Interface model is the CameraModule model. The design consists of a simple data class holding the current angles of the CameraModule, again using the observer structure along with the communications package to update the relevant fields-both client and server-side. The class diagram structure for this is shown in Figure 5.6.

Chapter 5. Software Design and Implementation

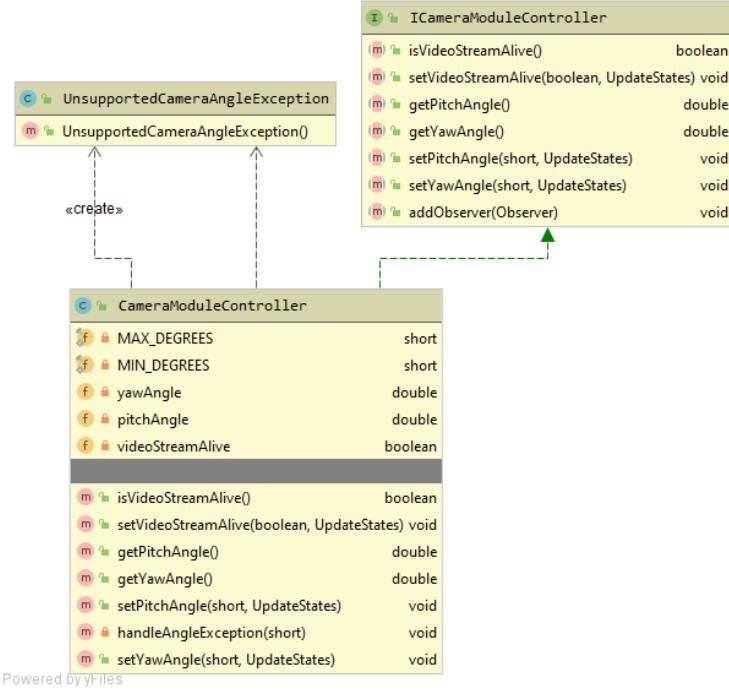


Figure 5.6: Camera Module Class Diagram

In order to show how the model communicates with the robot, a class diagram showing the struture of the packager architecture mixed with the CameraModule is shown in Figure 5.7.

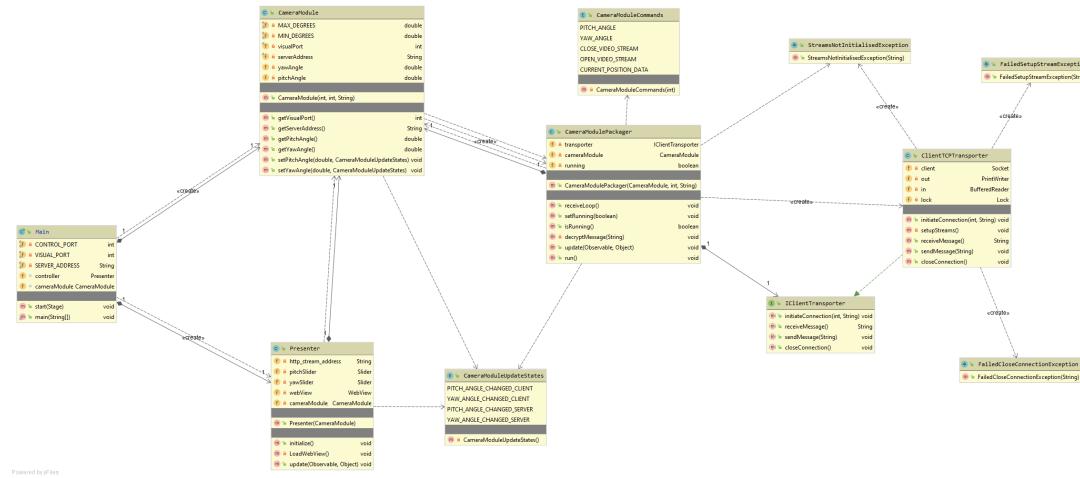


Figure 5.7: Camera Module Java Class Diagram

5.3.4 Observer Pattern Implementation

The Java control interface uses the observer pattern extensively in the design in order to update the relevant view elements and to limit unnecessary control messages between the robot and the control interface. This design is shown in Figure 5.8.

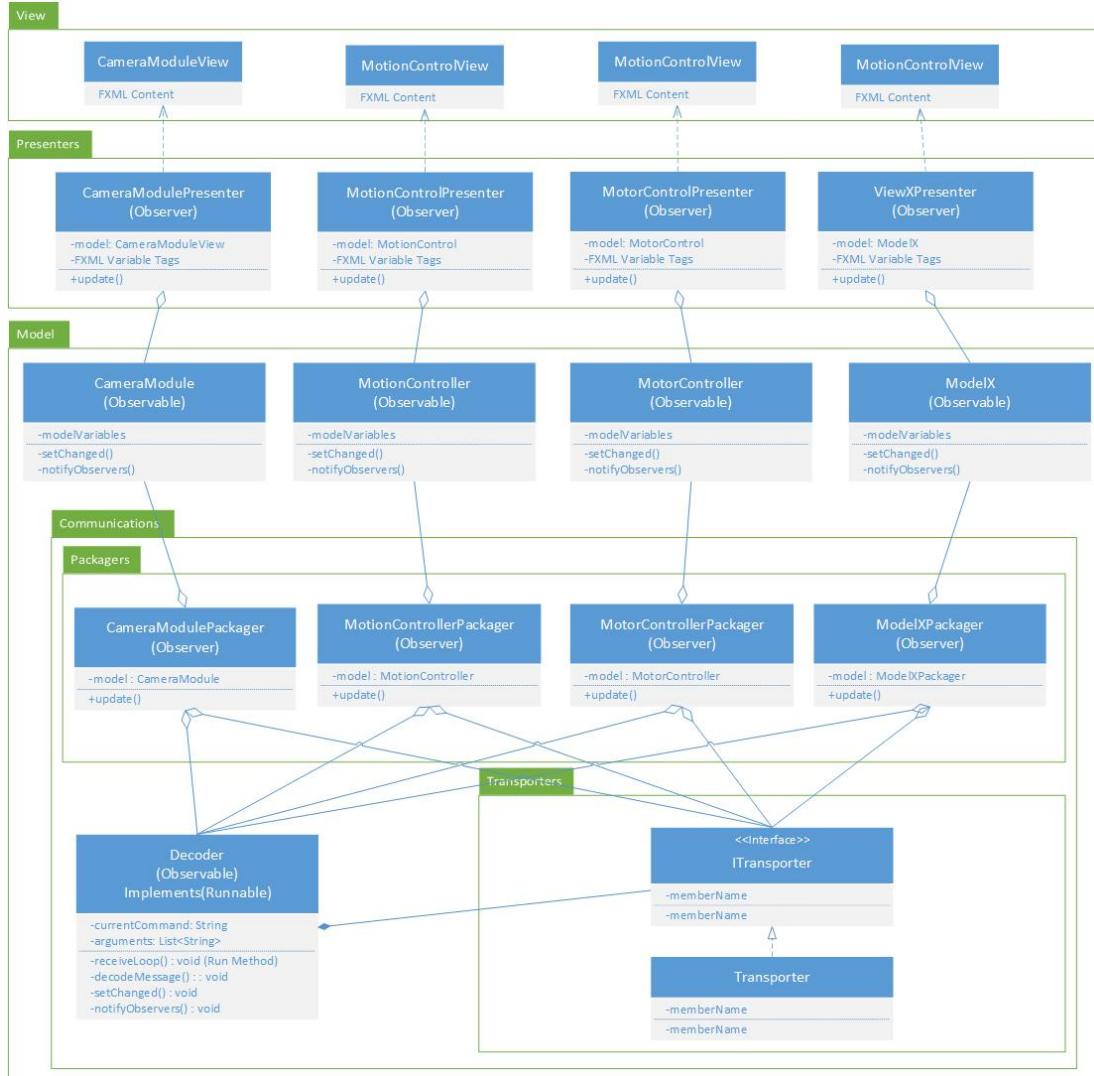


Figure 5.8: Java Interface Observer Pattern High Level Implementation

5.3.4.1 Communications

One of the most crucial systems within the Java Control Interface is the communications package. Its entire purpose is to handle the transmission and receival of data. How

Chapter 5. Software Design and Implementation

it does that is through a Transporter-Packager system. The class Diagram for this approach is shown in Figure 5.9.

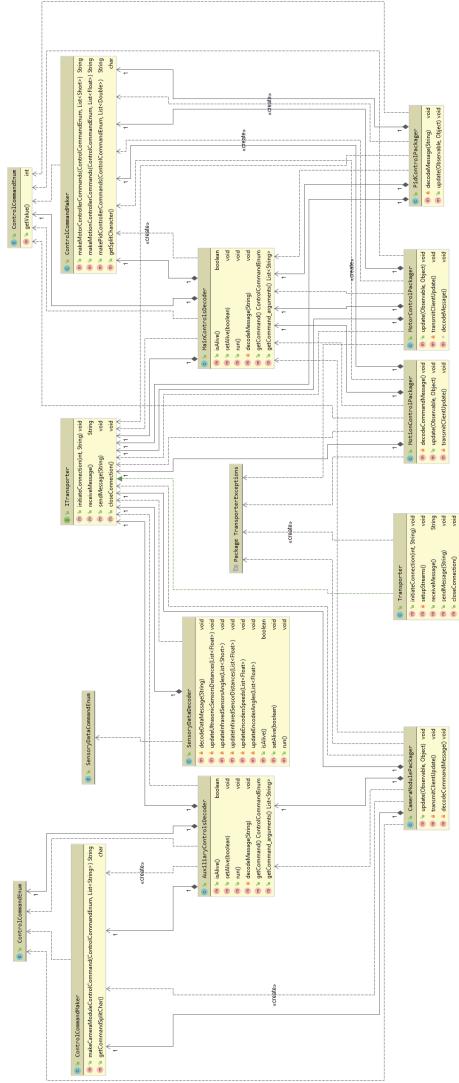


Figure 5.9: Java Control Interface Communication Package Class Diagram

The system splits the different communications described earlier, into different classes. The communication works by having a transporter class which is entirely responsible for retrieving and sending data. The decoder class then uses the transporter to decode any messages that come through and update the observer. The observer is the relevant packager which in turn updates the pertinent sections of the view. The packager also observes the model, and whenever an event has happened in the model,

the packager notices this and uses the decoder class to send the message through the transporter object. This design allows full decoupling from the model and the low-level communications.

5.3.4.1.1 Packagers and Decoders Dwelling deeper into the implementation of the packager class along with the decoder class. The architecture is designed such that there is a single decoder which will decode the incoming message and notify the relevant observer through an enum selection process. This design is most useful when it is implemented in the main controls communications. The design itself can be seen in the class diagram shown in Figure 5.10.

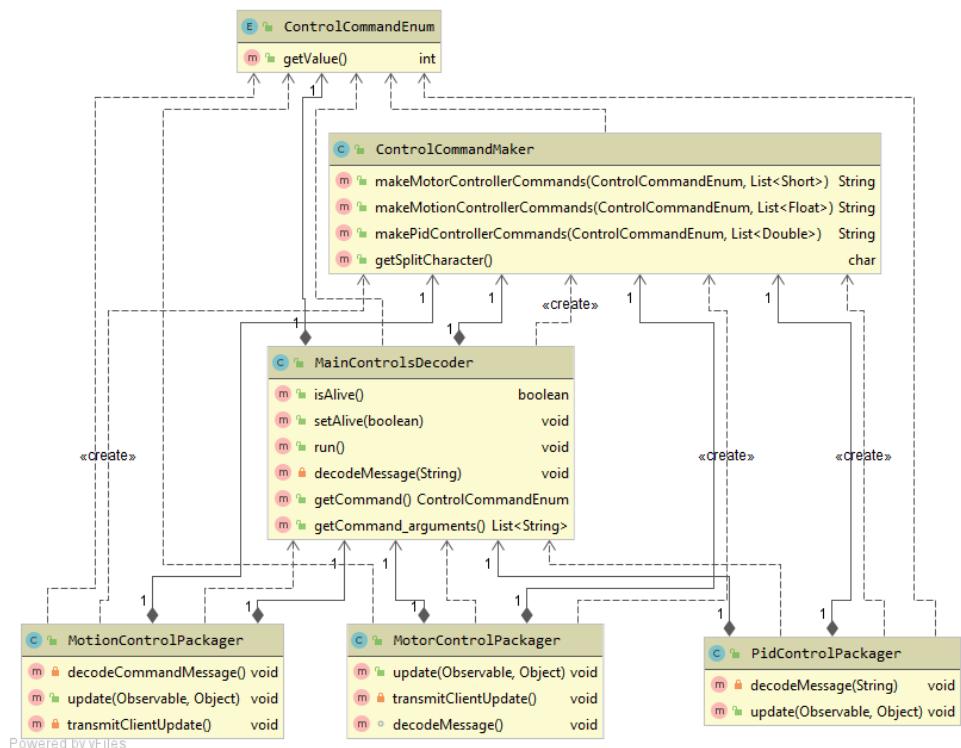


Figure 5.10: Primary Packager-Decoder Class Architecture

5.3.4.1.2 Transporter The final stage of communications is the transporter stage. The transporter stage uses TCP sockets to transfer all data. TCP was used as it is straightforward to set up along with ensuring the end to end communication. Unlike

UDP which can't provide end to end connection. The transporter class implements the ITransporter interface and as such, can be easily swapped out for other means. The class is also thread-safe to ensure that each message sent from the socket is correct and not corrupted.

5.3.5 View and Presenter Structure

Following the Model View presenter pattern presented earlier in chapter 3, the presenter class and view FXML code are fundamental, ensuring the model can be accessed by the user. This involved creating many different view scenes which in turn are described in the following sections. The presenter essentially acts as a controller for the view elements along with acting as an observer for the relevant model elements. For a more detailed analysis of this structure, refer to section 5.3.4.

5.3.5.1 CameraModule

As for the CameraModule and all subsequent Views they incorporate a very similar class structure as shown in Figure 5.11. The view associated with the CameraModule is also shown in Figure 5.12.

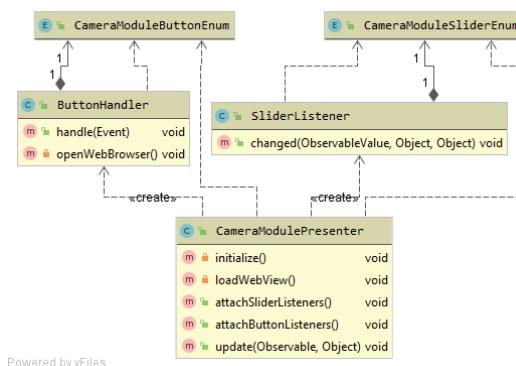


Figure 5.11: CameraModule Presenter Class Structure



Figure 5.12: Camera Module View

5.3.5.2 MotorControl

Another GUI was developed for controlling the motors by there corresponding turn and throttle values, the implemented design is shown in Figure 5.13.

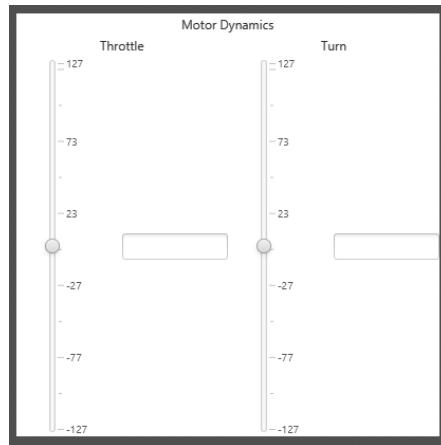


Figure 5.13: Motor Control View

5.3.5.3 MotionDynamics

As for controlling the motor dynamics in speed and current yaw angle. A GUI was devised but due to time constraints and other issues was not fully implemented, the FXML view is shown in Figure 5.14.

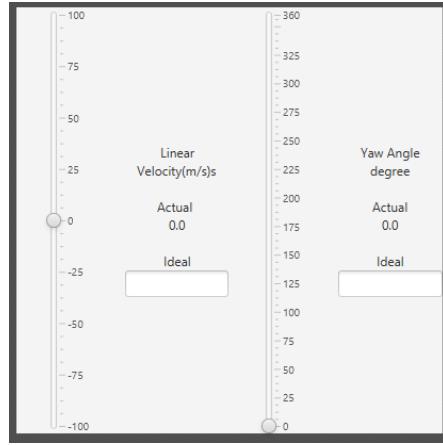


Figure 5.14: MotionDynamics View

5.3.5.4 SensoryData

In order to show all the different sensors, another GUI was developed and is shown in Figure 5.15. However due to timing constraints was not fully implemented in the presenter structure.

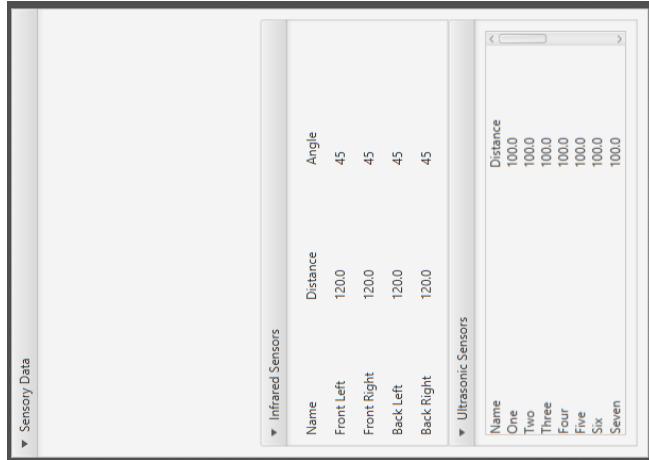


Figure 5.15: SensoryData View

5.3.5.5 TelemetryData

For dealing with telemetric data such as logging and PID control loop timing and various other features, a tab-like system was developed graphically and is shown in Figure 5.16.

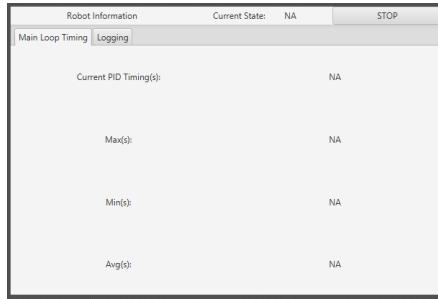


Figure 5.16: Telemetric FX View

5.3.5.6 WheelDynamics

For representing each wheel speed and current angle, a view was developed; however, the graphics for moving the wheel to describe the current angle were not developed due to timing constraints. The view as it stands is shown in Figure 5.17.

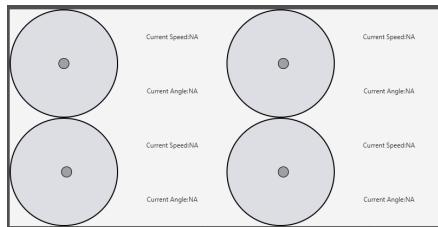


Figure 5.17: WheelDynamics View

5.4 Robot Controller

The final high-level application was the Python Robot Controller. Its purpose was to control all aspects of the robot; this includes sensors and actuators. As well as the control aspect, it's also responsible for holding all positional data.

Not entirely related to the Python Robot Controller application, for it to work correctly, a WiFi hot-spot procedure was followed to allow the robot to be controlled through a WiFi connection rather than an Ethernet connection. [63] For more information regarding the various software packages and requirements refer to appendix C.5.1

5.4.1 Requirements

The system must be able to access the I2C bus on the Raspberry Pi to communicate with the sensor controller discussed in chapter 6. Not only this, but the application must also be able to store sensor values, and navigational data while maintaining a connection with the Java Control Interface discussed earlier.

5.4.2 Python

The language chosen for the robot was python 3.7.3 [29], this was chosen as it is one of the most learnt programming languages in the world and as such has significant libraries and is directly targeted at Raspberry Pis. [64]

5.4.3 Observable Pattern

Similar to the Java Control Interface discussed earlier, the observer pattern is used extensively through the design in a multithreaded implementation. This was to allow all the different communication systems to run independently and not relying on synchronisation. The Observers in this design are the Packagers and decoders responsible for high-level communication with the control interface. The rest of the model runs their sub-models each on a different thread manipulating their data.

5.4.4 System Architecture

The entire system architecture is shown in Figure 5.18. It is split into communications, Manual Control, Movement, Controllers, Miscellaneous and finally Sensors. For the most part, each of these packages is running independently of each other with some loose coupling between packages. An example of the loose coupling being the packagers from the communications alongside the rest of the system.

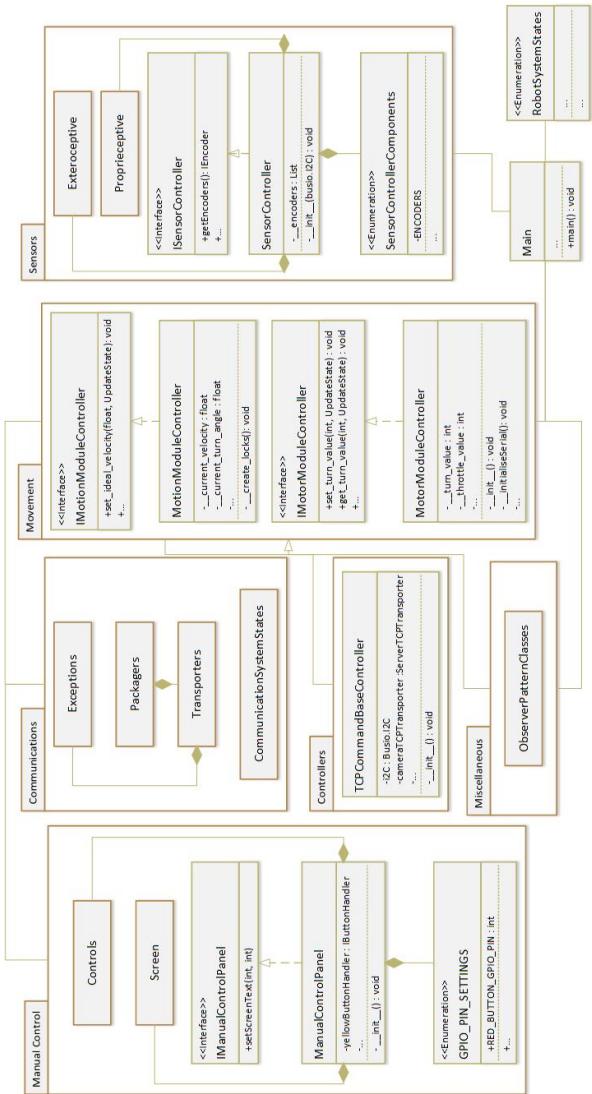


Figure 5.18: Python Module Structure

5.4.4.1 Communications

In terms of the communications package, it works in the same manner as the Java Interface equivalent as discussed previously in section 5.3.4.1. It incorporates the observer pattern along with using a separate class for constructing commands for the packagers to then send using the transporter.

Along with the packagers and decoders, there is another essential structure within communications, and that is the implementation of web sockets to transport the video data

from the Raspberry Camera Module. This implementation involves using the adapter pattern to effectively modify an external library into a usable class for the rest of the system. The library used was the Pistreaming library found on GitHub which allows the camera video to be encoded in FFmpeg and then broadcasted using an HTTP Server. Unfortunately, there was not enough time to further implement this into the code, and as such, to access the camera stream, a web browser is required. [65]

The complete port list along with the description of the port are listed in appendix D.

5.4.4.2 Controllers

The next section of the Python Robot controller is the controller's package. It's responsible for running and maintaining the system as it stands. Currently, due to timescale issues, the only implemented controller was the CommandBasedTCPController, which is responsible for opening up the server-side sockets to the control interface. The class works by creating objects of all aspects of the robot controller; this includes the relevant packagers and encoders and the controllers discussed later in this section. Each of these model elements run on their thread, continually updating their current values.

5.4.4.3 ManualControl

Another package developed for the Python Robot Controller is the ManualControl Package. Its purpose was to interact with the I2C LCD Screen and the buttons attached to the manual control interface. The design used for this implementation is shown in Figure 5.19. The design involves full decoupling from the rest of the system via IManualControlInterface and an Interface for the individual buttons. Each button has its own dedicated to which two functions can be assigned to the button using the ButtonFactory Class, the function to be called first is defined within each of the button classes, external functions are passed in via the factory method makeButton().

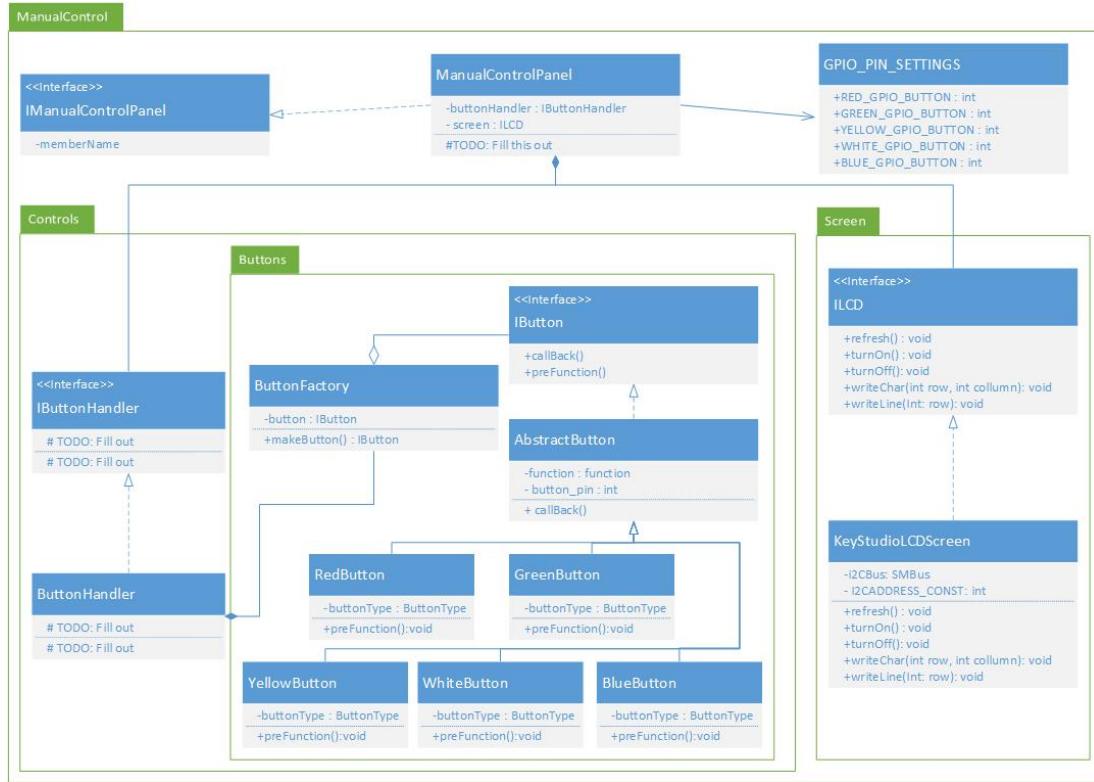


Figure 5.19: Manual Control Interface Package Class Diagram

5.4.5 Miscellaneous

An important feature is the Observer Interface and Observable pattern. The reason these are important is that Python does not have a direct implementation in the language. To remedy this problem, classes were developed to solve this issue and are held in the Miscellaneous package. The Observable class involves a threading aspect which allows each observer to update itself in its own thread; this is to allow the full system to run as fast as possible and avoid blocking methods through the update process.

5.4.6 Movement

As for the movement package, its priority is the navigation and maintaining specific speed and direction. Due to timing constraints along with numerous I2C issues, this could not be completed. However, the structure is still there, and a note should be taken of the design. The design itself works by having two classes accompanied by

interfaces. The first class to be discussed is the MotorModuleController, which is entirely responsible for communicating and controlling the Rampaging Chariots Motor Controller boards. It does this via thread that transmits the current throttle and turn values over UART to the motor controllers. The values are changed via thread-safe set methods. This provided open-loop control of the robot and was entirely functional. The next class is the MotionModuleController, its responsible for calculating and maintaining the exact speed and orientation of the robot; however, due to I2C issues, it was not tested and not currently functional. It was, however, running in its own thread which was thread-safe allowing the adjustment of the current speed and orientation variables using an outside observer class. The class was intended to incorporate a PID Controller using the Python PID library, but due to timing constraints, this was not met.

5.4.6.1 Sensors

The final package to be discussed is the sensors package. It consists of the SensorController class and sub-packages for both proprioceptive and exteroceptive sensors. The Sensor Controller class is a threaded class which is constantly querying the sensor controller and updating all sensor values on the python controller. The Sensor controller and all subsequent sensor classes extend the Observable Class allowing them to be observed any part of the system. They are thus enabling further decoupling and extending maintainability of the system.

As a final note, the CameraModule package incorporates a sub-package containing different servo implementations, hardware, software and finally I2C enabled servo control through the servo controller. Each version can be accessed and facilitated through the factory method in the ServoFactory Class.

5.4.7 Proportional Integral Derivative Controller

Due to time constraints and recurring Sensor Controller issues, the PID controller could not be implemented in the Python Robot Controller. The plan for implementation was to use the simple-pid library [66] to implement each of the terms discussed in chapter 3.

Chapter 5. Software Design and Implementation

This was going to primarily run off the encoders speed values, and using the Inertial Measurement Unit to gauges if the values are in an acceptable range. When they are not in an acceptable range, the IMU module would be used to compensate for false values. For more information regarding why this was not implemented, refer to chapter 7.

Chapter 6

Electronic Design and Implementation

6.1 Requirements

The Electronic aspect of the robot was required to be expandable and allow for modulation of a full design into smaller sub designs such that multiple systems can be added to the robot if necessary. This meant that both the firmware and hardware has to be expandable in order to be able to be upgraded without major revisions.

6.2 Design Process

The Electronic design process was structured such that hardware and firmware were developed side by side with an emphasis on hardware due to the developer's pre-existing knowledge and familiarity. However, due to the learning curve associated with HAL library [67] the firmware was lagging behind and not fully functional and implemented, for more information on this aspect refer to chapter 7.

6.2.1 Low Level Communication Network

As for communications between the Raspberry Pi Controller and the rest of the subsystems, the communication method chosen was Inter-Integrated Circuit(I2C) as this

provided the most benefits with its addressing scheme and throughput. The communication structure for all electronic components is shown in Figure 6.1. From this, we can see that there is both a 3.3V and 5V I2C network and multiple other devices.

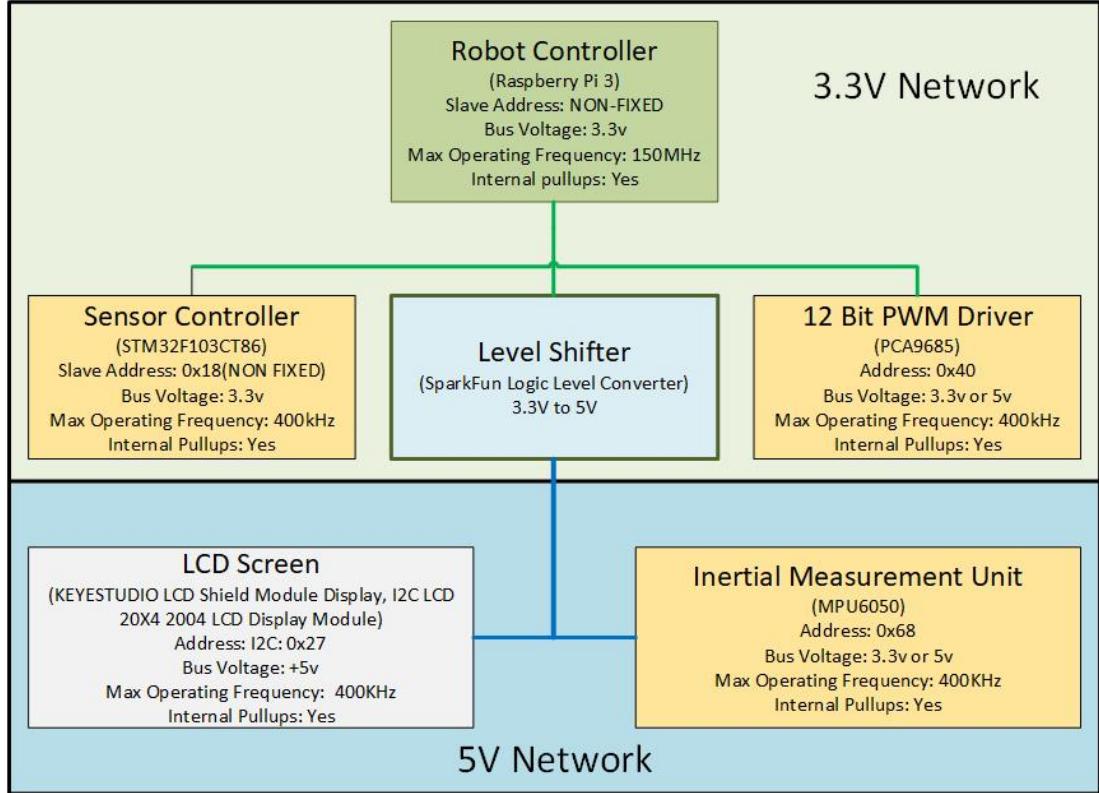


Figure 6.1: I2C Communication Network

6.3 Hardware

Since the project involves taking the radio-controlled Rampaging Chariot and turning it into an Autonomous Robot, there is a lot of hardware design in the project as everything has to be made in house. With that said, multiple Printed Circuit Boards(PCBs) were designed each for a particular purpose. In total, six designs were constructed with a need for eight PCBs in total to be constructed. The complete list of PCBs and a summary of responsibilities are shown in Table 6.1.

The software used to develop the PCB designs was Autodesk Eagle 9.5.1 [68], this was chosen for the reason that it is very familiar with the developer as well as it

Table 6.1: Electronic Hardware: PCB Design Summary

Electronic Hardware PCB Design Summary	
PCB Name	Responsibility
Power Converter Board	Converting and stabilising the standard 18V Battery Power to 5v and 12v for the different subsystems in the robot
Primary Power Distribution Board	Reducing any power transients at the heart of the power network, also responsible for powering all 5V Sensors and connecting 18V supply to motor controllers
Secondary Power Distribution Board	Splitting one of the main power networks into more sub-networks, used for additional power ports on the PCB stacks
Sensor Controller Board	Used to connect and control all sensors within the robot, this includes encoders for drive control, ultrasonic and infrared sensors for distance calculations and proximity detection
Raspberry Pi HAT	Responsible for interfacing the Pi into the robot systems such as I2C and various other connections
Inertial Measurement Unit External Hardware Components PCB	Acts as a place holder for the gyroscope as well as provides transient smoothing with onboard capacitors

does student licenses which again is ideal for the Rampaging Chariots as there target audience will be able to access the schematics using eagle.

6.3.1 Power Converter Board

The Power Converters Board is designed to convert the main 18V supply from the battery into 5V and 12V. It does this by using a set of Switch Mode Power Supplies discussed earlier in chapter 3 and a DC converter circuit.

6.3.1.1 Functional Diagram

In order to design this circuit, a functional diagram was made to ensure all the design criteria were met. This diagram is shown in Figure 6.2.

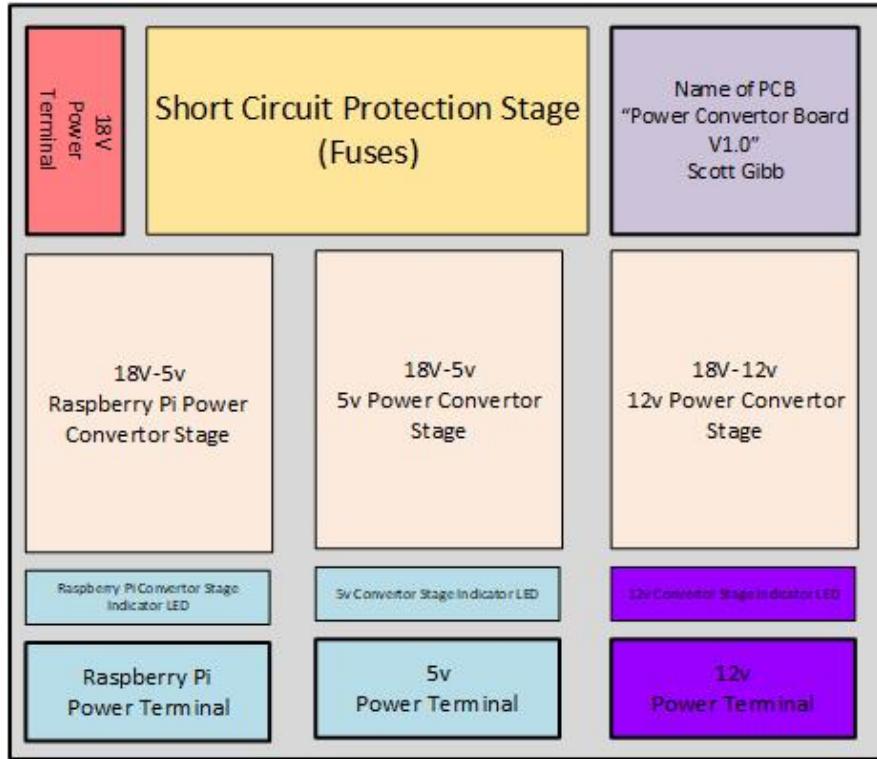


Figure 6.2: Power Converter Functional Diagram

The diagram consists of three voltage regulation and converter modules, along with this, there is short circuit safety modules and circuit taps for current and voltage measurement. Finally, the functionality ends with status LED's for the individual voltage regulators.

6.3.1.2 Implementation

A small section of the full schematic is shown in Figure 6.3. For the full schematic, refer to appendix F.1.3 and for the layout and component list refer to F.2.3 and table F.3. This subsection shows the basic converter circuit that is used for all conversions, the only difference between them being the component values. To prevent any short circuit damage to the powered circuitry, fuses were added to each of the output stages of the converters.

The design is also built for expansions purposes, there are two dupont headers on the system that allow for voltage and current measurement(a possible future upgrade to

the system). The current measurement terminal is put in series with the converter stage before the LED so that the power consumption of the LED is included. LEDs are added to each stage to give visual information that the converters are working. A trim-pot resistor is used on each converter to tune the voltage to the corresponding output 5V and 12V.

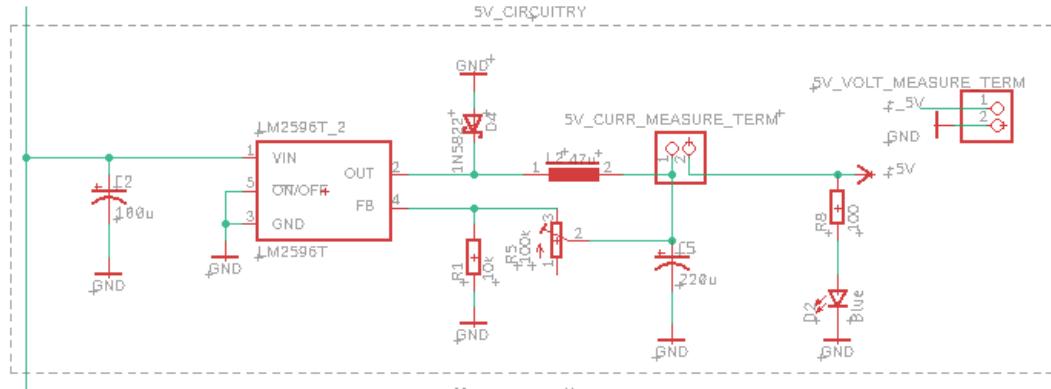


Figure 6.3: Power Converter Sub Section Circuit Diagram

The inductor and capacitor values for each circuit was worked out by using the values from the corresponding breakout boards that were used initially in the system. [69]. This meant that the system was mirroring this board, and thus the verification that the system would work was more straightforward. However, that board used surface mount components and thus, research was conducted to replace these parts with through-hole components.

6.3.1.3 Component Selection

In order to remain as close to the board as possible, the capacitor and inductor values were kept the same, as through-hole components were easily found for these parts. The LM2596 Module was kept as well; however, the datasheet was used to ensure the wanted operation. [70]. As for the diode and the inductor, the following components are shown in Figure 6.4 were chosen:

The reason these components were chosen in particular is for the following reasons,

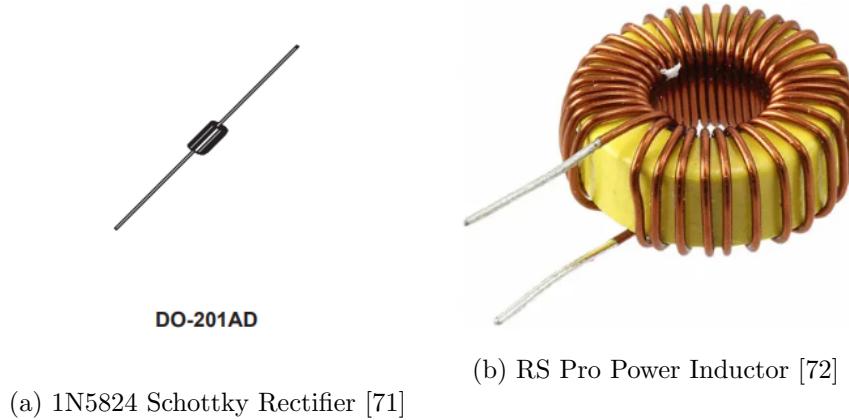


Figure 6.4: Power Converter Through Hole Components

the circuit shown in 6.3 is designed to output a maximum of 3A. The inductors current rating is 3A, as well as the diode chosen(1N5824), can handle 3A in forward bias mode, which is ideal to act as the flyback diode in the circuit.

6.3.1.4 Printed Circuit Board Manufacturing

A layout was then built for the full schematic and is shown in FigureF.12. Using Eagle Manufacturing a PCB visualisation of what the board would look like with silk skins and Solder masks applied to the board, the full design is shown in FigureF.18. This was printed and populated, the result of this process is shown in Figure 6.5, however the design does not entirely match the renders shown in Figure F.18, the reason for this is that the design that was sent for manufacturing had some errors in which the design did not incorporate current limiting resistors for the status LED's and the wrong eagle component was used for the fly-back diodes footprint, these issues were fixed by making the holes for the components slightly bigger and using surface mount resistors on the back of the PCB for the LED's.

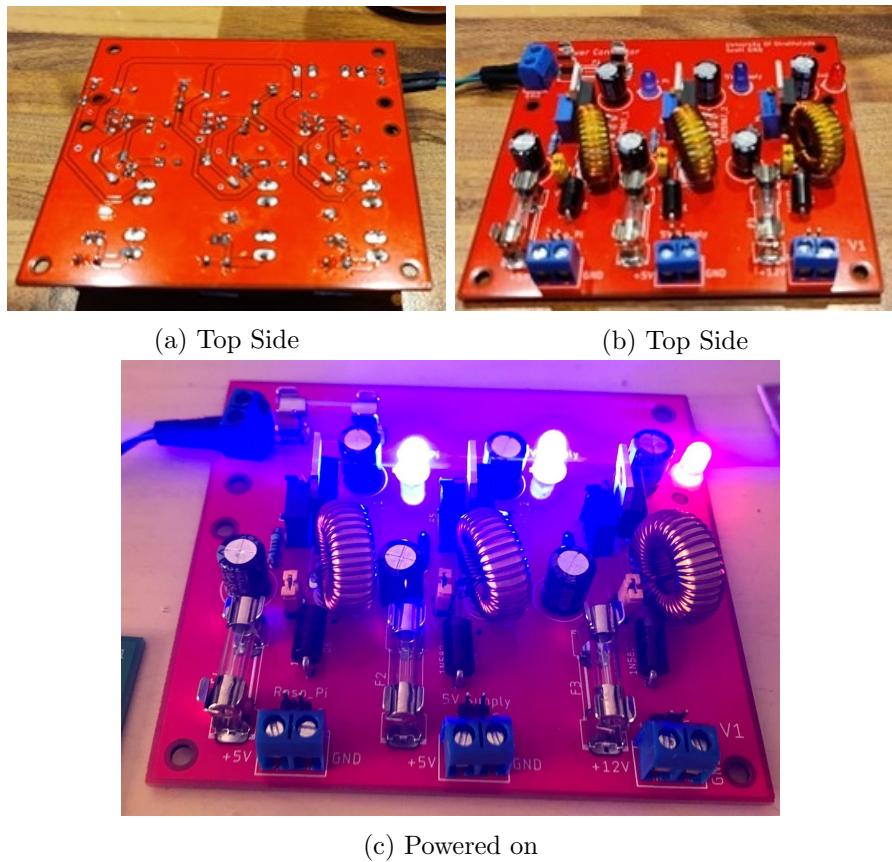


Figure 6.5: Populated Power Converter Board

6.3.2 Primary Power Distribution Board

One of the most critical and useful boards in the robot is that of the Primary Power Distribution Board. Its purpose is to distribute power to the 5V, 12V and 18V subsystems such as the motor controllers and sensors.

6.3.2.1 Functional Diagram

The design required three power pathways and a variety of terminals, along with these components, some form of short circuit protection is required for the 18V supply. The full functional diagram is shown in Figure 6.6.

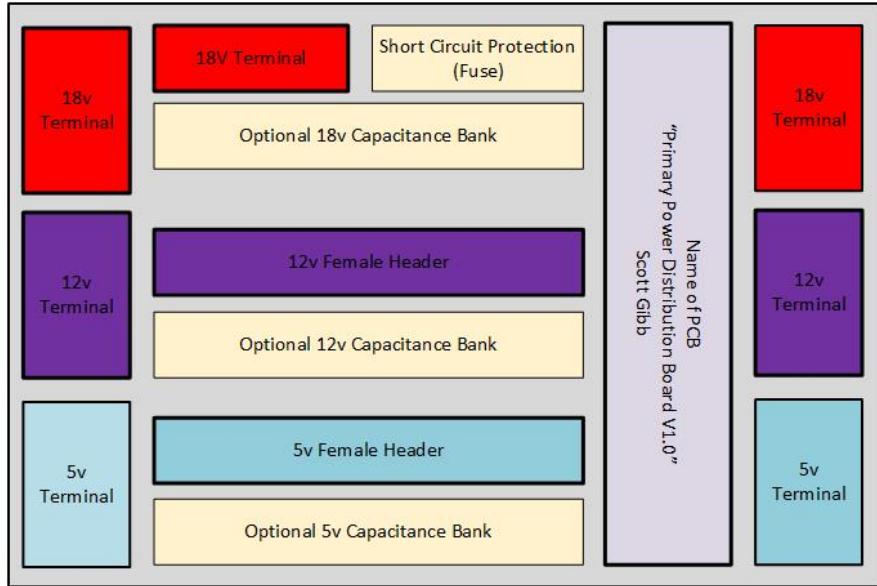


Figure 6.6: Primary Power Distribution Board Functional Diagram

6.3.2.2 Implementation

This is also the first board that the battery is connected too, due to this fact this board deals with a high amount of current and thus the traces widths for the power lines are more significant than other traces. The board also contains 18V fuse for the 18V Power Line. This board circuit diagram can be seen in appendix F.1.4 along with the layout in F.13. The full component list is also shown in table F.4

The design incorporates three $1000\mu\text{F}$ Electrolytic Capacitors each with the appropriate voltage rating. These are designed to smooth any low frequency transients in the system, it also helps with transients from the sensors due to all sensors being plugged into these boards. Due to the way the circuit is designed, this is an optional upgrade and is dependent on how much of the robots systems is implemented, if for some reason these capacitors are not needed or in fact a larger or smaller value is used, then this can be easily replaced with little effort required to the user. Dupont female headers are used for the 5v and 12v lines in order for easy plug in.

6.3.2.3 Component Selection

For selecting the right capacitor, the requirements were looked at for the system along with estimations for future extensions of the project. For each power line $1000\mu\text{F}$ capacitors were added in parallel with the voltage loads. This value was chosen as a standard value that will deal with low frequency voltage drops, the voltage rating of the capacitors were standard values available from the university's laboratory, 16V and 25V.

6.3.2.4 Printed Circuit Board Rendering

Again, using Eagle Manufacturing menu the PCB was rendered with the silk skins and solder masks, the result of which is shown in Figure F.19. The board was printed and populated and is shown in Figure 6.5.

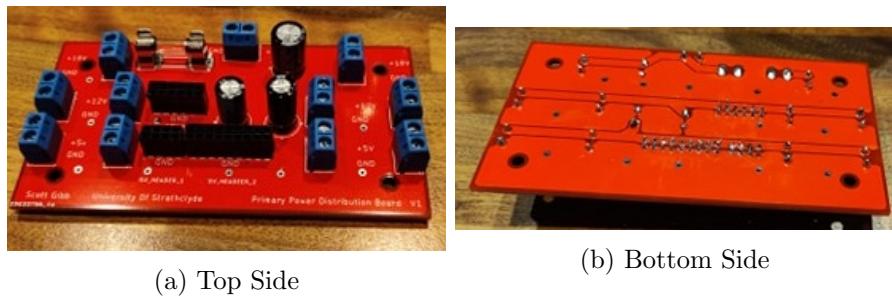


Figure 6.7: Populated Primary Power Distribution Board

6.3.3 Secondary Power Distribution Board

Another module that was created was that of a Secondary Power Distribution Board. Its focus was to allow power distribution for each of the main PCB stacks.

6.3.3.1 Functional Diagram

The required functionality is not as complicated as other designs and is shown in the diagram in Figure 6.8.

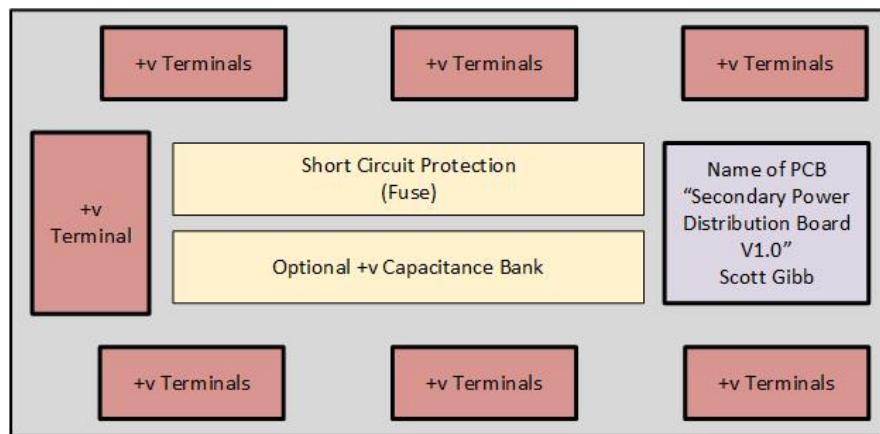


Figure 6.8: Secondary Power Distribution Functional Diagram

6.3.3.2 Implementation

The voltage going into the board was variable. The board contains a fuse holder for short circuit protection of each of the connected sub-power networks stemming from this board. It also contains an Electrolytic Capacitor as well for dealing with any transients stemming from the sub-networks. The circuit diagram for this design is shown in Figure 6.9. For the Eagle schematic, refer to appendix F.1.5 and appendix F.2.5. The component list is also shown in table F.5. The design utilises Screw terminals for the different sub-power ports, allowing higher current cables to be connected. Its use for distributing power to subsystems primarily.

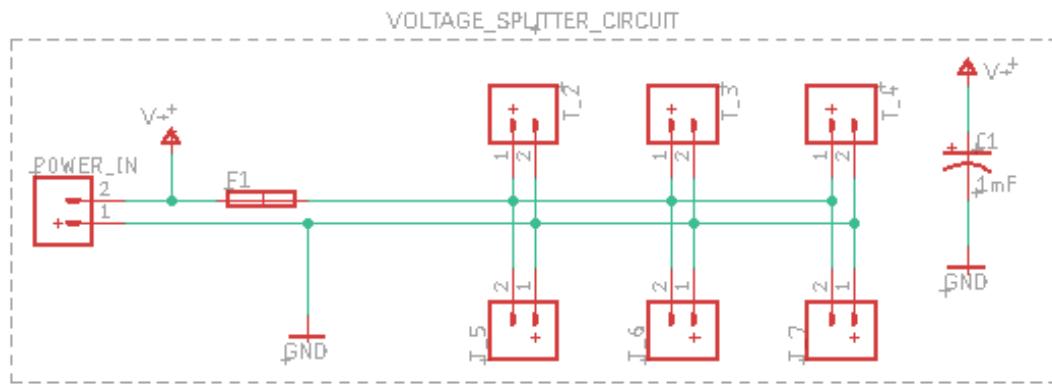


Figure 6.9: Secondary Power Distribution Circuit Diagram

6.3.3.3 Component Selection

The capacitor chosen for the transient smoothing was again $1000\mu F$, the reason for this is that its capable of dealing with larger voltage drops at lower frequencies, this is ideal for the functionality of this board, due to it supplying lots of internal subsystems. The voltage rating for the capacitor was chosen to be 25V, this allows the board to be used for any voltage currently present in the system.

6.3.3.4 Printed Circuit Board Rendering

Again using Eagle, the PCB was rendered and is shown in Figure F.20. This PCB was printed and populated and is shown in Figure 6.10.

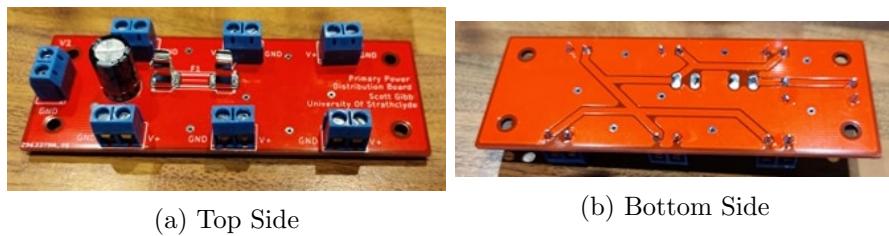


Figure 6.10: Populated Secondary Power Distribution Board

6.3.4 Sensor Controller Board

The most critical PCB for the robot is the Sensor Controller. Its purpose is to interact with all the sensors on the system, the design for this is rather intricate due to it being required to interact with both PWM and analogue sensors. This design uses the STM32F103CT6 [9] as the microcontroller used to interact with all the sensors. I2C is used as the Inter-board communications network, allowing the Raspberry to interact with the system. The board has multiple subsystems each with their own specific design criteria.

6.3.4.1 Functional Diagram

This is the most sophisticated hardware design of the full project and so requires a lot of different functionality. As a result, a functional diagram was made and is shown in

Figure 6.11.

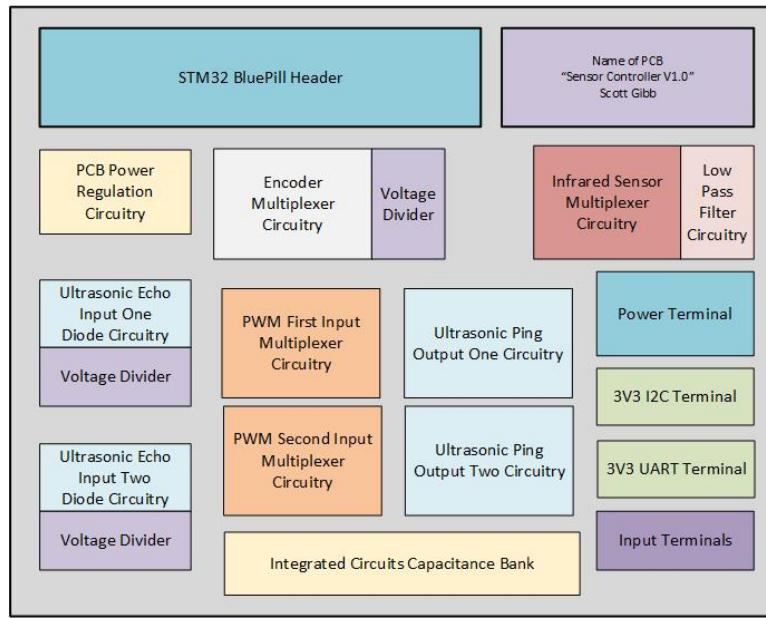


Figure 6.11: Sensor Controller Functional Diagram

6.3.4.2 Microcontroller Pin Utilisation

The pin utilisation of the microcontroller is shown in Figure 6.12. From this, it can be seen that the system is fully utilised except from two pins which are not in the breakout board [10].

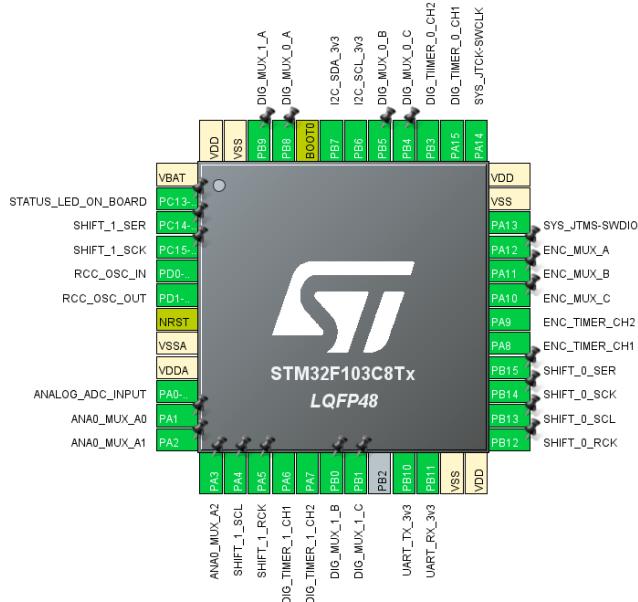
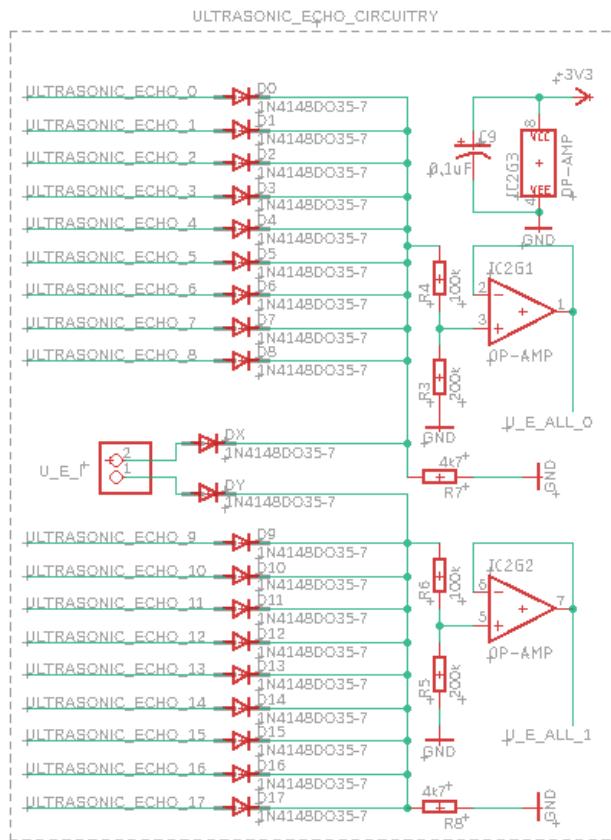


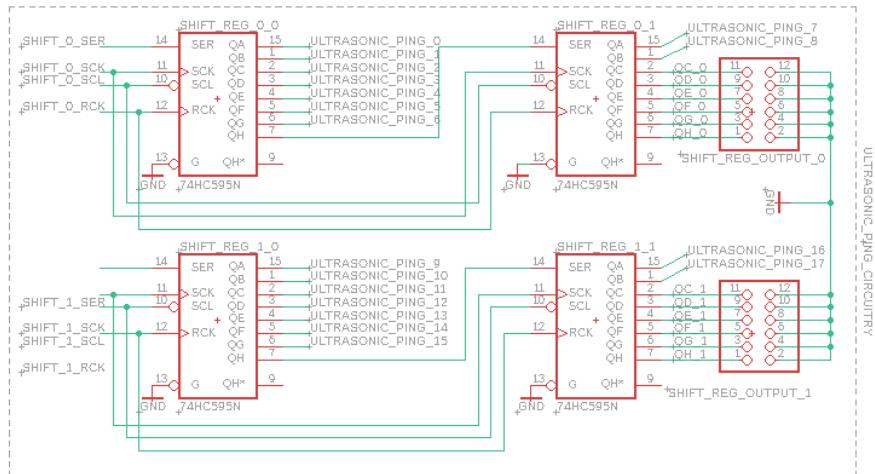
Figure 6.12: Sensor Controller Pin Utilisation

6.3.4.3 Ultra Sonic Circuitry Implementation

The system uses a total of 17 ultrasonic sensors, the microcontroller only has four Timers [73], so this required some form of multiplexing to attach all the systems. The chosen method was to use a series of shift registers connected to two timers and tie all the ultrasonic sensors together using a series of diodes. The circuits associated with this are shown in Figure 6.13.



(a) Ultrasonic Echo Circuitry



(b) Ultrasonic Ping Circuitry

Figure 6.13: Sensor Controller: Ultrasonic Circuitry

The diodes chosen for this task was the small signal fast switching diode (1N4148) [74]. This allowed for a maximum voltage drop of 1V to be applied to the incoming signals. It also allowed the creation of an eight input or gate. In order to protect the STM32 microcontroller pins a high resistance voltage divider was used to lower the Transistor-Transistor Logic(TTL) level to 3.3V. This voltage divider was then followed by an operational amplifier(LM358 [75]) in buffer configuration to help stabilise the incoming signal as well as provide a high impedance block between the rest of the sensor controller circuitry.

As for the Echo circuitry two shift registers(SN74HC595N [76] were used for each set of signals, this allowed each individual sensor to be triggered based on a serial command, sent from the microcontroller. $0.1\mu F$ Electrolytic Capacitors were added to the Integrated Circuits as well, to help with any transients the board may have.

6.3.4.4 Infrared Distance Sensor Circuitry Implementation

For the Infrared Distance Sensors(SHARP IR Sensor [47]), an analogue multiplexer(DG408DJ [77]) combined with a buffer operational amplifier(LM358 [75]) and a passive Low Pass RC filter was used, the schematic for this particular circuit is shown in Figure 6.14.

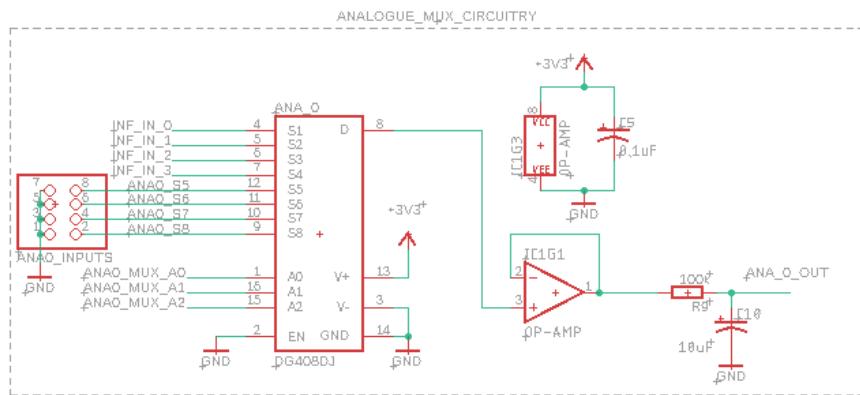


Figure 6.14: Sensor Controller: Infrared Input Circuitry

The Buffer operational Amplifier is used to stabilise and isolate the incoming analogue signals. After this stabilisation process it goes through the filter which is configured to remove frequencies above 100Hz. As a final consideration, a $0.1\mu F$ Electrolytic

capacitor was also added to the circuit to help with the multiplexer power line transients. The multiplexer along with the operational amplifiers were all powered off the 3v3 line this was a safety measure to stop any high voltages reaching the microcontroller pins.

6.3.4.5 Encoder Sensor Circuitry Implementation

The final set of sensors the sensor controller has to deal with, is the AS5600 Encoders [11]. The sensors output 5V PWM signals, as such a voltage divider was put in place to bring these down to 3.3V, this was then followed by a buffer operational amplifier(LM358 [75]) to stabilise the signal. This design ends with a $0.1\mu F$ Electrolytic capacitor attached to the power rails of the digital multiplexer [78].

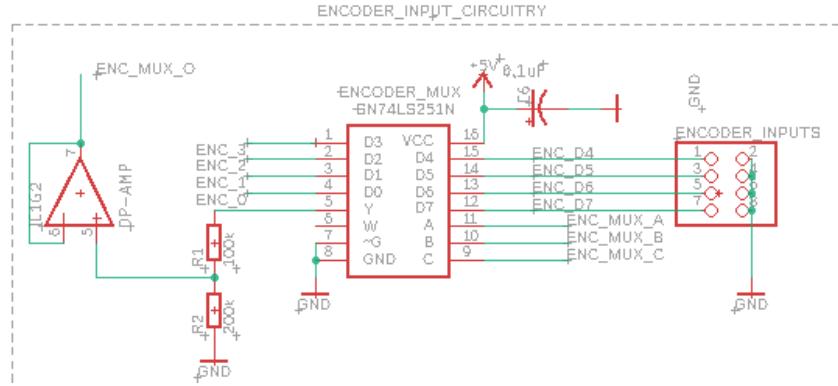


Figure 6.15: Sensor Controller: Encoder Input Circuitry

6.3.4.6 Power Regulation Circuitry Implementation

The last crucial circuitry for the sensor controller is the power regulation circuitry. Its purpose was to deal with any transients that can come from the 5V power line. The circuitry used for this transient smoothing is shown in Figure 6.16.

The circuitry involves using an electrolytic and ceramic capacitor in parallel with each other, the values of these capacitors are $100nF$ (Ceramic) and $100\mu F$ (Electrolytic). The electrolytic capacitor is targeting low frequency transients by providing power to these large low frequency voltage drops, whilst the smaller ceramic capacitor is targeting high frequency smaller voltage drops in the power line.

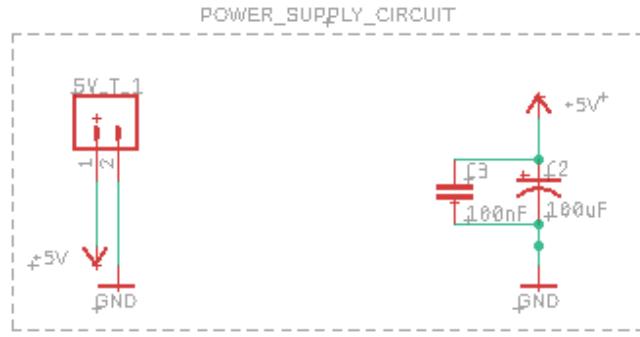


Figure 6.16: Sensor Controller: Power Regulation Circuitry

6.3.4.7 Printed Circuit Board Manufacturing

Finally the entire design was rendered using Eagle [68] and is shown in Figure F.17. A version of this board was also manufactured and populated however, the version built had some layout issues in which there was a missing IC Capacitor, as well as this there was a connection issue in which the the Analogue Multiplexer output was fed directly into the microcontroller rather than through the Low pass filter and buffer operational amplifier, the populated PCB is shown in Figure 6.17.

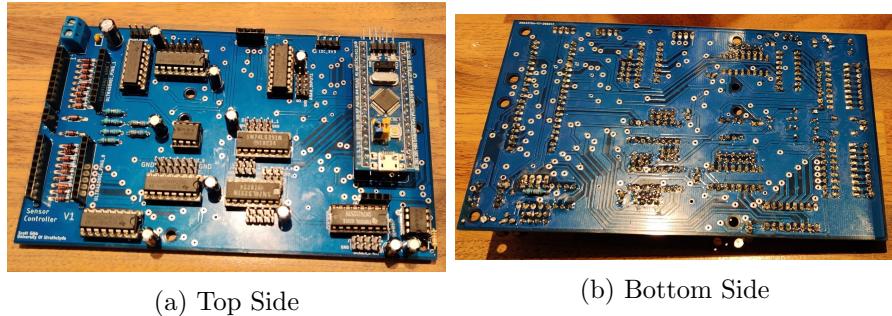


Figure 6.17: Populated Sensor Controller

6.3.5 Raspberry Pi Hardware Attached on Top

The design was first built by establishing a functional diagram, and this shows the intended functionality of this particular subsystem, the resulting diagram is shown in Figure 6.18

6.3.5.1 Functional Diagram

The design was first built by establishing a functional diagram. This shows the intended functionality of this particular subsystem. The resulting diagram is shown in Figure 6.18.

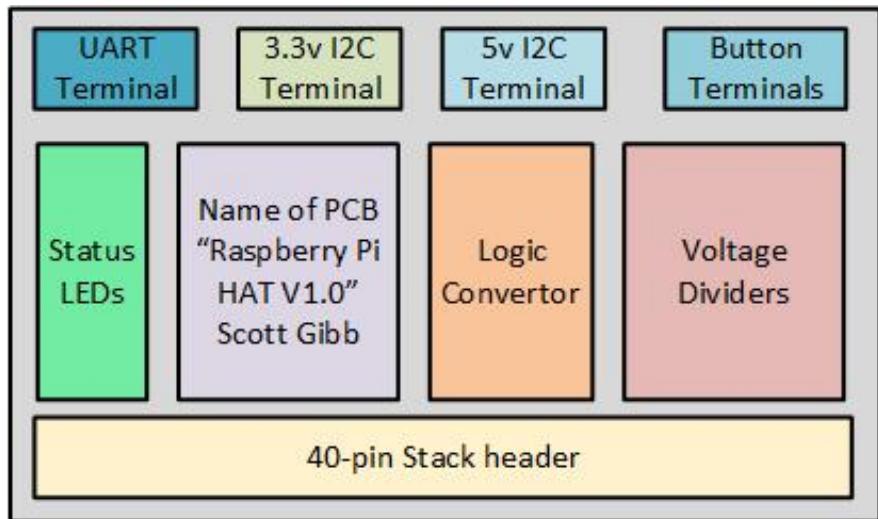


Figure 6.18: Raspberry Pi HAT Functional Diagram

6.3.5.2 Implementation

Since the system contains many different functionalities, circuits were designed for each of the relevant functionalities. The circuits designed were mainly for connecting the Raspberry Pi GPIO pins to the relevant headers, for these circuit diagrams refer to appendix F.1.1.

6.3.5.3 Component Selection

In order to implement these circuits, components were selected for the voltage dividers and the logic converters. The logic converter used was the SparkFun Bi-Directional logic converter [23]. The breakout board is shown in Figure 6.19.

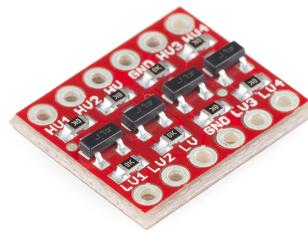


Figure 6.19: SparkFun Logic Converter [23]

Along with this logic converter, the resistor divider circuitry is shown in Figure 6.20. The dividers use $100k\omega$ in series with $200k\omega$ resistors to change the 5V button signals to 3.3V signals. Finally current limiting resistors were also used for the status LED's,

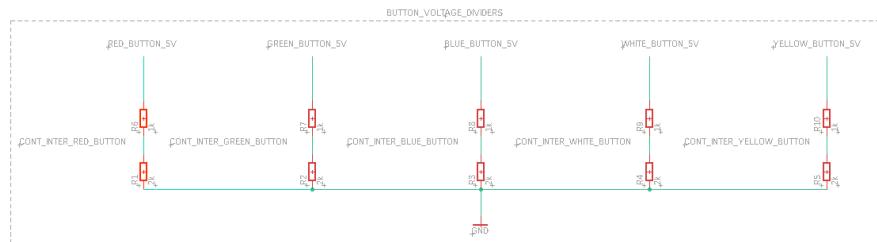


Figure 6.20: Raspberry Pi HAT Voltage Divider Circuitry

the values used were 330ω .

6.3.5.4 Printed Circuit Board Manufacturing

After the design was finalised and components selected, the PCB was rendered using Eagle [68], and the resulting renders are shown in Figure F.16. This design was then manufactured and populated, the result of this process is shown in Figure 6.21.

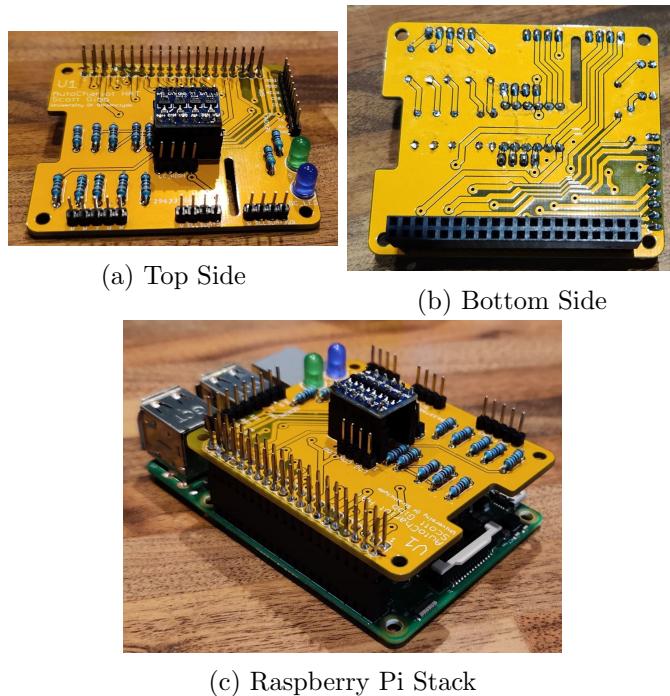


Figure 6.21: Populated Raspberry Pi HAT

6.3.6 Inertial Measurement Unit External Hardware

In order to ensure the exact placement of the MPU6050 Breakout board [44], a specially designed PCB was made in order to ensure the placement of the breakout board in the robot.

6.3.6.1 Functional Diagram

This is the projects simplest PCB, its functionality is very limited and is shown in the diagram in Figure 6.22.

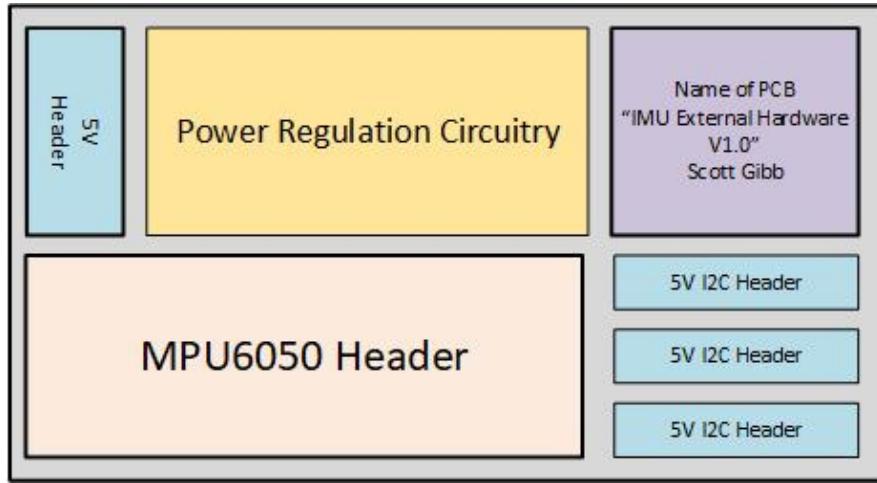


Figure 6.22: IMU External Hardware Functional Diagram

6.3.6.2 Implementation

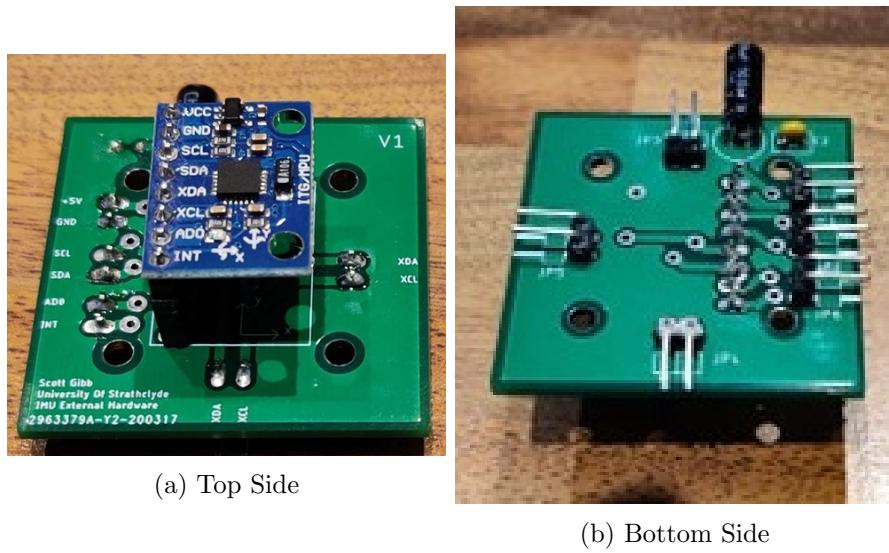
In order to reduce any switching transients in the power lines, power regulation circuitry was designed. It consisted of two capacitors in parallel, the circuitry for the entire PCB is shown in Figure F.9.

6.3.6.3 Component Selection

The capacitors chosen for transient smoothing were $100\mu F$ electrolytic capacitor and $100nF$ ceramic capacitor. This allows the low frequency higher voltage drops to be dealt with using the electrolytic capacitor and the high frequency lower voltage drops to be dealt with using the ceramic capacitor.

6.3.6.4 Printed Circuit Board Manufacturing

Finally, Eagle [68] was used to render the PCB design, and the rendered design is shown in Figure F.21. This design was then manufactured and populated, the finished product is shown in Figure 6.23.



(a) Top Side

(b) Bottom Side

Figure 6.23: Populated IMU External Hardware

6.4 Firmware

Following the hardware design discussed previously, the only aspect of firmware to be designed was the Sensor Controller firmware as this was the only component that required a form of intelligence to control and interact with the hardware.

The requirements of the Sensor Controller firmware were to be able to interact with a variety of proprioceptive and exteroceptive sensors via the appropriate signal decoding. The board was required to read in all the raw sensor values and convert them into their corresponding positional data values. During the conversion process, filtering should be applied to ensure the values are reasonable and not severely effected by noise and conversion errors. As well as processing and storage of this data, the board should be able to be accessed by the Raspberry Pi and the data store retrieved. The reason for the digital processing on the Sensor Controller is to relieve as much computational stress as possible from the Raspberry Pi. The firmware should be both modularised and easily expandable.

The microcontroller chosen for the sensor controller boards processing unit was the STM32F103CT86 [73]. This microcontroller is coded in pure C with the inclusion of the Hardware Abstraction Library [79]. For this reason, the design process involved researching different approaches to reading sensors and interacting with different hardware through the use of the HAL functions.

6.4.1 Implementation

The implementation of the Sensor Controller firmware took the approach of developing abstractions layers from the hardware up to the central processing loop. This allowed for extensive modularity as well as decoupling the lower level functions from the high-level processing. The design approach is shown in Figure 6.24. The overall design utilises three timers for input capture mode, and an Analogue to Digital Converter configured for interrupt mode. The firmware was split into three control mechanisms, DriveControl which was responsible for reading in encoder values and calculating the different

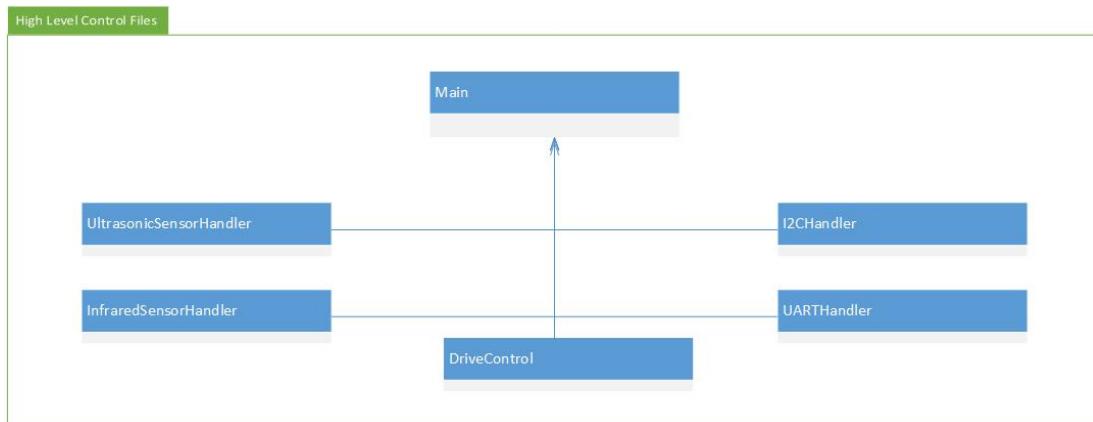


Figure 6.24: Sensor Controller High Level Firmware Diagram

wheel speeds, InfraredSensorHandler which is responsible for controlling and processing the Infrared Sensors and finally the UltrasonicSensorHandler which is responsible for controlling the related SensorController hardware and processing these values. There were two other handlers developed which were the I2CHandler for communication with the Raspberry Pi and the SensorController and finally the UARTHandler responsible for live feedback.

6.4.1.1 Drive Control

The DriveControl source file is associated with three other files. The DriveControl application loop uses each of the associated files to connect to the AS5600 modules [11] process the raw Pulse Width Modulation(PWM) values and store them as the calculated angles and speeds. The WheelEncoders source file is for storing and converting the PWM values. The hierarchy of this design is shown in Figure 6.25.

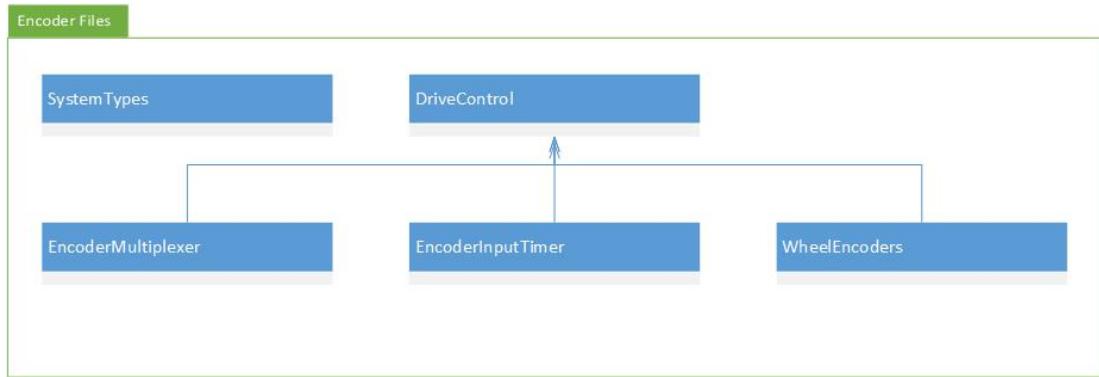


Figure 6.25: Sensor Controller Drive Control Firmware Diagram

6.4.1.2 Ultrasonic Sensors Handler

The UltrasonicSensorHandler source file follows a very similar structure to the DriveControl source file, except more hardware is required for this design due to it requiring Shift Registers for pulsing the Ultrasonic Sensors. The overall hierarchy of this particular handler is shown in Figure 6.26.

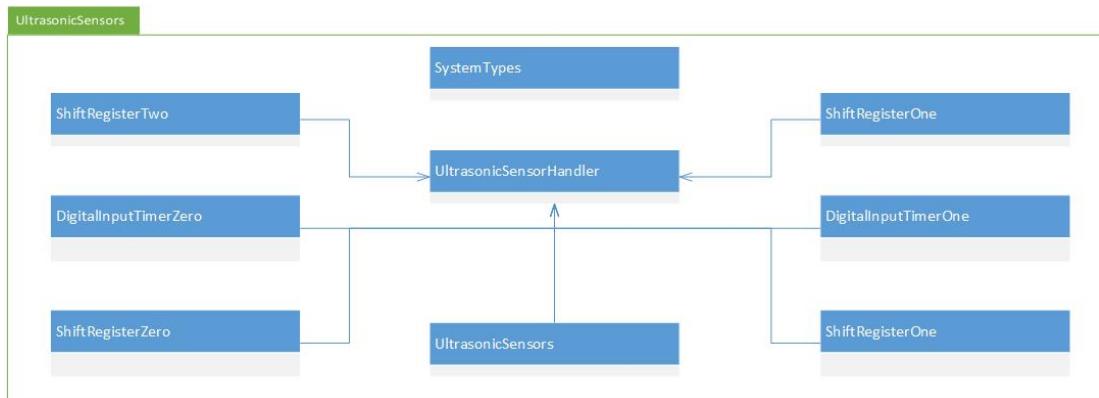


Figure 6.26: Sensor Controller Ultrasonic Sensors Handler Firmware Diagram

6.4.1.3 Infrared Sensors Handler

The last sensor handler developed for the sensor controller firmware was the InfraredSensorHandler. Its hierarchical structure is shown in Figure 6.27. It in itself is a unique system as it requires communication with the PCA9685 [80] Servo controller to move the infrared sensors. For this reason, this section of code had to have direct access to

Chapter 6. Electronic Design and Implementation

the I2CHandler to control the servos. The firmware, however, does follow the same structure as the previous handlers, using specific source file- InfraredSensors to convert and store all the related sensor values.

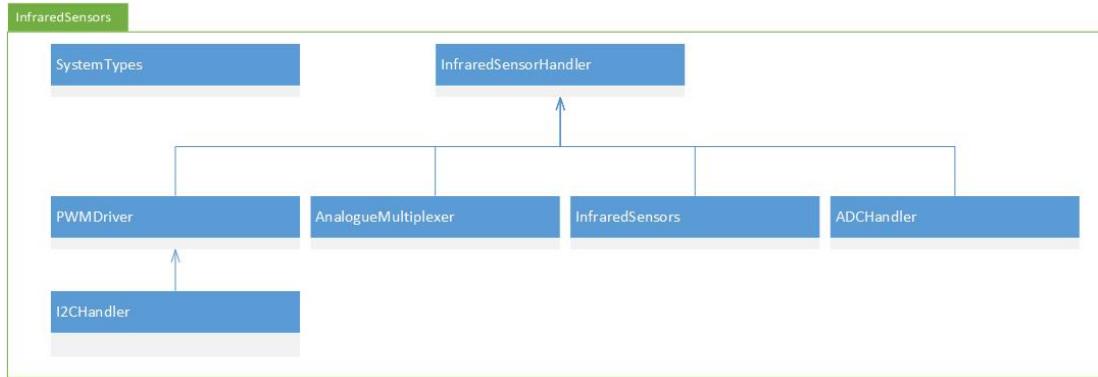


Figure 6.27: Sensor Controller Infrared Sensors Handler Firmware Diagram

Chapter 7

Discussion

7.1 Unforeseen circumstances and Project Delays

During the project, there was a variety of issues that arised, some of these issues took longer than others to fix, and some had a significant effect on the project. These issues were split into there relevant categories and are listed in the subsections below. Along with these specific setbacks, setbacks concerning COVID-19 are displayed in appendix appendix L.

7.1.1 Software

The software development of the project went fairly smoothly. However, there was some learning hurdles and documentation struggles. For example, the JavaFX [61] framework is not native in java and thus had to be installed; there was a lot of issues with IDE setup in that regard. The learning hurdle of learning JavaFX also came with issues due to it differing in structure from Swing [81]. When developing in FX issues was getting the Gluon scene builder [82] to work, it would seem it was designed for Apple products, and as such their Windows compatibility was unstable to some extent. There was also a magnitude of Python and Raspberry Pi problems, examples being a corrupted SD card in which the Raspberry Pi experienced voltage transients in the power line and corrupted the operating system. There was also IDE set up problems with Secure Socket Shell(SSH) in which the computer would reject the Pi as it would

detect it as an unsafe device.

7.1.2 Firmware

In terms of firmware development, there was numerous problems and setbacks with learning the Hardware Abstraction library. This was mostly due to errors in finding the right documentation and finding example code to base the system off. The HAL library has a very steep learning curve, and thus it was often difficult to find out what was breaking the implementations. As well as this one of the library functions does not operate as intended which created many issues with I2C, the library setup function for the I2C hardware did not operate as intended and gave a different address than what was given to the library function. This took a lot of time to solve and debug and thus, most of the time was taken away from firmware development. Along with this, there were issues with setting up System Workbench [83] in which it wouldn't detect the STM32 microcontrollers. The reason for this is that the batch of STM32 modules were clones and thus wouldn't be recognised by the IDE. Overall this caused significant setbacks with testing and ensuring the firmware was operational.

7.1.3 Hardware

In terms of hardware design, there were some setbacks in the design and manufacturing stages. One setback was converting some of the PCBs discussed in chapter 6 to PCBs that could be made by the university. The clearances tolerances lead to numerous manufacturing errors in which ground planes were disconnected and traces connected. This printing issue meant that out of the eight Printed Circuit Boards, only three were functional. There were also issues in which the Lab used did not have the correct power supplies and thus could not power the system. This caused a variety of testing problems since the system could not be powered on. The designs built on Veroboard were prone to mechanical failure in which the wires would come loose and took time to track down and fix these issues. As well as this there were issues with equipment. Specifically, the Dupont crimps could not be installed as the required crimping tool was not available; this meant female header cables were used and soldered on to the wires.

Chapter 7. Discussion

This connection then went on to cause more and more issues with bad connections. During the end of the project, a crimping tool was supplied; however, at that point, the sensors were already installed.

7.1.4 Mechanical

The mechanical aspect had significant setbacks. The university was not able to cut Medium Density Fibreboard(MDF) on the premises, and this meant alternatives methods had to be found to get the fibreboard cut to size. There were also multiple issues with the top panel shown in Figure H.18 in which the DXF file could not be read by the workshop, this meant alternative software had to be found to convert the file to the right format. As well as this the university did not have any wood for the top panel walls, and thus alternative sources had to be acquired. There were also problems in sourcing the correct sized screws for mounting the different CAD models. As for the 3D printing aspect, on the 28th of January there was a mechanical fault in the Wanhao I3 V2 Duplicator [58] in which the heated Nozzle malfunctioned and was severely damaged. This meant that there was a four week period in which no prototyping could be done due to the shipping restraints of the Prusa I3 Mk3s printer [59]. It also meant that time had to be taken out of the project to tune the new printer.

7.2 Improvements

Due to the multidisciplinary nature of the project, many different improvements can be made, and this was split up into four sections, each detailing specific improvements to the particular aspects of the system.

7.2.1 Software

Within the software systems, there are necessarily two applications, the Robot Controller, which is coded in Python and is running on the robot itself and the control interface made in Java. Both of which have areas of improvement.

7.2.1.1 CameraModule

The CameraModule on both software applications could use improvement, on the python side the code uses a modified library [84] and as such is more complex for what is needed. On the JavaFX side, the WebView JavaFX object is unable to render the HTTPS page generated by the modified library as such it cant be displayed in the controller and has to be displayed on an up to date internet browser. This in itself is a significant area of improvement as it takes away the functionality of the graphical user interface and adds unwanted complexity to the design. The reason for this problem is that the modified library creates a stream inside a webpage and as such the Java Side has to render the webpage and load the JavaScript for the stream to then render the stream, this is not optimised enough for this specific implementation. A solution for this would be to investigate how UDP can be used to transfer the video rather than through an HTTP web server as this would allow better throughput as no acknowledgements would be sent compared to TCP.

7.2.1.2 Java Model View Presenter Update Procedure

Due to using the observer pattern, to both refresh the view and check for model changes on the client-side and appropriately update the robot and vice versa. The efficiency of the Java control interface has gone down due to having to update entire sections of the view rather than specific fields. The issue with this is that the program will begin to slow down and become unusable as more functionality is added. This is a significant problem. One way of fixing this is creating an implementation of an observer like pattern to solve this one in which more information can be passed to the observer allowing the view to only updated when needed.

7.2.2 Firmware

In terms of firmware development, there are many ways in which the code could be improved. Due to the circumstances explained previously, the code did not undergo adequate testing and as such, can be improved by following a strict testing strategy using methods outlined in the appendix. Aside from testing the code can be improved by get-

Chapter 7. Discussion

ting the I2CHandler operational and implementing the missing I2C Slave transmission code so that the Python Robot Controller can use the onboard values.

7.2.3 Hardware

In terms of hardware, many aspects can be improved. Some can be considered improvements while others can be considered as improvements if the system requirements would change.

The system currently uses all Through Hole components, although this is good for beginners as they are easy to assemble and solder, it does create issues in which some components aren't through-hole components. A great example being the Adafruit PWM Driver [85], It contains the PCA9685 Surface mount driver chip [80], if the entire system were to move to surface mount design, the whole PCB sizes could be reduced significantly and may allow for smaller more compact designs. The problem with this, however, is that schools could no longer assemble the robot and thus detracts from the original specifications of the robot. Various hardware designs for the PCBS could also be significantly improved, and the corresponding improvements are listed below.

7.2.3.1 Raspberry Pi HAT PCB

Due to manufacturing constraints with the university Lab and tolerances, the PCBs were designed such that they could be built within the lab. This created uneven trace widths, in which the size of the widths drastically changes throughout the design, it also meant that the vias have to be bigger than needed to connect both sides of the PCB. Again this was due to workshop limitations in drill size and PCB manufacturing. To improve this design, the trace widths could be minimised significantly if JLCPCB [86] tolerances were used. This, in turn, would improve the overall design and if done right could reduce coupling between layers and traces.

7.2.3.2 Sensor Controller PCB

The Sensor Controller PCB can be vastly improved, in the design manufactured there was a layout which disconnected the low pass passive filter from the analogue line.

Chapter 7. Discussion

This meant that the signal could not be filtered and as such is a simple improvement to the design. The system also uses a passive filter rather than a reactive filter, which is an area of research that could be undertaken to improve the filtering efficiency of the design.

7.2.3.3 Power Converter PCB

The Power Converter PCB also suffers the same trace-width, and vias holes discussed previously. As before this can again be improved. The design itself suffers from the stacking effect. If a fuse were to blow in either of the secondary voltage lines on the board, the entire stack would have to take apart in order to replace it. With that said, a real improvement to the design would be component rearrangement, as this would greatly help with the maintainability aspect of the design while keeping surge protection in the system.

7.2.3.4 Primary Power Distribution PCB

The Primary Power Distribution PCB can be significantly improved in various aspects. The first problem with the design is that the grounds for each component are connected, meaning that a fault in the circuit can pass through into the different voltage systems. So ground isolation between the different power lines would be a significant improvement to the design.

More straightforward improvements to the design also exist, female Dupont headers were used for connecting the different sensors, although this seems to be an adequate solution. It is by far not the best, a better solution to this would be to use a Japanese Solderless Terminal(JST), as these would allow each sensor to be connected directly into the board and allows a clearer understanding of what wires belong to each sensor. The PCB also currently uses Screw Terminals for the main power lines, this creates an issue in which the board is difficult to install due to the angle that's required to tighten the terminals, these terminals could get replaced by a push on terminals instead.

Chapter 7. Discussion

Finally Power LEDS could also be added to the circuit to show that each power network is connected, giving immediate notification of whether the power is being distributed in this section.

7.2.3.5 Secondary Power Distribution PCB

The Secondary Power Distribution PCB also has the problem discussed in 7.2.3.4 in which the system uses screw terminals at awkward angles, making the system hard to work with. It also could be improved with the power indicator LED discussed earlier. As well as this Dupont Female Headers could also be added for easier add on expansion in which breadboards would be used with jumper wires.

7.2.3.6 Inertial Measurement Unit External Hardware PCB

The last PCB to get improvements would be the Inertial Measurement Unit External Hardware PCB, and now this is a very basic PCB solely designed to fix the IMU in place. This could be replaced by a surface mount PCB which contains mostly through-hole components except the MPU6050 [44]. This would minimise complexity in that fewer components will be required to build the PCB, and it could be easier placed inside the robot. This PCB like the sensor controller and Raspberry Pi HAT also have communication and power pins, again Dupont Male headers were used, a better solution would be the JST standard solderless crimps in which the connections would click into place allowing for safer assembly and less chance of wires getting pulled out mid-operation.

7.2.4 Mechanical

The mechanical part of this project mostly consists of 3D printing Computer generated Models. Now, this can be improved by using better printers as the models would come out more like there computer-generated counterparts. In some cases, the design uses nuts and bolts as well as 3D printed parts; this increases the complexity of the assembly so research could be done into investigation better designs. The designs also didn't undergo any stress testing or structural analysis. So possible improvements to

Chapter 7. Discussion

that aspect could be researched the design and making them more durable as they have to go through some somewhat harsh conditions within the Rampaging Assault Course shown in Figure 2.8. As well as this, the design contains many parts which might be able to minimise to save manufacturing complexity costs.

There was a recurring issue with ultrasonic sensors in which the HC-SR04 [87] could not be mounted due to limitations in the available bolts from the University. This meant that they could not be mounted in the same way as the Infrared sensors. So this is a definite area for improvement, which would involve acquiring smaller nuts and bolts(M1 size) and modifying the corresponding Ultrasonic Sensor Holder CAD models shown in Appendix H.9. There were also issues with the manufacturing of the encoder mounts(Appendix H.5, in which the flywheel encoder mount(Figure H.17 front chamfered screw hole couldn't be used post assembly of the robot due to the angle required to screw it in being impossible to achieve.

7.3 Further Work

The primary focus of the further work should be finalising the overall software and firmware implemented by first focusing on the I2C issue with the sensor controller not being able to transmit to the Raspberry Pi. After this is completed, the next stage would be to implement the PID control solution for allowing the robot to have full navigational control rather than open-loop control. Once this is completed, the Graphical User Interface should be expanded further to incorporate a better Graphical User Interface.

Although many improvements can be made to the project, there are many more potential avenues in which the project can be continued and expanded. These include building more systems for the Rampaging Chariots competitions and more academic research applications.

7.3.1 Rampaging Chariots Challenges

The robot can be further improved by making it able to do other competitions such as Sumo Challenge, Two A Side Football. This would require significant programming allowing the robot to compete in these challenges.

7.3.1.1 Sumo Challenge

The Rampaging Chariots Sumo Challenge consists of two robots joined together via a connected rope. This challenge requires the robots responding to wheel traction on the ground and apply the right amount of torque. This in itself is an exciting challenge, determining how much torque is required and applying enough to pull the other robot without losing traction and potentially sliding all over the place.

7.3.1.2 Two A Side Football

The Two A Side Football challenge consists of four robots playing football, the objective being to score as many goals as possible. Much like regular football coordination between the two robots is essential in winning the challenge. To make an autonomous robot to play by itself is already challenging, to make it communicate with another robot and play football in a team will take much work. This in itself, could be a perfect extension to the project and could contribute to robotic research in general as well.

7.3.1.3 Application Programming Interface Creation and Hosting

The Rampaging Chariots provide a construction manual for their robots. Although this is useful, as time moves on, everything is becoming digitalised and considering how many components the robot has both software and hardware. It would be increasingly difficult to represent the exact functionality of the robot in one of these manuals. One way the project could be extended would be to develop an interactive website which details all the functions and provides extension examples of how the functions can be used. This would introduce aspects of software engineering, such as using an API and being able to use libraries - a valuable skill to know.

7.3.2 Exploded View Video Rendering

In the project a variety of exploded view videos were made, the end pictures are shown in appendix H. These videos could be embedded inside the control GUI, and this would allow the direct indication of how to assemble all the components and therefore make the mechanical assembly easier for the young people building the robot.

7.3.2.1 Add-on Kits

In all aspects of the robot, software, mechanical and electronic, the design is such to be highly modularized, allowing for re-use of components and pieces of code. In terms of hardware, multiple PCBs are designed with modularity in mind such that there are other communication and power headers on most PCBs. This could allow the guild to add a robotic arm, more PCBs for different purposes etc.

7.3.3 Computer Vision

An exciting extension to the project would be using the Raspberry Pi Camera for Computer Vision, Open-CV [88] is widely adopted on the Pi, and thus there is plenty of examples. Edge detection could be used to make the robot solve mazes and essentially map its environment. Facial recognition could also be used, which would allow the robot to interact with humans and would be an interesting example for the Rampaging Chariots to show off to young people to get them interested in engineering.

7.3.4 Swarm Robotics

One interesting field in robotics is that of Swarm Robotics, and its primary focus is the control of multiple small robots in an extensive system to achieve a set task. Usually mapping an environment. Due to the variety of sensors, the robot has this makes it a perfect candidate for this in that, many of these robots could be used to map an environment. They also have the ability to communicate to each other with different mediums due to the many interfaces the Raspberry Pi offers such as WiFi and Bluetooth. If for example, they weren't enough, then the I2C network in the robot could be used to add more communication modules to the robot.

7.3.5 Robot Arm Add-on

One of the upgrades that add a bunch of functionality to the robot is the inclusion of a six degrees of freedom robot arm. This would allow us to interact with its environment and modify it. A whole new aspect could be added to the Rampaging Chariots games and their mission goal. By adding a robotic arm to the design we essentially involve a whole new aspect to engineering within the project, it would allow the Rampaging Chariots to give an overview of manipulators, kinematics, 3D printing and properly explore all aspects of robotics not just Mobile Robotics.

7.3.6 Suspension System

One problem with the encoders on each wheel is that for accurate speed estimations all four wheels should have contact with the ground. Due to the construction of the robot, this isn't possible as the wheels are directly mounted on to the frame. This could be majorly improved by investigating in ways to implement suspension like systems into the robot. One way to make this feasible is to investigate 3D printed suspension and implementing into the design.

7.3.7 Internal Robotic System Add-ons

Inside the robot, there is excellent potential for modularity due to the way each hardware design is developed, and they were all developed with expandability in mind. Thus allowing PCBs to be added to the individual stacks inside the robot.

7.3.7.1 Power Converter Measurement PCB Addon

A great example of expandability within the robot is the power converter PCB in which there are Dupont headers for current and voltage measurement. One such way to expand the project would be to add a current and voltage measurement PCB to the design as this would allow the measurement of each power line and analysis could be done to improve the systems operating efficiency. Even if that were not possible, it unleashes more possible applications of the system, in that the design could be made to act differently depending on power levels.

7.3.7.2 Battery Monitoring PCB Addon

Along with monitoring all the different networks, a PCB could be designed to monitor the battery levels and predict when the battery would fail and indicate students issues with the battery. This would be a simple I2C device which would be simply attached to the Battery terminals.

7.3.8 Transition to Fusion 360

The project used both Autodesk Inventor [54] and Eagle [68], at the time this seemed like the reasonable choice. However upon gaining more experience with Eagle it was found that Fusion 360 [89] would be the ideal candidate, this was due to it being able to render the PCBs developed in chapter 6 and be able to see what they would look like inside a design. This would allow a complete virtual model with all components to be simulated.

Chapter 8

Conclusion

The skills gap within the technology sector clearly shows that more methods for engaging young people in technology and engineering is required. The Rampaging Chariots Guild is an effective way to engage young people, and the need for an Autonomous Robot upgrade kit is clear.

This report detailed the development and construction of an autonomous upgrade kit for the Rampaging Chariots Radio controlled. Although not all requirements were met, the project was ambitious and required a tremendous amount of work to complete. With that said, the project successfully finalised all hardware aspects of the autonomous upgrade kit along with all construction aspects. As for the software aspect, the project successfully laid the groundwork for the completion of all objectives. As such the project has made a significant impact on developing an Autonomous Robot for the Rampaging Chariots Guild.

Bibliography

- [1] P. Bennett and R. Hill, “Technology - engineering - fun - rampaging chariots,” 2020. [Online]. Available: <http://www.rampagingchariots.org.uk/index.php>
- [2] “sojourner mars rover and beatty robotics,” 2020. [Online]. Available: <https://beatty-robotics.com/sojourner-mars-rover/>
- [3] “husky ugv - outdoor field research robot by clearpath,” 2020. [Online]. Available: <https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/>
- [4] “pioneer 3-dx,” 2020. [Online]. Available: <https://www.generationrobots.com/media/Pioneer3DX-P3DX-RevA.pdf>
- [5] “Boost converters,” 2020. [Online]. Available: <https://learnabout-electronics.org/PSU/psu32.php>
- [6] “How brushed motors work,” 2020. [Online]. Available: <https://www.roboteq.com/index.php/87-products/417-how-brushed-motors-work>
- [7] G. Dobbie, “Motorcontrol,” 2020.
- [8] S. W. Smith, *The scientist and engineer’s guide to digital signal processing*, 1st ed. California Technical Pub., 1997.
- [9] M. Microprocessors, S. MCUs, and S. MCUs, “Stm32f103c6 - stmicroelectronics,” 2020. [Online]. Available: https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32f1-series/stm32f103/stm32f103c6.html

Bibliography

- [10] “Stm32 bluepill pinout,” 2020. [Online]. Available: <https://idyl.io/wp-content/uploads/2018/07/stm32f103-pinout-diagram.png>
- [11] “As5600 data sheet,” 2020. [Online]. Available: https://ams.com/documents/20143/36005/AS5600_DS000365_5-00.pdf/649ee61c-8f9a-20df-9e10-43173a3eb323
- [12] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*, 2nd ed. MIT Press, 2011.
- [13] G. Dobbie, “Localisation,” 2020.
- [14] “i2c learn.sparkfun.com,” 2020. [Online]. Available: <https://learn.sparkfun.com/tutorials/i2c/all>
- [15] “serial peripheral interface (spi) learn.sparkfun.com,” 2020. [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all>
- [16] “serial communication learn.sparkfun.com,” 2020. [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-communication/all>
- [17] “creational design patterns,” 2020. [Online]. Available: <https://www.baeldung.com/creational-design-patterns>
- [18] “Observer design pattern,” 2020. [Online]. Available: https://sourcemaking.com/design_patterns/observer
- [19] “Adapter pattern,” 2020. [Online]. Available: https://sourcemaking.com/design_patterns/adapter
- [20] “Private class data,” 2020. [Online]. Available: https://sourcemaking.com/design_patterns/private_class_data
- [21] “mvc design pattern geeksforgeeks,” 2020. [Online]. Available: <https://www.geeksforgeeks.org/mvc-design-pattern/>

Bibliography

- [22] “model view presenter (mvp) design pattern and data binding,” 2020. [Online]. Available: https://www.c-sharpcorner.com/uploadfile/john_charles/model-view-presenter-mvp-design-pattern-and-data-binding/
- [23] S. Bi-Directional, “Sparkfun logic level converter - bi-directional - bob-12009 - sparkfun electronics,” 2020. [Online]. Available: https://www.sparkfun.com/products/12009?_ga=2.55786603.1791737581.1586020195-1822927537.1544008325
- [24] G. Dobbie, “Motors,” 2020.
- [25] “raspberry pi 3 model b+,” 2020. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>
- [26] H. Pozniak, “The skills crisis in engineering,” 2020. [Online]. Available: <https://www.telegraph.co.uk/education/stem-awards/energy/the-great-uk-engineering-shortage/>
- [27] 2020. [Online]. Available: <https://uk.leonardocompany.com/en/about-us/uk-locations/edinburgh>
- [28] Oracle, *Java SDK*. Oracle, 2020.
- [29] P. S. Foundation, *Python SDK*. Python Software Foundation, 2020.
- [30] “What is power electronics?” 2020. [Online]. Available: <http://www powerelectronics.ac.uk/what-is-power-electronics.aspx>
- [31] “Buck converters,” 2020. [Online]. Available: <https://learnabout-electronics.org/PSU/psu31.php>
- [32] “How servo motors work,” 2020. [Online]. Available: <https://www.jameco.com/Jameco/workshop/howitworks/how-servo-motors-work.html>
- [33] “Sg90 micro servo datasheet,” 2020. [Online]. Available: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

Bibliography

- [34] “All about stepper motors,” 2020. [Online]. Available: <https://learn.adafruit.com/all-about-stepper-motors/what-is-a-stepper-motor>
- [35] “Stepper motors - introduction and working principle,” 2020. [Online]. Available: <https://embedjournal.com/stepper-motors-introduction-and-working-principle/>
- [36] G. David, “lecture 10 pid controllers,” 2020.
- [37] “pid controller explained,” 2020. [Online]. Available: <https://pidexplained.com/pid-controller-explained/>
- [38] H. Garden, HowStuffWorks, Tech, Electronics, and Electronics, “How microcontrollers work,” 2020. [Online]. Available: <https://electronics.howstuffworks.com/microcontroller1.htm>
- [39] “frontpage raspbian,” 2020. [Online]. Available: <https://www.raspbian.org/>
- [40] S. Monk, *Programming the Raspberry Pi3: Getting Started with Python, Second Edition*, 2nd ed. McGraw-Hill Professional, 2013.
- [41] L. Petropoulakis, “Robotic sensors,” 2020.
- [42] “Inertial measurement units,” 2020. [Online]. Available: <https://www.analog.com/en/products/sensors-mems/inertial-measurement-units.html>
- [43] “mpu6050 pinout configuration features arduino interfacing and datashee,” 2020. [Online]. Available: <https://components101.com/sensors/mpu6050-module>
- [44] “Mpu6050 data sheet,” 2020. [Online]. Available: <https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/>
- [45] “Hc-sr04 first data sheet,” 2020. [Online]. Available: <https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf>
- [46] “Hc-sr04 second data sheet,” 2020. [Online]. Available: <https://www.electroschematics.com/wp-content/uploads/2013/07/HC-SR04-datasheet-version-2.pdf>

Bibliography

- [47] “Gpy2y0a21yk0f sharp ir sensor data sheet,” 2020. [Online]. Available: <https://www.pololu.com/file/0J85/gp2y0a21yk0f.pdf>
- [48] G. Dobbie, “Kinematics,” 2020.
- [49] L. Petropoulakis, “Robotics path planning,” 2020.
- [50] K. . ROSS., *Computer networking: a top-down approach 6th ed*, 6th ed. PEARSON, 2013.
- [51] “what is http (hypertext transfer protocol)?” 2020. [Online]. Available: <https://www.computerhope.com/jargon/h/http.htm>
- [52] “design patterns and refactoring,” 2020. [Online]. Available: https://sourcemaking.com/design_patterns
- [53] “mvc: model, view, controller,” 2020. [Online]. Available: <https://www.codecademy.com/articles/mvc>
- [54] Autodesk, *Autodesk Inventor*, Autodesk, Ed. Autodesk, 2020.
- [55] “inventor mechanical design and 3d cad software autodesk,” 2020. [Online]. Available: <https://www.autodesk.com/products/inventor/overview>
- [56] “keyestudio i2c lcd 20x4 2004 lcd display module for arduino uno r3 mega 2560 r3 white letters on blue backligh,” 2020. [Online]. Available: <https://www.keyestudio.com/free-shipping-keyestudio-i2c-lcd-20x4-2004-lcd-display-module-for-arduino-uno-r3-mega-2560-r.html>
- [57] “Raspberry pi homepage,” 2020. [Online]. Available: <https://www.raspberrypi.org/>
- [58] 2020. [Online]. Available: <https://www.wanhao3dprinter.com/Unboxin>ShowArticle.asp?ArticleID=70>
- [59] J. Prusa, “Original prusa i3 mk3s - prusa3d - 3d printers from josef prusa,” 2020. [Online]. Available: <https://www.prusa3d.com/original-prusa-i3-mk3/>

Bibliography

- [60] “java software oracle,” 2020. [Online]. Available: <https://www.oracle.com/java/>
- [61] Oracle, *JavaFX*, Oracle, Ed. Oracle, 2020.
- [62] “JavaFx download page,” 2020. [Online]. Available: <https://openjfx.io/>
- [63] “setting up a raspberry pi 3 as an access point,” 2020. [Online]. Available: <https://learn.sparkfun.com/tutorials/setting-up-a-raspberry-pi-3-as-an-access-point/all>
- [64] “welcome to python.org,” 2020. [Online]. Available: <https://www.python.org/>
- [65] D. Jones, “waveform80/pistreaming,” 2020. [Online]. Available: <https://github.com/waveform80/pistreaming>
- [66] “Simplepid python library,” 2020. [Online]. Available: <https://pypi.org/project/simple-pid/>
- [67] “Stm32f1 hal driver manual,” 2020. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/user_manual/72/52/cc/53/05/e3/4c/98/DM00154093.pdf/files/DM00154093.pdf/jcr:content/translations/en.DM00154093.pdf
- [68] Autodesk, *Eagle*. Autodesk, 2020.
- [69] “Lm2596 dc-dc buck converter breakout board,” 2020. [Online]. Available: <https://www.aliexpress.com/item/33023128916.html?spm=a2g0s.9042311.0.0.27424c4dPMzjBE>
- [70] “Lm2596 switching regulator datasheet,” 2020. [Online]. Available: <http://www.ti.com/lit/ds/symlink/lm2596.pdf>
- [71] “1n5824 datasheet,” 2020. [Online]. Available: http://www.farnell.com/datasheets/1689683.pdf?_ga=2.222837403.1614726073.1584274832-836347807.1577643715
- [72] “Rs power inductors data sheet,” 2020. [Online]. Available: <https://docs.rs-online.com/5b58/0900766b81505722.pdf>

Bibliography

- [73] “Stm32f103ct6 data sheet,” 2020. [Online]. Available: <https://www.st.com/resource/en/datasheet/stm32f103c6.pdf>
- [74] “1n4148 small signal fast switching diode datasheet,” 2020. [Online]. Available: <https://www.vishay.com/docs/81857/1n4148.pdf>
- [75] “lmx58-n low-power, dual-operational amplifiers,” 2020. [Online]. Available: <http://www.ti.com/lit/ds/symlink/lm158-n.pdf>
- [76] “Sn74hc595n shift register data sheet,” 2020. [Online]. Available: <http://www.ti.com/lit/ds/symlink/sn74hc595.pdf>
- [77] “Dg408dj analogue multiplexer,” 2020. [Online]. Available: <https://www.vishay.com/docs/70062/dg408.pdf>
- [78] “Sn74hc251n digital mutliplexer data sheet,” 2020. [Online]. Available: <http://www.ti.com/lit/ds/symlink/sn74hc251.pdf>
- [79] “Stm32f103ct6 user manual,” 2020. [Online]. Available: https://www.st.com/resource/en/reference_manual/cd00171190-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-application-manual.pdf
- [80] “Pca9685 data sheet,” 2020. [Online]. Available: <https://www.nxp.com/docs/en/data-sheet/PCA9685.pdf>
- [81] “getting started with swing,” 2020. [Online]. Available: <https://docs.oracle.com/javase/tutorial/uiswing/start/about.html>
- [82] “scene builder,” 2020. [Online]. Available: <https://gluonhq.com/products/scene-builder/>
- [83] A. Tools, *System Workbench for STM32*, A. Tools, Ed. AC6 Tools, 2020.
- [84] “low latency and high fps camera stream with a raspberry pi,” 2020. [Online]. Available: <https://nerdhut.de/2018/12/17/low-latency-and-high-fps-camera-stream-with-raspberry-pi/>

Bibliography

- [85] A. Industries, “Adafruit 16-channel 12-bit pwm/servo driver - i2c interface,” 2020. [Online]. Available: <https://www.adafruit.com/product/815>
- [86] “Jlpcb manufacturing tolerances,” 2020. [Online]. Available: <https://jlpcb.com/capabilities/Capabilities>
- [87] “Hc-sr04 data sheet,” 2020. [Online]. Available: <https://www.electroschematics.com/hc-sr04-datasheet/>
- [88] “Opencv,” 2020. [Online]. Available: <https://opencv.org/>
- [89] “cloud powered 3d cad software for product design fusion 360,” 2020. [Online]. Available: <https://www.autodesk.co.uk/products/fusion-360/overview>
- [90] 2020. [Online]. Available: <https://junit.org/junit5/>
- [91] 2020. [Online]. Available: <https://wiki.python.org/moin/PyUnit>
- [92] Jetbrains, *InteliJI 2019 Edition*, Jetbrains, Ed. JetBrains, 2019.
- [93] ——, *Pycharm 2019 Edition*, Jetbrains, Ed. JetBrains, 2019.
- [94] “sw4stm32 stmicroelectronics,” 2020. [Online]. Available: <https://www.st.com/en/development-tools/sw4stm32.html>
- [95] “easy to use 3d printing software 2020,” 2020. [Online]. Available: <https://ultimaker.com/software/ultimaker-cura>
- [96] J. Prusa, “Prusaslicer - prusa3d.com - 3d printers by josef prusa,” 2020. [Online]. Available: <https://www.prusa3d.com/prusaslicer/>
- [97] “eagle pcb design software autodesk,” 2020. [Online]. Available: <https://www.autodesk.com/products/eagle/overview>
- [98] “epilog laser engraving and cutting machine systems - etching, cutting and marking systems,” 2020. [Online]. Available: <https://www.epiloglaser.com/>
- [99] 2020. [Online]. Available: <https://www.3dprima.com/filaments/pla/primavalue/>

Bibliography

- [100] J. Prusa, “Prusament pla,” 2020. [Online]. Available: <https://prusament.com/>
- [101] ——, “Prusa pla,” 2020. [Online]. Available: <https://shop.prusa3d.com/en/21-pla>

Appendix A

System Testing

In terms of testing, different strategies were used for various aspects of the robot. However, due to a variety of issues that arose with the project, not all testing strategies were completed. As such, an overview of the types of processes that would be used is given.

A.1 Mechanical Testing

For mechanical testing, Inventor assemblies and exploded assemblies were used to ensure all parts could fit together. Along with this, a rapid prototyping approach was undertaken to refine each part to ensure a good fit. Given more time, material analysis could be applied to analyse how much material is needed and how the sizes of each component can be minimised. This will reduce not only manufacturing time but reduce the cost of the 3D printed parts. Material analysis could also be used to determine the stress loads of each section and improve overall stability and rigidity of the robot.

A.2 Software Testing

In terms of software testing, multiple methods can be used such as Unit Testing, Black-box and Whitebox testing for each of the model elements of the Java application and some aspects of the python controller. However, due to time constraints, this could not be achieved and the testing strategy involved user testing through the Graphical User

Appendix A. System Testing

Interface. In terms of Unit testing, for the Java side of the software JUnit [90] would be used and a dummy version of the Robot Python Controller would be made, in order to make easier tests for the user interface. PyUnit [91] would also be applied to the Python RobotController.

A.3 Firmware Testing

Due to circumstances outwith the control of the project, extensive testing could not be applied to the firmware of the project. This was partially due to not having access to the equipment to test it properly. However, the strategy that would have been employed would be to use an oscilloscope and measure each of the different sensor inputs and compare this with the raw values on the sensor controller. After confirmation of the values being correct, the conversion functions would be tested by setting up distance tests and comparing the calculated distance value to the actual value. Along with this, a Logic Analyzer would be used to debug the I2C communication on the I2C network and locate precisely what is being sent on the network.

A.4 Hardware Testing

Finally, the approach taken to testing the hardware was to use probe tests and monitoring the power consumption of the system. Short circuit probe tests were applied to each PCB and Veroboard solution to ensure that the board's traces are connected correctly. In terms of the I2C network, some rudimentary Logic Analyzer tests were performed to monitor the I2C bus, and loose connections were found and replaced. If the setbacks discussed in appendix L did not happen, an Oscilloscope and Digital Multimeter would be used to ensure the intended functionality of the PCBs. For testing the Power Converter boards, a dynamic load test would be used to monitor how the regulation and conversion stages reacted to increased switching noise in the circuitry. The hardware would also be tested with a full system stress test, in which everything would run at maximum levels. The Raspberry Pi running stress tester software and all motors rotating against a high torque load as this would test all short circuit protection

Appendix A. System Testing

mechanisms and transient smoothing capacitors.

Appendix B

User Guide

Regarding the operation of the current system, there is a lot that needs to be checked to ensure the safety of each PCB due to the fact that the currently implemented system is in the prototype stage.

B.1 System Setup

This part of the user guide is focused on making sure the hardware is connected correctly, using Steps 1-2 described in appendix B.2. Turn on the system and using a multimeter probe all voltage points across the system to ensure the correct power line voltage.

B.2 Control Interface Instructions

Regarding the initialisation of the system, the process is rather simple. Following the steps listed below the system can be initialised in prototype mode.

- Step 1: Ensure all power lines are not short-circuited by using a multimeter on short circuit detection mode, this can be done easily by plugging one probe into the ground and the other into the positive voltage currently being tested.
- Step 2: Once step 2 is completed, it is recommended to use a current limiting power supply to ensure the safety of the system. Connect the power supply(set

Appendix B. User Guide

to 18V and 0.5A) to the 18V power line, and press the power button located on the manual control panel.

- Step 3: Once step 2 is completed, using an appropriate laptop/PC with Inteliji [92] and appropriate java libraries(refer to table C.1 for more information), and Pycharm [93] installed connect to the robot hot-spot through the laptop/PCs WiFi connection.
- Step 4: Upon connection to the WiFi hot-spot open the AutoChariot PyCharm project and run it through the SSH IDE Interpreter
- Step 5: using Inteliji select the correct JavaFX view from the Main Java file and run the application, the different sockets should connect and allow control over the various systems incorporated in the robot.
- Step 6: At this point, the system should be operational and controllable through the java control interface, increase the current limiter to 3A, as this is the maximum draw for the system's motors. This should allow the robot to be fully used.

Appendix C

Maintenance Guide

In terms of maintenance and expandability, the system was designed to allow for easy modification and inspection of all subsystems. In order to represent these features, the maintenance guide is split into four sections, Mechanical, Software, Firmware and finally hardware.

C.1 Software

In terms of software maintenance, the models are very decoupled to allow the design to be as flexible as possible. For any additions to the software, the process described in the sections below should be used.

C.1.1 Python Robot Controller

Since the Python Robot Controller was not entirely finished there is a lot that may need to change. However, with that said, the groundwork of the system was built, and thus, the relevant features are there for expansion. For any additions or areas of modification, the general information listed below about the package structure should be followed to allow the system to be modified.

C.1.1.1 Adding/modifying High Level Communications

For maintaining and adding additional high-level communications, the implementations should follow the general structure outlined within the Communications Package. That is they should be split into packagers, transporters and decoders. The Transporter should support the low-level communications such as putting messages on the "wire". The Packagers are responsible for updating the relevant sections in the model implementing the observer pattern. Finally, the decoder is responsible for dissecting the messages into the relevant variables, as well as creating new messages through a commandMaker class. Not only this if the communication protocol requires an internet-protocol port it must align with the already utilised ports detailed in appendix D.

C.1.1.2 Implementing other Controllers

Regarding adding more controllers that control every aspect of the robot such as a predefined path should be written in the Controllers package. Making sure to use of all the other packages for navigation and sensor readings. If the user interface is required so that the sensor values can be read whilst on the move, the different packagers for localisation and control should also be instantiated in the implemented controller class. Making sure to start threads before entering the main functionality of the implemented controller algorithm.

C.1.1.3 Manual Control Panel Modification

For writing to the LCD Screen on the manual control interface, the appropriate class must be instantiated and used. This should be done through the IManualControlPanel interface. This is due to the required thread locks and ensuring the screen does what is expected. Finally, if any button functionality is required, such as shutting down connections or general algorithms, these functions must be added in the ManualControlPanel class and passed to the relevant button using the ButtonFactory class.

C.1.1.4 Adding new Structure Patterns

For any structural design patterns that need to be added, that is general enough for the full system to use, it is recommended that they are placed inside the Miscellaneous package under their own sub-package. This allows the system to maintain different structures in such a way that the whole system can use them in a similar fashion to the way the observer implementation was used.

C.1.1.5 Movement

In terms of movement modification and additional control techniques such as PID controllers and Kalman filters, it is recommended that the code is implemented inside the Movement Package. It is also recommended that it implements an interface allowing the functionality to be swapped in and out quickly.

C.1.1.6 Sensors Modification

The addition of extra sensors can make for a rather complex structural change. To minimise this, the sensor being implemented should be categorised as either exteroceptive or proprioceptive and placed in the relevant sub-package within the Sensors package. Once done, the implemented sensor class should extend the observable class to allow easy observation of the class and use within the system.

C.1.1.7 Adding additional I2C Devices

The final aspect of being covered in the maintenance guide regarding the Python Robot Controller is the addition of other I2C devices and how this implementation must go to keep the rest of the functionality operational. Due to the already implemented I2C devices the multiple threads that they run on, any implemented device using I2C must use the busio.I2C class to interact with the I2C bus as well as using the appropriate accessor methods: *try_lock()*: and *unlock()*.

C.1.2 Java Control Interface

In terms of the Java Control Interface, many areas should remain the structure they are. The way the system can be maintained is by following the general procedure listed below:

C.1.2.1 Adding more Graphical User Interfaces

To maintain the structure of the control interface, any subsequent modifications and/or addons must adhere to the Model View Presenter Pattern. This can be done quite quickly by using the central packages Controller, Model and finally View. To achieve this goal, it is helpful to use the Java Observer, and Observable classes as this allows the system to remain decoupled.

C.1.2.2 Adding Communications

To add more high-level communications, the communication packages should be used and maintained following the same structures, Decoder, Packager and Transporter classes.

C.2 Firmware

In terms of firmware, there is only one programmable microcontroller in the system, and the maintenance of this can be directly related to the structures discussed in chapter 6. The relevant maintenance guides were split into different categories and listed below. SystemWorkbench [94] is also required to modify the firmware.

C.2.1 Adding Ultrasonic Sensors

For adding Electronic Sensors to the firmware, a variety of files need to be modified. The hardware process listed in the next section also needs to be followed. Firstly the new ultrasonic sensor added has to be HC-SR04, and the numbering parameters inside UltrasonicSensors Header and source file also need to be changed. The code is dynamic

Appendix C. Maintenance Guide

and should allow for easy changing. The architecture to be followed for adding more sensors is shown in 6.26

C.2.2 Adding Infrared or Analogue Sensors

If adding more Infrared Sensors are required, the structure outlined earlier in chapter 6 should be explicitly followed, making use of the source files 6.27. Inside InfraredSensors the parameters will need to be changed and once changed the for loops inside the application loop will account for this new sensor.

If a new type of analogue sensor is adding a separate handler and accompanying data storage source and header file will be required, These should make sue of the ADCHandler and AalogueMultiplexer file to change and acquire their data and set up the sensor.

C.2.3 Adding Encoders

The addition of encoders again involves changing some parameter within the different source and header files associated with encoders. The structure that needs to be modified is shown in Figure 6.25. There is also a hardware aspect that is required for this modification and is discussed in the next section.

C.2.4 Adding Digital Sensors

If other types of sensors are to be added to the SensorController, the digital multiplexers are to be used, and Timer 2 and Tmer3 are to be used as input capture timers. However, care must be taken into these modifications as this will slow down the refresh rate of the Ultrasonic sensors as they rely on these timers. In any circumstance, a separate sensor handler is required.

C.2.5 Maintaining Communications

The microcontrollers communications consist of both a UARThandler and an I2CHandler, modification of any aspect of communication should be done in these files to allow continued decoupling of the designs.

C.2.6 Changing Main Application Control

The final aspect of any firmware modification is the main source file. This file, in particular, is responsible for executing and managing all handlers in the firmware. It consists of a setup and loop method. This is where any additional handler should be set up and executed.

C.3 Hardware

The system consisted of six different PCBs; each of these PCBs have ways of ensuring their operation, and this process is described in the relevant sections listed below:

C.3.1 Power Converter Board

To ensure the correct operation of the power converter board, it is recommended to use an 18V power supply and check the voltages of each power converter stage using the jumper connectors at each terminal, ensuring they are the appropriate voltage. After this, it is recommended that the current is measured in each converter stage to ensure that the different power networks are not short-circuiting or higher than expected. This can be done by using the current header pins located between the inductor and fly-back diode, for exact location look at Figure C.1.



Figure C.1: Power Converter Test Probe Points

C.3.2 Primary Power Distribution Board

The Primary Power Distribution Board can be easily checked for operation by probing each of the terminals and ensuring the correct voltages are present on the terminals. If fuses are being blown replace with a high rating fuse and inspect the system using a current limited power supply to ensure there are no short circuits. If the system is having switching noise problems, increase the value of the decoupling capacitors to compensate for this switching noise.

C.3.3 Secondary Power Distribution Board

The secondary Power Distribution Board follows the same maintaining and testing strategy as the Primary Power Distribution Board.

C.3.4 Sensor Controller

The Sensor Controller Hardware was designed to be incredibly modular. With that in mind, if more sensors were to be added, they would need to follow the procedure described below:

C.3.4.1 Adding Ultrasonic Sensors

To add more Ultrasonic Sensors to the sensor controller, appropriate sensor hardware would have to be developed to add more diode circuitry to the echo inputs shown in Figure C.2 using the input headers(Yellow). As well as this, each additional sensor would need to be numbered and the trigger pin added to the appropriate male header(Green) which is directly connected to the relevant shift register output. If there are more than five ultrasonic sensors that need to be added, external hardware would need to be built attaching the final output of the shift register to another shift register. As well as this, some soldering to the sensor controller board to connect the serial lines to the shift register as well, the lines that need to be added are SCLK, SRCLK, SCLR and SER, the headers for these were missed in the PCB design stage.

Appendix C. Maintenance Guide

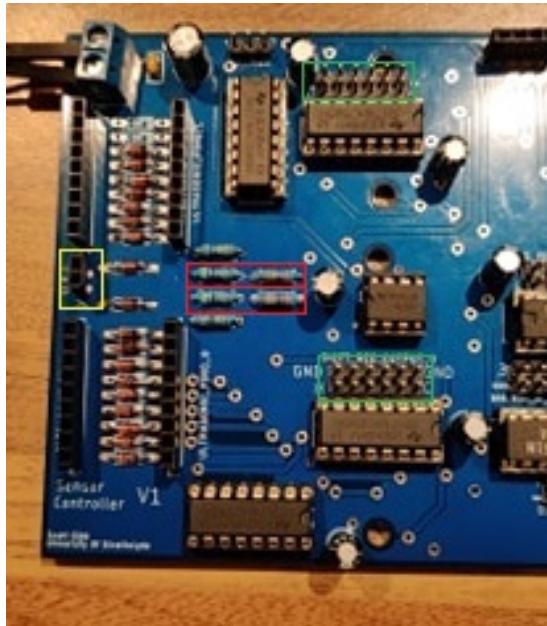


Figure C.2: Adding More Ultrasonic Sensors Connection Points

To ensure the circuitry is working it is recommended that the ultrasonic sensors are unplugged and that the trigger female headers are attached to LED's in series with a 330ω resistor, if the system is working, the LED's should light up in sequence. To ensure the echo circuitry is working an Oscilloscope is recommended and some form of slow signal generator. Attach the signal generator to one of the echo inputs shown, then using the oscilloscope attach a probe in between the two resistors shown in red in Figure C.2. Once this the appropriate signal from the signal generator should be shown on the oscilloscope, if this is not happening, then it is likely that the input diode circuitry has been damaged.

C.3.4.2 Adding Infrared Distance Sensors or other Analogue Sensors

To add more analogue sensors, the sensor outputs must be limited to a maximum of 3.3V to protect the microcontroller. This can be done using a simple voltage divider. Once this is done, attach the sensor outputs to the corresponding analogue multiplexer inputs shown in Figure C.3 by the red box. Whilst connecting the output of the sensors, it is vital that the sensors share the same ground as the sensor controller.

Appendix C. Maintenance Guide

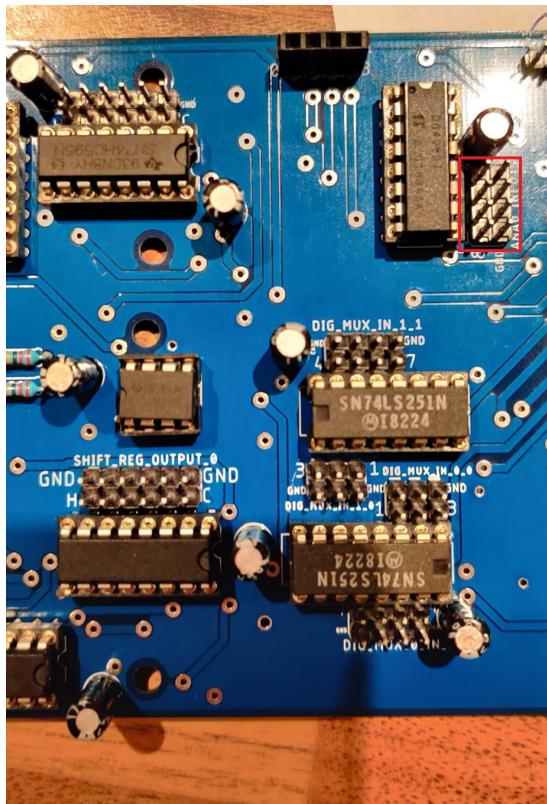


Figure C.3: Adding Analogue/SHARP IR Sensors to the Sensor Controller

C.3.4.3 Adding More Encoders

In order to add more encoders, the outputs of the encoders (and the ground signals) must be plugged into the encoder male headers shown in Figure C.4 highlighted by the red box.

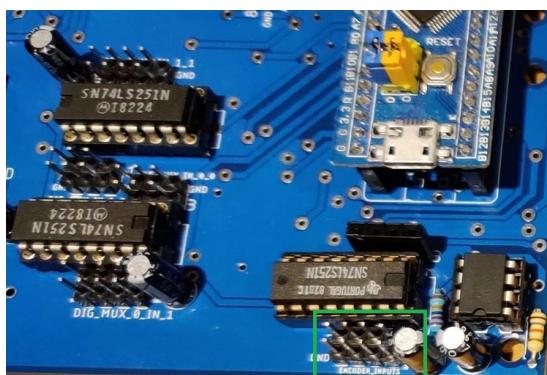


Figure C.4: Adding More encoders

C.3.5 Adding Digital Sensors

Finally, to add more digital sensors, there is a specific criterion that needs to be met. They must all be 3V3 logic to not damage the Sensor, up to seven sensors on each timer can be added, in total fourteen digital sensors can be added without having to add any more hardware or change pin functionalities. The male headers that the outputs need to be connected to can be seen in Figure C.5 highlighted with a red box.

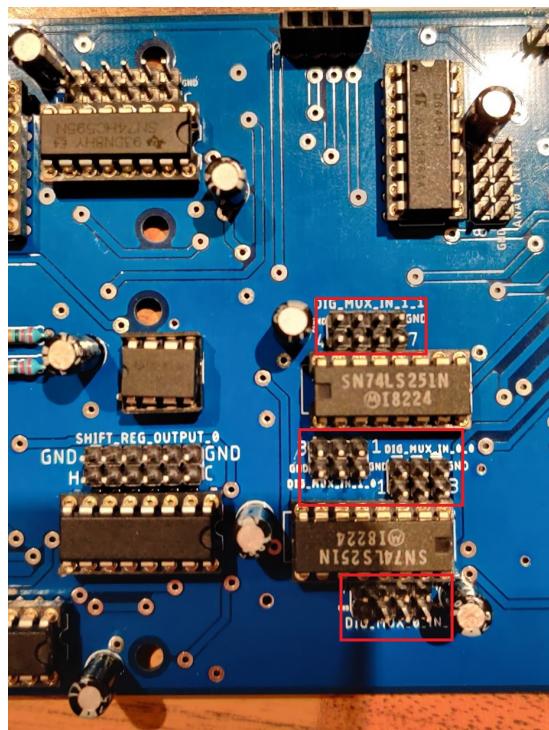


Figure C.5: Adding more digital Sensors

C.3.6 Raspberry Pi Hardware Attached on Top

For maintenance of the Raspberry Pi HAT module, anything that connects to the Pi has to be 3v3 logic level to prevent any damage to the GPIO pins. The HAT incorporates a Serial Peripheral Interface(SPI) header which any SPI device can be connected to and used by the Pi itself. As well as this, the pins from the 3v3 to 5V logic converter are broken out on the HAT to allow the unused converter channels to be used by future alterations to the robot.

Appendix C. Maintenance Guide

To ensure the correct operation of the HAT, the appropriate debugging strategies are listed below:

C.3.6.1 I2C Testing

To ensure the I2C network is working on the HAT, it would be beneficial to use a logic analyser and connect it to the 3v3 I2C pin header highlighted with the green box in figure C.6 and connect another channel to the 5V I2C header(blue box). This should reveal the same signal if the logic converter is operating correctly.

C.3.7 Ensuring correct button input voltage

To ensure the voltage going into the Pi is not exceeding the 3V3 limit, it is recommended the HAT is not connected to the Pi when doing this testing. Firstly connect a Voltage Meter probe to each of the resistors shown in FigureC.6 with a red box and ensure that the voltage measured does not exceed 3V3 logic level if it does there is a higher voltage being input to the button headers shown by a purple box in Figure C.6.



Figure C.6: Raspberry Pi HAT Maintenance Diagram

C.3.8 Inertial Measurement Unit External Hardware

For the final PCB, the Inertial Measurement Unit External Hardware PCB incorporates multiple I2C headers which allow the 5V I2C network to be expanded in almost every direction. To ensure correct operation, the board must be powered by 5V, and the I2C

Appendix C. Maintenance Guide

pins must be connected to a 5V I2C network; otherwise, the connected devices could be damaged by this board or vice versa.

C.4 Mechanical

In terms of mechanical maintenance and adding more implementation to the designs, it is recommended that they adhere to PCB standards and use the same mounting mechanisms as the PCB stacks use. It would be ideal to use the space directly above the drill motors. For the modification of any the existing models, it is recommended that Inventor [55] is used. For more information regarding software to be used refer to appendix C.5.2.

C.5 Program Versions and Libraries

Throughout the project, multiple different types of software's were used. Not only were different software applications used, but different frameworks were also used for the coding of the Python Robot Controller and the Java Control Interface. All necessary products are listed below along with their versions known to work.

C.5.1 Programming Software Libraries

The required Software Libraries and software for the different applications: Python Robot Controller and Java Control Interface are listed in tables C.1,C.2 and C.3.

Table C.1: Required Java Control Interface Software Versions

Java Control Interface Software Versions	
Name	Version
Java	1.8.0_01
JavaFX	14

Appendix C. Maintenance Guide

Table C.2: Required Python Software and libraries part 1

Python Software and Libraries part 1	
Name	Version
Python	3.7
Adafruit-Blinka	3.9.0
Adafruit-PlatformDetect	2.1.0
Adafruit-PureIO	1.0.4
PyGObject	3.30.4
RPLCD	1.2.2
RPi.GPIO	0.7.0
SecretStorage	2.3.1
adafruit-circuitpython-busdevice	4.1.4
adafruit-circuitpython-lis3dh	5.0.2
adafruit-circuitpython-motor	2.0.2
adafruit-circuitpython-pca9685	3.2.6
adafruit-circuitpython-register	1.7.4
adafruit-circuitpython-servokit	1.1.1
asn1crypto	0.24.0
certifi	2018.8.24
cffi	1.14.0
chardet	3.0.4
cryptography	2.6.1
decorator	4.2.2
entrypoints	0.3
ffmpeg	1.4
i2cled	0.1.0
idna	2.6
keyring	1.7.1.1
keyrings.alt	3.1.1
numpy	1.16.2
picamera	1.13
pip	18.1
public	2019.4.13
pyparser	2.20
pycrypto	2.6.1
pyftdi	0.42.2
pyserial	3.4
python-apt	1.8.4.1
python-interface	1.5.3

Appendix C. Maintenance Guide

Table C.3: Required Python Software and libraries part 2

Python Software and Libraries part 2	
pyusb	1.0.2
pyxdg	0.25
requests	2.21.0
rpi-ws281x	4.2.3
self	2019.4.13
setuptools	40.8.0
six	1.12.0
smbus	1.1.post2
smbus2-asyncio	0.0.5
spidev	3.4
ssh-import-id	5.7
sysv-ipc	1.0.1
urllib3	1.24.1
wheel	0.32.3
ws4py	0.5.1

C.5.1.1 Raspberry Pi Required Software

As for the Raspberry Pi a set of programs are required to be installed for various reasons such as turning the Pi into a WiFi hot spot. These libraries along with the Operating System version is shown in table C.4.

Table C.4: Required Raspberry Pi Software

Raspberry Pi Software and Libraries	
Name	Version
Raspbian Buster Lite	4.19
hostapd	2:2.7
dnsmasq	2.80

C.5.2 Software Tools Versions

Throughout the project a variety of different software was used for different purposes, the software used along with the versions are listed below in table C.5. [54] [95] [68] [96]

Table C.5: Software Tools and Versions

Software Tools and Versions	
Autodesk Inventor [55]	2020
Autodesk Eagle [97]	9.5.1
Ultimaker Cura [95]	4.4
Prusa Slicer [59]	2.1.1
Inteliji	2019.3.1
Pycharm	2019.3.1

C.5.3 Raspberry Pi Start Up Script

Due to the complexity of the system, the Raspberry Pis, the performance was boosted by overclocking the CPU, the resulting startup script for the Operating System is shown in Figure C.7.

Appendix C. Maintenance Guide

```
arm_freq=800
dtparam=i2c_arm=on, i2c_arm_baudrate=400000
dtparam=spi=on

dtoverlay=dwc2
start_x=1
gpu_mem=256
enable_uart=1
dtoverlay=w1-gpio
```

Figure C.7: Raspberry Pi Boot Script

Appendix D

Communication Ports and I2C Addresses

D.1 I2C Addresses

For adding more I2C devices, the system would require an I2C Multiplexer if the address collided with a currently used address. The current addresses used by the system are listed in table D.1.

Table D.1: I2C Currently Utilised Addresses

Currently Utilised I2C Addresses	
Device Name	Address
Sensor Controller	0x1F
MPU6050	0x68
LCD Screen	0x27
PWM Driver Base Address	0x40
PWM Driver Second Address	0x70

D.2 Internet Protocol Ports

The system uses a variety of different internet protocol ports, as such any additional ports must not use the ones currently or else the system would cease to work. The complete set of ports now utilised is listed in table D.2.

Appendix D. Communication Ports and I2C Addresses

Table D.2: High Level Internet Protocol Utilised Ports

High Level Internet Protocol Utilised Ports	
Use	Port Number
Visual HTTP	8082
Visual Web Socket	8084
Visual Control TCP	3000
Movement Control TCP	3001
Sensor Control TCP	3002
Sensor Data TCP	30003

Appendix E

Wiring Colouring Scheme

Due to the complexity of the robots wiring harness, a colouring scheme for the cables was devised so that it was easy to work with and develop. The chosen colour scheme of the wires is shown in table E. Due to limitations in university resources, the analogue output wire colour had to be the same as the I2C Data Line. However, this is not noticed due to the I2C wires being zip-tied together. The same is said for the Raspberry Pi GPIO button wires being the same as the I2C clock line.

Table E.1: Autonomous Robot Wiring Colour Scheme

Cable Colour	Voltage (v)	Purpose
Power Network		
Red Cable	+18	Main System Positive
Black Cable	GND	Main System Negative
Purple	+12	Auxillary One System Positive
Blue	+5	Auxillary Two System Positive
Ultrasonic Sensors		
Green	+5	Trigger Signal
Yellow	+5	Echo Signal
Infrared Sensors		
Grey	0 - (+5)	Analogue Output
Inter-Integrated Circuit		
Grey	5/3.3	I2C Data Line
White	5/3.3	I2C Clock Line
Other Networks		
White	3.3	Raspberry Pi Push Buttons

Appendix F

Hardware Schematics, Layouts and Renders

This appendix contains all hardware schematics, layouts and renders for the different PCBs in the system for a more in-depth look at the design, refer to chapter 6.

F.1 Hardware Schematics

This section contains all eagle schematics for the various hardware designs in the project.

For Raspberry Pi HAT, refer to appendix F.1.1.

Regarding the Sensor Controller look at appendix F.1.2

The Primary and Secondary Power Distribution Board schematics can be seen in appendix F.1.4 and appendix F.1.5 respectively.

The Power Converter full schematic can be seen in appendix F.1.3.

Finally, the IMU External Components Schematic can be seen in appendix F.1.6.

F.1.1 Raspberry Pi Hardware on Top Schematic

Appendix F. Hardware Schematics, Layouts and Renders

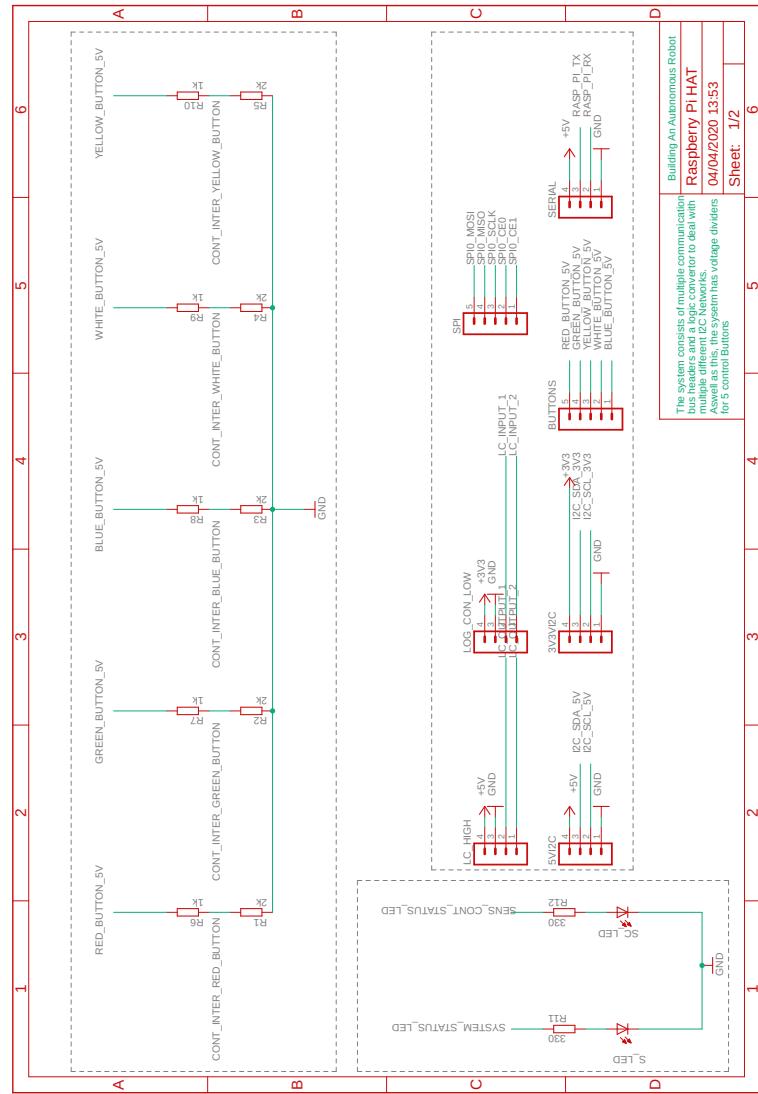


Figure F.1: Raspberry Pi Hat Schematic Sheet 1

Appendix F. Hardware Schematics, Layouts and Renders

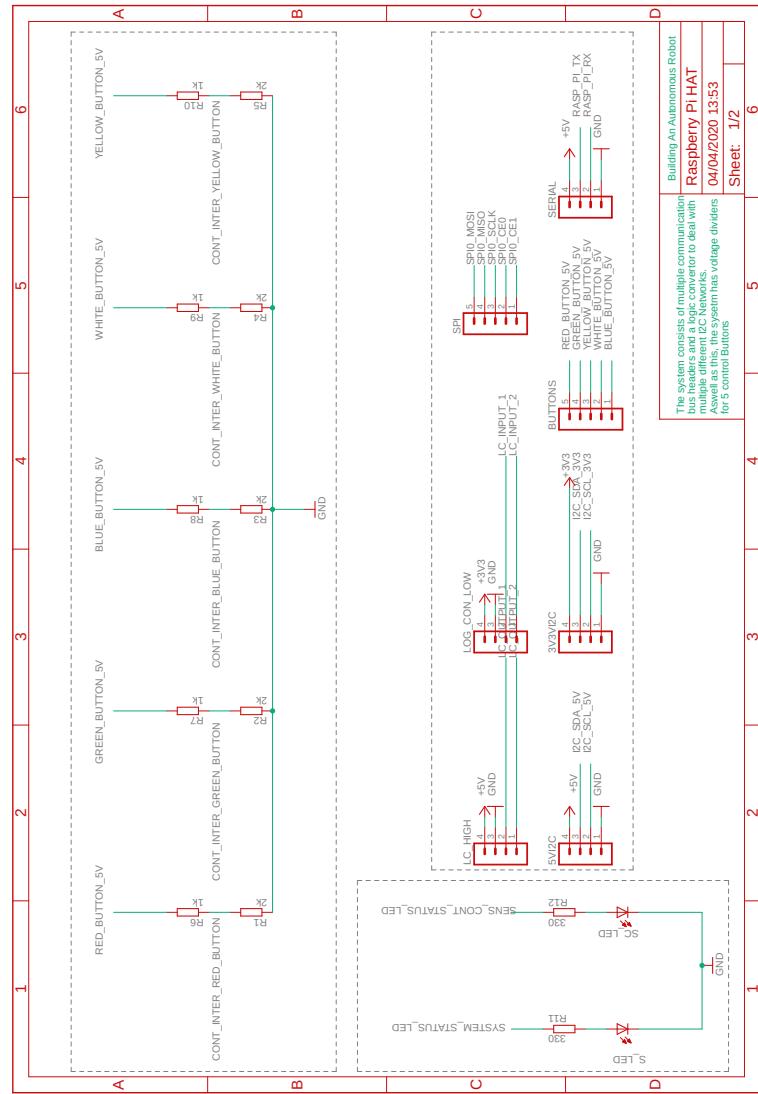


Figure F.2: Raspberry Pi Hat Schematic Sheet 2

F.1.2 Sensor Controller Schematic

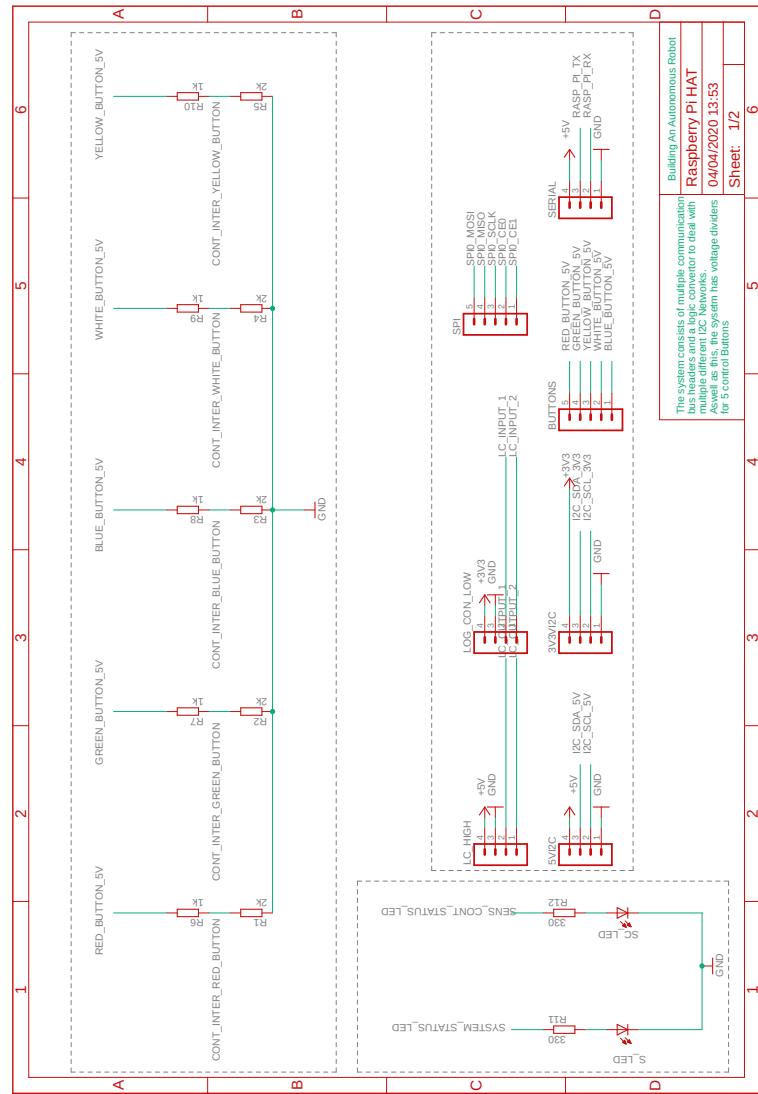


Figure F.3: Sensor Controller Schematic Sheet 1

Appendix F. Hardware Schematics, Layouts and Renders

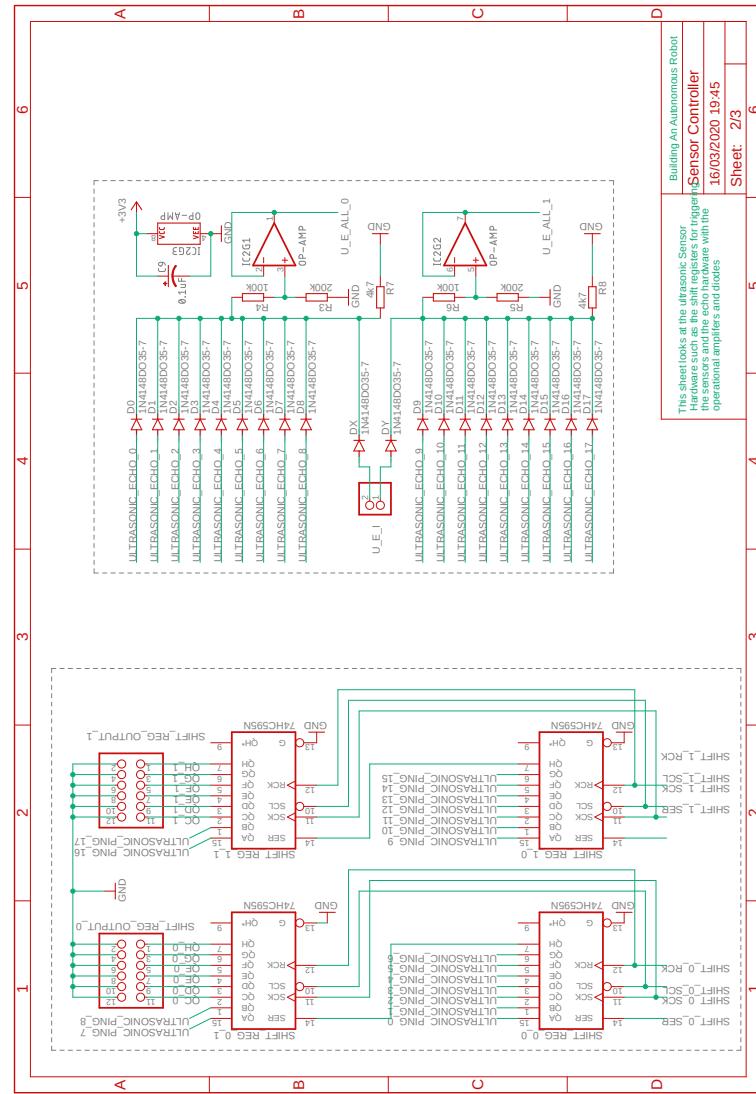


Figure F.4: Sensor Controller Schematic Sheet 2

Appendix F. Hardware Schematics, Layouts and Renders

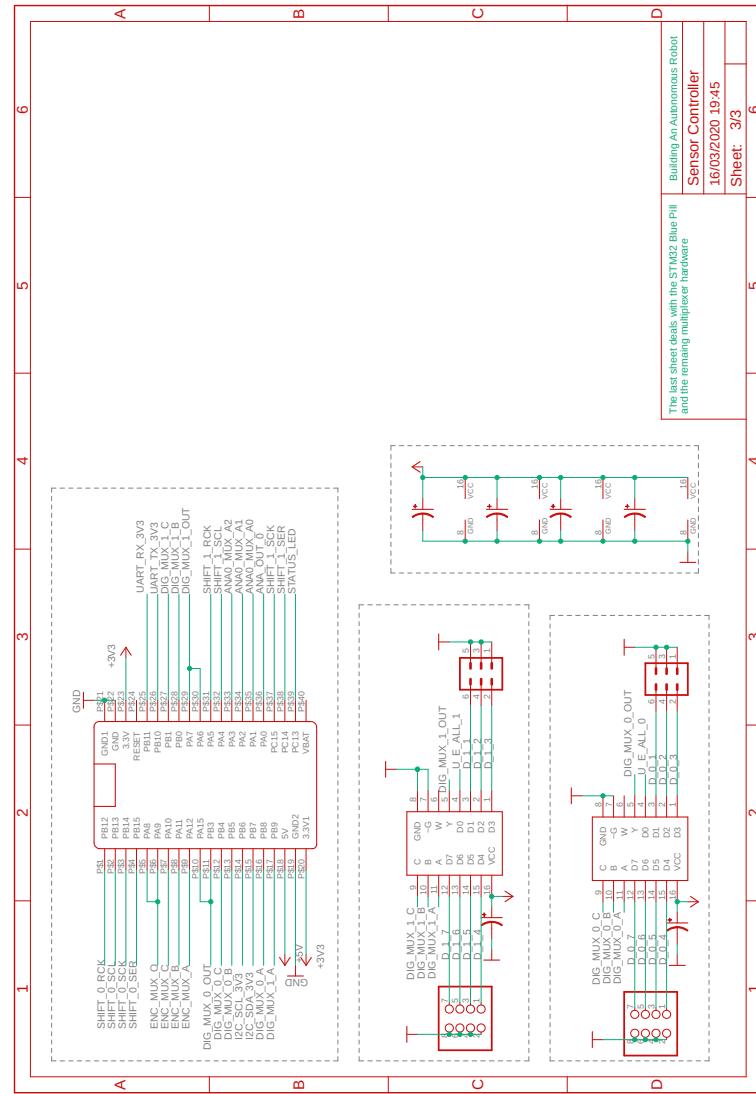


Figure F.5: Sensor Controller Schematic Sheet 3

F.1.3 Power Convertor Schematic

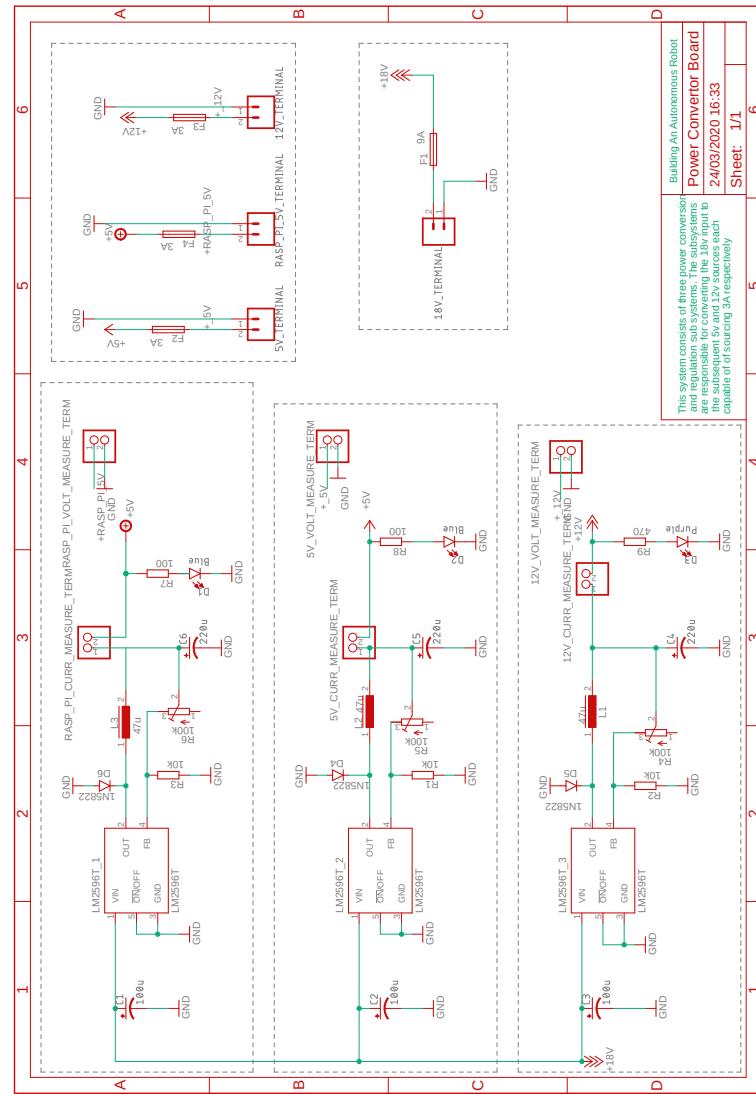


Figure F.6: Power Convertor Schematic

F.1.4 Primary Power Distribution Schematic

Appendix F. Hardware Schematics, Layouts and Renders

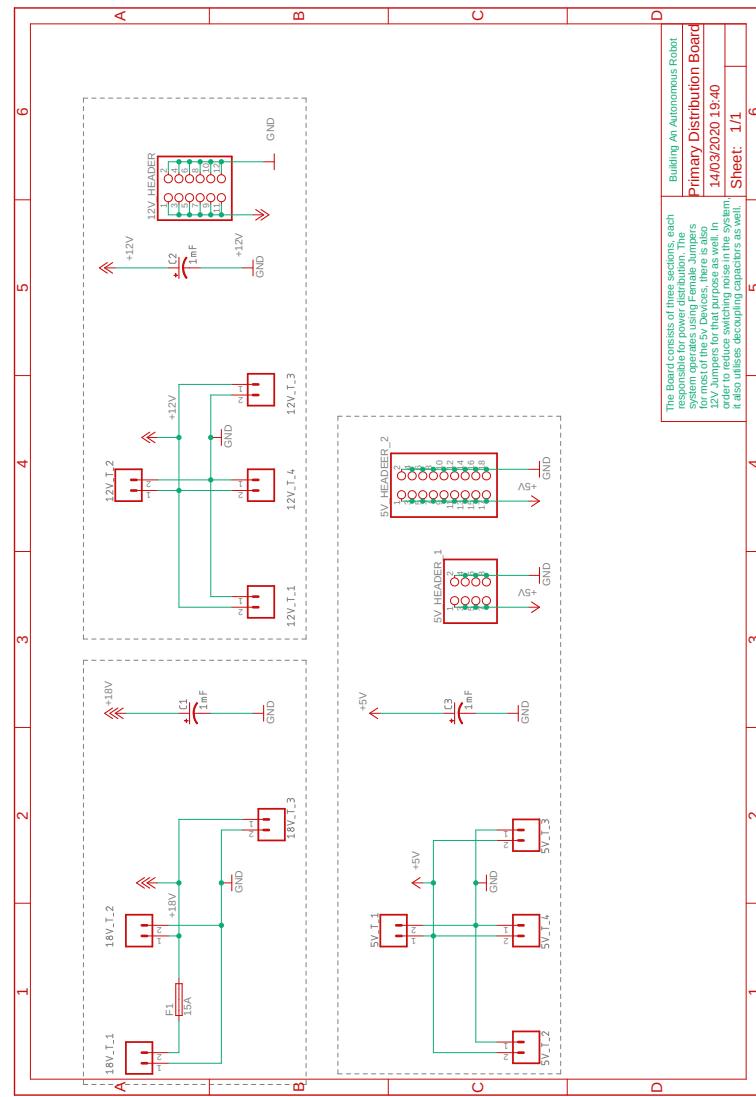


Figure F.7: Primary Power Distribution Schematic

F.1.5 Secondary Power Distribution Schematic

Appendix F. Hardware Schematics, Layouts and Renders

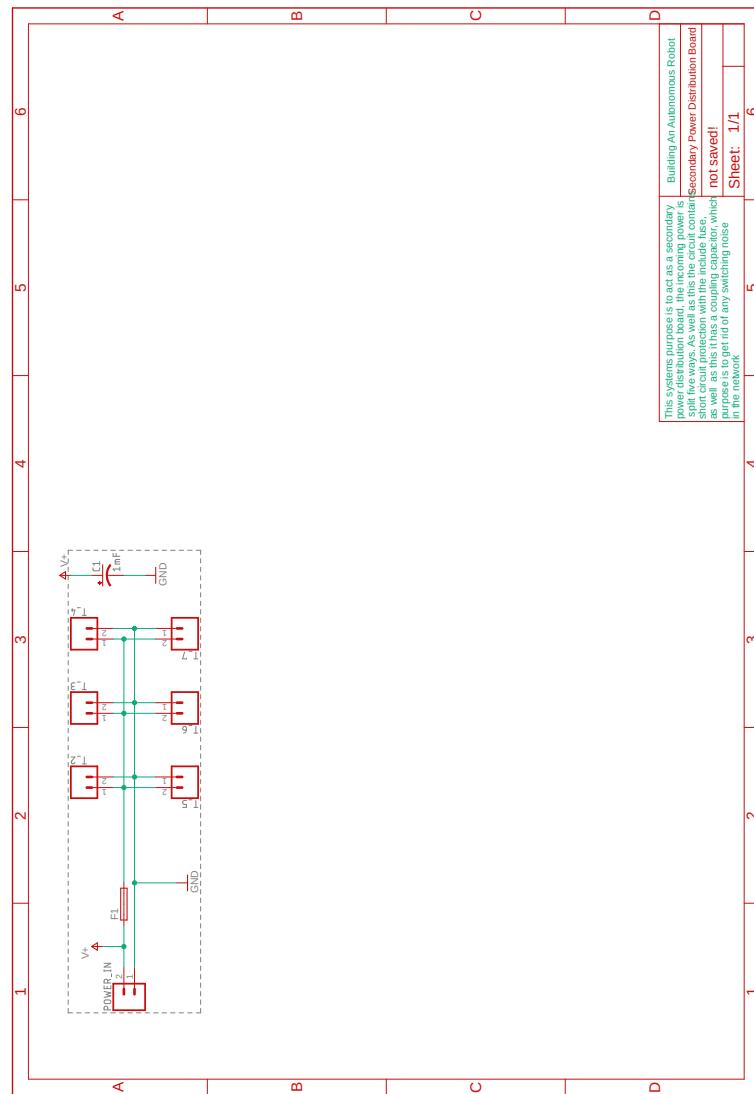


Figure F.8: Secondary Power Distribution Schematic

F.1.6 Inertial Measurement Unit External Hardware Schematic

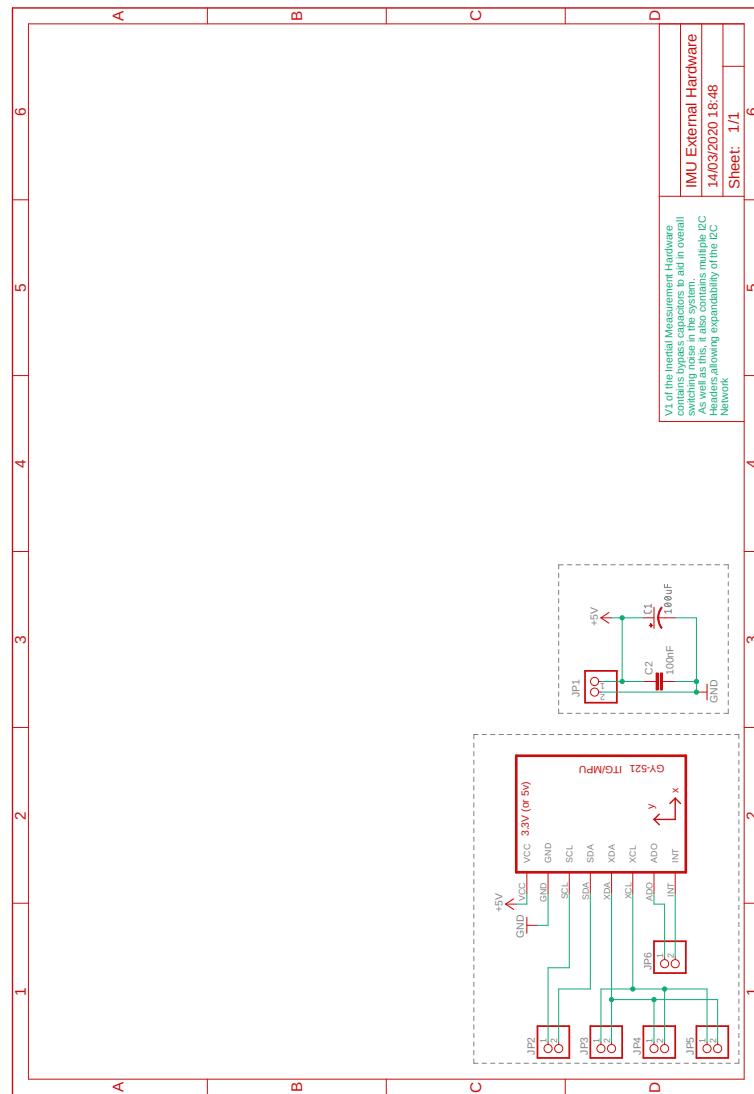


Figure F.9: Inertial Measurement Unit External Hardware Schematic

F.2 PCB Layouts

This section contains all the layouts for the PCB Schematics shown in appendix F.1.

The Raspberry Pi HAT Layout can be seen in appendix F.2.1

The Primary and Secondary Power Distribution PCB layouts can be seen in ap-

Appendix F. Hardware Schematics, Layouts and Renders

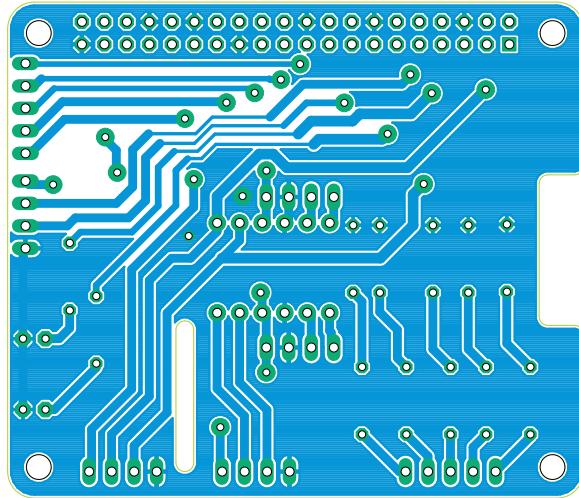
pendix F.2.4 and appendix F.2.5.

The Power Converter PCB Layout is shown in appendix F.2.3.

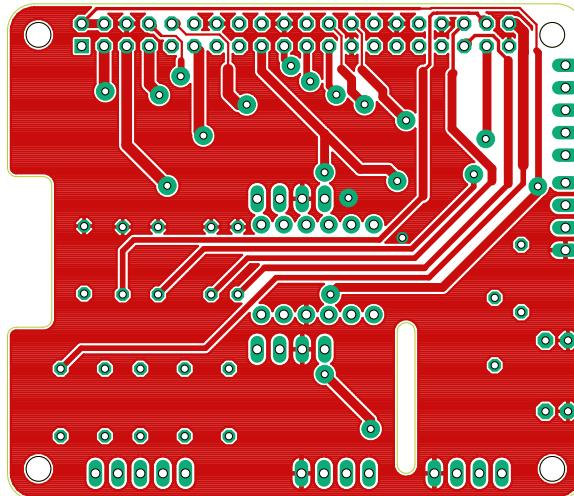
The remaining PCBs, Sensor Controller and Inertial Measurement Unit External Hardware PCB layouts are shown in appendix F.2.2 and appendix F.2.6.

F.2.1 Raspberry Pi Hardware on Top Layout

Appendix F. Hardware Schematics, Layouts and Renders



(a) Raspberry Pi HAT Board (Bottom)

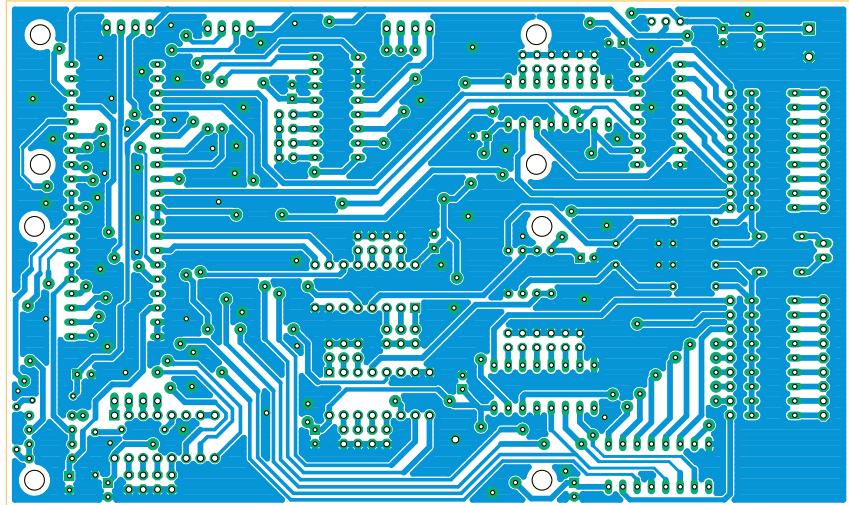


(b) Raspberry Pi HAT Board (Top)

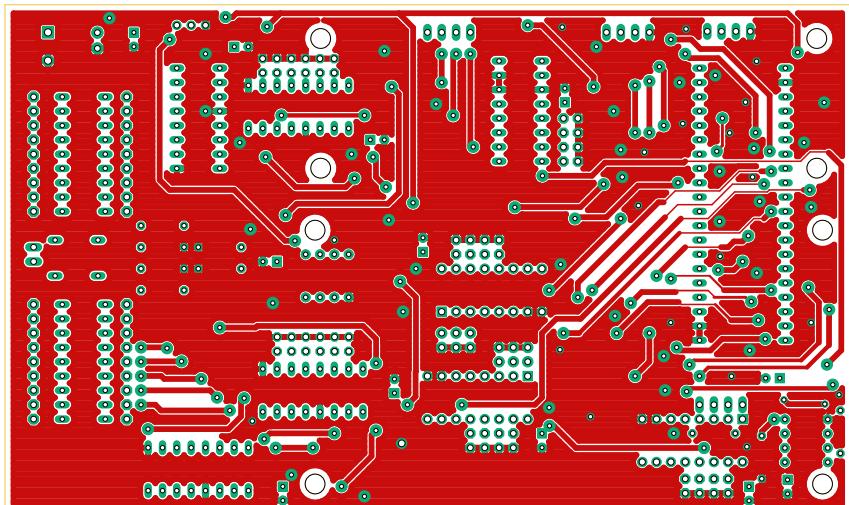
Figure F.10: Raspberry Pi Hardware on Top PCB Layout

F.2.2 Sensor Controller Layout

Appendix F. Hardware Schematics, Layouts and Renders



(a) Sensor Controller Board (Bottom)

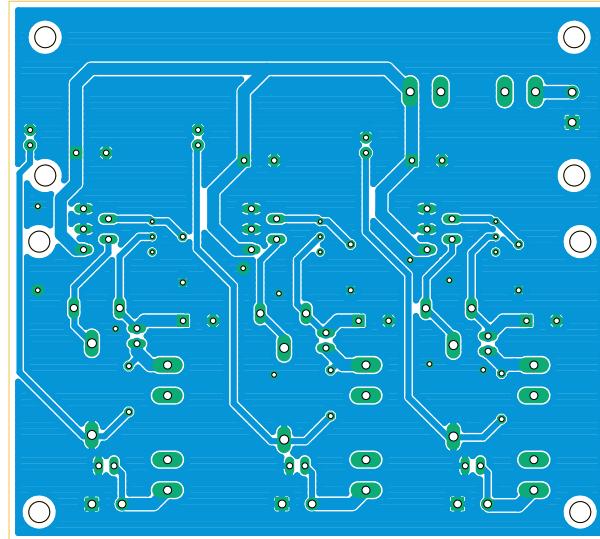


(b) Sensor Controller Board (Top)

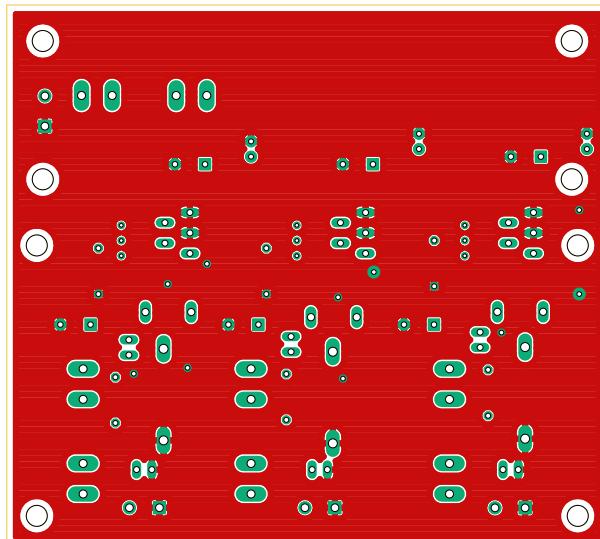
Figure F.11: Sensor Controller PCB Layout

F.2.3 Power Converter Layout

Appendix F. Hardware Schematics, Layouts and Renders



(a) Raspberry Pi HAT Board (Bottom)

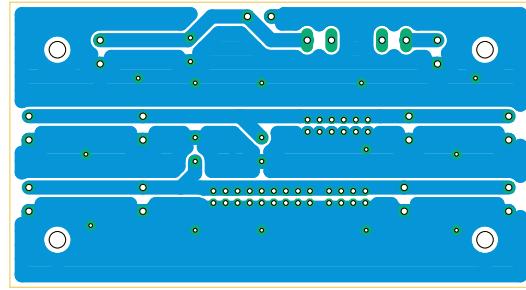


(b) Raspberry Pi HAT Board (Top)

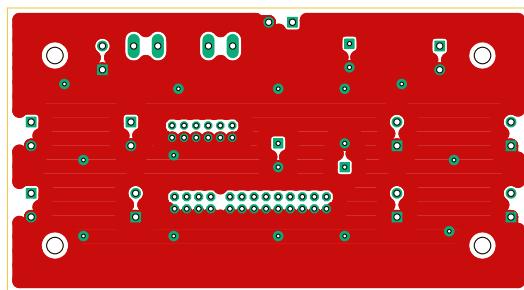
Figure F.12: Power Converter PCB Layout

F.2.4 Primary Power Distribution Layout

Appendix F. Hardware Schematics, Layouts and Renders



(a) Primary Power Distribution Board (Bottom)

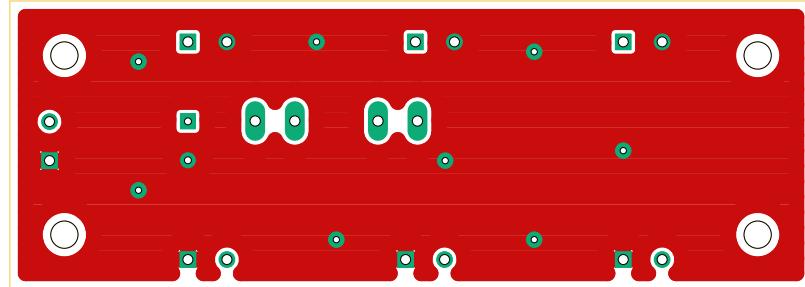


(b) Primary Power Distribution Board (Top)

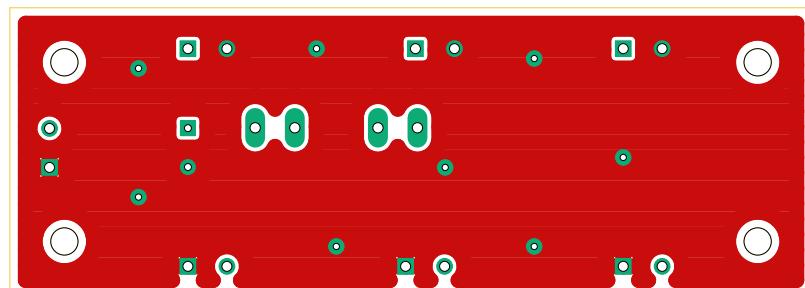
Figure F.13: Primary Power Distribution PCB Layout

F.2.5 Secondary Power Distribution Layout

Appendix F. Hardware Schematics, Layouts and Renders



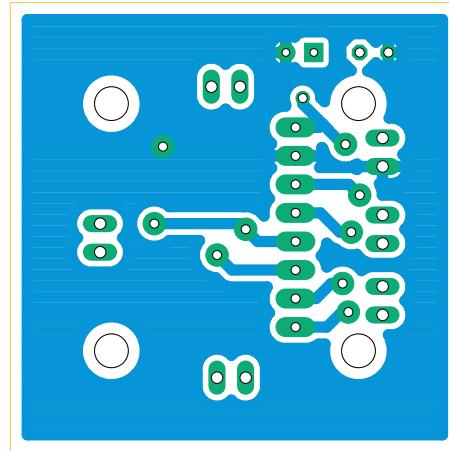
(a) Secondary Power Distribution Board (Bottom)



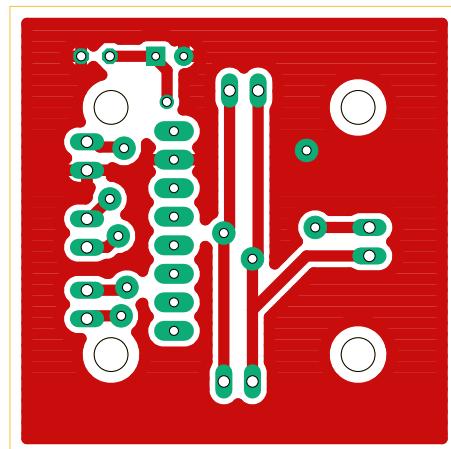
(b) Secondary Power Distribution Board (Top)

Figure F.14: Secondary Power Distribution PCB Layout

F.2.6 Inertial Measurement Unit External Hardware Layout



(a) IMU External Hardware Board (Bottom)



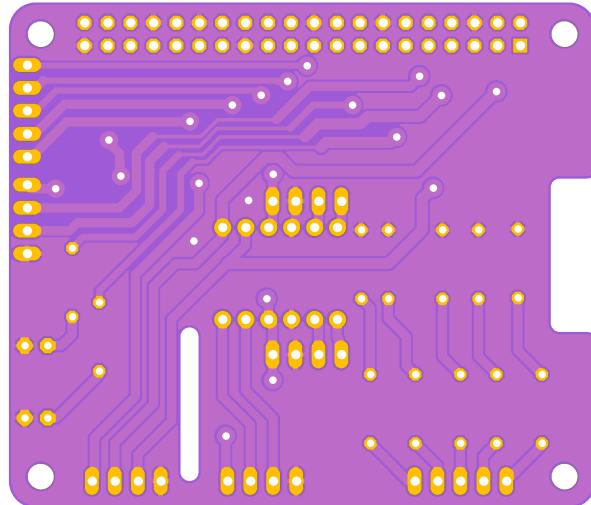
(b) IMU External Hardware Board (Top)

Figure F.15: Inertial Measurement Unit External Hardware Layout

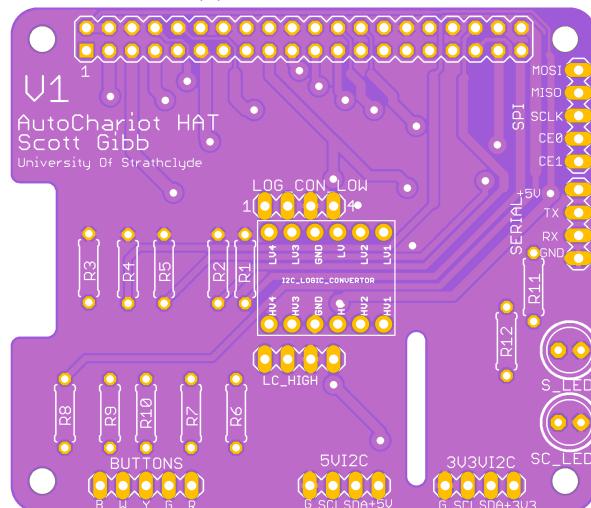
F.3 PCB Eagle Renders

Finally, the last aspect this appendix covers is the renders of the PCBs covered previously in chapter 6. The different renders are categorised below in the relevant sections.

F.3.1 Raspberry Pi Hardware Attached on Top Renders



(a) Bottom Render

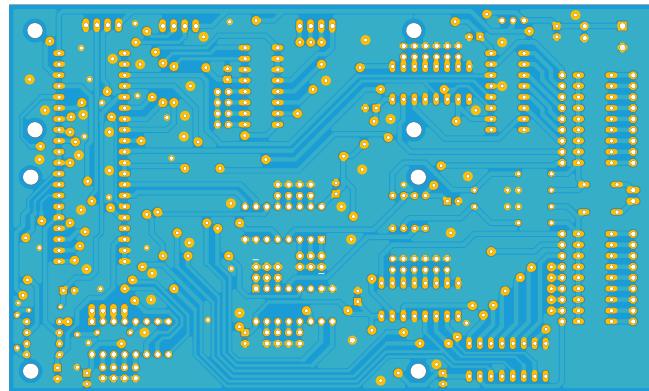


(b) Top Render

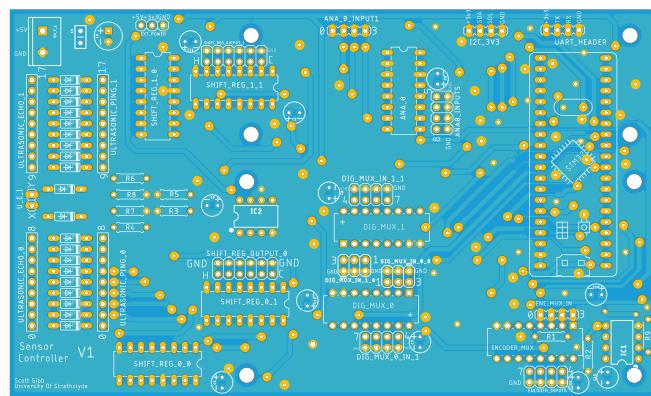
Figure F.16: Raspberry Pi HAT PCB

F.3.2 Sensor Controller Renders

Appendix F. Hardware Schematics, Layouts and Renders



(a) Bottom Render

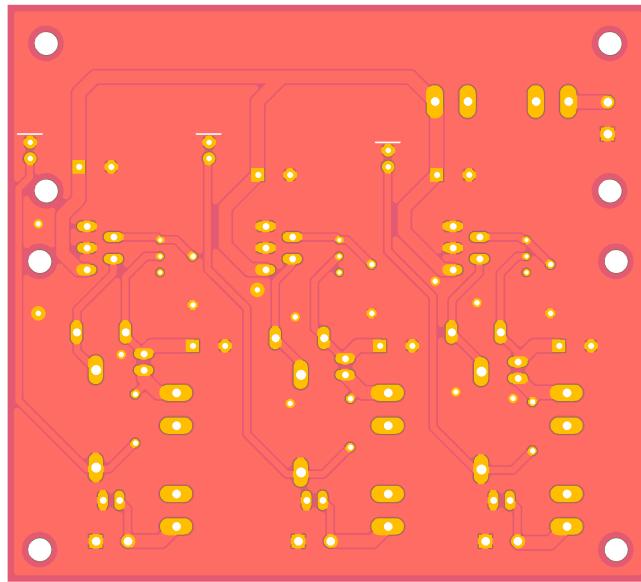


(b) Top Render

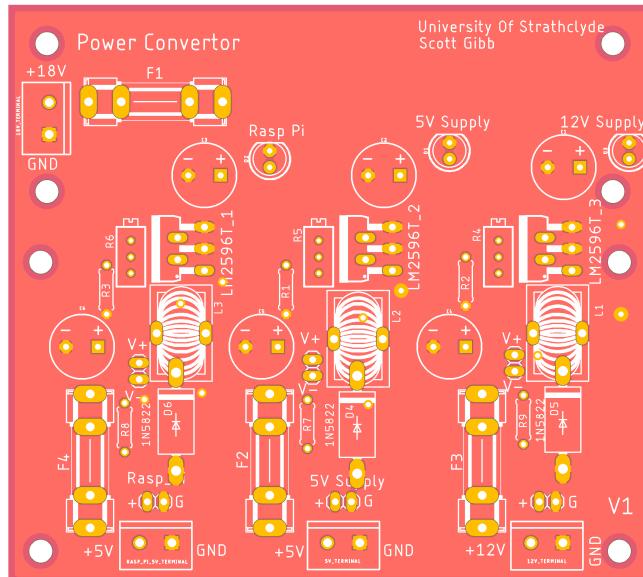
Figure F.17: Sensor Controller PCB

F.3.3 Power Converter Renders

Appendix F. Hardware Schematics, Layouts and Renders



(a) Bottom Render

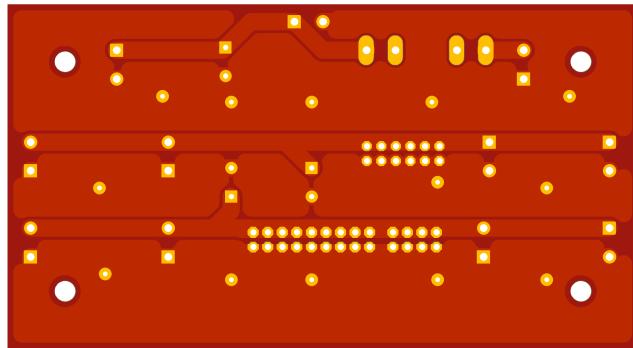


(b) Top Render

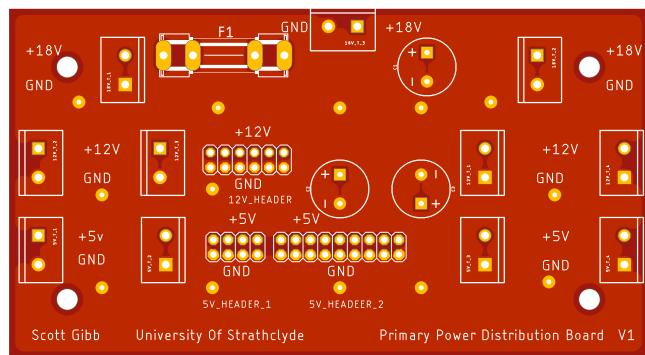
Figure F.18: Power Converter PCB

F.3.4 Primary Power Distribution Board Renders

Appendix F. Hardware Schematics, Layouts and Renders



(a) Bottom Render

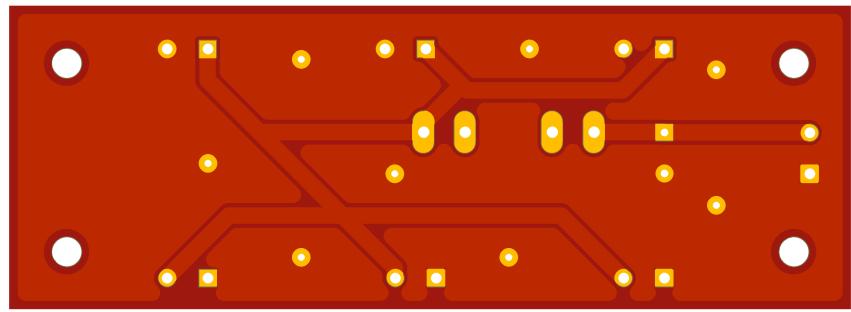


(b) Top Render

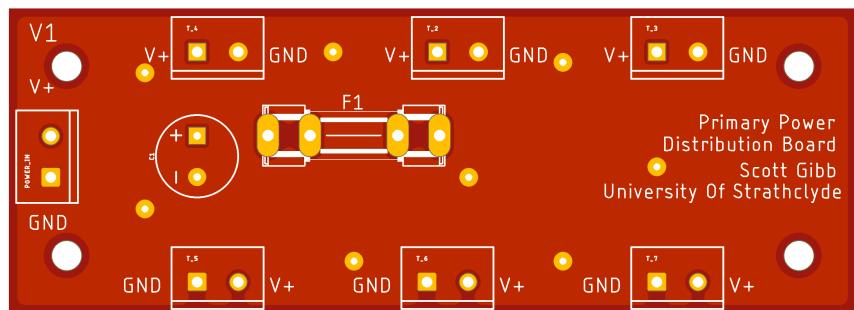
Figure F.19: Primary Power Distribution PCB

F.3.5 Secondary Power Distribution Board Renders

Appendix F. Hardware Schematics, Layouts and Renders



(a) Bottom Render

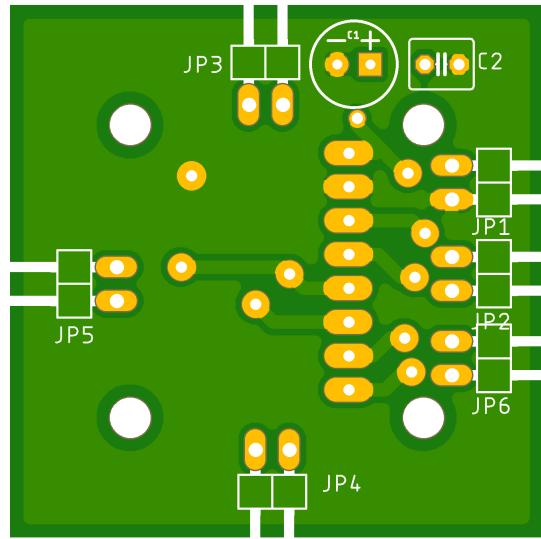


(b) Top Render

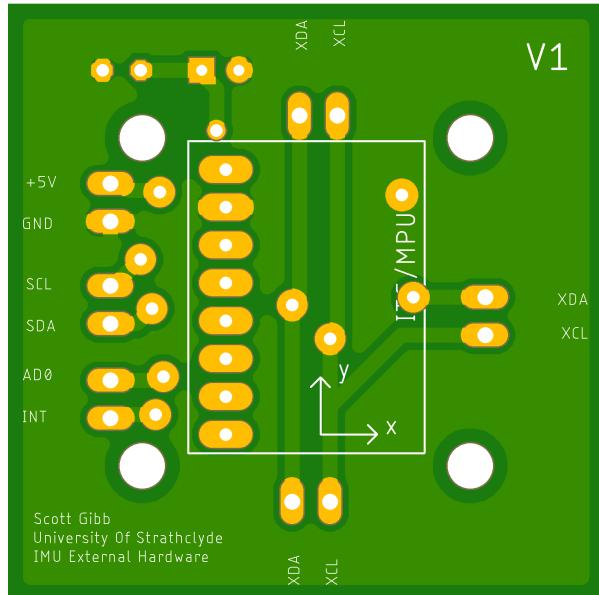
Figure F.20: Secondary Power Distribution PCB

F.3.6 Inertial Measurement Unit External Hardware Renders

Appendix F. Hardware Schematics, Layouts and Renders



(a) Bottom Render



(b) Top Render

Figure F.21: Inertial Measurement Unit External Hardware PCB

F.4 Required PCB Components

F.4.1 Raspberry Pi Hardware Attached on Top PCB Components

All resistors within the Raspberry Pi HAT are standard 1/4W Through Hole resistors.

Table F.1: Raspberry Pi Hardware On Top PCB Component List

Raspberry Pi HAT PC Components List		
Component	Quantity	
Raspberry Pi HAT PCB	1	
2x20 Stack Header	1	
1x4 Male Header	5	
1x5 Male Header	2	
2.2kΩ Through Hole Resistor(0207)	5	
3.3kΩ Through Hole Resistor(0207)	5	
330Ω Through Hole Resistor(0207)	2	
SparkFun Logic Convertor	1	
1x6 Female Header	2	

F.4.2 Sensor Controller PCB Components

All resistors within the Sensor Controller are standard 1/4W Through Hole resistors.

Appendix F. Hardware Schematics, Layouts and Renders

Table F.2: Sensor Controller PCB Component List

Sensor Controller Components PCB List	
Component	Quantity
Sensor Controller PCB	1
STM32F103CT6	1
1x15 Female Header	2
5mm Screw Terminal	1
2x3 Female Header	2
1x4 Male Header	4
1x2 Female Header	1
1x3 Male Header	1
1x9 Female Header	4
2x4 Male Header	4
2x6 Male Header	2
200k Ω Through Hole Resistor(0207)	3
4k7 Ω Through Hole Resistor(0207)	2
100k Ω Through Hole Resistor(0207)	4
100 μ F 5V Electrolytic Capacitor(2.5mm)	10
100 μ F 5V Ceramic Capacitor(2.5mm)	1
10/F 5v Electrolytic Capacitor(2.5mm)	1
LM358 Dual Operational Amplifier(DIP-8)	2
1N4148D035-7 Through Hole Diode	20
74HC595N Shift Register	4
DG408DJ Analogue Multiplexer	1
SN74LS251N Digital Multiplexer	3

F.4.3 Power Converter PCB Components

All resistors within the Power Converter are standard 1/4W Through Hole resistors unless otherwise stated.

Appendix F. Hardware Schematics, Layouts and Renders

Table F.3: Power Converter Component PCB List

Power Converter PCB Components List	
Component	Quantity
Power Converter PCB	1
5mm Screw Terminal	4
1x2 Male Header	6
100kΩ Through Hole Trimpot(3296)	3
10kΩ Through Hole Resistor	3
220μF 30V Electrolytic Capacitor(5mm)	3
100μF 30V Electrolytic Capacitor	3
Purple 5mm Through Hole LED	1
Blue 5mm Through Hole LED	1
47μH 3A Through Hole Power Inductor(0207)	3
1N5822 Schottky Rectifier Diode	3
LM2596T SMPS	3
Fuse Holder Clip Set	4
3A Fast Blow Fuse(20x5mm)	3
?A Fast Blow Fuse(20x5mm)	1
470Ω Through Hole Resistor(0207)	2
100Ω Through Hole Resisto(0207)r	1
Jumper Cap	3

F.4.4 Primary Power Distribution PCB Components

Primary Power Distribution PCB Components	
Component	Quantity
Primary Power Distribution PCB	1
5mm Screw Terminal	11
2x4 Female Header	1
2x6 Female Header	1
1x9 Female HEader	1
1μF 18V Through Hole Electrolytic Capacitor(5mm)	1
1μF 12V Through Hole Electrolytic Capacitor(5mm)	1
1μF 5V Through Hole Electrolytic Capacitor(5mm)	1
Fuse Holder Clip Set	1
15A Fast Blow Fuse (20x5mm)	1

Table F.4: Caption

F.4.5 Secondary Power Distribution Components

Appendix F. Hardware Schematics, Layouts and Renders

Table F.5: Secondary Power Distribution PCB Component List

Secondary Power Distribution PCB Components	
Component	Quantity
1 μ F 18V Through Hole Electrolytic Capacitor(5mm)	1
5mm Screw Terminal	7
Fuse Holder Clip Set	1
Fat Blow Fuse(20x5mm) Current Rating dependent on application	1

F.4.6 Inertial Measurement Unit External Hardware PCB Components

Table F.6: Inertial Measurement Unit External Hardware PCB Component List

Inertial Measurement Unit External Hardware PCB Component List	
Component	Quantity
1x2/90 Male Header	6
100nF Ceramic Through Hole Capacitor	1
100 μ F Electrolytic Through Hole Capacitor	1
MPU6050 Breakout Board	1
1x8 Male Header	1
1x8 Female Header	1

Appendix G

Required Parts Overview

The Autonomous Robot Upgrade requires a variety of different parts, and this section focuses on the required 3D printed parts and high-level assembly parts such as what types of bolts are required to assemble the modules. For PCB related components, please look at appendix F.

G.1 Full System Assembly

The full component list for the full Autonomous Upgrade is shown in multiples tables.

For 3D printed components see G.1.

Wooden and laser cut parts are shown in table G.2.

For PCB related parts, see table G.3.

Regarding actuator parts such as servos, see table G.4.

For the total amount of screws, nuts and bolts see table G.5.

Finally for all remaining parts, please see table G.6.

Appendix G. Required Parts Overview

Table G.1: 3D Printed components List

Autonomous Robot Full Component List	
3D Printed Components	
Component Type	Quantity
Battery Holder	1
Camera Case Part 1	1
Camera Case Part 2	1
Camera Mount Part 1	1
Camera Mount Part 2	1
Camera Mount Part 3	1
Control Panel Model	1
Encoder Flywheel Stand	2
Encoder Wheel for Flywheel	2
Encoder Wheel	2
Gyroscope Holder	1
Infrared Sensor Mount Case Part 1	4
Infrared Sensor Mount Case Part 2	4
Infrared Sensor Corner Mount	3
Infrared Sensor Front Left Mount	1
Infrared Sensor Front Left Cover	1
Ultrasonic Sensor Single Mount	7
Ultrasonic Sensor Dual Mount	2
Ultrasonic Sensor Corner Mount	2
PCB Screw Frame Holder	12
Battery Management System Layout Holder	1
Voltage Splitter Layout Holder	1
PWM Driver Holder	1
Side Cable Management Divider	1
Side Cable Management Divider with RC Receiver Holder	1
Single Mounting Tack	4
Double Mounting Tack	4
Triple Mounting Tack	4
Roof Mounting Tack	13

Table G.2: Wooden And Laser Cutting Parts

Autonomous Robot Full Component List	
Wood Working and Laser Cutting Parts	
Component Type	Quantity
Top Panel Part 1	1
Top Panel Part 2	1
Top Panel Part 3	1

Appendix G. Required Parts Overview

Table G.3: Printed Circuit Board Components

Autonomous Robot Full Component List	
Printed Circuit Boards	
Component Type	Quantity
Primary Power Distribution Board	2
Raspberry Pi Camera V2	1
Keyestudio LCD Shield Module Display 2004 LCD Display	1
AS5600 Breakout Board	4
Inertial Measurement Unit PCB	1
GP2Y0A41SKOF IR Sensor	4
HC-SR04 Ultrasonic Sensor	17
Raspberry Pi 3 Model B +	1
Raspberry Pi HAT	1
Power Converter PCB	1
Auxiliary Power Distribution PCB	1
Adafruit 16 Channel 12 Bit PWM/Servo Driver	1
Sensor Controller PCB	1

Table G.4: Total Actuator Parts

Autonomous Robot Full Component List	
Actuator Parts	
Component Type	Quantity
SG90 Servo Motor	6
SG90 Servo Motor Blade	7
MakerHawk 320mm Servo Extension Lead	3

Appendix G. Required Parts Overview

Table G.5: Required Screws, Bolts and Nuts

Autonomous Robot Full Component List	
Screws, Bolts and Nuts	
Nylon Parts	
M3X6mm Nylon Screw	8
Female-Female M3 Nylon 10mm Spacer	8
Male-Female M3 Nylon Space(6mm+6mm)	20
M3x12mm Nylon Screw	10
M3 Nylon Hex Spacers	22
M3x20mm Nylon Screw	20
M2.5 Nylon Screw	4
M2.5 Nylon Nut	4
M2.5 Male-Female(6mm+6mm)	4
M3x20mm Nylon Male-Female(20mm+6mm)	8
M3 Nylon Male-Female(20mm+6mm)	2
M3x20mm Female-Female Nylon Spacer	16
Wood Screws	
3mmx12mm	72
4mmx12mm Screw	14
2mmx12mm Screw	34
2mmx16mm Screw	12
Metal Parts	
M2x8mm Socket Screw	2
M2x12mm Socket Screw	8
M2x20mm Socket Screw	4
M2 Nut	14
M3x8mm Socket Screw	13
M3x12mm Socket Srew	8
M3x20mm Socket Screw	8
M3 Nut	29
2mmx8mm	5

Table G.6: Miscellaneous Parts

Autonomous Robot Full Component List	
Miscellaneous Parts	
Component Type	Quantity
Generic Zip Ties	Roughly 200
16mm Push Button On/Off Momentary Power Switch	5
3 Pin Switcher Rectangular Latching Rocker Switch W/LED on/off	1
2mm Push On Crimp	10

G.2 Battery Holder Assembly

The full component list for the Battery Holder is shown in table G.7.

Table G.7: Battery Holder Full Component List

Battery Holder Full Component List	
Component Type	Quantity
Battery Holder	1
M3X6mm Nylon Screw	8
Female-Female M3 Nylon 10mm	8
Male-Female M3 Nylon Spacer(6mm +6mm)	8
Primary Power Distribution Board	2
4mmx12mm Screw	6

G.3 Camera Module Assembly

The camera module consists of many different parts which are listed in table G.8.

Table G.8: Camera Module Full Component List

Camera Module Full Component List	
Component Type	Quantity
Camera Case Part 1	1
Camera Case Part 2	1
Camera Mount Part 1	1
Camera Mount Part 2	1
Camera Mount Part 3	1
M2x8mm Socket Screw	2
M2x20mm Socket Screw	4
M2 Nut	6
2mmx8mm Screw	1
SG90 servo motor	2
SG90 servo motor Rotor Blade	3
MakerHawk 320mm servo Extension Lead	1
Raspberry Pi Camera V2	1
3mmx12mm Screw	2

Appendix G. Required Parts Overview

G.4 Control Panel Assembly

The control panel consists of 12 parts, and the different parts are listed in table G.9.

Table G.9: Control Panel Full Component List

Control Panel Full Component List	
Component Type	Quantity
Control Panel Model	1
16mm Push Button On/Off Momentary Power Switch	5
3Pin Switcher Rectangular Latching Rocker Switch W/ LED on/off	1
Keyestudio LCD Shield Module Display 2004 LCD Display	1
2mm Push-On Crimp	10
M3X12mm Nylon Screw	2
M3 Hex Nylon Spacers	2
3mmX12mm Screw	2

G.5 Encoder Modules Assembly

The encoder modules consist of two sub-assemblies consisting of 22 parts. The parts list is shown in table G.10.

Table G.10: Encoder Modules Full Component List

Encoder Modules Full Component List	
Component Type	Quantity
Flywheel Encoder Module	
Encoder Flywheel Stand	2
Encoder Wheel for Flywheel	2
AS5600 Encoder Magnet	2
AS5600 Breakout Board	2
4mmx12mm Screw	4
Drive-wheel Encoder Module	
Encoder Wheel	2
AS5600 Encoder Magnet	2
AS5600 Breakout Board	2
4mmx12mm Screw	4

Appendix G. Required Parts Overview

G.6 Top Panel Assembly

The top panel consists of 21 parts. The component list is shown in table G.11.

Table G.11: Top Panel Assembly Full Component List

Top Panel Assembly Full Component List	
Component Type	Quantity
Top Panel Part 1	1
Top Panel Part 2	2
Top Panel Part 3	2
3mmX12mm Screw	16

G.7 Inertial Measurement Unit Assembly

The Inertial Measurement Unit Assembly requires 10 parts, the complete list of parts is shown in Table G.12

Table G.12: Inertial Measurement Unit Full Component List

Top Panel Assembly Full Component List	
Component Type	Quantity
Inertial Mesasurement Unit PCB	1
Gyroscope Holder	1
M3X20mm Nylon Screw	4
M3 Hex Nylon Spacer	4

G.8 Infrared Sensor Modules

For the complete set of Infrared Sensors the parts list is shown in table G.13.

Appendix G. Required Parts Overview

Table G.13: Infrared Sensor Modules Full Component List

Infrared Sensor Modules Full Component List	
Component Type	Quantity
Infrared Sensor Mount Case Part 1	4
Infrared Sensor Mount Case Part 2	4
Infrared Sensor Corner Mount	3
Infrared Sensor Front Left Mount	1
Infrared Sensor Front Left Cover	1
SG90 Servo Motor	4
SG90 Servo Motor Rotor Blade	4
2mmX8mm Screw	4
M2X12mm Socket Screw	8
M2 Nut	8
M3X20mm Socket Screw	8
M3 Nut	16
M3X12mm Socket Screw	8
GP2Y0A41SKOF IR Sensor	4
3mmx12mm Screw	16
MakerHawk 320mm servo Extension Lead	2

G.9 Ultrasonic Sensor Modules

For the complete ultrasonic Sensor Modules to be installed on the robot, the parts required are shown in Table G.14.

Table G.14: Ultrasonic Sensor Modules Full Component List

Ultrasonic Sensor Modules Full Component List	
Component Type	Quantity
Ultrasonic Sensor Single Mount	7
Ultrasonic Sensor Dual Mount	2
Ultrasonic Sensor Corner Mount	2
HC-SR04 Ultrasonic Sensor	17
2mmx12mm Screw	34
2mmx16mm Screw	12

G.10 Cable Management and Mounting Modules

Not all components are designed for specific PCBs and other modules etc. The table is shown in table G.15 summarises the remaining required parts.

Appendix G. Required Parts Overview

Table G.15: Cable Management and Mounting Modules Component List

Cable Management and Mounting Modules Component List	
Component Type	Quantity
Raspberry Pi Stack Components	
Raspberry Pi 3 Model B +	1
Raspberry Pi HAT	1
PCB Screw Frame Holder	4
3mmx12mm Screw	8
M3x20mm Nylon Screw	4
M3 Nylon Hex Spacer	4
M3x20mm Female-Female Nylon Spacer	4
M3 Male-Female(6mm+6mm)	2
M3 Male-Female(20mm+6mm)	2
Front Motor Controller Stack Components	
PCB Screw Frame Holder	4
3mmx12mm Screw	8
M3x20mm Nylon Screw	4
M3 Nylon Hex Spacer	8
M3 Nylon Male-Female(6mm+6mm)	6
Battery Management System Layout Holder	1
Primary Power Converter PCB	1
M3x20mm Nylon Female-Female Nylon Spacer	8
Voltage Splitter Layout Holder	1
Auxiliary Power Distribution PCB	1
M3x20mm Nylon Male-Female(20mm+6mm)	8
PWM Driver Holder	1
Adafruit 16 Channel 12 Bit PWM/Servo Driver	1
M2.5 Nylon Screw	4
M2.5 Nylon Nut	4
M2.5 Nylon Male-Female(6mm+6mm)	4
Bottom Motor Controller Stack Components	
PCB Screw Frame Holder	4
3mmx12mm Screw	8
M3x20mm Nylon Screw	4
M3 Nylon Hex Spacer	4
M3x20mm Nylon Female-Female Nylon Spacer	4
M3 Nylon Male-Female(6mm+6mm)	4
Sensor Controller PCB	1
TODO: FINISH THIS	0
Cable Management Components	
Generic Zip Ties	Roughly 200
Side Cable Management Divider	1
Side Cable Management Divider with RC Receiver Holder	1
Single Mounting Tack	4
Double Mounting Tack	4
Triple Mounting Tack	4
3mmx12mm Screw	12
Roof Mounting Tack	13
M3X8mm Socket Screw	13
M3 Nut	13

Appendix G. Required Parts Overview

Appendix H

Mechanical Model Renders

This part of the appendix consists of all CAD Model renders and exploded views of each mechanical subsystems. The subsystems include the Battery Holder, Camera Module, Encoder Mounts, Top Panel, Gyroscope Mounts, Infrared Sensor Mounts, Ultrasonic Sensor Mounts, Mounting Frame Parts and finally PCB Footprints.

H.1 Base Rampaging Chariots CAD Models

The base radio-controlled Rampaging Chariots kit was modelled so that an entire system model could be created in CAD software. The models were simplified in many aspects such that the measurements needed were present, however, the detail in the models was not added in some cases. The models are split into multiple sections. The Assembled CAD model for the Radio Controlled Rampaging Chariots robot is shown in Figure H.1

For Strengthening brackets the CAD Models are shown in Figure H.2

The individual MDF Chassis Panels are shown in Figure H.3

Parts concerning wheel and motors can be seen in Figure H.4.

Finally the electronic specific parts are shown in Figure H.5.

Appendix H. Mechanical Model Renders



Figure H.1: Rampaging Chariots Base Kit Assembled CAD Model

H.1.1 Strengthening Brackets

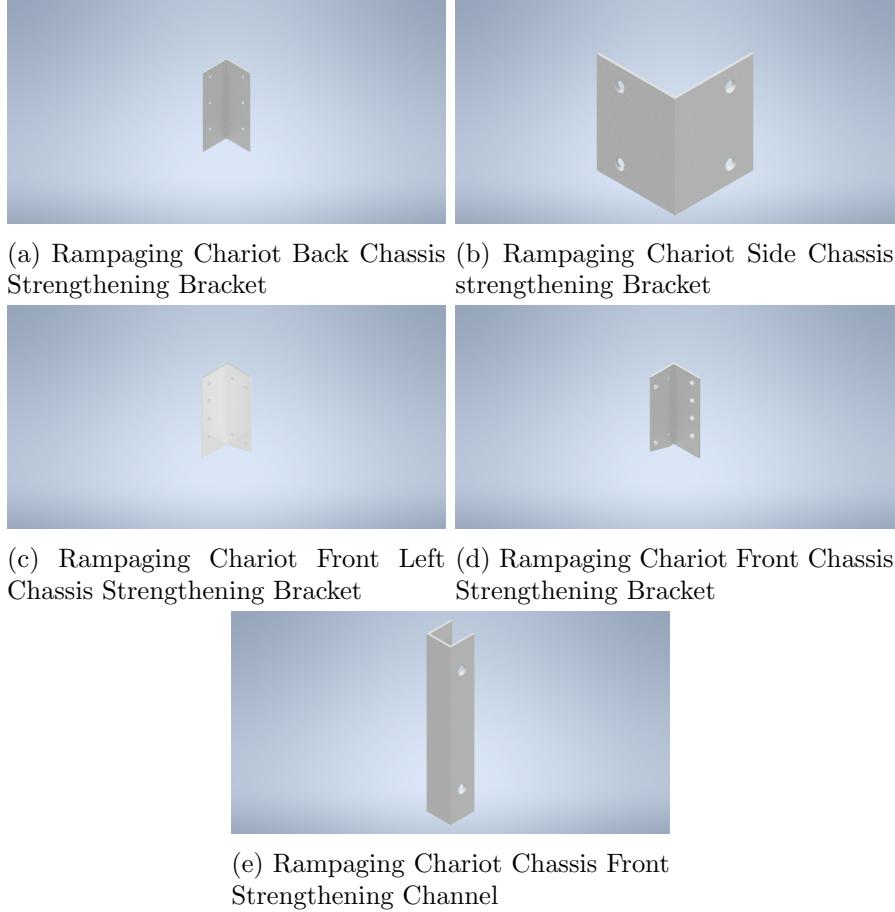
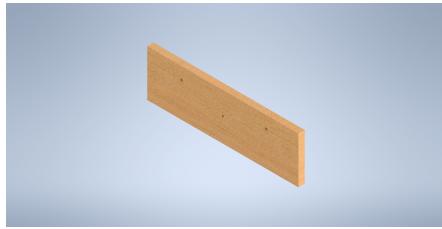


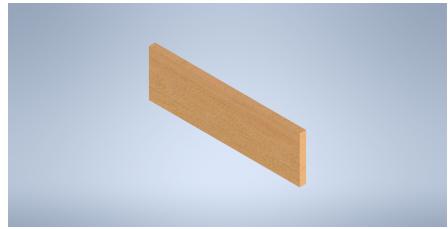
Figure H.2: Rampaging Chariots Strengthening Brackets CAD Models

H.1.2 Chassis Panels

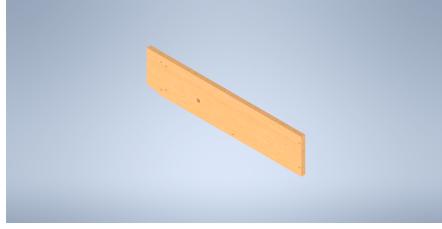
Appendix H. Mechanical Model Renders



(a) Rampaging Chariot Back MDF Panel



(b) Rampaging Chariot Front MDF Panel



(c) Rampaging Chariot Left Side MDF Panel



(d) Rampaging Chariot Right Side MDF Panel



(e) Rampaging Chariot Bottom MDF Panel

Figure H.3: Rampaging Chariot Chassis Panels

H.1.3 Wheel Assembly Parts

Appendix H. Mechanical Model Renders

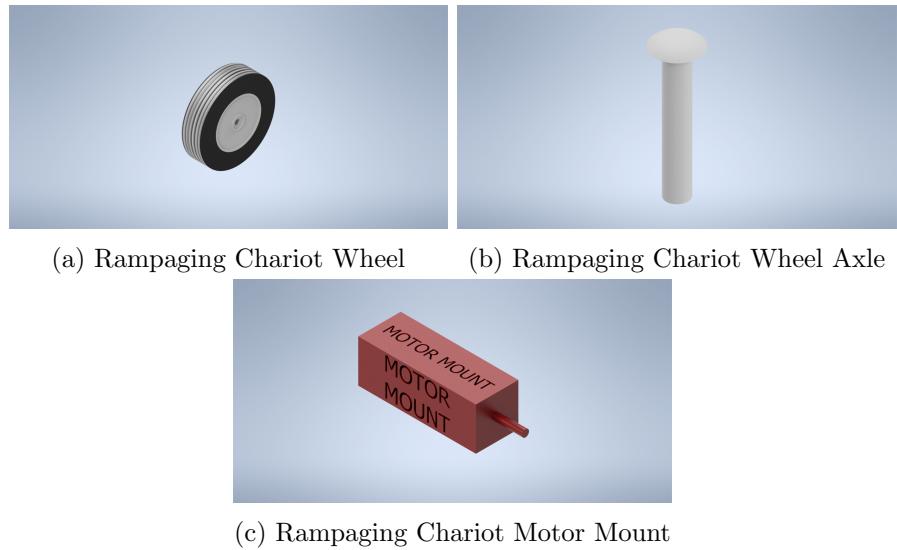


Figure H.4: Rampaging Chariots Wheel Assembly Parts

H.1.4 Electronic Specific Components

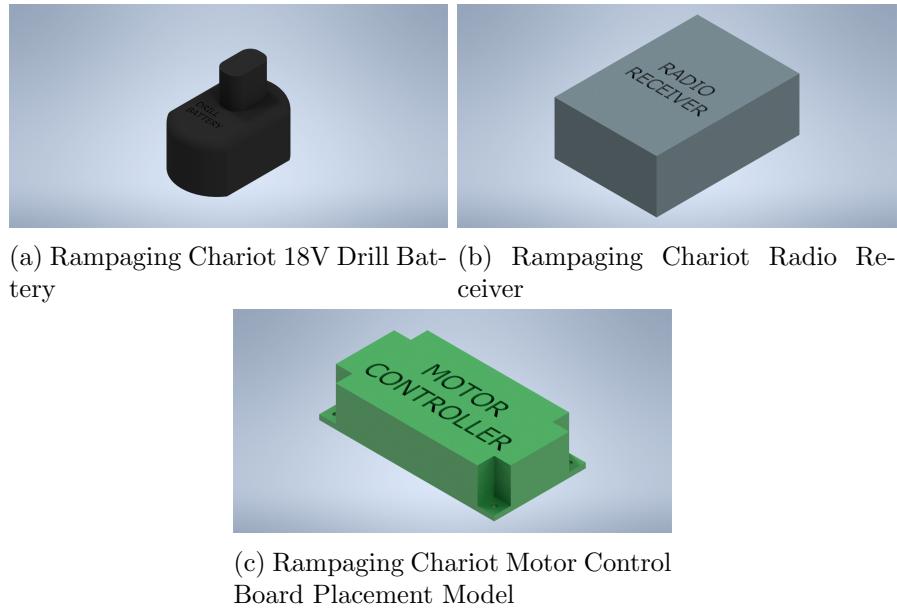
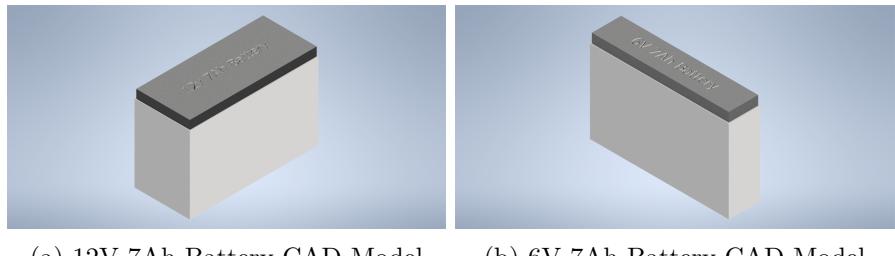


Figure H.5: Rampaging Chariots Electronic Specific Components

Appendix H. Mechanical Model Renders

H.2 Battery Holder

This section consists of all the renders regarding the Battery Holder CAD Model. It consists of multiple renders of the following parts Battery Holder, 12v 7Ah Battery and 6v 7Ah battery. The model renders shown in Figure H.6 are the individual battery renders.



(a) 12V 7Ah Battery CAD Model (b) 6V 7Ah Battery CAD Model

Figure H.6: Battery CAD Models Renders

The Battery Holder model itself is shown in Figure H.7. Finally the exploded and static view of the assemblies are shown in Figure H.8

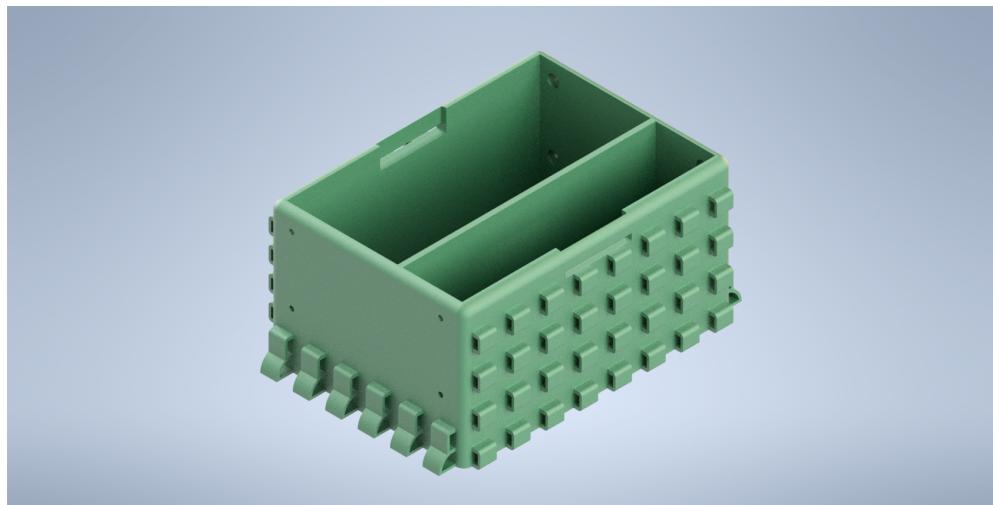
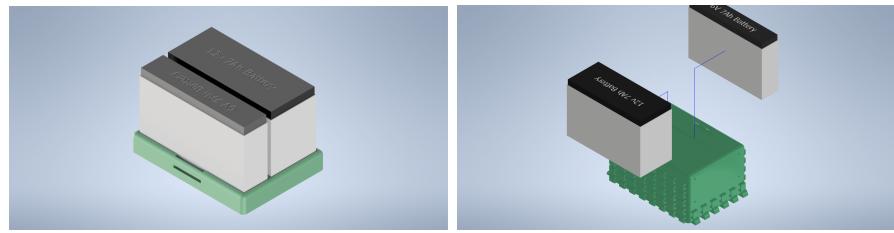


Figure H.7: Battery Holder CAD Model

Appendix H. Mechanical Model Renders

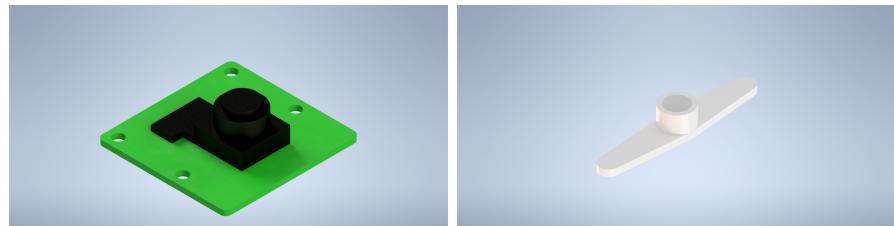


(a) Battery Holder CAD Model Assembly (b) Battery Holder CAD Model Exploded Assembly

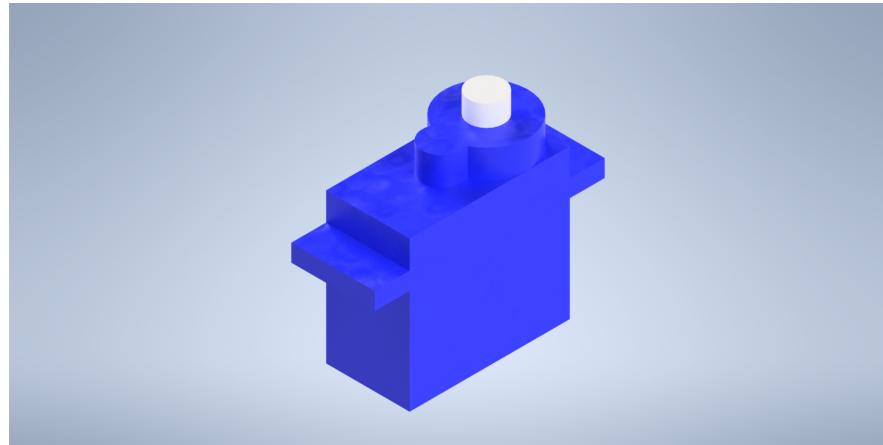
Figure H.8: Battery Holder CAD Model Assemblies

H.3 Camera Module

The camera module consists of six 3D printed parts and three predesigned parts. The predesigned parts include the Raspberry Pi Camera V2, SG90 micro servo motor and its corresponding rotor blade. Figure H.9 shows the predesigned parts.



(a) Raspberry Pi Camera V2 CAD Model (b) SG90 micro servo motor Rotor Blade CAD Model



(c) SG90 Micro-servo motor CAD Model

Figure H.9: Camera Module Predesigned Parts

Finally, Figure H.10 shows the camera module 3D printed parts and the correspond-

Appendix H. Mechanical Model Renders

ing assembly view for the camera module, along with the exploded view, is shown in Figure H.11.

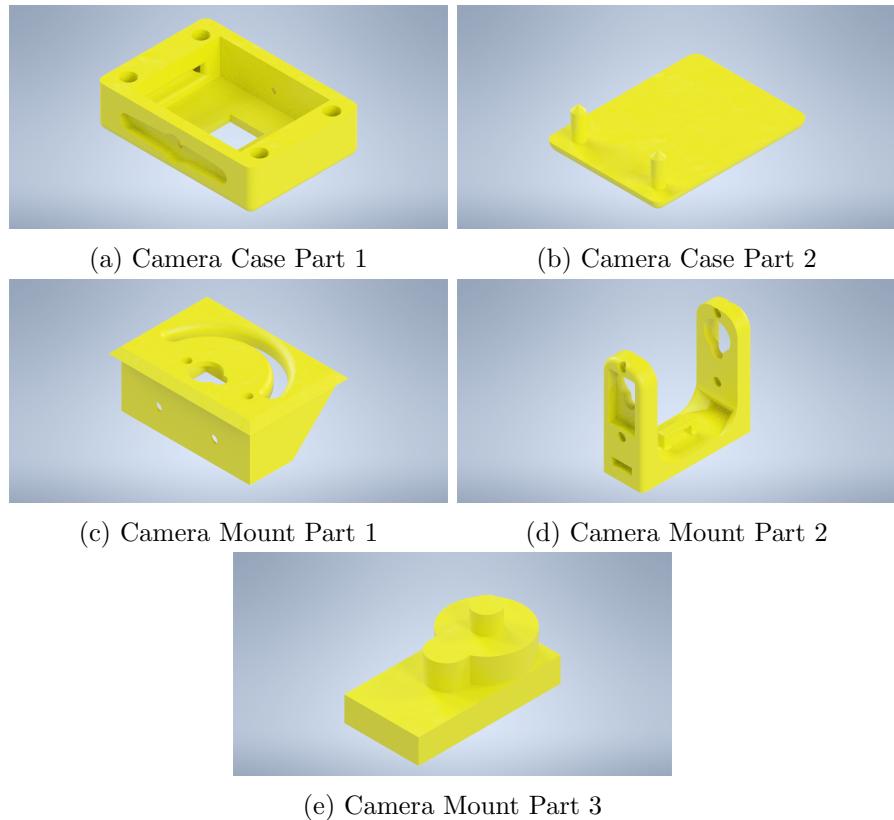
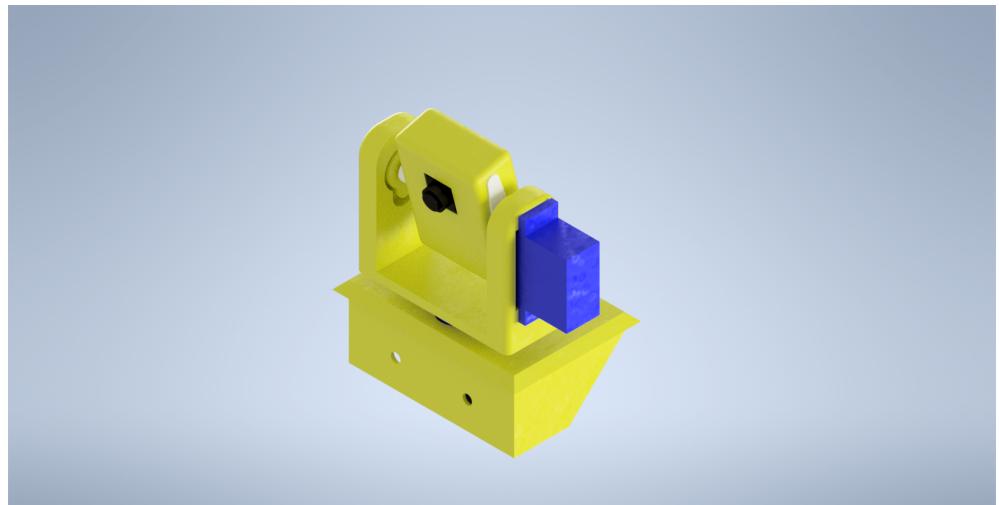
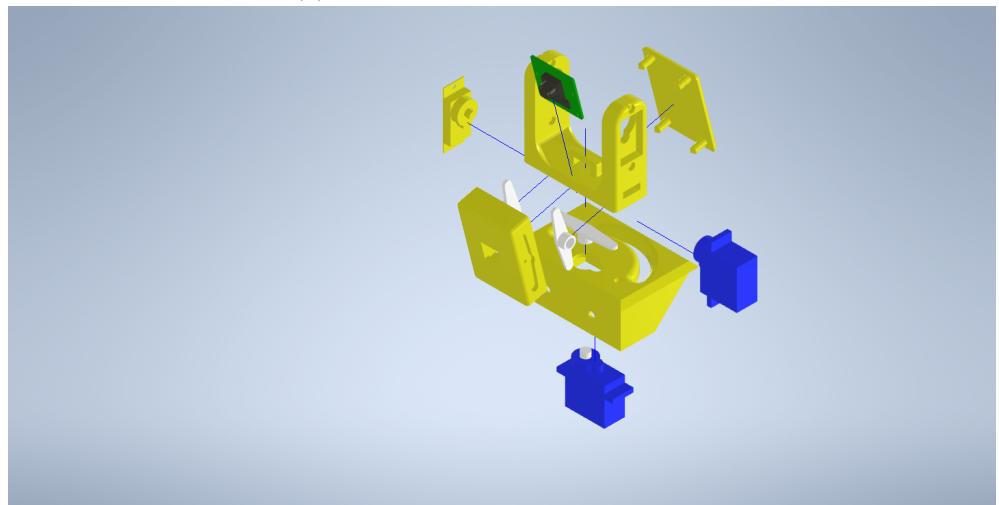


Figure H.10: Camera Module CAD Models

Appendix H. Mechanical Model Renders



(a) Camera Module CAD Assembly



(b) Camera Module CAD Exploded View

Figure H.11: Camera Module CAD Assemblies

H.4 Control Panel

The control panel consists of seven main parts(excluding screws, nuts and bolts), one 3D printed and the others are mechanical buttons used to control various aspects of the Robot along with a 16x4 LCD Screen. The button, switch and screen CAD models are shown in figure H.12.

Appendix H. Mechanical Model Renders

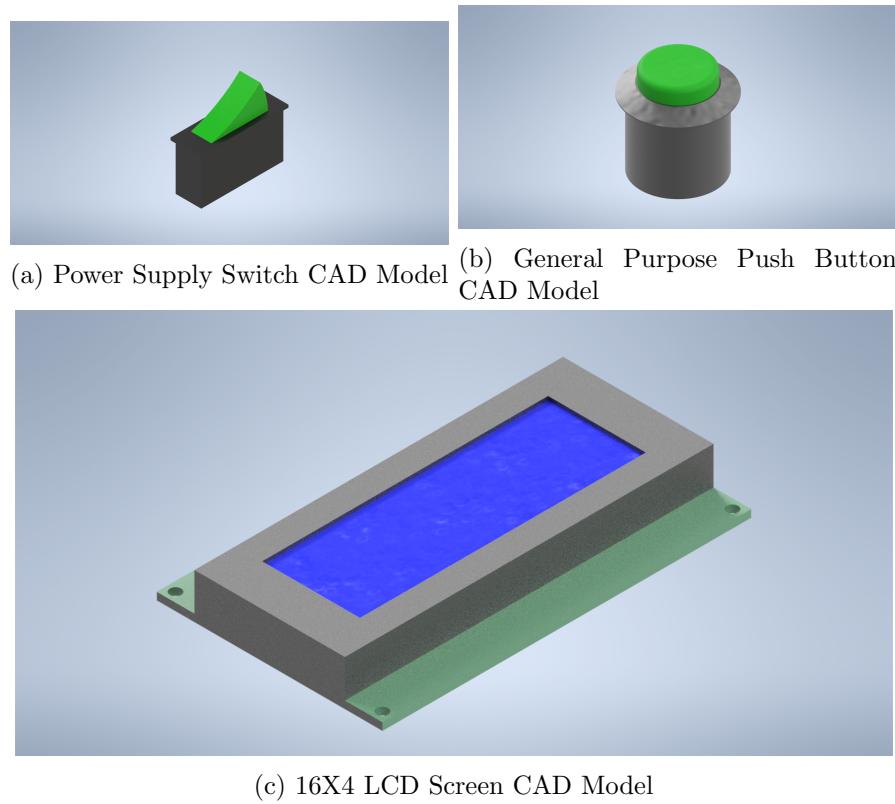


Figure H.12: Control Panel Button, Switch and Screen CAD Models

The control panel CAD Model is shown in figure H.13. The corresponding system Assembly along with the exploded view of the assembly is shown in figure H.14

Appendix H. Mechanical Model Renders

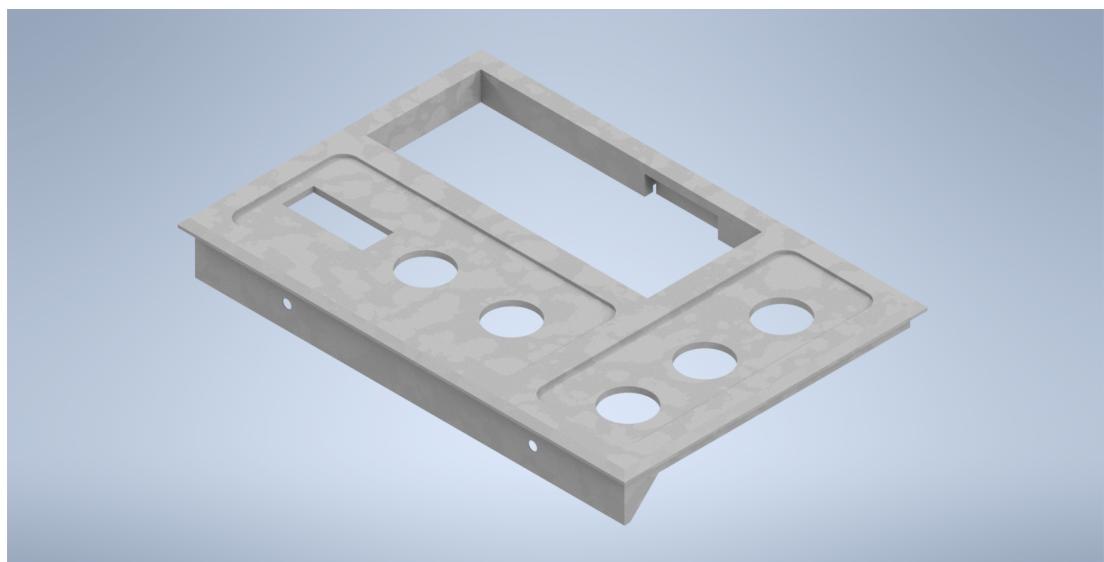
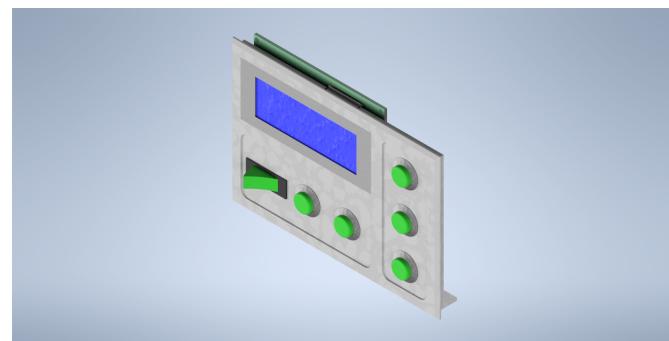
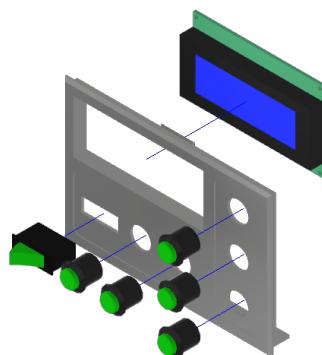


Figure H.13: Control Panel CAD Model

Appendix H. Mechanical Model Renders



(a) Control Panel Assembly

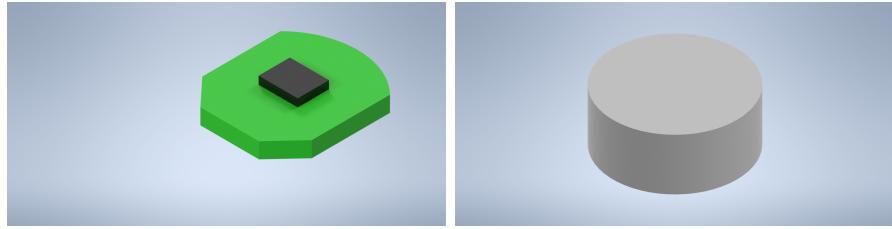


(b) Control Panel Exploded Assembly

Figure H.14: Control Panel CAD Model Assembly Views

H.5 Encoder Mounts

The robot requires four encoders for each wheel this requires two different mounting mechanisms, the resulting mechanisms are shown in Figures H.16 and H.17. The Sensor, along with the magnet, was also modelled and are shown in Figure H.15.



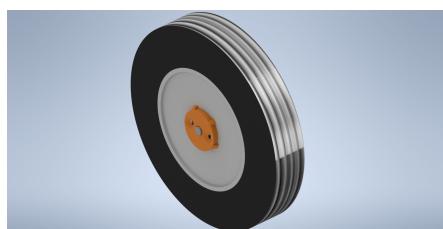
(a) AS5600 PCB

(b) AS5600 Encoder Magnet

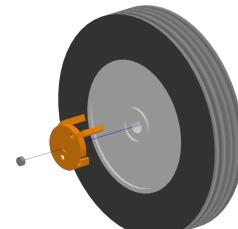
Figure H.15: AS5600 and Magnet CAD Models



(a) Drive Wheel Magnet Mount



(b) Drive Wheel Encoder Mount Assembly



(c) Drive Wheel Encoder Mount Exploded View

Figure H.16: Drive Wheel Encoder Mount CAD Models and Assemblies

Appendix H. Mechanical Model Renders

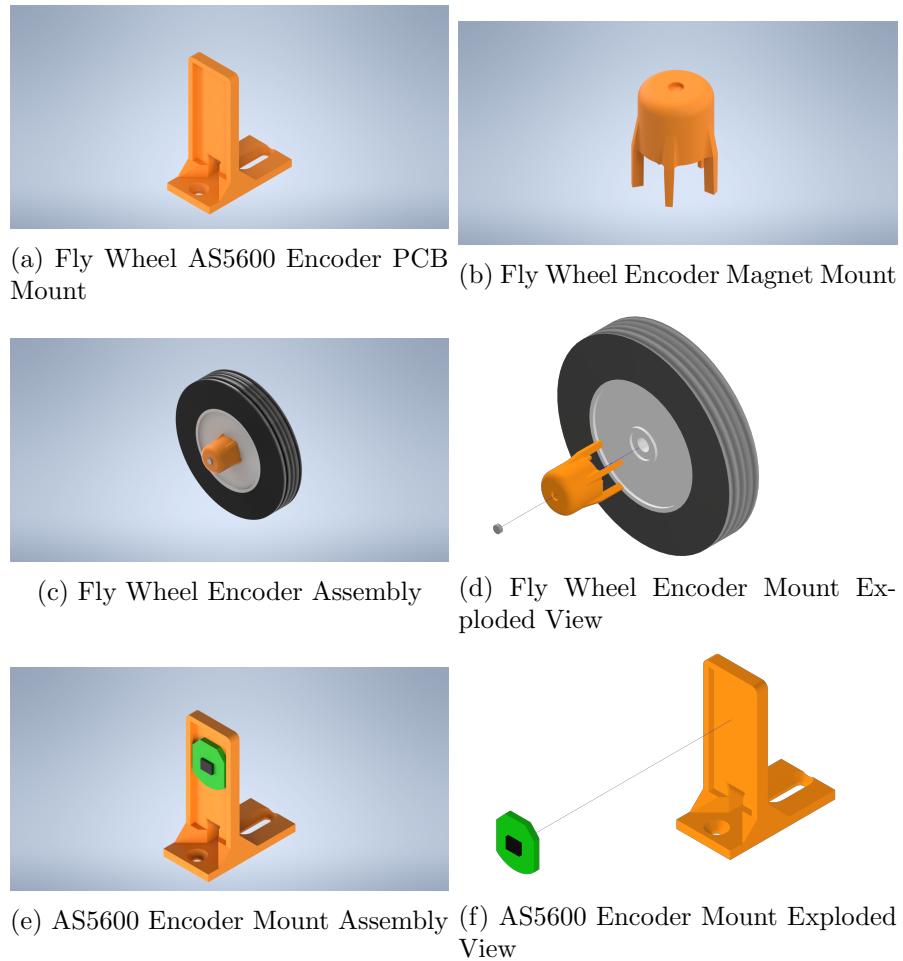


Figure H.17: Fly Wheel Encoder Mount CAD Models and Assemblies

H.6 Top Panel

The top panel section consists of three models, two of which are meant to be cut in MDF using appropriate woodworking techniques. The remaining model (Top Panel) should be cut using a laser cutting machine or other suitable devices. The Models are shown in Figure H.18.

Appendix H. Mechanical Model Renders

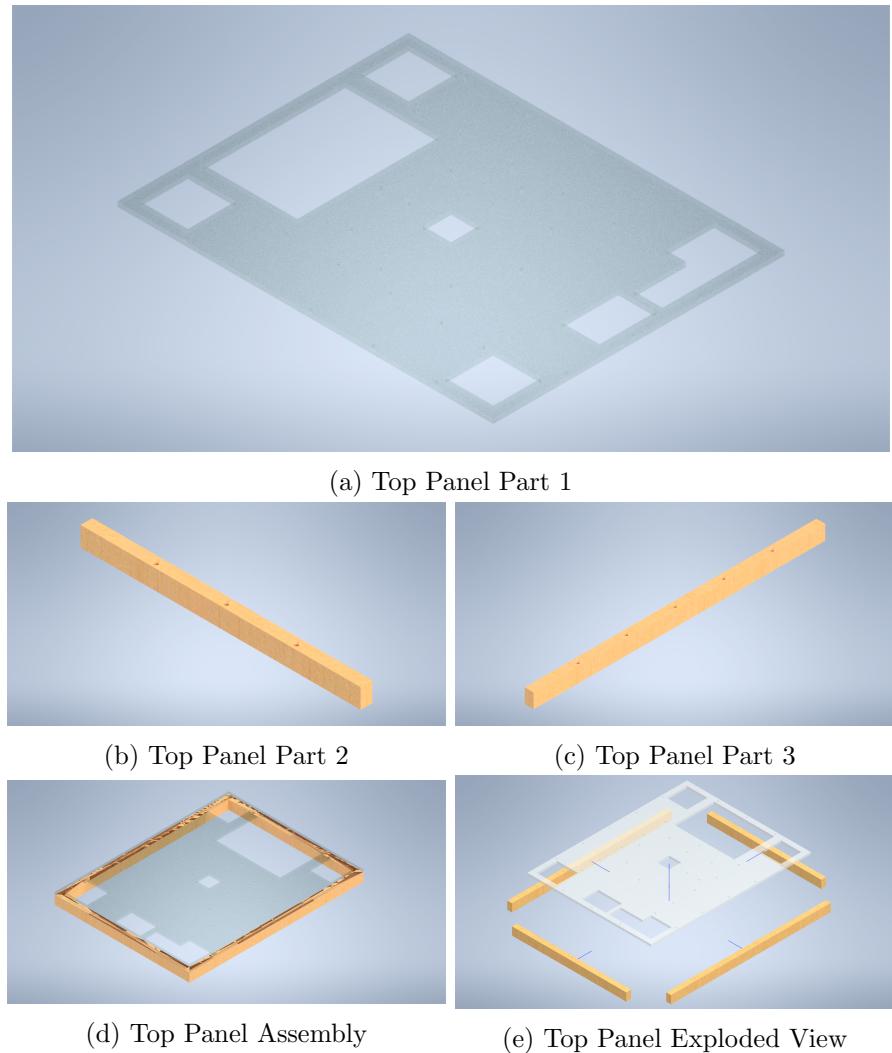


Figure H.18: Top Panel CAD Models and Assemblies

H.7 Inertial Measurement Unit Mount

The Inertial Measurement Unit(IMU) module was modelled along with its holder, these CAD Models along with the assembly and exploded view, is shown in Figure H.19.

Appendix H. Mechanical Model Renders

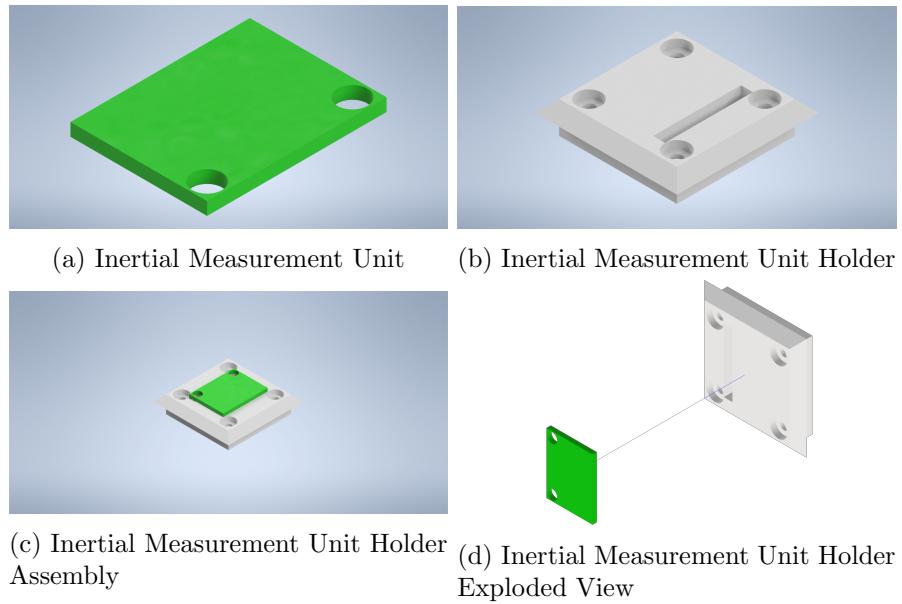


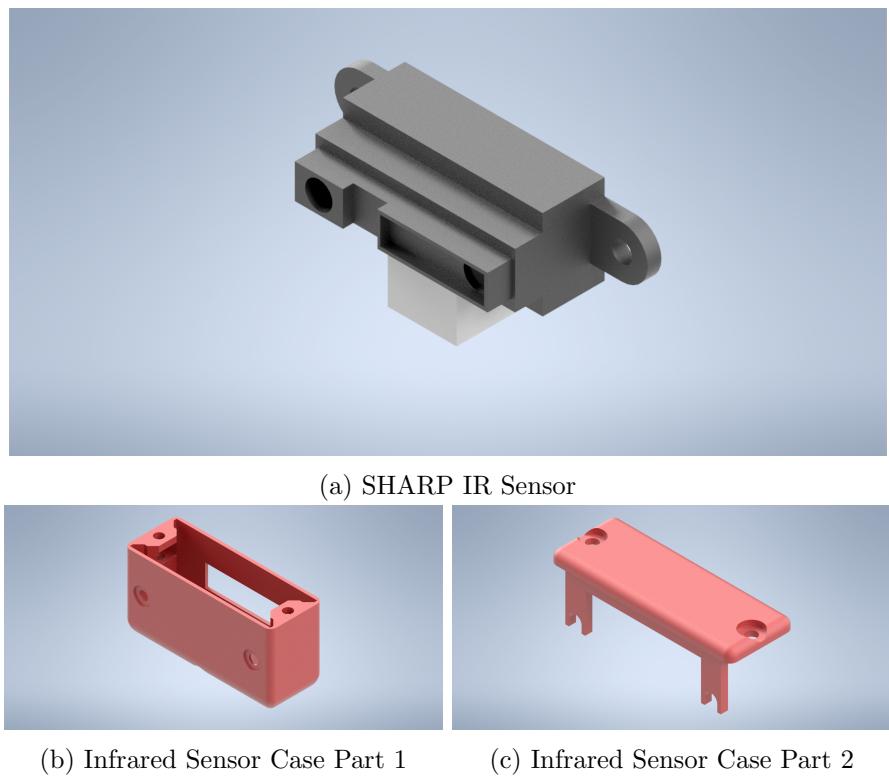
Figure H.19: Inertial Measurement Unit CAD Models and Assemblies

H.8 Infrared Sensor Mounts

To use the SHARP IR Sensor, CAD Models were created for the required mounts and the sensor. Due to the design of the robot, there are two designs, one for the front left corner and the other design for the remaining corners.

newline Figure H.20 shows the infrared sensor and the required casing parts.
newline The corresponding Front left mount and assemblies is shown in FigureH.21.
newline Finally the remaining mount and its assemblies are shown in FigureH.22.

Appendix H. Mechanical Model Renders



(a) SHARP IR Sensor (b) Infrared Sensor Case Part 1 (c) Infrared Sensor Case Part 2

Figure H.20: Infrared Sensor and Required Casing CAD Models

Appendix H. Mechanical Model Renders

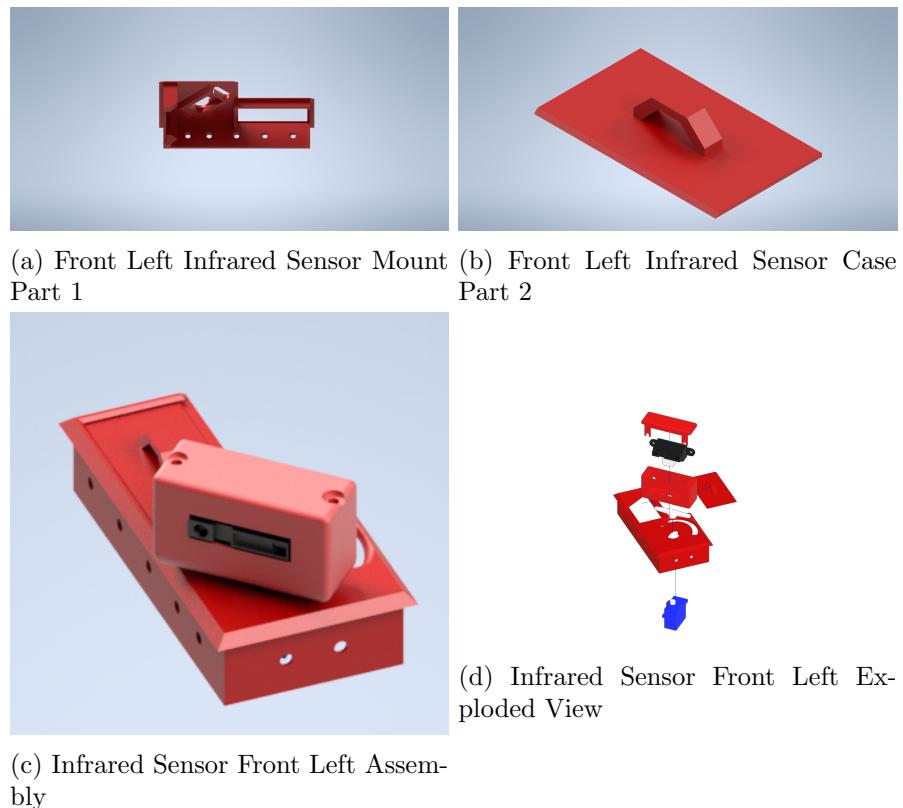


Figure H.21: Infrared Sensor Front Left Mount CAD Model and Assemblies

Appendix H. Mechanical Model Renders



(a) Infrared Sensor Corner Mount



(b) Infrared Sensor Corner Mount As-
sembly



(c) Infrared Sensor Corner Mount As-
sembly Exploded View

Figure H.22: Infrared Sensor Corner Mount and Assemblies

H.9 Ultrasonic Sensor Mounts

Due to the amount and positioning of the ultrasonic sensors, multiples designs were used, for Single Mount sensors see Figure H.24. For Front Corner Mounts see H.25 and finally for back corner mounts see H.25. The HC-SR04 was also modelled and is shown in Figure H.23.

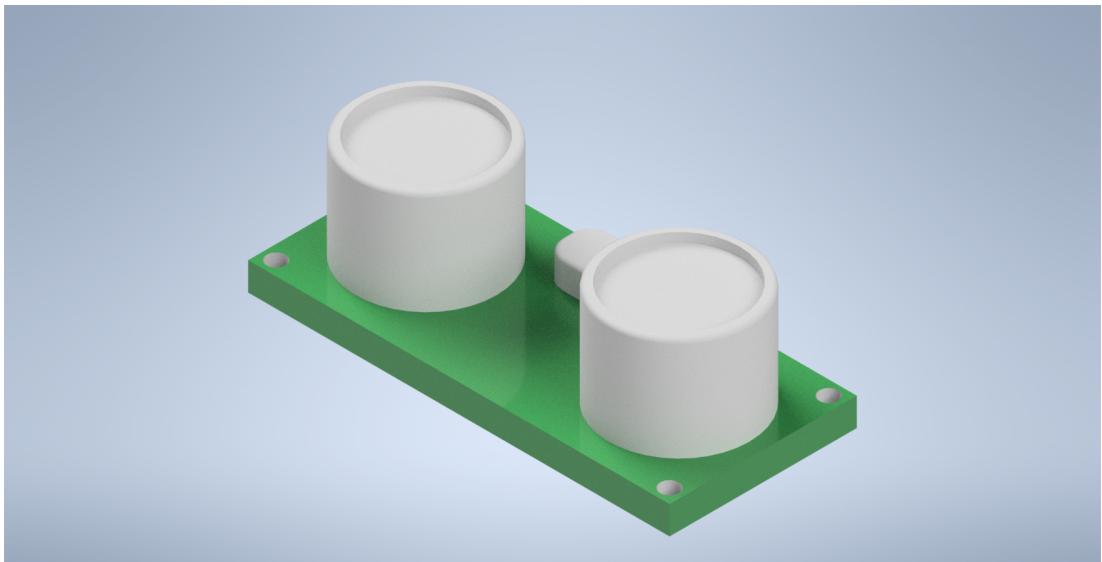
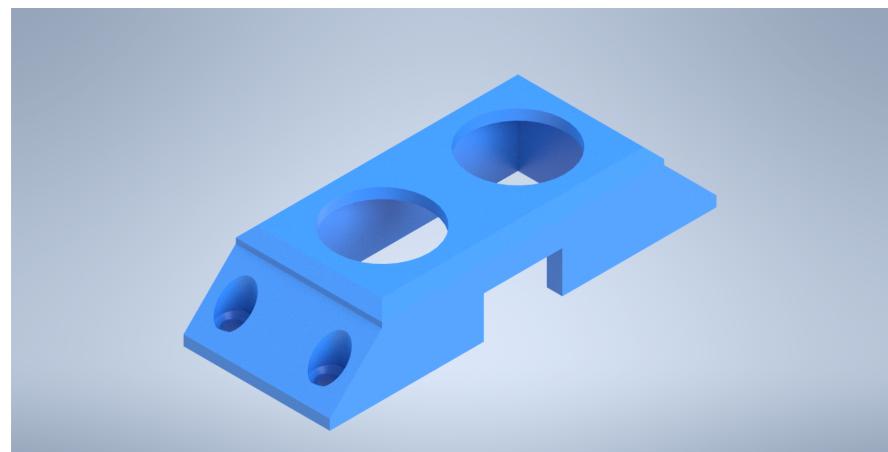


Figure H.23: HC-SR04 Ultrasonic Sensor CAD Model

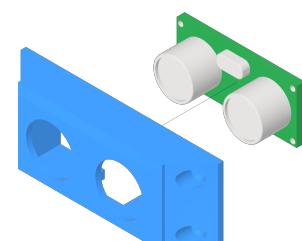
Appendix H. Mechanical Model Renders



(a) Ultrasonic Sensor Single Mount



(b) Ultrasonic Sensor Single Mount Exploded View
Assembly



(c) Ultrasonic Single Mount Assembly

Figure H.24: Single Mount Ultrasonic Sensor CAD Model and Assemblies

Appendix H. Mechanical Model Renders

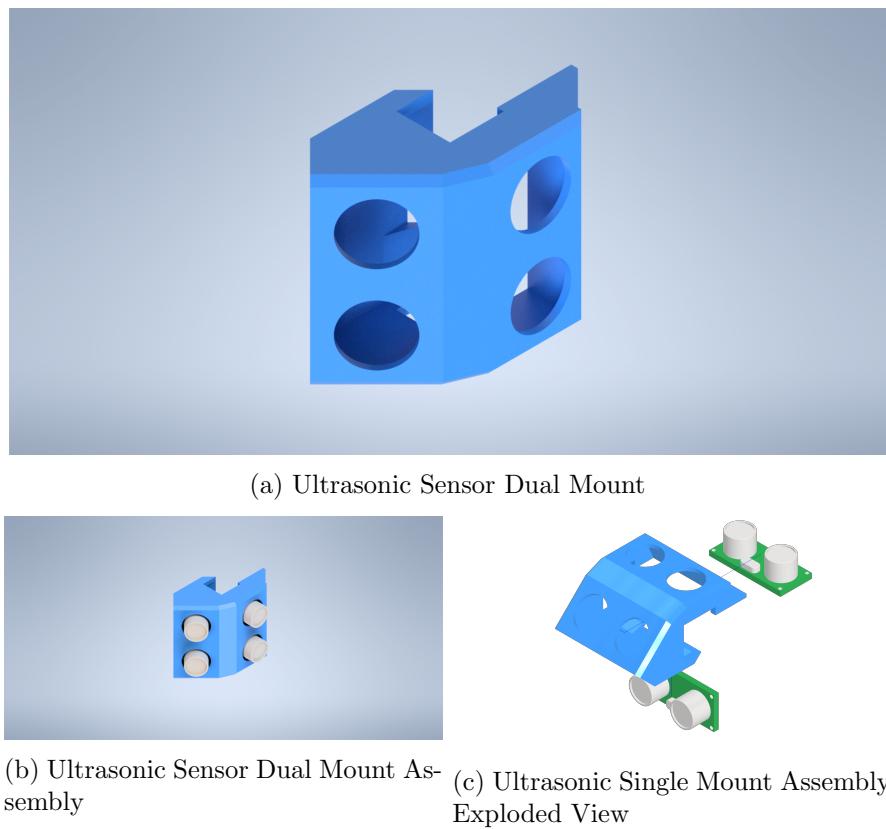


Figure H.25: Dual Mount Ultrasonic Sensor CAD Model and Assemblies

Appendix H. Mechanical Model Renders

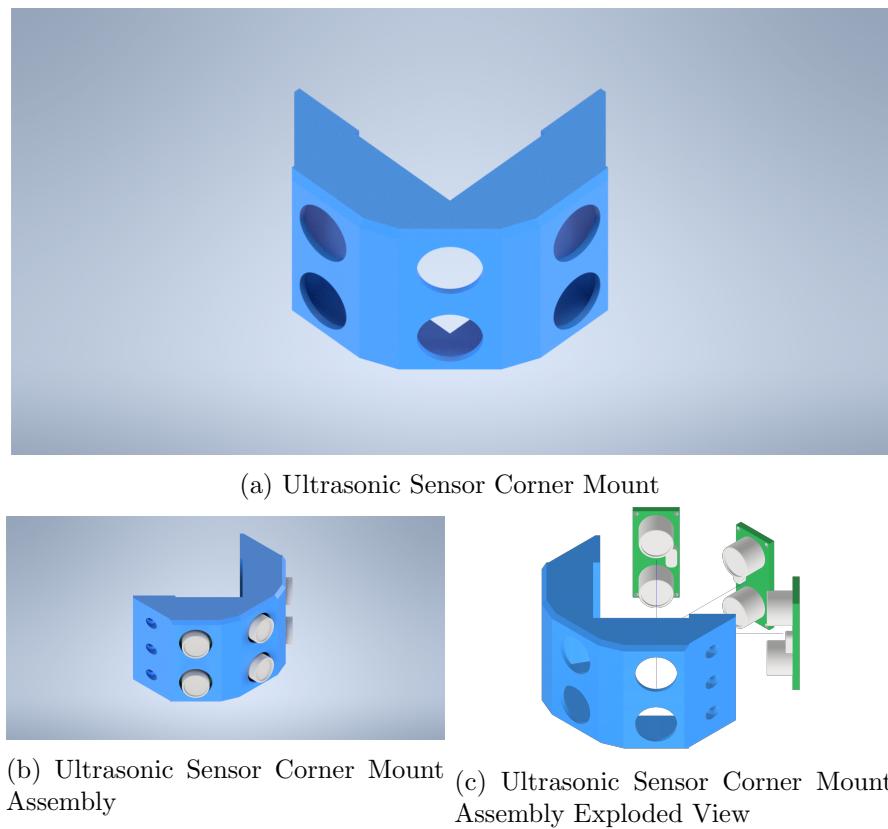


Figure H.26: Ultrasonic Sensor Corner Mount CAD Model and Assemblies

H.10 Cable Management Parts

Multiple different mounting parts are used for cable management purposes along with PCB mounting. The CAD models are shown in Figure H.27. As well as cable mounting parts, two cable divider parts were also designed and are shown in Figure H.28.

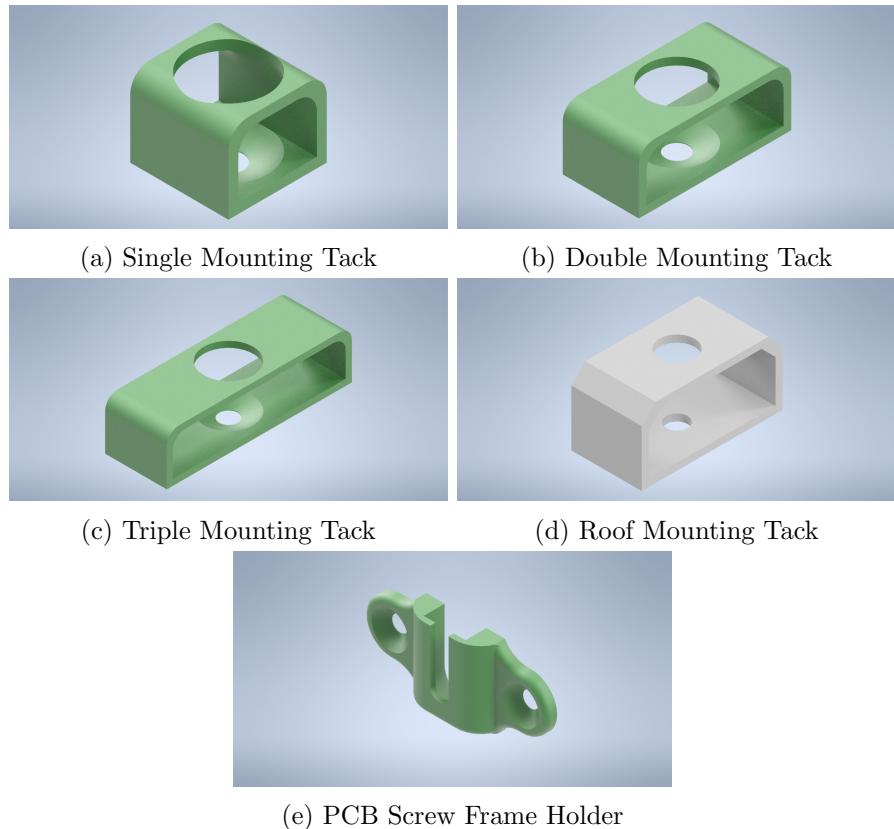


Figure H.27: Mounting Tack CAD Models

Appendix H. Mechanical Model Renders

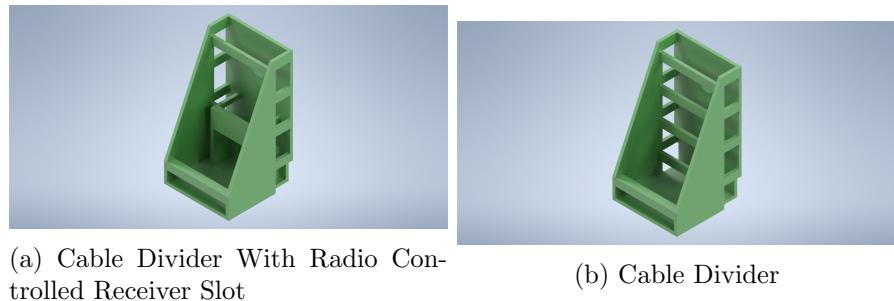


Figure H.28: Cable Divider CAD Models

H.11 PCB Footprints

The developed PCB Footprints for holding the different PCB's and Veroboard hardware solutions are shown in Figure H.29.

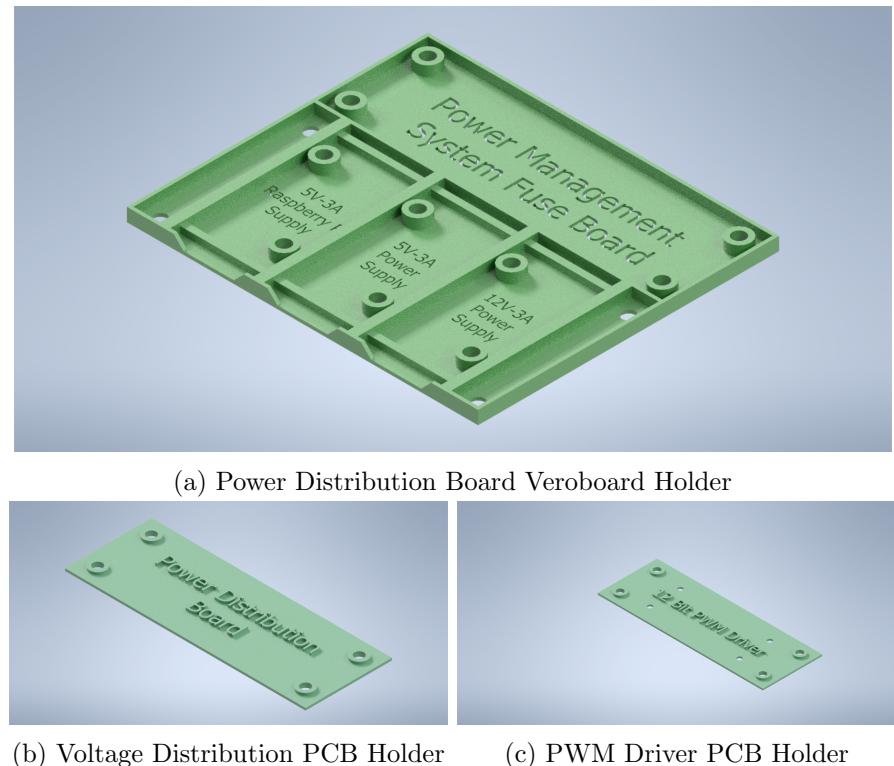
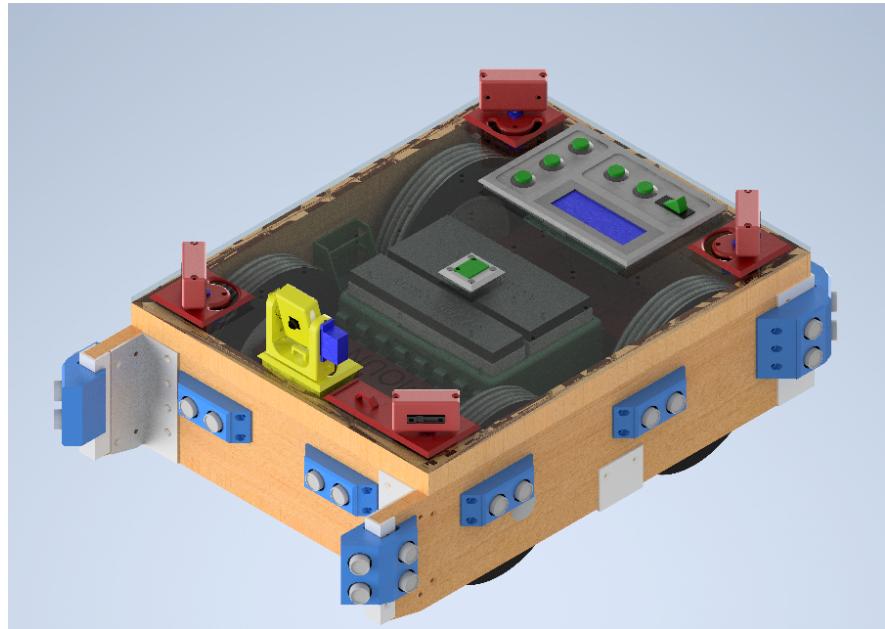


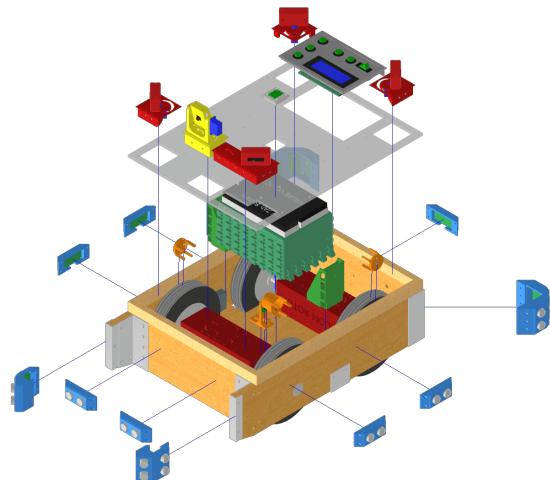
Figure H.29: PCB and Veroboard Holder CAD Models

H.12 Autonomous Robot Assembly

The completed Autonomous Robot Assembly is shown in Figure H.30 along with its exploded view showing all the individual sub-assemblies.



(a) Autonomous Robot Full Assembly



(b) Autonomous Robot Exploded View

Figure H.30: PCB and Veroboard Holder CAD Models

Appendix I

Robot Construction

The following appendix contains a variety of different pictures of the various components of the constructed Autonomous Robot.

I.1 Battery Holder

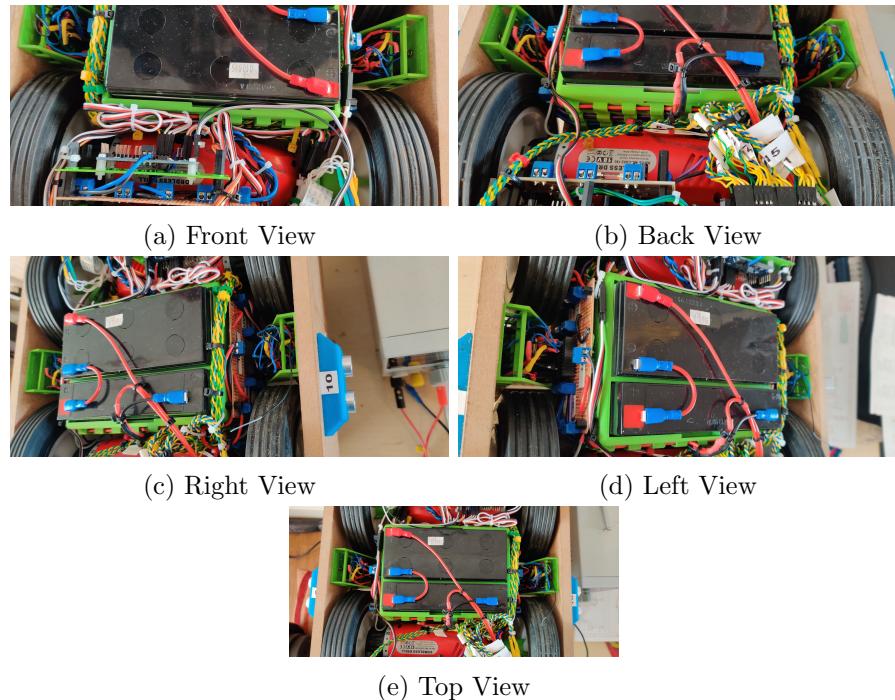


Figure I.1: Autonomous Robot Battery Holder Pictures

I.2 Camere Module Construction



Figure I.2: Camera Module Picture

I.3 Control Panel



Figure I.3: Control Panel

I.4 Inerial Measurement Unit

Appendix I. Robot Construction

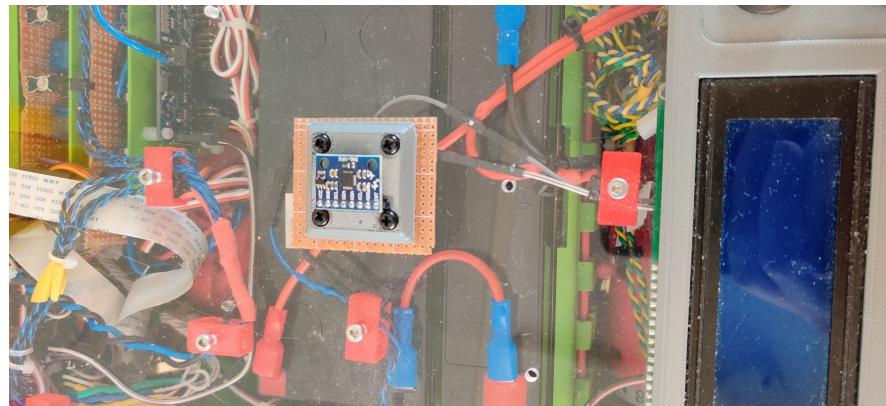
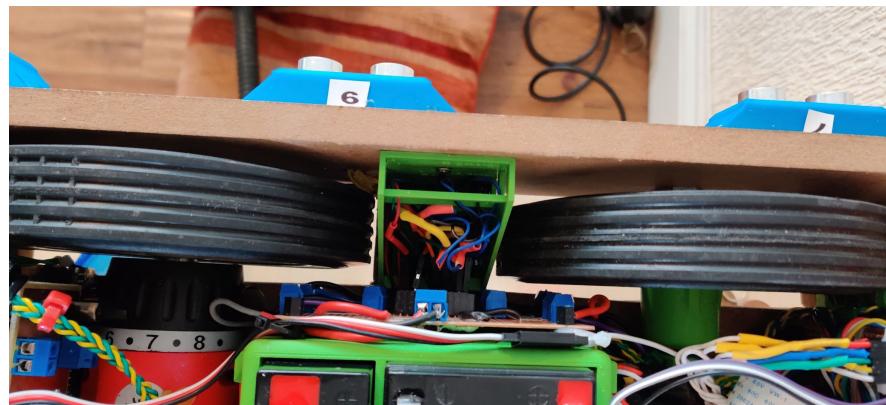
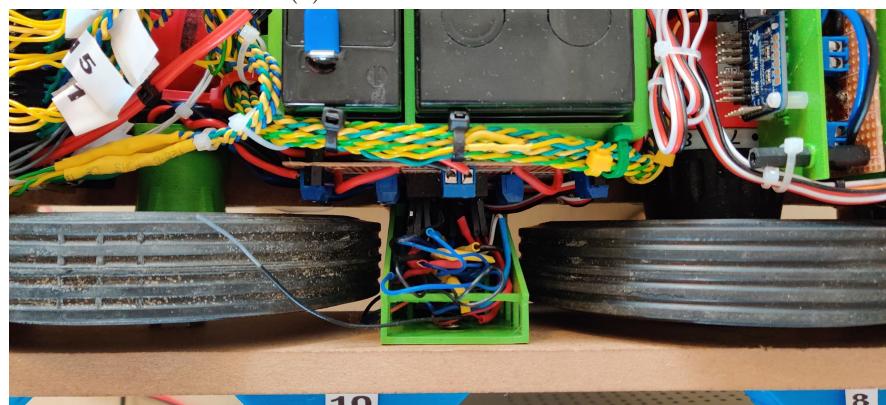


Figure I.4: IMU External Hardware Top View

I.5 Cable Management Dividers



(a) Left Hand Side dividers



(b) Right Hand Side dividers

Figure I.5: Robot Implemented Dividers

Appendix J

Printed Circuit Board Manufacturing Guide

J.1 Introduction

In order to make PCB's there were two options available for the project, using the in-house PCB manufacturing or to go outside of the university and find a PCB supplier suitable for the needs of the project.

J.2 University of Strathclyde PCB Workshop

The University Of Strathclyde does not provide silk skins or built-in vias and thus can only do two-layer PCB's at low resolution. The full details are shown in Table J.1.

Appendix J. Printed Circuit Board Manufacturing Guide

Table J.1: University Of Strathclyde PCB Tolerances

PCB Manufacturing Limits	
Dimension Limitations (mm)	
Maximum Board Size	80x100
Minimum Trace width	1.5
Minimum Clearance	1.5
Minimum Pad Size	2mm
Drill Size	0.8
Other Limitations	
Number of layers	2
Silk Skin	No
Solder Mask	No
Pre-made vias	No

J.3 JLCPCB PCB Prototype and Fabrication

The company chosen to make PCB's was JLCPCB(JiaLiChuang (HongKong) Co., Limited). The reason for this company is that they are relatively cheap and can do multiple layers. The full details are shown in Table J.2.

Table J.2: JLCPCB PCB Tolerances

PCB Manufacturing Limits	
Dimension Limitations (mm)	
Maximum Board Size	400x500
Minimum Board Size	5x5
Minimum Trace width	0.127
Minimum Clearance	0.127
Minimum Pad Size	0.45
Drill Size	0.2
Other Limitations	
Number of layers	6
Silk Skin	Yes
Solder Mask	Yes
Pre-made vias	Yes

Appendix K

3D Printing and Laser Cutting Guide

K.1 Introduction

Throughout the project, two 3D printers were used, the Wanhao I3 Duplicator V2 [58] and the Prusa I3 Mk3S [59]. Due to this reason, two different slicer programs were used Cura 4.4 [95] for the Wanhao and PrusaSlicer 2.1.1 [96] for the Prusa.

The project didn't just use 3D printing. The project also utilised laser cutting specifically Epilog Laser [98] from the Design Manufacture and Engineering Management(DMEM) department. This required Drawing Exchange Format(.dxf) files to be made of the Top Panel shown in figure H.18.

K.1.1 Prusa I3 Mk3S

The Prusa I3 Mk3S [59] used was the stock model and the relevant print settings are shown in Table K.1.

Appendix K. 3D Printing and Laser Cutting Guide

Table K.1: Prusa I3 Mk3S Print Settings

Prusa I3 Mk3S Print Settings	
Printing Temperature	
Nozzle Temperature	210 degrees
Bed Temperature	60 degrees
Layers and Perimeters	
Layer Height	0.15mm
First Layer Height	0.2mm
Perimeters	2mm
Top Solid Layers	7
Bottom Solid Layers	5
Infill Settings	
Fill Density	15%
Fill Pattern	Gyroid
Top Fill Pattern	Rectilinear
Bottom Fill Pattern	Rectilinear
Support Material	
Generate Support Material	Yes
Auto Generated Supports	Yes
Overhang Threshold	55 degrees
Pattern	Rectilinear
Print Move Speeds (mm/s)	
Perimeters	45
Small Perimeters	25
External Perimeters	25
Infill	80
Solid Infill	80
Top Solid Infill	40
Support Material	50
Bridges	30
Gap fill	40
Travel	180
First Layer	20

K.1.2 Wanhao I3 Duplicator V2

The Wanhao I3 Duplicator V2 was modified to include Z brace stabilisers and tightened belt tensioners. It was also contained in an enclosure to enhance prints. The relevant print settings are shown in Table K.2. [58]

Appendix K. 3D Printing and Laser Cutting Guide

Table K.2: Wanhao I3 Duplicator V2 Print Settings

Wanhao I3 Duplicator V2 Print Settings	
Quality (mm)	
Layer Height	0.15
Initial Layer Height	0.3
Line Width	0.4
Shell	
Wall Thickness	1.2mm
Wall Line Count	3
Top Surface Skin Layers	1
Top/Bottom Thickness	1.2mm
Top/Bottom Layers	8
Top/Bottom Pattern	Lines
Infill	
Infill Density	10%
Infill Line Distance	8mm
Infill Pattern	Grid
Material	
Printing Temperature	210 degrees
Bed Temperature	40 degrees
Retraction Distance	5mm
Retraction Speed	25mm/s
Speed (mm/s)	
Print Speed	70
Infill Speed	110
Wall Speed	60
Outer Wall	47
Inner Wall	94
Top Surface Skin	40
Top/Bottom	40
Support	70
Travel	150
Initial Layer	30
Initial Layer Travel	64.2857
Skirt/Brim	30
Number of Slower Layers	2
Support	
Generate Support	Yes
Support Placement	Everywhere
Support Overhang Angle	50 degrees
Support Pattern	Grid
Support Density	8%

K.1.3 University of Strathclyde Design Manufacture and Engineering Management Workshop

The workshop used one of the Epilog Laser cutters [98], with the following setting shown in Table K.3.

Table K.3: DMEM Laser Cutter

Design Manufacture and Engineering Management Laser Cutter Settings	
Cutting Speed	14%
Power	100%
Laser Fire Rate	5000Hz
Focal Length	2 inches

K.1.4 Filament Requirements

For best printing results using the Prusa and Wanhao Prints, the recommended filament [99–101] for each assembly is shown in Table K.4.

Appendix K. 3D Printing and Laser Cutting Guide

Table K.4: Autonomous Robot CAD Assembly Filament Information

CAD Assembly Filament Table			
CAD Module	Colour	Filament Brand	Filament Type
Battery Holder	Green	PrimaValue 1.75mm	PLA
Camera Module	Yellow	Prusa Flexfill 1.75mm	PLA
Control Panel	Grey	Prusa Flexfill 1.75mm	PLA
Encoder Assembly	Green	PrimaValue 1.75mm	PLA
Inertial Measurement Unit Assembly	Grey	Prusa Flexfill 1.75mm	PLA
Infrared Sensor Assembly	Red	PrimaValue 1.75mm	PLA
Ultrasonic Sensor Assembly	Azure Blue	Prusameent 1.75mm	PLA
Cable Management Parts	Red and Blue	PrimaValue 1.75mm	PLA

Appendix K. 3D Printing and Laser Cutting Guide

Appendix L

COVID-19 Setbacks

From the 16th of March, all face to face activities and teaching was stopped. As part of the project, a COVID-19 Impact form was to be filled out. The following sections give a detailed response to the form provided by the University

L.1 Cancellation of Face-To-Face Activities

The cancellation of face to face activities had a severe detriment to the completion of the project. During the final three weeks hardware was being finalised and the final versions of the design being constructed and tested. Since the university was closing the parts ordered through the Electronic and Electrical Engineering department Workshop were unable to be retrieved. This created a significant strain on the project. The week before the closure modified PCB designs were made in case the JLCPCB PCBS wouldn't make it to Scotland. This meant that they were and as such a scaling error caused all PCBs except two to be unusable. As well as this firmware was being tested and experiments requiring oscilloscopes and logic analysers were being run. These processes were ruined by the university closure in that the firmware could not be adequately tested as lab conditions were required to analyse the system. As well as this, the high-level software testing and implementation progress was reduced due to not being able to properly interact with all parts of the system.

L.2 Steps Undertaken to Mitigate Technical Risks to the Project

To mitigate the risk to the project, components were ordered online, and A soldering Kit was obtained from family. The focus was on developing more software and ensuring high-level communications worked. However, this was increasingly difficult due to the tight coupling between hardware and software on the robot. Although attempts were made to mitigate the risk to the project, the project did suffer much because of adverse working conditions. One such situation was the inability to focus due to the quarantine measure forcing the report to be completed in the developer's residence. The residence is a noisy tenement building with limited space and quickly crowded with four tenants in a kitchen designed for two. A new way of life was also adopted, which created much anxiety and as such slowed the progress.

L.3 Impacts of Regular Project Management Activities

COVID-19 affected project management activities significantly. During the University closure period, lecturers were unable to help students due to having to organise alternative methods of examination and coursework etc. This meant that Dr John Levine responses were limited. As well as this on the 1/4/20 John went into self-isolation as he developed symptoms of the Coronavirus and as such was unable to read and comment on the dissertation. However, contact was later resumed but to a limited extent due to other commitments as well as the ongoing symptoms. As such refactoring of the report was limited, and the structure could have been improved otherwise