Multiple Configuration Shell-Core Structured Robotic Manipulator with Interchangeable Mechatronic Joints

A thesis presented in partial fulfilment of the requirements
for the degree of

Masters of Engineering
in
Mechatronics

at
Massey University, Turitea Campus
Palmerston North
New Zealand

Jacques J.P. Janse van Vuuren
March 2017

Abstract

With the increase of robotic technology utilised throughout industry, the need for skilled labour in this area has increased also. As a result, education dealing with robotics has grown at both the high-school and tertiary educational level. Despite the range of pedagogical robots currently on the market, there seems to be a low variety of these systems specifically related to the types of robotic manipulator arms popular for industrial applications. Furthermore, a fixed-arm system is limited to only serve as an educational supplement for that specific configuration and therefore cannot demonstrate more than one of the numerous industrial-type robotic arms.

The Shell-Core structured robotic manipulator concept has been proposed to improve the quality and variety of available pedagogical robotic arm systems on the market. This is achieved by the reconfigurable nature of the concept, which incorporates shell and core structural units to make the construction of at least 5 mainstream industrial arms possible. The platform will be suitable, but not limited to use within the educational robotics industry at high-school and higher educational levels and may appeal to hobbyists.

Later dubbed SMILE (Smart Manipulator with Interchangeable Links and Effectors), the system utilises core units to provide either rotational or linear actuation in a single plane. A variety of shell units are then implemented as the body of the robotic arm, serving as appropriate offsets to achieve the required configuration. A prototype consisting of a limited number of *'building blocks'* was developed for proof-of-concept, found capable of achieving several of the proposed configurations.

The outcome of this research is encouraging, with a Massey patent search confirming the unique features of the proposed concept. The prototype system is an economic, easy to implement, plug and play, and multiple - configuration robotic manipulator, suitable for various applications.

Acknowledgements

I would like to extend a special thanks to my supervisor Liqiong Tang for her invaluable support and guidance throughout. I am very grateful to be part of her vision, in further developing her initial concept, conceptually and from an implementation perspective. Her assistance in helping me attain further financial support in the form of scholarships has been much appreciated and is the reason I am able to pursue this project further. I look forward to working with her in the future.

The contribution Huub Bakker has made to this project has also been noted. His useful control advice has saved me many hours I would have otherwise spent coming to the same conclusion, allowing me to spend my time elsewhere.

Thanks also to Trish O'Grady for the professional standard that she has uphold when ordering parts for the project.

I would also like to thank Will Haarhoff, whom I shared an 'office' with. I really enjoyed the occasional project and non-project related discussions. Your contribution may not have been direct, but has certainly helped to shape the outcome.

Thank you to my wife, Katy Johnston, who has been a great source of support. Her unique perspective on technology has influenced this project in ways I would not have considered.

Finally, I would like to thank all of those people involved in this project over the past year that I have not mentioned directly, they have helped make this research possible and contributed to its success.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3D: 3-Dimensional
CAD: Computer Aided Design
CAM: Computer Aided Manufacturing
CAN: Controller Area Network
CD: Compact Disc
CMY: Cyan Magenta Yellow
CNC: Computer Numerical Control
CPR: Commonplace Robotics
DC: Direct Current
DOF: Degrees Of Freedom
FEA: Finite Element Analysis
FSR: Force Sensing Resistor
GUI: Graphical User Interface
HP: High-Power
I^2C: Inter-Integrated Circuit
ID: Identification
IO or I/O: Input Output
LED: Light Emitting Diode
LSC: Laser Slit-scan Camera
MOSFET: Metal-Oxide Semiconductor Field-Effect Transistor
MP: Medium-Power
MPC: Model based Predictive Controller
MTTR: Mean Time To Repair
PC: Personal Computer
PCB: Printed Circuit Board
PI: Proportional Integral
PIC: Peripheral Interface Controller
PID: Proportional Integral Derivative
PSU: Power Supply Unit
PUMA: Programmable Universal Machine for Assembly
PWM: Pulse Width Modulation
RGB: Red Green Blue
ROS: Robotic Operating System
RPM: Revolutions Per Minute
SCARA: Selective Compliance Assembly Robot Arm
SDA: Serial Data Line
SLC: Serial Clock Line
SMILE: Smart Manipulator with Interchangeable Links and Effectors
SMT: Surface Mounted Technology
STEM: Science, Technology, Engineering and Mathematics
TITO: Two Input Two Output
UK: United Kingdom
US: United States
USB: Universal Serial Bus
USD: United States Dollar
VP: Velocity Profile

Chapter 1

Introduction

1.1 Background

Robotic manipulators have been an ever-growing aspect of industry since the robotic-industrial relationship was first documented in 1937, when the earliest known industrial robot was completed by 'Bill' Grifith P. Taylor, published in Meccano Magazine (Meccano Ltd, 1938, p.172). The first robotic patents were applied for in 1954 by George Devol, who later co-founded Unimation with Joseph F. Engelberger, installing GM's first spot-welding robots in 1969 (IFR Germany, 2012). Unimation later licensed their technology to Kawasaki Heavy Industries and GKN, resulting in what would later be referred to as the Japanese 'robot boom', peaking in 1984.

In 2006, the Japanese government estimated its robotic industry to be worth about $US 5.2 billion, further increasing to $US 26 billion in 2010. According to Tabuchi 2008, this figure is expected to rise to $US 70 billion by 2025. This translates to over 370,000 operational industrial robots in 2005, with the Japanese Trade Ministry calling for 1 million industrial robots to be installed throughout the country be 2025.

In an article by Kowles-Cutler et al. (2014), it is concluded that 35 per cent of today's jobs in the UK and 30 per cent in London are at high risk of becoming non-existent within the next two decades as a result of the impact of computers and robotics (fig. 1-1).

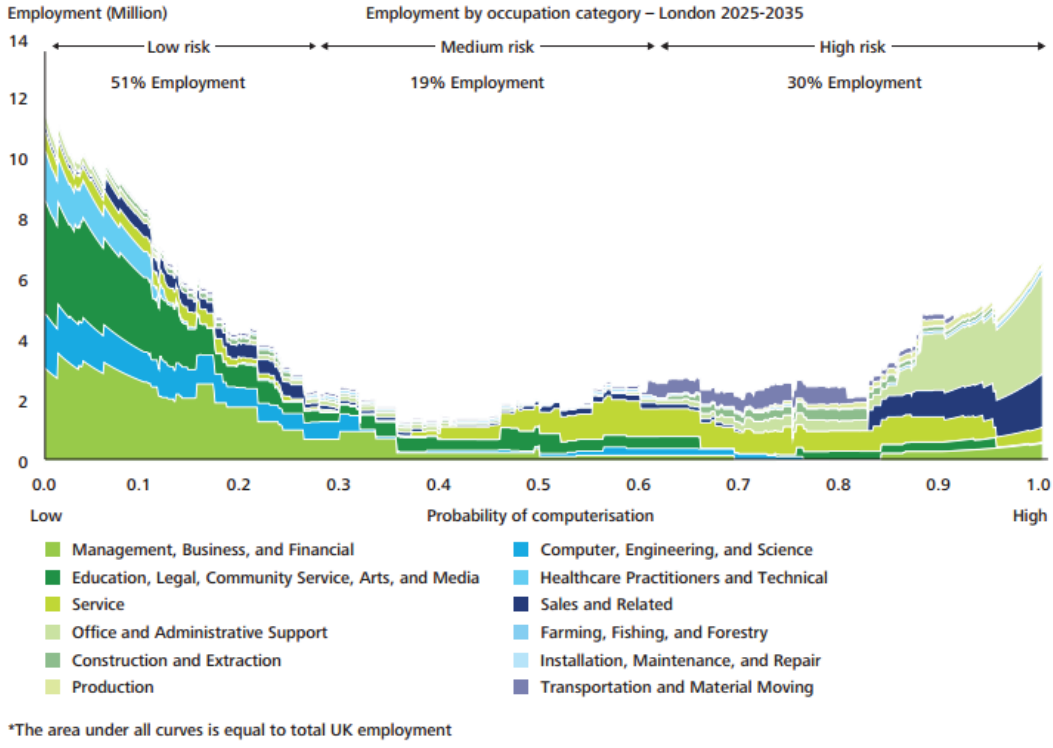


Figure 1 - 1. **London jobs at risk of computerisation. (Kowles-Cutler, et al., 2014). Copyright 2014 by Deloitte.**

The article states that London jobs with a pay rate of at least £100,000 per annum are more than eight times likely to survive automation after a period of two decades, when compared to jobs earning £30,000 or less (Kowles-Cutler et al., 2014). Frey and Osborne have also identified a change in the UK's workforce over the last 15 years, as workers have shifted from low skill, routine jobs to higher-skill, non-routine jobs.

Although over 800,000 jobs have been lost over this period, approximately 3.5 million have been created as a direct result of automation and computerisation (Kowles-Cutler, at al., 2014), with new jobs annually paying on average £10,000 more than their predecessor, boosting the net economy by approximately £140 billion. As a result, the skillsets prioritised by London business employers have shifted (fig. 1-2).

Figure 1 - 2. **Employer desired skill-sets. (Kowles-Cutler, et al., 2014). Copyright 2014 by Deloitte.**

As a result of the influx of robotics and automation in industry, the presence of robotics-related schooling has significantly increased at secondary and tertiary educational institutes around New Zealand, with Massey University first offering a Mechatronics Engineering degree in 1999. This has provided a stable platform for pedagogical robotic products at the high-school level, most popularly Vex Robotics. Vex introduces students to robot-like design aspects and implementation and encourages the type of generalised critical thinking required for robotic problem solving, however this does not expose the user to industrial-type manipulator arms.

Although a fixed-arm robotic system can be very useful serving as an education supplement, the scholastic value of such a system is limited to the robotic theory relating to the specific manipulator arm and therefore cannot physically demonstrate more than one of the varying types of arms currently popular for industrial use. This gained insight seemed to suggest a potential market gap and lead to the development of the reconfigurable robotic arm concept.

The Shell-Core Structured Robotic Manipulator Arm concept aims to incorporate modularity to the extent that all mainstream manipulator arms may be constructed with the system through the use of two types of actuators, linear and rotational.

1.2 Research Topic

Robotic manipulators have different configurations, shown in Figure 1-3. Each has its own unique features. The aim of this research is to develop a future-focused educational robotic manipulator system that incorporates modern design concepts and methodologies capable of supporting all mainstream robotic manipulator construction and learning. Such a robotic platform aims to be suitable not only to educational organisations and hobbyists but may also be a considered for non-high-precision applications.

1.3 Scope of Research

The scope of this research extends to the development of a prototype for the purpose of investigating the viability and methodologies surrounding the related physical implementation aspects of a robotic manipulator system that is flexible enough for mainly robotic manipulator pedagogical purposes.

- Investigate the current pedagogical robotic manipulator market and understand the limitations and drawbacks for future robotic education. Study the configurations offered by the current educational robotic industry. Research the available control methodologies used by educational robotic manipulator systems.
- Propose a robotic system that could better meet the current demand on robotic learning. Although fixed manipulator arms can help teach underlying robotic theory, they are inherently limited to only expose the user to that specific type of robot and therefore cannot physically demonstrate the multiple types of arms currently popular for industrial use. The proposed outcome of this research may be very useful in serving as a supplement for robotics education.
- Design an initial system platform capable of achieving different robotic manipulator configurations, flexible enough to accommodate for varying effectors.
- Design and implement an end-effector gripper for the proposed system.
- Develop a robust control basis for the proposed robotic system, which may allow for future advanced control of multiple elements acting in unison.
- Develop a physical prototype of the proposed robotic system for proof-of-concept, capable of achieving at least 2 configurations.

The initial concept includes two types of actuators, prismatic and revolute, with each providing either linear or rotational motion respectively. Structural units will also be included to inter-connect these actuators, providing the offsets needed to produce the required configurations, eg: 90° bend, straight off-set connection, flipped straight off-set connection. Varying end-effectors may be included depending on application, eg: electro-magnetic, 2-fingered gripper, 3-fingered gripper, drawing utensil attachment. Passive actuators may also be considered.

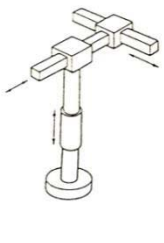
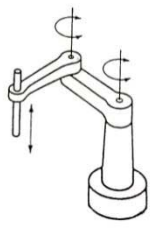
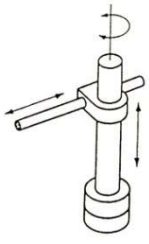
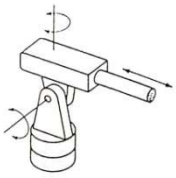
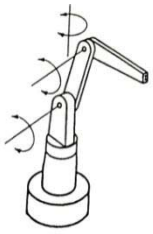
Cartesian	SCARA	Cylindrical	Spherical (Polar)	Articulated
				
3 linear	1 linear 2 rotational	2 linear 1 rotational	1 linear 2 rotational	3+ rotational

Figure 1- 3. **Basic manipulator arms and the actuators needed to achieve configuration. (Meier, 2016).**

Due to the nature of inter-connecting, modular robotics, it is preferable that each actuator will include its own, on-board control electronics as well as pass along both data and power to the next actuator in the queue. Each type of structural element has a unique ID

correlating to its specific dimensions, which are passed along to the control unit that will automatically adjust its control methodology to suit the unique configuration.

1.4 Organisation of Dissertation

This dissertation contains nine chapters and one appendix. Each of the following passages briefly describes the contents of each chapter or appendix in this thesis.

Chapter 1 provides brief context and outlines the introduction and scope of research.

Chapter 2 presents and studies relevant products currently popular for use in a robotics related educational field. This chapter also includes applicable control methodologies, examining the way in which common control problems have been solved in the past.

Chapter 3 details the Shell-Core reconfigurable manipulator concept at the core of this thesis.

Chapter 4 focuses on Core-type components, covering their design, componentry and programming implementation.

Chapter 5 covers the 2-fingered gripper design, componentry used and programming control methodologies.

Chapter 6 describes Shell-type structural design considerations.

Chapter 7 covers the implemented system communication protocols and provides design and componentry details relating to the control box and circuit-boards.

Chapter 8 provides details regarding the final produced prototype in terms of the physical outcome and control results.

Chapter 9 discusses the novelty of the proposed concept, control difficulties and prototype advantages and disadvantages. This chapter also discusses the viability for future development of the prototype and concept.

Appendix A is an accumulation of prototype pictures.

Chapter 2

Literature Review

2.1 Background

The nature of this project may be described as a combination of new product development and the evaluation of the viability and practical implementation requirements of a reconfigurable, modular robotic manipulator for pedagogical purposes. As such, market research is crucial and key in determining novelty of the proposed concept. Since it is intended that this concept will eventuate into a product, the purpose of the literature review in this context is to assess not only design aspects of products on the market, but their accompanying documentation and consumer related support as well.

Fine manipulator control is very important for robotic applications, therefore a large focus has been put on examining the way in which relevant control problems have been solved by others. Future control additions have also briefly been touched upon, to assure the final prototype system is capable of supporting further advanced control.

It is not the explicit intent of this literature review to assess or critique material, but rather to present any content relevant to research in a way that will make it useable in producing the final system and any associated methodologies.

2.2 Market Research - Industrial Robotic Systems

2.2.1 SCORBOT-ER 9PRO

Figure 2 - 1. **Scorbot-ER 9Pro picture. (Intelitek, 2008).**

Intelitek is a privately owned company originating from Derry, New Hampshire. Founded in 1982, the company has since spread worldwide, with customers in North America, South America, Europe, Asia, the Middle East and Australia. With a strong concentration on pedagogical technology, Intelitek has immersed itself in fields that include engineering, robotics, advanced manufacture, automation, agriculture and STEM, providing systems which cater to CAD / CAM, robotics and machine vision learning.

In 2002, Intelitek provided a $350,000 USD grant to Gilbert High School, Arizona, installing a state-of-the-art robotics lab in the school's former wood work shop. Intelitek is also heavily involved in VEX, sponsoring the world championship in 2011. Despite the strong educational aspect to the company, some of the available products meet industry standards and thus, are used for such purposes. The Scorbot-ER 9Pro is one of those robots. This system offers advanced path control, speed and accuracy, resulting in major industrial uses including CNC machining, CNC routing and most popularly, laser engraving. At a cost of approximately $20,000 USD, this system sits at the high end of the price spectrum for educational robots. However, as far as industrial robotics is concerned, this system is extremely cost-effective.

All Scorbot manipulator arms function through the use of the Intelitek USB-PRO controller, enabling multi-tasking, real time control and synchronisation of up to 8 axes. 16 digital IO and 4 analogue IO are included (Intelitek, 2008, p.13). While the Scorbot-ER 9PRO integrates with industry-standard robotic control software, Intelitek includes proprietary software with each purchase. SCORBASE is a graphically orientated control and simulation software package that allows arm manipulation through the use of 3D control methodologies.

This 5 DOF manipulator arm incorporates the use of optical switches and an encoder index pulse on each axis, enabling for a smooth and accurate homing calibration. Positional feedback is provided by the same incremental optical encoders used for homing.

Weight	54.4 kg
Speed	1.9 ms^{-1}
Operating radius	691 mm
Payload (Full acceleration)	2 kg
Payload (Reduced acceleration)	5 kg
Repeatability	± 0.05 mm
Positional feedback	Optical encoders

Table 2 - 1. **Scorbot specifications. (Intelitek, 2008, p.13).**

Scorbase

Scorbase is a proprietary software package which works in conjunction with the Intelitek USB-PRO controller through the USB interface to control the connected robotic manipulator arm. It should be noted that the control methodology implemented is not graphically orientated. A pseudo-spreadsheet command-line is used to determine movement instructions with each cell containing values relating to arm manipulation.

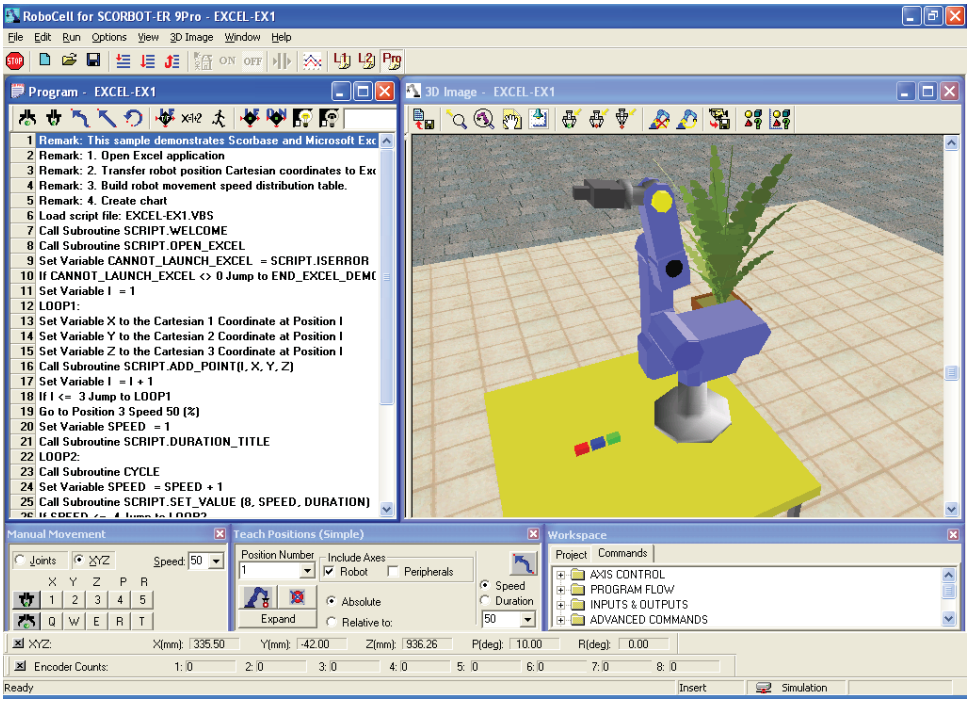


Figure 2 - 2. **Scorbase graphical user interface. (Intelitek, 2009).**

Several graphical representations are made available by Scorbase. The manipulator arm is constructed in a 3D environment to simulate movement online or offline, providing the user with a visual aid. Charts are also provided. Scorbase charts can be configured to display encoder counts, positional error and PWM value at controller output. Positional error is calculated as the difference between required axis position and the actual axis position (Intelitek, 2009, p.29) obtained through encoder counts.

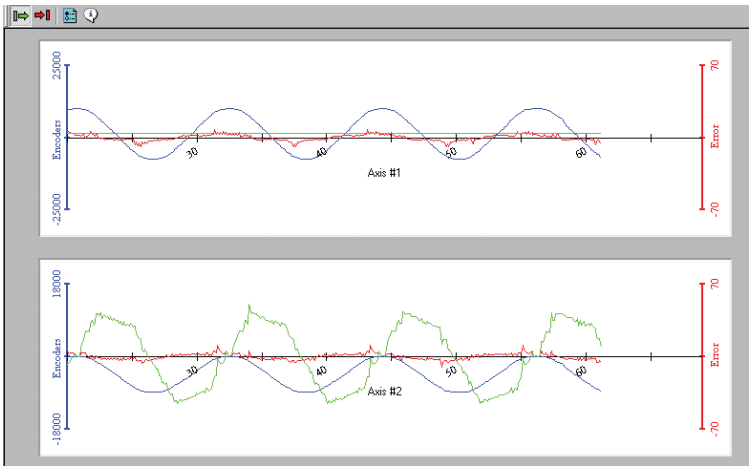


Figure 2 - 3. **Scorbase positional feedback. (Intelitek, 2009).**

The programming package allows for two homing features. 'Search Home' is the procedure for homing, which moves all axes to home position and initialises encoder counts. The 'Go Home' command simply moves all axes back to home position; however the encoder count remains unchanged. Peripherals and individual axes may be homed separately. Scorbase supports two coordinate systems, Cartesian (X,Y,Z) and joint angle positions (Intelitek, 2009, p.40).

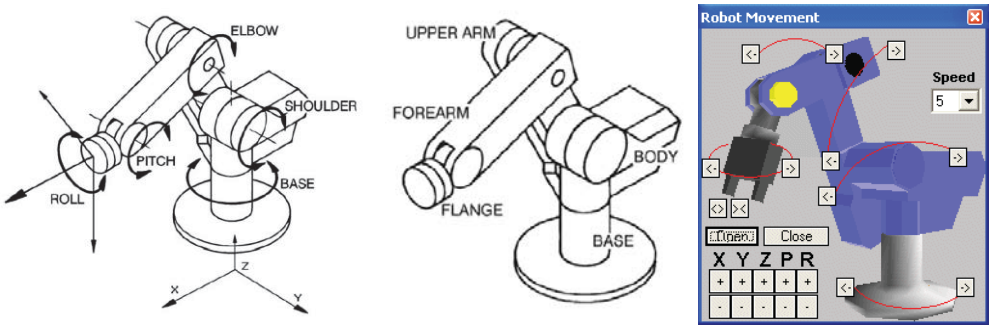


Figure 2 - 4. **Scorbot coordinate system overviews. (Intelitek, 2009).**

The 'Robot Movement' command dialogue box (fig. 2 - 4) allows for real-time movement of the manipulator arm by the user, both joint angle and Cartesian coordinates are adjustable. It should be noted that there is no mention of any singularity / collision control throughout the user manual accompanying Scorbase. All 20 input and output pins on the USB-PRO controller are also programmable through the use of the spreadsheet style programming language.

USB-PRO Controller

The USB-PRO controller is at the core of all Scorbot systems, providing both power and advanced control to the manipular arms, while enabling human interaction through the GUI interface. The correct version of Scorbase is required to allow PC-to-controller communication (Intelitek, 2008, p.13). This is somewhat specific, as both the controller and software accompany the manipulator arm version closely.

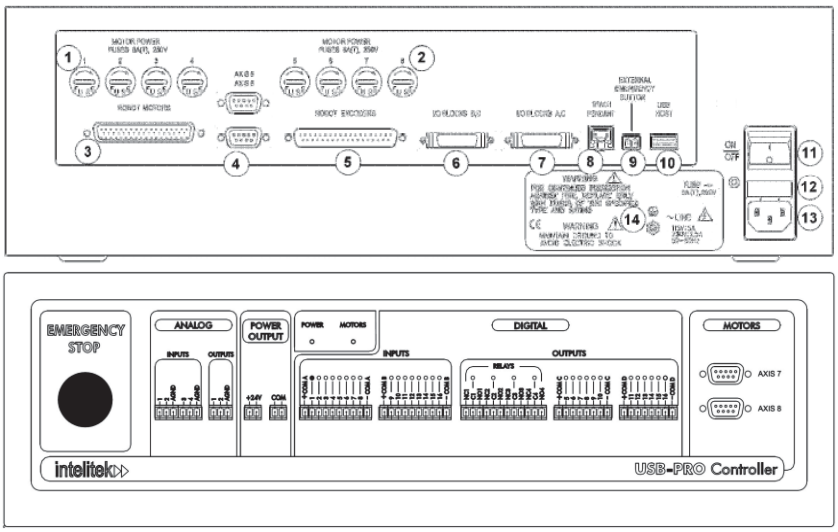


Figure 2 - 5. **USB-Pro controller input and output diagram. (Intelitek, 2008).**

Servo axis drivers	8
Path control	Joint, linear, circular
Speed control	10 speed settings
Power usage	1000 W
Weight	7.5 kg
Digital IO	16
Analogue IO	4

Table 2 - 2. **USB-Pro specifications. (Intelitek, 2008, p.13-14).**

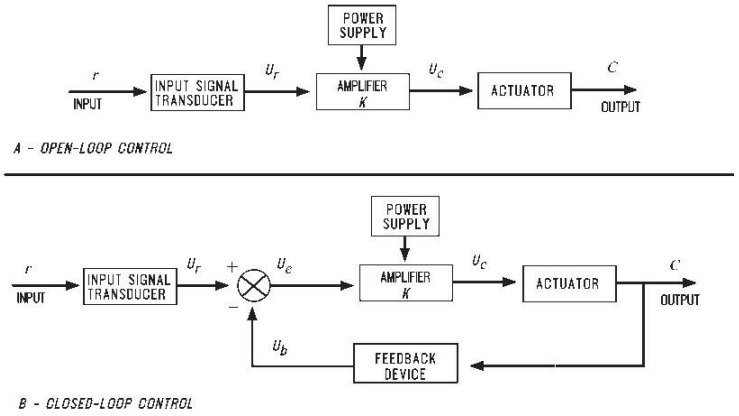


Figure 2 - 6. **USB-Pro available control loops. (Intelitek, 2008).**

This controller supports both open-loop and closed-loop control (Intelitek, 2008, p.45). In open-loop control mode, the system does not check whether or not the actual output (position or velocity) has reached the desired output. Since no feedback exists, the system is unable to correct output errors. Closed-loop control mode manipulates the system based on measured feedback. Comparing the desired output with the actual output, it is able to correct any errors.

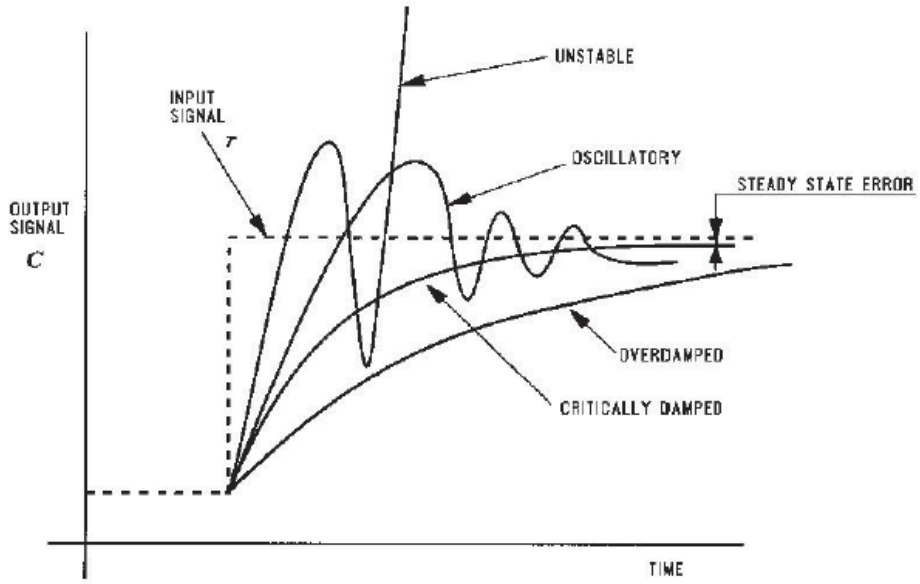


Figure 2 - 7. **PID control overview. (Intelitek, 2008).**

A control loop is used to determine fine motor control and provide smoother manipulator arm movement. The entire control cycle takes roughly 10ms (Intelitek, 2008, p.49). A pulse width modulation (PWM) signal is produced as the output signal, with encoder positions as the input. Proportional, integral, differential (PID) control is used, with all values configurable by the user.

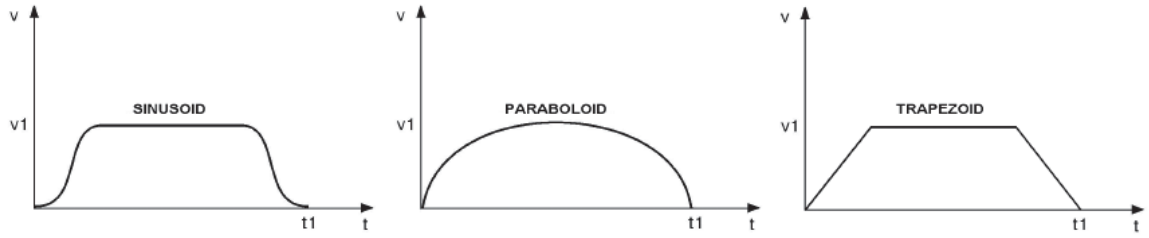


Figure 2 - 8. **Typical velocity curves. (Intelitek, 2008).**

Due to the nature of motor hardware, acceleration limitations are imposed on the system. This leads to the introduction of velocity profiles that aim to control motor velocity in a way which maximises acceleration. All curves are pre-calculated in the controller right before actuation depending on positional, velocity and acceleration requirements set by the user. Both path and point-to-point arm movement is implemented by the controller. Path control is composed of thousands of separate positions, depending on resolution (Intelitek, 2008, p.50).

Design Aspects

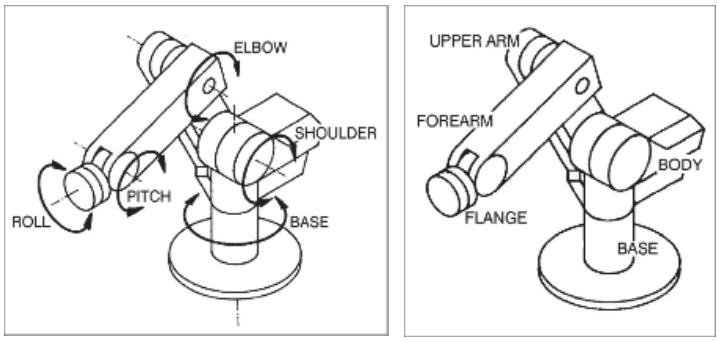


Figure 2 - 9. **Scorbot joint definitions. (Intelitek, 2008).**

The Scorbot-ER 9PRO is a vertically articulated robot. The base actuator (fig. 2 - 9) allows for rotational movement up to 270° (Intelitek, 2008, p.13). The range of this joint is quite important as it is a large factor in determining the total workspace. The shoulder joint (fig. 2 - 9) is capable of 145° of actuation. This joint, in conjunction with the elbow joint increases total reach of the arm. This also allows the arm to manipulate to an area underneath the robot (fig. 2 - 10), which may be beneficial in certain industrial scenarios.

Axis	Range	Effective Speed	Maximum speed
1 - Base rotation	270°	80°/sec	140°/sec
2 - Shoulder rotation	145°	69°/sec	123°/sec
3 - Elbow rotation	210°	77°/sec	140°/sec
4 - Wrist pitch	196°	103°/sec	166°/sec
5 - Wrist roll	737°	175°/sec	300°/sec

Table 2 - 3. **Scorbot joint actuation specifications. (Intelitek, 2008, p.13).**

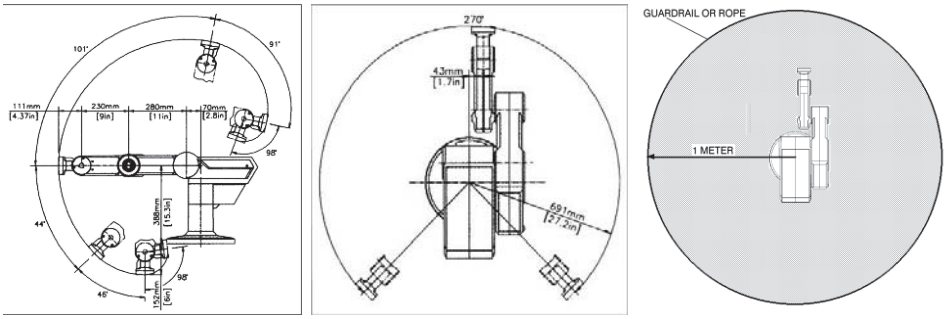


Figure 2 - 10. **Scorbot workspace. (Intelitek, 2008).**

Workspace is quite important as it determines the effective usefulness of a robotic manipulator arm. Generally, it is useful to have a large workable area directly in front of the robot which is not hindered by singularity or arm collision limitations.

Intelitek have gone to great lengths to ensure that the installation of their product is fairly straightforward. They have introduced a 'robot safety range' of 1m (fig. 2 - 10), allowing enough free space for the robot to move, with an added safety margin (Intelitek, 1996, p.19). At the edge of this boundary it is recommended that a guard rail is installed, or at minimum some form of robot workspace indicator generally in the form of a rope (Intelitek, 1996, p.20). At the base of the robot are three mounting locations, each consisting of M9 threaded holes. The diameter of the base is 330mm, providing ample

surface area to cater to the constantly changing momentum of the robot. Allowing three bolts for fastening also helps to distribute the load of the robot.

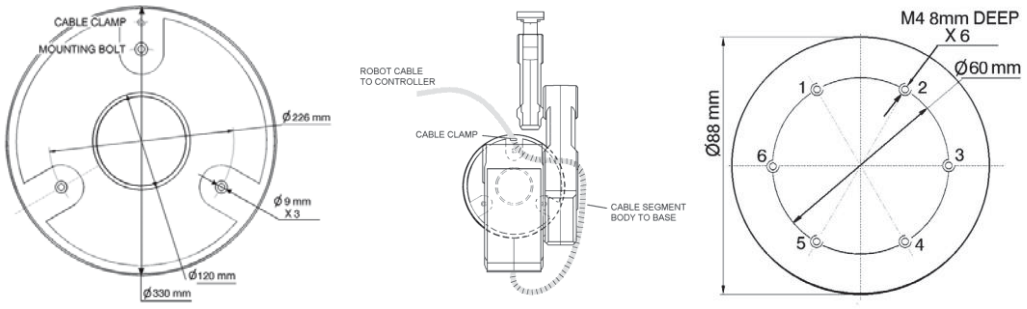


Figure 2 - 11. **Scorbot mounting diagram. (Intelitek, 2008).**

A mounting bolt for the cable clamp is also provided (fig. 2 - 11). This is useful to secure the robot cable which leads to the controller. Connectivity of the manipulator arm to the controller is relatively simple. One D37 connector, one D50 connector and a ground cable lead from the robot and connect with the controller. Each connector is unique. Once the robot has been linked and the controller has been interfaced correctly with a PC, via USB and Scorbase, it is a simple matter performing a 'Hardware Check' (Intelitek, 2008, p.20). If there are no errors in hardware, the robot may be operated as usual after homing. The gripper mount consists of six M4 mounting bolts, integrated with an 88mm diameter flange. Both power (24V) and pneumatic sockets are located at the top of the 'forearm' link (fig. 2 - 10).

There are three main elements to the Scorbot-ER 9PRO drive system (fig. 2 - 12). These include a permanent magnet DC motor, harmonic drive gear and timing belt and pulleys (Intelitek, 2008, p.29). Each robotic manipulator arm contains five of these drive systems, however they may vary individually to suit their specific actuation requirements.

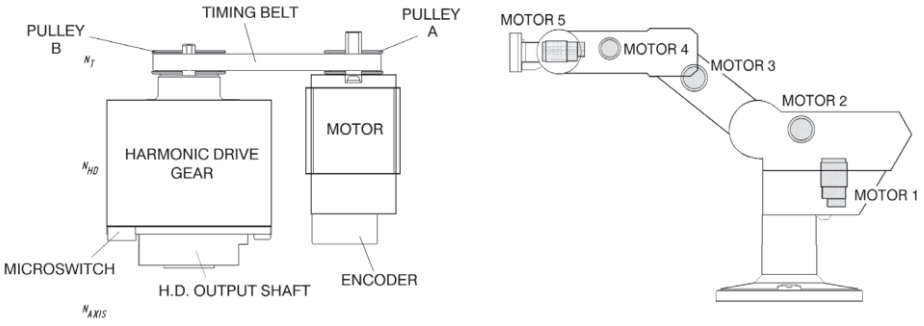


Figure 2 - 12. **Motor and motor location diagram. (Intelitek, 2008).**

Each DC motor is fitted with its own optical encoder (fig. 2 - 12). This encoder is used to provide rotational feedback to the controller, which in turn can be used to determine position.

	Motor Axes 1,2,3	Motor Axes 4,5
Peak rated torque	143 oz-in (10.3 kg-cm)	27.8 oz-in (2.00 kg-cm)
Rated torque	32 oz-in (2.30 kg-cm)	12.5 oz-in (0.90 kg-cm)
Maximum operating speed	4000 rpm	4500 rpm
Weight	1.29 kg	0.28 kg

Table 2 - 4. **Scorbot motor axes properties. (Intelitek, 2008, p.32).**

The harmonic drive gear consists of a circular spline, wave generator, flexspline and dynamic spline (Intelitek, 2008, p.33). The flexspline (fig. 2 - 13) has two more teeth than the circular spline, which only mesh once the wave generator forces these to interlock. This means that for one complete clockwise rotation of the wave generator, the flexspline moves anti-clockwise by two teeth. The dynamic spline couples directly with the flexspline.

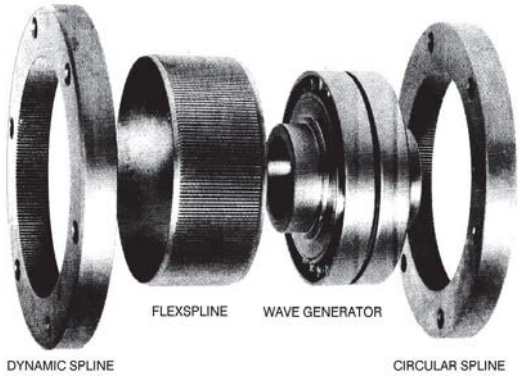


Figure 2 - 13. **Harmonic drive exploded view. (Intelitek, 2008).**

For every revolution of input shaft, output shaft will rotate by $\left(\frac{2}{Nf}\right)$ of a revolution from harmonic gear drive (Intelitek, 2008, p.34).

$$N_{HD} = \frac{1}{\left(\frac{2}{Nf}\right)} = \frac{Nf}{2}$$

$$N_T = \frac{PulleyB}{PulleyA}$$

$$N_{axis} = N_T \times N_{HD}$$

N_{HD} = Harmonic gear ratio
Nf = Number teeth flexspline
N_T = Belt drive ratio
N_{axis} = Overall effective gear ratio of axis

	N_T	N_{HD}	N_{axis}
Axis 1	1.33 : 1	160 : 1	213.33 : 1
Axis 2	1.52 : 1	160 : 1	243.8 : 1
Axis 3	1.33 : 1	160 : 1	213.33 : 1

Axis 4	1.8 : 1	100 : 1	180 : 1
Axis 5		100 : 1	100 : 1

Table 2 - 5. **Scorbot axes gear ratios. (Intelitek, 2008, p.35).**

The rotational movement of each axis is measured by an optical encoder. Each DC motor is of such design that the drive shaft extends past the motor itself, allowing for the encoder disk to couple directly onto the shaft at one end without hindering rotation at the other end (fig. 2 - 14). Each encoder uses a single light emitting diode (LED). Opposite the LED is a light detecting integrated circuit containing photodetectors and componentry to produce a digital signal. A perforated, rotating disc (fig. 2 - 14) is located between the emitter and detector circuit. As the disc rotates, the beam of light from the LED is broken, which can be detected and thus, create a digital signal (fig. 2 - 4) which is dependent on rotational movement of the motor. Each encoder has 512 slots generating 2048 counts per revolution, allowing for a final resolution of 0.000825° (Intelitek, 2008, p.39).

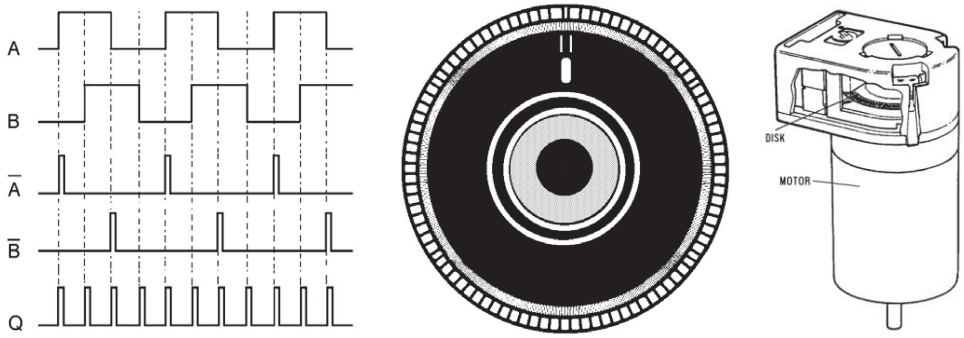


Figure 2 - 14. **Encoder signal output, encoder slot, encoder to motor mounting. (Intelitek, 2008).**

Axes 1-4 each contain two limit switches (fig. 2 - 15). Axis 5 does not, which means that it may rotate indefinitely, but is bounded by software control. Limit switches are located at the end of arm travel (fig. 2 - 15) in both directions, right before the hard stops. This allows for maximum actuation of the arm.

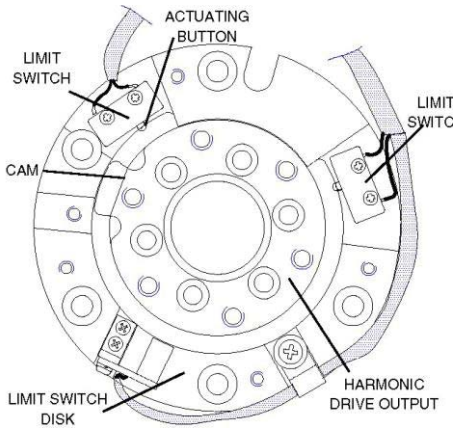


Figure 2 - 15. **Scorbot actuator cam design cross-section. (Intelitek, 2008).**

As seen in Figure 2 - 15, a cam design is used which consists of a circular rotating disc with an offset to allow for positional feedback and hardware limits. The mechanical limits are over-designed. This has been done to accommodate for motor actuation even once the hard stops have been reached, preventing permanent damage to the axis. An auto 'COFF'

software protocol is also in effect (Intelitek, 2008, p.41), which is based on encoder counting, cutting power once a collision has been detected. An optical home switch (fig. 2 - 15) has been implemented in each axis to identify a fixed reference, allowing for a consistent home position.

2.2.2 Kawasaki Industrial Robotics

Because of the size and scope of Kawasaki, there is very little information regarding internal componentry or design of their robotic manipulators. Understandably, this seems to be classified, as these key design principles are at the core of Kawasaki's success. User manuals or any documentation of that nature also appear to be unavailable. Taking note of these limitations, this segment will only briefly touch on three main industry robotic types utilised by Kawasaki, a leader in robotics technology.

YS002N

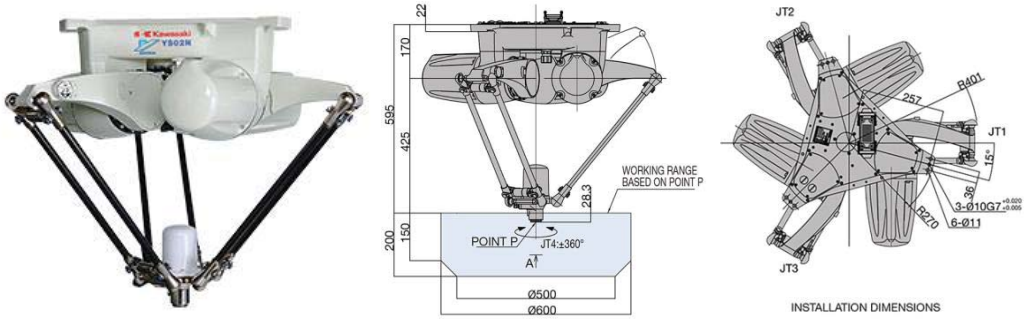


Figure 2 - 16. **YS002N robot, robot workspace, robot dimensions. (Kawasaki, 2015).**

Payload	2 kg
Horizontal reach	600 mm
Vertical reach	150 mm
Repeatability	$\pm$ 0.1 mm
Maximum linear speed	3,3000 mms^{-1}
Power consumption	1000 W

Table 2 - 6. **YS002N specifications. (Kawasaki, 2015, p.3)**

The YS002N is a delta robot. This design makes use of parallelograms to construct a robot with three translational degrees of freedom. A rotating actuator is fitted to the end-effector. According to Bonev (2001), the first delta robot was patented in 1987 by Raymond Clavel.

This design tends to be popular among rapid sorting and pick and place applications and has spread throughout assembly and material handling industrial areas. This manipulator has a rather limited workspace (fig. 2 - 16) relative to some of the other types of robot available, but what this robot lacks in workable area it makes up for in speed. The haste at which this robot completes tasks is unmatched, mainly due to the unsurpassed acceleration of the configuration.

Each delta robot requires an E94 controller for power and control (Kawasaki, 2015, p.9). The controller is accompanied by a 'Tech pendant' which acts as a human interface and allows direct user control. End-effector attachments come in a large variety, depending on

application. Pneumatics and power are available for gripper use. Kawasaki installs, configures, troubleshoots and maintains all of their products. Usually a factory with a significant number of Kawasaki robots will have at least one Kawasaki technician on-site full-time.

RS005N

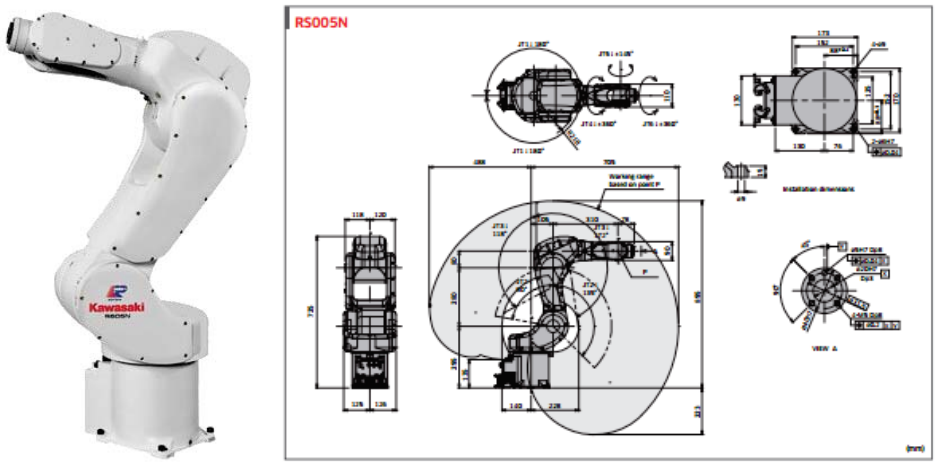


Figure 2 - 17. **RS005N robotic arm and workspace. (Kawasaki, 2015).**

Payload	5 kg
Horizontal reach	705 mm
Vertical reach	1,118 mm
Repeatability	± 0.2 mm
Maximum linear speed	9,1000 mms^{-1}
Power consumption	1,500 W

Table 2 - 7. **RS005N specifications. (Kawasaki, 2015, p.3).**

The Kawasaki RS005N is a medium payload industrial robot. Applications include assembly, machine tending, material handling, material removal, sealing and dispensing. As with many other 6 DOF robotic manipulator arms, the workspace overlaps with an area underneath the arm (fig. 2 - 18), providing extra utility in certain situations.

The E0X controller accompanies this robot (Kawasaki, 2015, p.9), including the Teach Pendant. It should be noted that due to the modular design of this manipulator arm, the cabling is limited, simplifying maintenance and diagnostics. Kawasaki has created a Mean Time To Repair (MTTR) rating, which compares the time taken to repair each of their products (Kawasaki, 2015, p.9). The controller comes pre-programmed with numerous programming functions. Functions may be combined to fit the particular application and if the user requires a wider range of capabilities, each controller supports the Kawasaki AS programming language.

Three external axes may be added onto the RS005N manipulator arm. The controller is also expandable and has the capabilities of handling up to nine axes, not including end-effector peripherals (Kawasaki, 2015, p.9).

CP500L

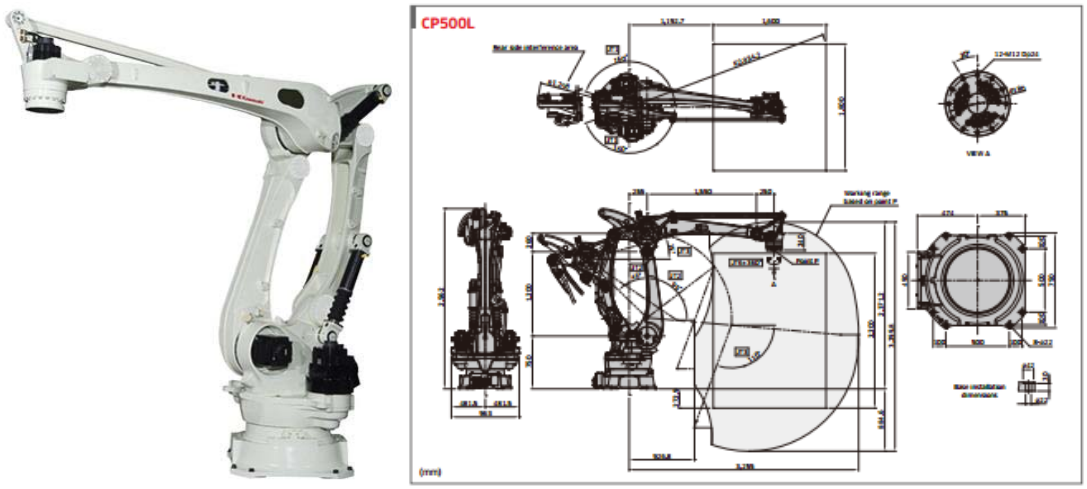


Figure 2 - 18. **CP500L robotic arm and workspace. (Kawasaki, 2015).**

Payload	500 kg
Axes	4-5
Repeatability	± 0.5 mm
Power consumption	12,000 W
Weight	1,650 kg

Table 2 - 8. **CP500L specifications. (Kawasaki, 2015, p.9)**

Kawasaki's CP500L is a robust robotic arm developed mainly for palletising applications. With a payload of 500kg, this arm is ideal for heavy lifting and stacking. Safety is a big concern when manipulator arms of this magnitude are involved, hence Kawasaki have developed a 'Safety function' which cuts power to the robot in the event of human interaction or motion detection in the 'Safety fence' area (Kawasaki, 2015, p.6).

The E03 is required for control of this manipulator arm. 'K-ROSET' is the software package used with this system and includes several pre-programming palletising patterns (fig. 2 - 19). The layout of the surrounding environment may also be configured in software, enabling auto collision-avoidance and control. Depending on end-effector, Kawasaki caters to a large range of packing applications.

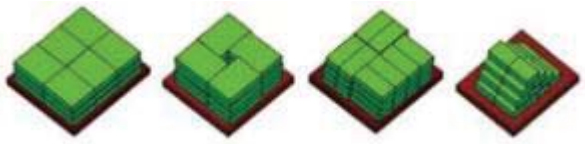


Figure 2 - 19. **Kawasaki palletising patterns. (Kawasaki, 2015).**

A Laser slit-scan camera (LSC) is generally used in conjunction with the arm in de-palletising applications (Kawasaki, 2015, p.5).

2.3 Market Research - Service Robotics

2.3.1 Kinova JACO

Figure 2 - 20. **Kinova Jaco picture. (Kinova Robotics, 2009).**

Founded in 2006, Kinova Robotics is a Canadian based company specialising in the production and implementation of service and assistive robotics. Their first major release came in 2010, with the launch of JACO, a six degree of freedom (DOF) robotic manipulator arm. Although the JACO arm has shown success in both industrial and assistive applications, it has been heavily marketed to help the wheelchair bound, physically impaired, with the original intended use to enhance the independence of the user.

The manipulator arm comes with a choice of two end-effectors, 2 fingered, or 3 fingered, depending on operation requirements. This product has put great emphasis on the needs of the user, tailoring each unit specifically. This system acts as an add-on, integrating directly with an electric wheelchair and drawing power from the same battery. This introduces unique design challenges, resulting in a light-weight arm, with relatively low power consumption.

Weight	5.2 kg
Payload	1.6 kg
Reach	900 mm
Power consumption	25 W, 5W standby
Construction	Carbon fiber
Linear speed	200 mms^{-1}

Table 2 - 9. **Kinova Jaco arm specifications. (Kinova Robotics, 2014, p.4).**

While focussing on 'activities for daily living', the JACO system is unique in its ability to provide an interface personalised for the user. Control can include, but is not limited to: hand-interfaced joystick, foot-interfaced joystick, mouth actuated (sip and puff), head array (Kinova Robotics, 2015, p.2).

Helping those with (Kinova Robotics, 2015, p.4):

- Cerebral palsy (CP)
- Spinal muscular atrophy (SMA)
- Spinal cord injury (SCI)
- Amyotrophic Lateral Sclerosis (ALS)
- Multiple Sclerosis (MS)

Intended use covers tasks such as eating, drinking, using a microwave and general household responsibilities which may be difficult for those suffering with compromising disabilities. The arm is generally mounted to the side of a wheelchair seat frame.

Customer Support

Kinova Robotics offers a range of comprehensive support to their customers. It is fairly obvious to note that this should be an integral part of the product, due to the nature of their intended clients. According to the JACO Assistive Robotic Arm e-brochure made available by Kinova Robotics (2015), a free pre-installation home or on-site evaluation is provided on request. This seems to be ideal for opening the initial client-company relations. Because of the tailoring nature of the JACO system, this is also a necessity.

Figure 2 - 21. **Kinocare logo. (Kinova Robotics, 2015).**

Kinocare 'BASIC' is a supporting package included in the purchase of any of Kinova's service robotic products. The coverage ranges depending on the selected bundle, with 'PLUS' and 'PREMIUM' upgrades available. Kinocare 'BASIC' includes minimum technical support, with a 24-hour contact reply time guaranteed by Kinova. This package also comes with a 2-year limited warranty, which warrants that each new product shall be free from defects in material and workmanship, as well as operate correctly as described by the User Manual for a continuous period of 24 months. This does not include damage caused to the system by the user.

The Kinocare 'PLUS' features an extended 3-year limited warranty, with the same terms as prescribed by the 'BASIC' package. The 24-hour contact response time is also included. A repair discount of 15% is applied to the invoice of any repairs done by Kinova on the product which includes the Kinocare 'PLUS' coverage.

Most extensive, Kinocare 'PREMIUM' is the maximum coverage available. The limited warranty is extended by 2 years, providing the user with a total of 4 years of security. The repair discount is increased to 25% and one 'Complete tune up and maintenance' contract is added to the package. This includes product inspection and diagnoses, as well as software or hardware updates. Damaged or defective parts are also replaced, not excluding aesthetic faults.

Pricing:

Kinocare 'BASIC': Free with product purchase
Kinocare 'PLUS': $3000
Kinocare 'Premium': $7500

The Kinovo website includes a 'Knowledge Centre' section. The resources in this section are somewhat limited to a handful of videos, including 'How I do it', 'Home/Retracted position' and 'Drinking mode'. These videos are instructional, informing the viewer of basic robot control and features. The videos are dated January 2015, so this section has been stagnant for some time.

Kinova Jacosoft

Kinova Jacosoft is software designed specifically for use with the JACO or MICO systems. The purpose of this software ranges from general information to diagnosis, depending on the licence type acquired by the user (tab. 2 - 10):

	RESEARCH	REHAB	USER	SERVICE
General Information	✓	✓		✓
Configurations	✓	✓		✓
Protection zones	✓	✓		✓
Trajectory	✓			
Diagnosis	✓	✓	✓	✓
Service[1]				✓
Roboticist Central	✓			

Table 2 - 10. **Kinova Jacosoft available packages. (Kinova Robotics, 2015).**

Jacosoft allows the user to position the robot in specific configurations, which may be saved, loaded, imported and exported (Kinova Robotics, 2014, p.8). The software also includes angular and Cartesian control methodologies. Once the required trajectory and positional settings have been configured as desired by the user, this information may be exported to the control unit, for quick use. Jacosoft also allows the default 'Retracted' position as well as trajectory settings to be altered.

Other settings may also be changed. These include joystick sensitivity, arm linear speed, drinking mode, spasm filter and mapping info. Mapping info allows the controller's outputs to be customised, depending on input.

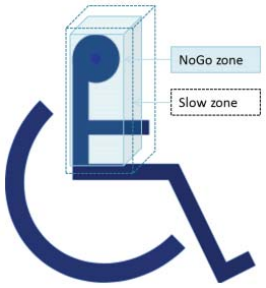


Figure 2 - 22. **Jacosoft NoGo and Slow zone diagram. (Kinova Robotics, 2015).**

One unique aspect to Jacosoft and the JACO system is the 'NoGo Zone' (Kinova Robotics, 2014, p.26). This is a user-defined zone (fig. 2 - 22) which the robot will take into account when executing a movement command. The intention is to provide a safe zone for the operator, which the robotic arm may not enter. There is also a grace zone, slowing robot activity at the border of the 'Slow Zone'.

The 'Roboticist Central' tab in Jacosoft includes options such as 'Set PID filter', 'Set PID', 'Set zero position', 'Set actuator address' and 'Force control' (Kinova Robotics, 2014, p.32). These options allow for finer control and are recommended for advanced users.

Finally, Jacosoft includes a 'Diagnosis' tab. This tab shows firmware information, as well as any information regarding faults or inaccuracies detected by the actuators within the arm. An 'appendix' is added at the end of the Jacosoft User Manual. This includes a comprehensive 'Step-by-step' example referring to 'Control Mapping'.

Actuators

Interestingly, Kinova Robotics have made available some of their internal hardware for purchase. In particular, there is an entire product range devoted to the sales of Kinova actuators (Kinova Robotics, 2014). There are three main models, including: K-58, K-75 and K-75+. The main differences between these models are their size, however the + model provides an increase in other properties relative to the lower models.

Although access to internal workings and electronics are limited, a significant amount of technical specifications are provided.

	K-58	K-75	K-75+
Nominal Torque	3.6 Nm	9.2 Nm	12 Nm
Nominal Speed	15 rpm	7 rpm	9.4 rpm
Positional sensor resolution	0.068°	0.055°	0.047°
Torque sensor precision	$\pm$ 0.4 Nm	$\pm$ 0.4 Nm	$\pm$ 0.4 Nm
Weight	357 g	587 g	570 g
Picture			

Table 2 - 11. **Kinova actuator specifications. (Kinova Robotics, 2015).**

Each actuator consists of two halves which rotate relative to one another. One half as in figure 29 must be stationary, to allow the other half to provide rotational actuation. The K-58 mounting screws are designed to house M3 bolts, while the K-75 and K-75+ models require M4. Each unit includes an integrated torque sensor and encoders to provide positional accuracy. It should be noted that brushless DC motors are used to provide rotational input.

Design Aspects

The design constraints imposed by the nature of the product has resulted in a light-weight arm with low power consumption. The way in which Kinova have reduced overall weight is by reducing the total material used. Their design comprises of hollow structures, which provide enough strength for the manipulator arm, while not over-designed significantly. The use of carbon fibre further reduces arm weight. The final weight of the 6 DOF JACO model is 5.4kg, excluding the end-effector (Kinova Robotics, 2014, p.4).

Power usage is a priority. The unique actuators manufactured by Kinova do not incorporate the use of stepper motors. Instead brushless DC motors are used in conjunction with the Kinovo ratio 136 Harmonic Drive. It is unclear due to lack of information, but this may influence power consumption.

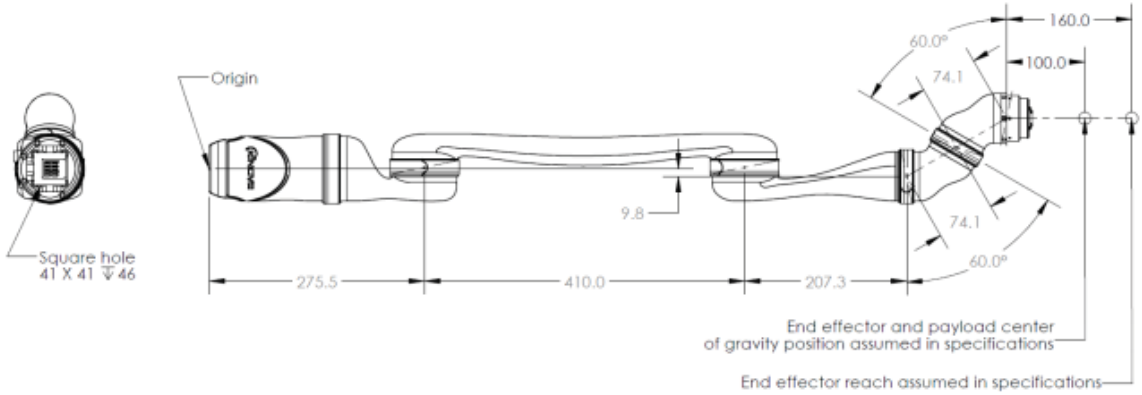


Figure 2 - 23. **6 DOF JACO dimensions. (Kinova Robotics, 2015).**

The 6-DOF manipulator arm has several actuated joints (fig 2 - 23). A rotating joint is used at the base of the robot, allowing movement for majority of the arm. Two additional joints are used, separated significantly to improve working space. Interestingly, to allow for finer end-effector control, two joints are used, each offset by 60° relative to arm 2. A final rotating actuator is used at the base of the end-effector. It should be noted that all joints required for actuation of the JACO manipulator arm are rotationally actuated. This design permits robot interaction at floor level, enabling the user to work with objects on the floor.

JACO includes a somewhat modular design. Each arm segment is separate from one another, with actuation devices not integrated with the arm. Actuators are attached to the arm through the use of 8 fastening screws (fig 2 - 24). This simplifies the system significantly and allows for relatively uncomplicated repairs. This does mean however that all parts must be manufactured to a certain precision, as to avoid misalignment.

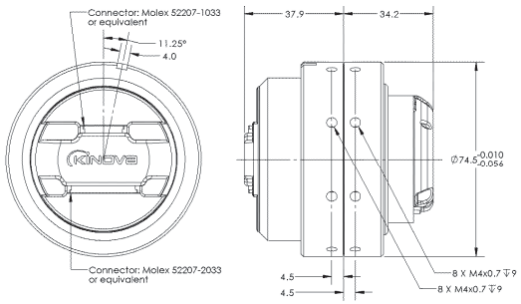


Figure 2 - 24. **JACO actuator dimensions. (Kinova Robotics, 2015).**

End-effector grippers are under actuated and include temperature sensors, as well as rotational encoders. With an opening range of 175mm and a gripping force of 40N (Kinova Robotics, 2015, p.2), the 3-fingered grippers are able to accommodate majority of tasks. The gripper includes rounded contact surfaces for better grip as this increases the effective friction area. A plastic insert at the centre of the gripper increases its ability to handle circular objects.

Figure 2 - 25. **Kinova JACO 2-fingered gripper, 3-fingered gripper. (Kinova Robotics, 2015).**

2.3.2 Bestic

Figure 2 - 26. **Bestic picture. (Bestic AB, 2015).**

Bestic AB specialises in assistive eating devices. Founded in 2004, the Swedish company developed Bestic over seven years with the help from Swedish organisations and sectors such as the Swedish Institute of Assistive Technology, Robotdalen and Promobilia. The company has spread to include customers in Denmark, Finland, Noway, Holland, English, Spain, France and the USA.

Launched is 2011, Bestic is a simple manipulator arm, designed solely for aiding those who may have trouble eating from diminished arm or hand functionality. The 'one button - one press - one bite' interface allows the user to control the arm with comfort. Noting that this manipulator is also battery operated and fits inside a Bestic backpack, enabling the user to eat on their own terms and on their own time. The system includes the Bestic manipulator arm, a spoon end-effector attachment and the application specific plate. The utensils provided by Bestic are dishwasher friendly.

This 4 DOF robot is controllable through either a simple one-button interface or the more advanced five-button Picasso MINI controller. The one-button interface is adapted to the user, depending on individual requirements. Although the five-button control device allows for functionality, it may not always be suited for the user.

Bestic is a robust robot from necessity and consequently makes use of this to include several features. It is recommended that the user handle the robotic arm at the lower end of the middle link. The arm is also water-resistant.

Height	340 mm
Width	220 mm
Depth	200 mm
Weight	2.3 kg
Operating time	4 hours

Table 2 - 12. **Bestic specifications. (Bestic AB, 2015).**

Design Aspects

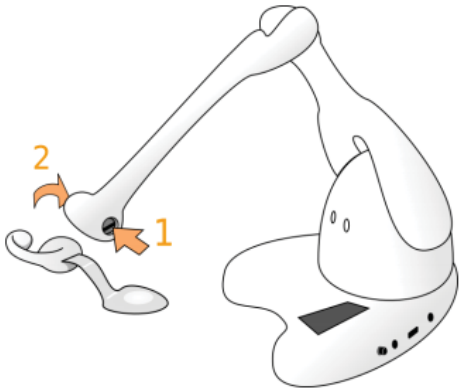


***Figure 2 - 27*. Bestic diagram. (Bestic AB, 2015).**

As a result of the unique design constraints enforced on the Bestic manipulator arm by its target audience, the end-effector design is required to be simplistic and easily detachable (Bestic AB, 2015, p.13). The spoon attachment incorporates the use of a living hinge and slot configuration (fig. 2 - 27).

Through the use of elastic deformation, the spoon attachment is able to bend around the outer limits of its allotted slots to snap into position (fig. 2 - 28). The deformation is reversible and does not permanently damage the attachment. Inside the homing slot, there is a smaller slit (fig. 2 - 28) which provides rotational actuation to the end-effector.

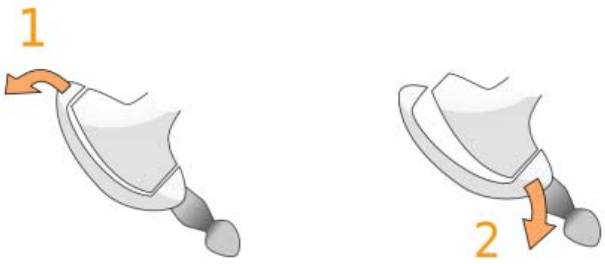


***Figure 2 - 28.* Bestic spoon attachment diagram. (Bestic AB, 2015).**

In an effort to reduce complexity of the system, Bestic have provided their own specific plate. This has been done to both standardise the system and provide a consistent operating environment for the robotic manipulator arm. Bestic's circular base design enables positional control of the plate, once the plate has been pushed up against the base (fig. 2 - 29), its position will always be constant. This allows for pre-programed positions and command chains to be established. A non-stick mat is included.

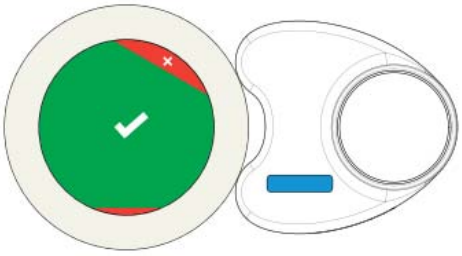


***Figure 2 - 29.* Bestic plate workspace diagram. (Bestic AB, 2015).**

Bestic supports and provides two main types of control devices (Bestic AB, 2015, p.20). The one-button interface (fig. 2 - 30) is essentially one large button which provides a single one-step command to the arm. Each press of this button corresponds to one eating cycle. The arm will swing down, allowing the end-effector to gather food, the food will then be transported upward to the user for consumption after a scrape cycle. The scrape cycle brushes the spoon end-effector past the brim of the plate, as to remove any debris from the bottom of the attachment before passing it along to the user. Prior to the scrape cycle, the spoon is gently swayed from side to side, eliminating any food which would tend to fall off mid transference.

Figure 2 - 30. **Bestic Picasso MINI. (Bestic AB, 2015).**

Bestic also offers a more advanced control device, the Bestic Picasso MINI (fig. 2 - 30). This is a USB operated pad, with five unique control buttons. The robotic arm may be manipulated in a Cartesian-like fashion, with the middle button changing which cycle the arm is in. In the case of the Picasso MINI, the arm has several more cycles than the one-button operation and may be set-up depending on user preference. The default setting allows XY movement of the end effector, once the middle button is pressed, the scrape and feed function and executed.

Bestic includes a limited display (fig. 2 - 31). This provides the user with the state of the battery power and several other unique pieces of information. This is also how the menu is accessed. The menu is navigated through the use of the Picasso MINI controller. Individual profiles are included in the settings, enabling the user to configure end-effector feeding height and XY position. Control options are also changeable, which sees the user pick the 1-click or 2-click option. The 1-click option runs through the entire scrape-feed cycle, while the 2-click option splits this into two.

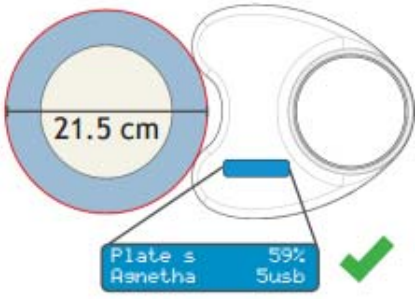


Figure 2 - 31. **Bestic control display diagram. (Bestic AB, 2015).**

Two control modes are pre-programmed into Bestic (Bestic AB, 2015, p.18-19). Single two-step mode and single one-step mode (fig. 2 - 32). These configurations determine the movement of the end-effector over numerous cycles.

Figure 2 - 32. **Bestic control step mode diagrams. (Bestic AB, 2015).**

2.4 Market Research - Educational Robotics

2.4.1 Mecademic DexTAR

Figure 2 - 33. **Mecademic DexTAR picture. (Mecademic, 2015).**

Mecademic is an industrial automation company specialising in the design and manufacture of high-accuracy desktop robotic arms for research and educational purposes. Founded in 2013, they have released two major products, including:

Meca500
DexTAR

DexTAR (Dextrous Twin Arm Robot) is a 3-DOF pick and place robot developed solely for pedagogical purposes. Essentially a five-bar mechanism, this parallel robot operates with two passive joints. This does however, complicate the inverse kinematics of the system significantly. The robot comes with two standard end effectors, an electromagnet for simple pick and place of Ø11mm ferromagnetic balls and a pen tool, which allows the user to make illustrations on a paper medium with the correct trajectory plan. The product is accompanied by its own 3D simulation and offline programming software, which allows the user to simulate the outcome of or send custom programs to DexTAR, which are then executed using the integrated touch screen. The robot is precision-machined aluminium, driven by two 90W Maxon servo motors and one stepper motor which allows for linear actuation of the Z axis, this however does not permit for orientation control of the end-effector about the vertical-axis. All four links are of equal length (90mm, axis to axis).

Footprint	435 x 420 mm
Weight	10.2 kg
XY workspace	Larger than circular area of diameter 242mm
Axes 1 & 2 encoder resolution	0.011°
Axes 1 & 2 max speed	$720°s^{-1}$
XY max linear speed	$300\ mms^{-1}$
Z axis stroke	19 mm
Z axis resolution	0.025 mm
Z axis max speed	120 mm
Position repeatability	$\pm$ 0.025 mm
Power usage	24 V, 9.17 A
Touch screen	480 x 320 ppx (16M colours)
Communication	USB 2.0 port

Table 2 - 13. **Mecademic DexTAR specifications. (Mecademic, 2014, p.2)**

Internal Software

At every start-up of DexTAR, the system needs to be homed, which is done through the integrated touch-screen interface (fig. 2 - 34). To prevent damage to the robot, the angle between the distal links must be within 30° and 150° (Mecademic, 2015, p.4). While a program is executed, there is a yellow indicating LED which flashes at 60Hz to alert the user of robot movement. The program can be paused and resumed at any time through the on-board interface. If any motor control errors occur such as misalignment due to friction from belt control or collisions, the robot will cut power to the motors and go into an error mode, which requires a restart and re-home.

The robot can be jogged manually through the touch-screen interface, which also allows manual deactivation of both motors. It should be noted that when the robot is jogged through a type-2 singularity the controller cannot compensate and will produce a control error, requiring the system to be reset and the arms to be moved to a safe position manually (Mecademic, 2015, p.8).

Figure 2 - 34. **Mecademic DexTAR control screenshots. (Mecademic, 2015).**

Assembly modes

Due to the unique design of the robot, there are, theoretically, eight working configurations (fig. 2 - 35) which determine the behaviour of the end effector relative to the two arms. These have been labelled as 'working modes'.

Generally, for any given base motor angles, there are two solutions as to end-effector position. The software has categorized these as 'assembly modes' (fig. 2 - 35), for which there is a positive and negative assembly mode for all four working modes. To switch from positive to negative assembly modes, the system must pass through a type-2 singularity. DexTAR does not include a sensor or mechanism for detecting which assembly mode it is currently in.

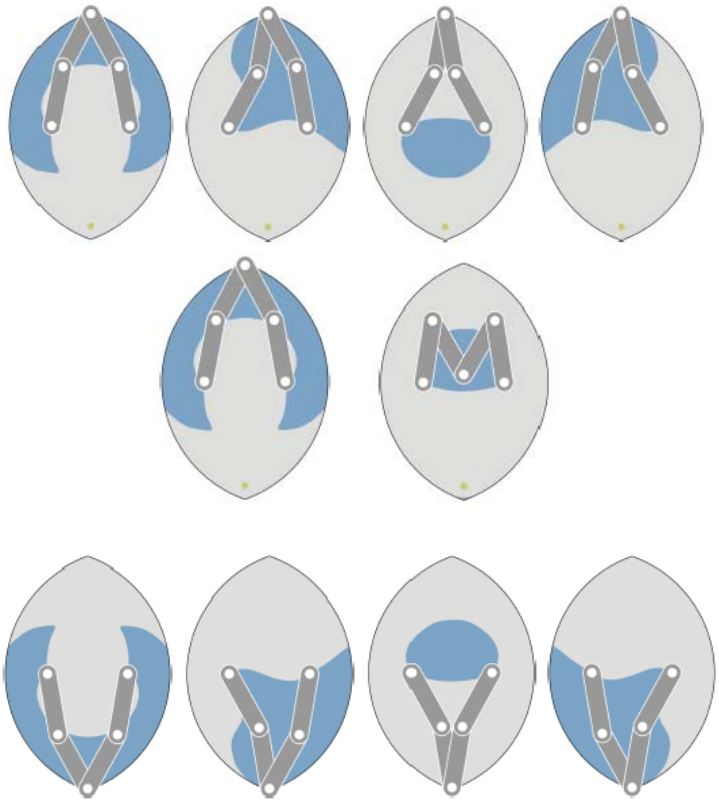


Figure 2 - 35. **Various DexTAR assembly mode diagrams. (Mecademic, 2015).**

End-Effectors

DexTAR is shipped with the electromagnetic tool pre-installed. The base end-effector consists of a screw mechanism, so the head accessory can be exchanged as necessary. There is a 24V, 1A socket which supplies power to the tool (Mecademic, 2014, p.2).

The drawing attachment is designed to be used with standard D1 ballpoint pen refills. It is required that the robot is without power to switch tools, followed by the Z-axis' linear actuator manually returned to its upper extreme position. The installation of the pen tool also requires the acrylic base plate to be temporarily removed. It should be noted that it is recommended to use a tool to wedge the acrylic plate to a point where it can be manually handled, due to the four neodymium magnets keeping it in place (Mecademic, 2015, p.9). Two standard acrylic bases are included, one with laser-cut holes to house the ferromagnetic balls and the other a solid base for the pen tool. A standard A4 sheet does not cover DexTAR's complete workspace, however, Mecademic does supply their own laser-cut drawing sheets.

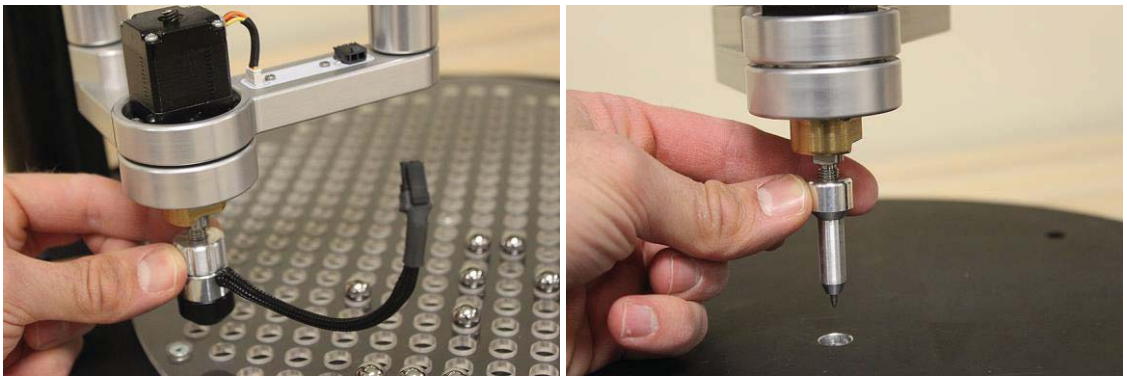


Figure 2 - 36. **DexTAR magnetic attachment, pen attachment. (Mecademic, 2015).**

Desktop Software

The DexTAR simulation and offline programming software package (fig. 2 - 37) is designed specifically for use with the DexTAR system and can be downloaded from the Mecademic website for free. The robot is programmed offline using Mecaprol, Mecademic's proprietary robotic programming language, which does not support on-line features such as drive-through or lead-through methods. Once code has been written, it is then compiled and sent to the robot through the DexTAR Sim graphical user interface. This software currently only supports Windows.

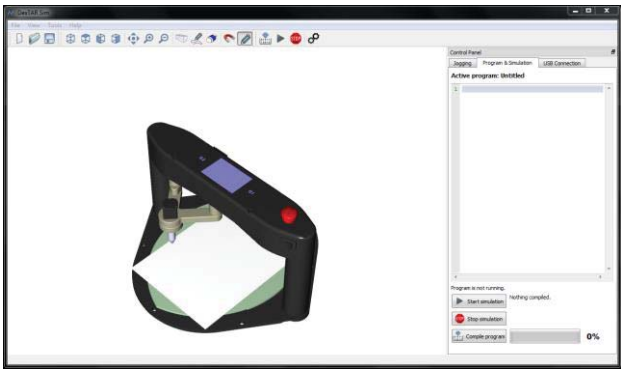


Figure 2 - 37. **DexTAR simulation software overview. (Mecademic, 2015).**

Using the DexTAR Sim graphical interface (fig 2 - 38), the user can simulate the movement of the real robot based on the trajectory planned in the software. End-effector trajectory and control with the pen tool can be manually achieved through the 'trace' function which allows the user to control the robot without Mecaprol. The software also allows for the addition of 'code snippets' to be added as part of the code to be uploaded. Pick & place command runs can be constructed visually and added to the code automatically.

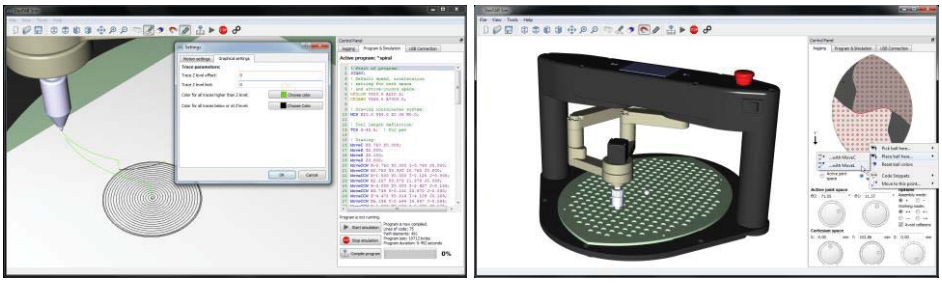


Figure 2 - 38. **DexTAR Sim interface screenshot. (Mecademic, 2015).**

To add to the educational aspect of the product, DexTAR Sim comes with a jogging menu. This allows the axis-by-axis jogging of individual arms in the cartesian workspace.

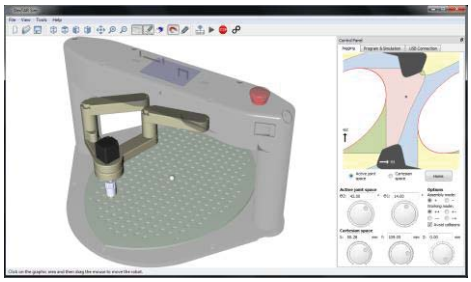


Figure 2 - 39**. DexTAR Sim jogging menu screenshot. (Mecademic, 2015).**

Mecaprol is a compiled language, similar to G-code (Mecademic, 2015, p.19), a common language used for programming CNC machine tools. The language consists of generally naming a function, followed by setting the parameters of that function, from 'Output OFF;' it is clear to see that the code is setting the 'OUTPUT' functions' parameter to 'OFF'. It is also necessary to end each line with ';' to let the compiler know that it is the end of that line, similarly 'START' and 'STOP' are necessary.

```
! The electromagnetic tool and the acrylic plate with holes must be installed.
! Three 11 mm steel balls must be placed at the front extremity of the plate.
! The distance between the centres of adjacent holes is 14.75 mm.

START;
CFGLIN V40.0 A10000.0; ! Speed and acceleration settings for MoveL, etc.
CFGANG V360.0 A10000.0; ! Speed and acceleration settings for MoveJ and MoveC.

! Definition of the world coordinate system
WCS X0.0 Y0.0 Z-2.50 W0.0; ! Z-2.50, because of the holes
! Definition of the tool coordinate system
TCS Z-39.00;

! Start manipulating the three balls
MoveC X0.00 Y162.25 M1; ! pick the first ball in the ++ working mode
MoveZ Z0.00;
Output ON; !turn the electromagnet on
MoveZ Z12.00; !lift the tool at least the height of a ball
MoveL X0.00 Y14.75;
MoveZ Z0.00;
Output OFF; ! turn the electromagnet off
MoveZ Z12.00;
WaitTime T1000; ! Pause for 1 second, DexTAR is almost in a Type 2 singularity
MoveC X14.75 Y162.25;
MoveZ Z0.00;
Output ON;
MoveZ Z12.00;
MoveC X-118.00 Y14.75;
MoveZ Z0.00;
Output OFF;
MoveZ Z12.00;
MoveC X-14.75 Y162.25;
MoveZ Z0.00;
Output ON;
MoveZ Z12.00;
MoveC X118.00 Y14.75;
MoveZ Z0.00;
Output OFF;

! Move DexTAR to its home configuration
MoveZ L0.0; ! it's always best to start with the linear actuator
MoveJ A0.0 B0.0;
END;
```

Figure 2 - 40. **Mecaprol G-code screenshot. (Mecademic, 2015).**

Kinematics

The kinematics of this robot ignore the Z-axis completely and treats the system as Cartesian. The Z-axis is handled separately, as it is a simple linear actuator for pick and place only, kinematics are redundant.

Inverse Kinematics (Mecademic, 2015, p.31).

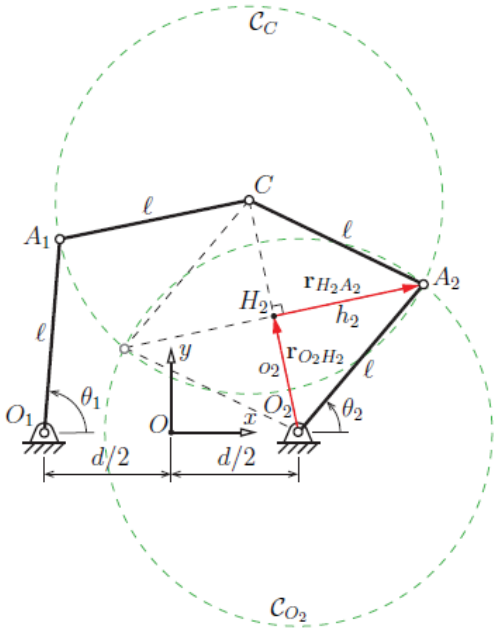


Figure 2 - 41. **DexTAR Inverse kinematic mathematical definitions. (Mecademic, 2015).**

Oxy: Reference frame base coordinates (BCS)
O_1: Axis of revolution joint 1, arm 1
O_2: Axis of revolution joint 2, arm 2
A_1: Passive revolute joint 1 from arm 1
A_2: Passive revolute joint 2 from arm 2
C Tool centre position (TCP)
θ_1: Active joint angle 1, with respect to BCS
θ_2: Active joint angle 2, with respect to BCS

H_2: Midway point between C and O_2
o_2: Length segments O_2H_2 and H_2C
h_2: Length segment H_2A_2

From Figure 2 - 41:

$$r_{O_2A_2} = r_{O_2H_2} + r_{H_2A_2}$$

$$r_{O_2H_2} = \frac{1}{2}(r_{OC} - r_{OO_2}) = \frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} d/2 \\ 0 \end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix} x - d/2 \\ y \end{bmatrix}$$

It should be noted that there are two possible positions for point A_2 depending on the orientation of vector $r_{H_2A_2}$, which can be manipulated by rotating vector $r_{O_2H_2}$ 90° clockwise or 90° anti-clockwise. In the anti-clockwise position, the vector is normalized. Multiplying this resultant vector by the length of the H_2A_2 (h_2) segment produces:

$$r_{H_2A_2} = -\delta_2 \frac{h_2}{o_2} E r_{O_2H_2} = -\delta_2 \frac{h_2}{o_2} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} r_{O_2H_2}$$

Where E is the matrix that rotates vectors 90° anti-clockwise. Resulting in:

$$h_2 = \sqrt{l^2 - o_2^2}$$

δ_2 is the branch index of arm 2 with:

$\delta_2 = +1$ if A_2 is on the right side of vector r_{O_2C}

$\delta_2 = -1$ if A_2 is on the left side of vector r_{O_2C}

Therefore for arm 2:

$$\theta_2 = atan2\left(y + \delta_2 \frac{\sqrt{l^2 - o_2^2}}{o_2}(d/2 - x), \quad x - d/2 + \delta_2 \frac{\sqrt{l^2 - o_2^2}}{o_2} y\right)$$

Similarly for arm 1:

$$\theta_1 = atan2\left(y - \delta_1 \frac{\sqrt{l^2 - o_1^2}}{o_1}(d/2 + x), \quad x + d/2 + \delta_1 \frac{\sqrt{l^2 - o_1^2}}{o_1} y\right)$$

δ_1 is the branch index of arm 1 with:

$\delta_1 = +1$ if A_1 is on the right side of vector r_{O_1C}

$\delta_1 = -1$ if A_1 is on the left side of vector r_{O_1C}

The set $\{sign(\delta_1),\ sign(\delta_2)\}$ defines the working modes: $++, +-, -+, --$.

Exceptions and singularities:

$o_2 \geq l$	Outside working space
$o_2 = 0$	Type 1 singularity
$o_2 = l$	Arm fully stretched

All singularities are handled by the software and are not permitted for the robot to follow as there are mechanical constraints.

Direct Kinematics (Mecademic, 2015, p.34).

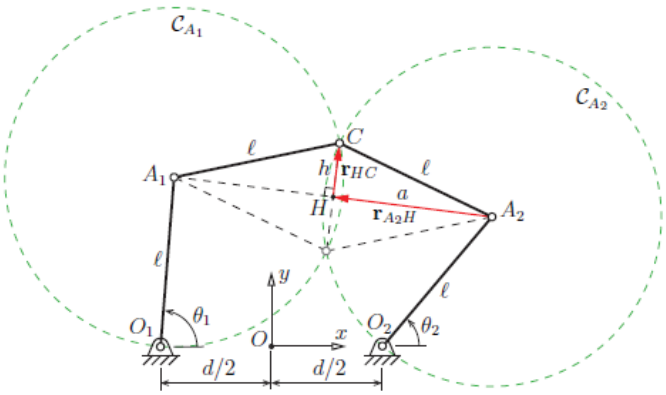


Figure 2 - 42. **DexTAR Direct kinematics mathematical definitions. (Mecademic, 2015).**

Oxy: Reference frame base coordinates (BCS)
O_1: Axis of revolution joint 1, arm 1
O_2: Axis of revolution joint 2, arm 2
A_1: Passive revolute joint 1 from arm 1
A_2: Passive revolute joint 2 from arm 2
C Tool centre position (TCP)
θ_1: Active joint angle 1, with respect to BCS
θ_2: Active joint angle 2, with respect to BCS

H: Midway point between A_1 and A_2
h: Length of segment HC
a: Length of segments HA_1 and HA_2
r_{A_1H}: Vector connecting points A_1 and H
r_{HC}: Vector connecting points H and C

From Figure 2 - 42:

$$r_{OC} = r_{OO_2} + r_{O_2A_2} + r_{A_2H} + r_{HC} = r_{OA_2} + r_{A_2H} + r_{HC}$$

Where:

$$r_{A_1H} = \frac{1}{2}(r_{OA_1} - r_{OA_2})$$
$$r_{OA_1} = \begin{bmatrix} -d/2 + l\cos\theta_1 \\ l\sin\theta_1 \end{bmatrix}$$
$$r_{OA_2} = \begin{bmatrix} d/2 + l\cos\theta_2 \\ l\sin\theta_2 \end{bmatrix}$$

It should be noted that there are generally two possible positions for point C depending on the orientation of vector r_{HC}, which can be manipulated by rotating vector r_{A_2H} 90° clockwise or 90° anti-clockwise. In the anti-clockwise position, the vector is normalized. Multiplying this resultant vector by the length of the HC (h) segment produces:

$$r_{HC} = -\gamma \frac{h}{a} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} r_{A_2H}$$

Where:

$$h = \sqrt{l^2 - a^2}$$

γ is the assembly mode index with:

$$\gamma = +1 \text{ if } C \text{ is on right side of vector } r_{A_2A_1}$$
$$\gamma = -1 \text{ if } C \text{ is on left side of vector } r_{A_2A_1}$$

Therefore:

$$x = \frac{1}{2}(-d + lcos\theta_1 + lcos\theta_2) + \frac{1}{2}\gamma\frac{\sqrt{l^2 - a^2}}{a}(lsin\theta_1 - lsin\theta_2)$$

$$y = \frac{1}{2}(lsin\theta_1 - lsin\theta_2) + \frac{1}{2}\gamma\frac{\sqrt{l^2 - a^2}}{a}(d - lcos\theta_1 + lcos\theta_2)$$

Exceptions:

$a = 0$	Type 2 singularity
$a > l$	No mechanical solution
$a = l$	Type 2 singularity

Design Aspects

The 5-bar parallel robot design adopted by DexTAR has several unique aspects which are accompanied by mechanical and implementation advantages.

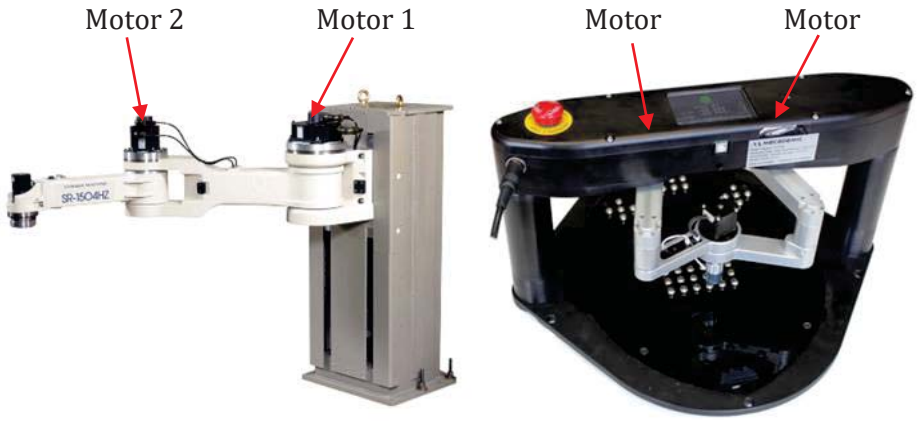


Figure 2 - 43. **SCARA motor locations, DexTAR motor locations. (Mecademic).**

Requiring only two motors to provide the Cartesian co-ordinates for DexTAR (fig. 2 - 43) is quite common among robots, however the placement of these motors provide simplicity in implementation. Since both motors needed to provide angular actuation of the arms sit at the base of the design, there is no need for complex bearing systems or transmission of this actuation through some medium such as belts, drive shaft, gear system, etc. This can come into significant effect for the SCARA design, as both arms require individual actuation (fig. 2 - 43), which can only be achieved through motor placement at the base of the subsequent arm or angular transmission of a motor through some medium, to allow that motor to be placed at the base. This is eliminated by the 5-bar arrangement. The three passive joints incorporated in this configuration are also relatively easy to implement compared to conventional robots which require actuation at another point on the arm. This does mean however, that there is no orientation control of the end-effector about the vertical axis.

The tool screw-in design is very useful. Mecademic have taken advantage of the lack of end-effector orientation control to utilise the simplicity accompanied by the pick and place method. Since the end-effector heads available consist of a drawing tool and an electromagnetic tool, orientation is of no consequence. This does limit the usefulness of the end-effector to applications which do not require orientation since this system cannot efficiently place asymmetrical objects for example. However, this also allows for the tool head to be screwed directly into the end-effector without taking orientation into account, simplifying the mechanism significantly.

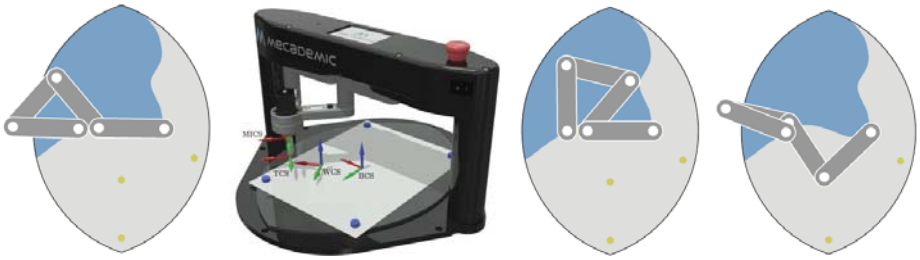


Figure 2 - 44. **DexTAR singularity positions. (Mecademic, 2015).**

The 5-bar configuration implemented by Mecademic has been refined to utilise as much of the working space available. The links have been designed to be 'multi-level'. From the Cartesian perspective, this allows links to overlap one another freely (fig. 2 - 44) without resulting in collision. The Z-axis motor is also of such height, that it does not collide with the two main actuating arms when overlapping. Both actuating arms have also been placed to maximise efficiency. This can be observed when both arms are normalized with the body of DexTAR, as seen in (fig. 2 - 44). These design features contribute significantly to the end-effector mobility throughout the workspace, although collisions and singularities must still be considered when the robot is in specific configurations (fig. 2 - 44).

2.4.2 Mover6

Figure 2 - 45. **Mover6 picture. (Commonplace Robotics).**

Commonplace Robotics (CPR) is a German-based company, specialising in the development, manufacture and distribution of robotic arms. Their aim is to provide the consumer with cost-effective equipment for pedagogical research and industrial applications. Two main production lines are available, including Mover and Worker, which focus on higher education/research or industrial applications respectively.

Throughout the CPR educational range, the Mover6 robotic manipulator arm is top-of-the-line. With the improved 2016 version recently released, the product generally sells for $5,599 USD. All CPR manipulator arms are based on the same mechanical principles, with similar configurations and hardware.

Mover6 communicates via USB, through the use of CPRs proprietary software CPRog. Robot Operating System (ROS) packages have been adapted to allow for direct control of the arm. The system supports the controller area network (CAN Bus) protocol.

Weight	3.5 kg
Payload	200 g
Reach	600 mm
Power consumption	30 W
Joint 1-3 velocity	$45°s^{-1}$
Joint 4-6 velocity	$60°s^{-1}$
Repeatability	$\pm$ 1.0mm (no load)

Table 2 - 14. **Mover6 specifications. (Commonplace Robotics, 2016).**

This product is only available for purchase from the CPR website or Robotshop.com. Each Mover6 unit is sold as a set, which includes several other accessories required to function.

Starter Set:

- Robotic manipulator arm
- Stand
- Two-fingered gripper
- PSU (12V, 5A)
- USB to CAN adapter
- CPRog Software (Installation CD)

Software

CProg is the proprietary software accompanying the Mover6 system. Prior to installation, the PCAN-USB driver must be installed to allow for communication with the arm controller at the base of the robot (Commonplace Robotics, 2014, p.12). This is included on the installation CD. It should be noted that all relating software is available for download at no charge from the CPR website.

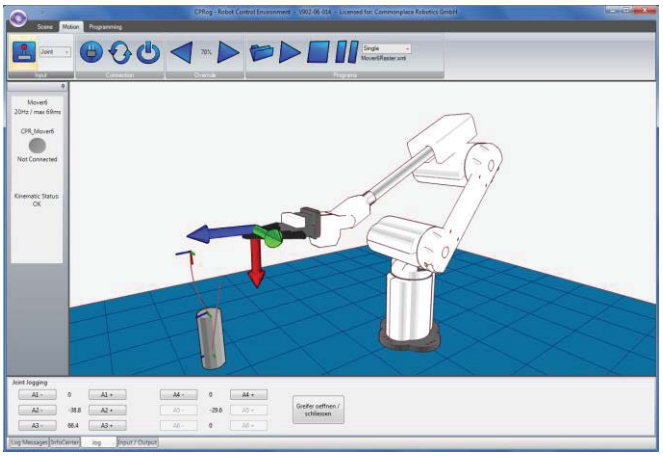


Figure 2 - 46. **CProg GUI screenshot. (Commonplace Robotics, 2014).**

The CProg programming environment allows for complete robot control through a graphically based interface. Both online and offline work is supported. The robot may be 'jogged' manually through the use of the joystick button, seen in the top left hand corner of figure 52. The 'override' button allows for robot speed control, ranging from 0-100%. A Logitech joy-pad is also available which interfaces directly with CProg, enabling real-time control (Commonplace Robotics, 2014, p.20).

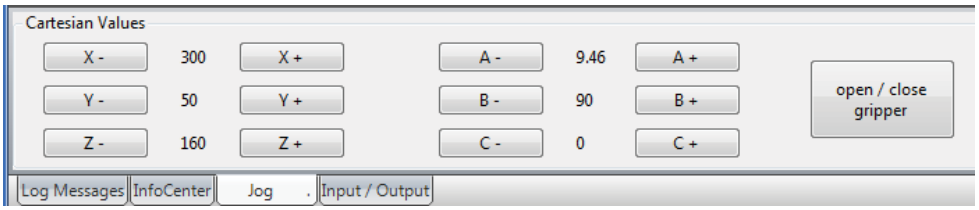


Figure 2 - 47. **CProg XYZ coordinate control. (Commonplace Robotics, 2014).**

CProg also allows for direct XYZ coordinate and gripper control (fig. 2 - 47). It should be noted that software limits have been enforced to avoid arm and obstacle collision. These may however be manually overridden.

Alternatively, the user may drag-and-drop the robotic arm in the 3D environment, followed by use of the refresh button to communicate the new position to the robot.

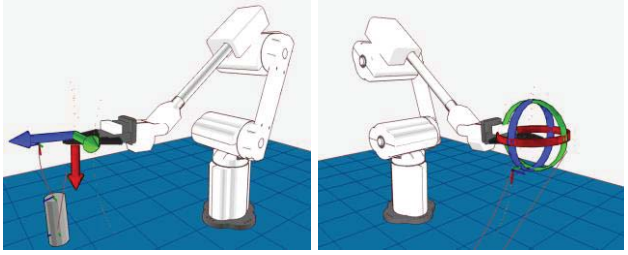


Figure 2 - 48. **Mover6 end-effector coordinate control screenshot. (Commonplace Robotics, 2014).**

CProg includes a graphical-command-chain style of programming called 'GraphEdit' which enables the user to string together a set of instructions for the robot to execute in order.

Figure 2 - 49. **CProg graphical command chain. (Commonplace Robotics, 2014).**

Linear	This command moves the robot from the current location directly to the target.	
Joint	Interpolates the axis from current position to target position with respect to joint coordinates.	
Wait	Basic wait command.	
DigitalOut	Sets digital output of the gripper or current end-effector. This is a binary command, allowing only 'open' and 'closed' states of the gripper.	
Loop	Basic loop command.	

Table 2 - 15. **Graphical command chain button definitions. (Commonplace Robotics, 2014).**

A 'record' feature is also available, which allows for location recording depicted by the user through the 3D interface. While 'record' is active, any actions made by the user in the 3D environment are recorded and may be communicated to the robot. These commands may also be saved, loaded and replayed as necessary.

More advanced than the graphical programming method, CProg includes a 'TextEdit' editor (fig. 2 - 50), allowing for more precise control of the arm. This spread-sheet style programming language works in conjunction with the 'record' feature. Once a command string has been recorded, it may be edited using 'TextEdit'. Individual values may also be altered or entered directly into a blank document. Windows default copy and paste commands are also supported.

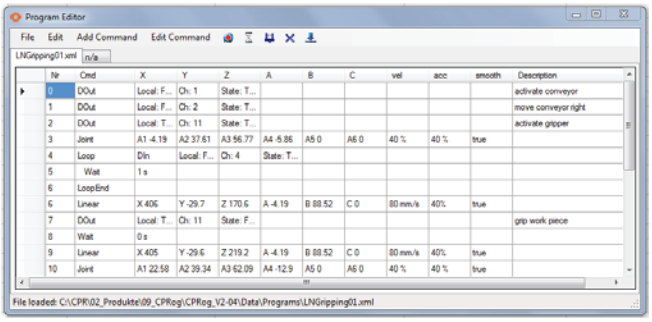


Figure 2 - 50. **CProg TextEdit window. (Commonplace Robotics, 2014).**

Robot Operating System (ROS) enables direct communication with Mover6 via the CAN protocol (Commonplace Robotics, 2014, p.31). This allows for robotic commands to be sent directly to the controller. Kinematics are handled by the controller, so it is only necessary to send coordinates and speed, which may be done only once set-up has been completed to calibrate the system and set acceleration and velocity values.

Command examples:

To robot

'Dir 95.0 14.1 -30.0 0.0 0.0 0.0 open 8\n'
'Pos 234.0 123.1 987.3 0.0 90.0 0.0 8\n'

From Robot

'Done \n'

Once the robot has communicated *'Done \n'*, it will begin to execute the given commands. These commands include: XYZ position, gripper state, velocity value and acceleration value, all followed by an 8 and Newline character to denote end of line.

Design Aspects

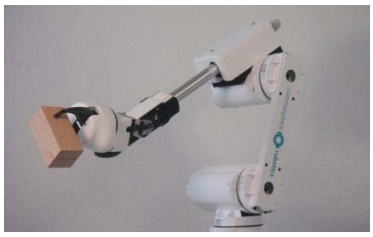


Figure 2 - 51. **Mover6 operational picture. (Commonplace Robotics, 2014).**

The Mover6 system incorporates the use of a link which operates through extension. This is a unique attribute among all educational robotics at this level. A flexible cabling sleeve is used to allow wiring through the metallic structure of the arm (fig. 2 - 51). It is unclear how this actuation is achieved, but it is added that absolute encoders are used.

Excluding the extension arm, all other links are actuated through the use of servo motors (Commonplace Robotics, 2014, p.8). Stepper motors are not used. This simplifies the manufacture and design of the manipulator arm immensely. Most servo motors operate via pulse width modulation (PWM), with certain duty cycles corresponding to distinct angular locations.

The end-effector mount is of such design to enable the attachment of any number of customised gripers. A mounting flange is supplied, with 4 x M3 fastening locations (fig. 2 - 52). A 6-pin connector is also available for use; supplying 12V 0.5A and two digital input output pins.

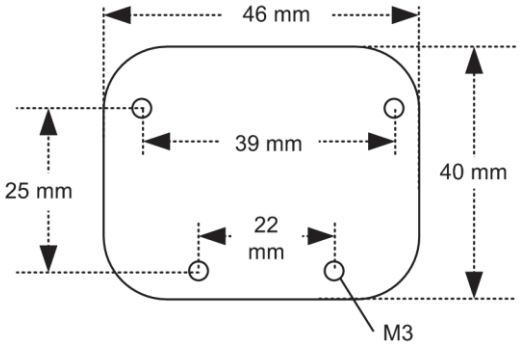


Figure 2 - 52. **Mover6 mounting diagram. (Commonplace Robotics, 2014).**

2.5 Market Research - Modular Robotics

2.5.1 Cublets

Figure 2 - 53. **Cublets system picture. (Modrobotics).**

Officially founded in 2009, Modrobotics is a Colorado, US based company focused on the design and manufacture of robotic toy construction systems for kids and education. Initially roBlocks, the Cublets system was publically announced in August of 2010 after 3 years of development.

Cublets are building blocks which make use of electronic components to provide a robotic aspect to the system. Magnets are used to create a 'snap' feature, which attract blocks to one another, simplifying construction significantly. Blocks communicate with their neighbour and pass along electrical power and data. There is a wide range of blocks which offer varying functions and unique roles. Behaviour of a collection of blocks is dependent on the configuration, type of block and placement.

Focused on ages 4+ the system is initially very simplistic, but complexity is amplified as the number of blocks interconnected is increased. There are three main types of Cublet; sense cubes, think cubes and action cubes. Sense cubes acquire information on their surroundings or from the user, while think cubes use this information to make decisions based on their default settings or user-assigned values. Action cubes provide physical outputs, including movement, light and sound.

Drive  The drive Cublet contains a single motor, interconnected through a gear system to the roller wheels providing horizontal movement. Although varying speed, this Cublet accommodates for movement in a single direction only.	Rotate  This Cublet allows for single face rotation.
Speaker  This block contains an amplifier and speaker arrangement. It should be noted that it is a tweeter speaker and does not support complex audio.	Flashlight  An LED is used to emit light. The light source can vary its output intensity.

Bar Graph  This is an array of LED strips which act as a bar graph, displaying information provided to it by its neighbor blocks.	Knob  A simple potentiometer block, which gains a value from the user depending on the orientation of the knob.
Brightness  Through the use of an analogue photocell, this block detects the amount of light hitting the sensor and converts this into useful data.	Distance  Based on infrared, this blocks detects distances of objects relative it itself. It is claimed to be accurate between 10 and 80cm.
Temperature  Containing a thermistor, this block is able to detect temperature. With a claimed range of around 0 to 35°C	Inverse  This block inverts the data it receives and passes it along. Specifically, it will take a weighted average of its inputs and output a value of one minus that average.
Minimum  This block will accept any amount of data input, but only output that which is the lowest.	Maximum  Accepting any amount of inputs, this block only outputs the highest of those received.
Battery  This Cublet provides electrical energy for the collection of blocks which are interconnected with it. Contains a rechargeable battery with a micro USB socket.	Passive  This block is simply a building component. It passes on data and power.
Blocker  The blocker Cublet blocks the transmission of data, however it does pass on power.	Bluetooth Cublet  The Bluetooth Cublet is the newest addition among the Cublets. This allows for firmware updates of those blocks it is interconnected with, as well as provide information to the user concerning block values. It also enables the user for direct block value manipulation, which allows for remote control capabilities.

Table 2 - 16. **Cublet type definitions. (Modrobotics, 2012).**

Cublets Education

The Modrobotics website offers a large range of pedagogical resources, claiming that the curricula on offer delivers 40+ hours of learning time (Modrobotics). The credibility of this claim is grossly supported by the numerous and comprehensive lesson plans available for free from the 'Education' section on the website.

Lesson Plans on offer:

- Robot creatures and their behaviour
- Robots and sensing
- Characteristics and categories with Cublets
- Cublets and cause and effect
- Easy Cublets robotics
- Robotics with Cublets
- Class Unit: Engineering and design principles
- Class Unit: Computational thinking

Cublet Activities on offer:

- Cublets Challenge Part 1 – 4
- 10 Cool things to do with Cublets!
- 10 More cool things to do with Cublets!

Upon further investigation of the lesson plans (Modrobotics*)*, it is clear to see that a significant amount of resources have gone into the development of these educational activities. The planners are produced from an educator's point of view, allotting time for each segment. Not only do these plans provide the educational resource to be conveyed to the student, they also include strategies on how to transfer this information in an engaging manner, stressing student involvement, all in the context of the Cublets system.

At the end of each segment there is a 'Suggested age variations/progression' section. This section aims to cater to varying difficulty based on age and includes '4 years old to 8 years old', '8 years old to 11 years old' and '11 years old and up' with the more difficult subjects like 'Engineering and design principles' recommended for '12+'. Under each heading, there are varying difficulty suggestions, which warp the current activities to increase or decrease difficulty to appeal to the varying ages. All lesson plans conform to current STEM (Science, Technology, Engineering and Mathematics) education. Some of the more advanced planners also include a Q&A section.

Cublets Studio

Cublets Studio is a proprietary programming environment developed solely for use with the Cublets System. The software currently supports Mac OS X and Windows. Based on Arduino, the language is very similar to C, but contains its own syntax and API references (Modrobotics).

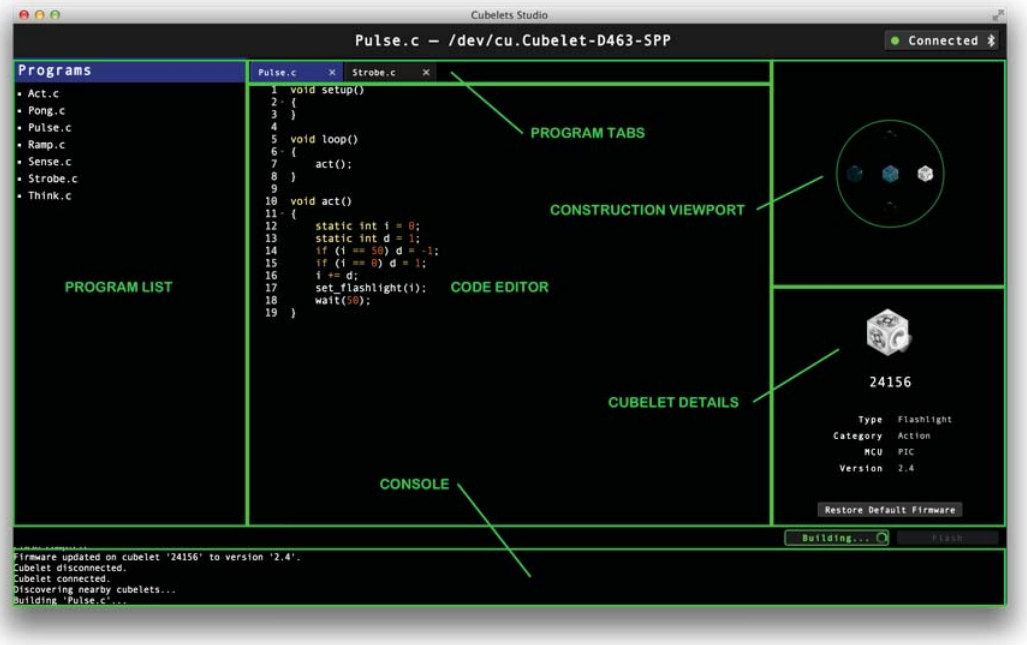


Figure 2 - 54. **Cublets studio screenshot. (Modrobotics).**

The software communicates with the Cublet system through the Bluetooth Cublet. The addition of this Cublet allows for full control of the interconnected Cublets, with individual block control.

Additions to Arduino (Modrobotics):

get_block_value()
set_block_value()
inverse()

block_type
block_value
neighbour_data

act()
loop()
sense()

Including these additions to the Arduino language which Cublets Studio have made, there are numerous more, which deal with individual Cublet type addresses, such as 'BAR_GRAPH_CUBLET' and 'PASSIVE_CUBLET' which communicate data to those types specifically. The software also enables the user to see current block values and change default values. It should also be noted that the new firmware 'OS4' for Cublets are updated through the Bluetooth Cublet.

Cublets Mobile App

Working in conjunction with the Cublet Studio software, Modrobotics have released a mobile app. Although this app is not as comprehensive and does not deliver as much control over the Cublets system as Cublets Studio, it does offer a more simplistic dimension to robot interaction and understanding. The app is more visually focused, and provides a more graphical representation of Cublet data and configuration. Individual block values can be manipulated through the Bluetooth block as per usual, however a slider is provided to increase ease of use on mobile devices. It is also possible to read current block values through the application.

Figure 2 - 55. **Various Cublets mobile app screenshots, Cublets logo. (Modrobotics, 2014).**

The 'Culbets' mobile application is available on both Apple and Android devices. It should be added that this app is downloadable at no cost.

Design aspects

Unfortunately there is limited to no information regarding the internal workings or design of Cublets supplied by Modrobotics. Luckily, there have been third party reviews of the product, which investigate the internals (Robot Test Kitchen, 2014).

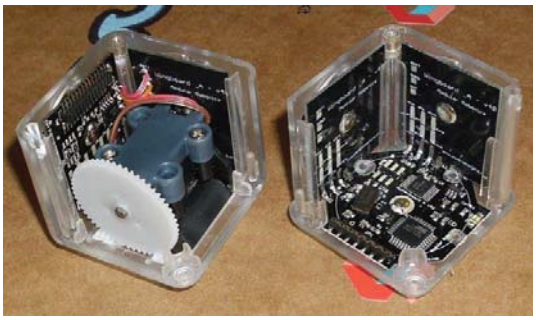


Figure 2 - 56. **Drive cube. (Robot Test Kitchen, 2014).**

Figure 2 - 57. **Rotate cube. (Robot Test Kitchen, 2014).**

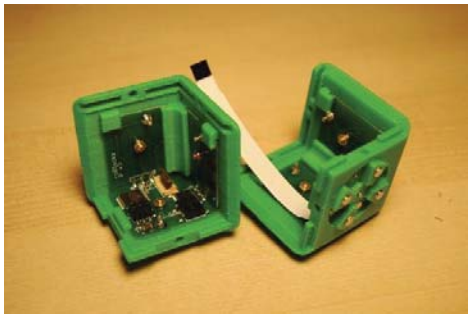


Figure 2 - 58 **Passive cube. (Robot Test Kitchen, 2014).**

From Figures 2 - 56 through to 2 - 58, we are able to ascertain some of the unique design decisions made by Modrobotics to produce such a successful product. It is clear that plastic injection moulding would be a good option for producing the outer cube covering. Although plastic is used for the shell of the blocks, it should be noted that there is a single metallic insert, as seen in the left cube in Figure 2 - 56 or 2 - 57. This allows for the holding screw to fasten itself in threaded metal, rather than plastic. This is a great design choice as it somewhat prevents over-tightening as well as provides a smoother screw insert.

The slide-in design allows for two separate halves of the cube to complement one another. Once the electronic circuitry has been fastened to the plastic, the two halves simply slide together, with one construction screw securing the cube. The electronics accommodate the slide-in design through the use of a socket system, as seen in Figure 2 - 56 and 2 - 57. When the two halves are attached, the contacts of one circuit board connect to the other

via the multi-socket (fig. 2 - 56). Although this is not the case for the 'passive' Cublet, where a manual connector has been used and there appears to be no socket (fig. 2 - 58).

Cublets incorporate 'plug and play' capabilities. While the system is in operation, another Cublet may be added without any need to disconnect power, nor does the addition of an extra cube disrupt current data communications. This leads to the assumption that a microcontroller is used to allow this feature. The communications throughout the interconnected Cublets must also either be addressed or share a common line, with sense Cublets producing an output, while action Cublets simply wait for an input.

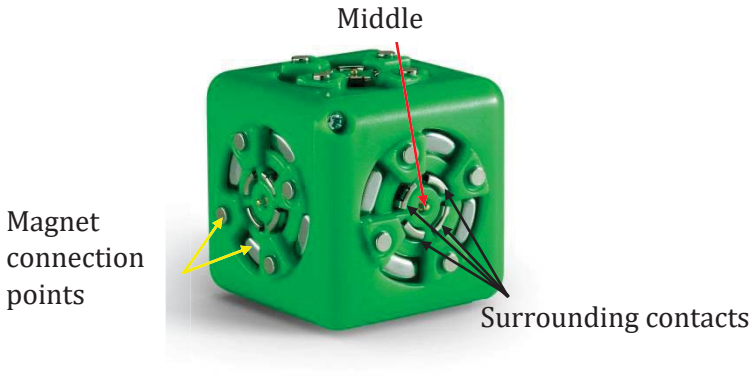


Figure 2 - 59. **Passive cube contact diagram. (Modrobotics).**

The electronics utilised in the Cublets system seem fairly uncomplicated to manufacture. From the 'How are Cublets Made?' instructional video provided on their website, they inform the viewer that surface mount technology (SMT) is used to produce the circuit boards used for their product. Componentry is fitted to their designated position on a flexible printed circuit board (PCB), followed by the addition of liquid solder. Once complete, the board is run through a kiln, to allow the solder to solidify. This process is automated.

Cublets are able to snap together in any orientation. The magnet connection points used interconnect (fig. 2 - 59). This allows for a strong link between Cublets, as the magnets will attract one block to another, but due to the plastic interconnecting style of the design, the links will prevent rotational movement. Along with data transmission, Cublets share power to their neighbours regardless of orientation. Although uncertain, it would seem logical that the middle pin seen in Figure 2 - 59 is the ground pin, with the cathode connected to either two of the four surrounding contacts or through a magnet contact point (which would assume not all magnet points are magnetic). Another assumption can be made which states that the middle pin is the data pin, with the four surrounding contacts arranged in an alternating anode-cathode series.

The Bluetooth module block presents several noteworthy features. Once connected to power, each Bluetooth block exhibits its own unique colour sequence. A block which flashes red-green-blue will appear with name 'Cublet-RGB' to a corresponding Bluetooth device. Similarly a block which flashes Cyan-Magenta-Yellow appears as 'Cublet-CMY'. Once powered, blocks intercommunicate with one another to establish separate Bluetooth addresses. Two-way communication is also permitted by the system. This concept, although simple, provides a unique and excellent solution to the problem of multiple transmission devices interconnected to the same agglomeration of blocks.

Modrobotics have implemented LEGO adaptability to the Cublets system. The consumer may purchase adaptors that allow for Cublet to LEGO interconnect ability. This is a great marketing strategy as LEGO is a very popular and well established company, exposing Cublets to mainstream construction toys.

2.6 Market Research - Price Comparison

Industrial
Intelitek Scorbot-ER 9 PRO - $20,000
Intelitek Motoman MH5 - $45,000

Service
Kinova JACO 6 DOF - $29,950
Kinova Jaco 4 DOF - $22,950
3-Fingered Gripper - $4,950
2-Fingered Gripper - $3,500

Bestic 36 Month Rental - $110 / month ($3,960)
Bestic 24 Month Rental - $158 / month ($3,792)
Bestic 12 Month Rental - $303 / month ($3,636)

Educational
Mecademic DexTAR - $4999

CPR Mover6 - $5699.05
CPR Mover4 - $3868.42

Modular
Cublets Twelve-Kit - $329.95
Cublets Educator Pack - $1,138.00
Engineering Expansion Pack - $299.95
Computational Thinking Expansion Pack - $235.95
Life Science Expansion Pack - $220.95

2.7 Control Methodologies

2.7.1 PID Control

As claimed by Minorsky (1992), 'three-term' controllers, later known as Proportional Integral Derivative (PID) controllers (fig. 2 - 60) were first analysed and discussed by Russian American engineer Nicolas Minorsky in 1922 for the purpose of automatic ship steering, but it was not until 1939 when the Taylor Instrument Companies released its re-designed version of the 'Fulscope' pneumatic controller, implementing 'pre-act' for the first time, that PID controller popularity grew in mainstream industry (Bennet, 1993).

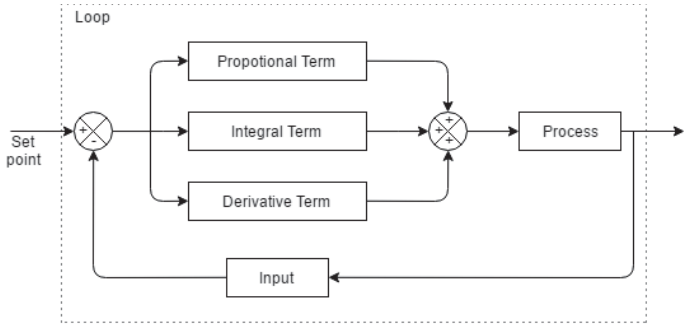


Figure 2 - 60. **PID control loop diagram.**

With the increasing number of field-adjustable devices in use, the problem of loop tuning arose. Bennet (1993) states this was first recognised as a weakness by the Taylor Instrument Companies, which carried out extensive investigations in an effort to produce a repeatable method for finding optimal gain settings, resulting in two separate papers published in 1942 and 1943 by J.G Ziegler and N.B Nichols. This provided the classical method of PID loop tuning, based on the open-loop response of the system.

$$Output = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t)$$

$e(t) = SP - PV(t)$	error value (set point - process variable)
K_p	proportional gain
K_i	integral gain
K_d	derivative gain
t	time or instantaneous time

Figure 2 - 61. **PID algorithm ideal form.**

$$Output = K_p \left(e(t) + \frac{1}{T_i} \int e(t)dt - T_d \frac{d}{dt} PV(t) \right)$$

$e(t) = SP - PV(t)$	error value (set point - process variable)
K_p	proportional gain
T_i	integral time
T_d	derivative time
t	time or instantaneous time
$PV(t)$	change in process variable

Figure 2 - 62. **PID algorithm standard form.**

Despite the slightly increased mathematical complexity of the standard form (fig. 2 - 62), it is most commonly used in industry as it tends to be more robust when compared to the ideal form (fig. 2 - 61). This is mainly due to the elimination of the derivative spike with set point change, as this term is based on the change in process variable, rather than calculated error.

Ziegler and Nichols' tuning scheme provides a methodological approach to PID loop tuning (Zielger & Nichols, 1942). Although the heuristic nature of the method does not guarantee optimal loop settings, Microstar Laboratories have determined it will provide sufficient immediate loop control, which may later be refined depending on system response requirements. It should be noted that Ziegler-Nichols tuning rules assume that the output response is a first order system with dead time of the following transfer function (Åström & Hägglund, 1995):

$$y(s) = \frac{K_P e^{-Ls}}{(\tau s + 1)}$$

$y(s)$	output response
K_P	process gain
e^{-Ls}	dead time
s	time constant

***Figure 2 - 63*. First order transfer function with dead time. (Åström & Hägglund, 1995).**

To find the gain values for the standard PID algorithm using this method, both the integral and derivative terms must be turned off. K_p is then slowly increased from zero until the system reaches stable and consistent oscillations. At this point, the gain is labelled 'ultimate gain' (K_u) and the period of oscillation is measured (T_u). Once these two system properties have been attained, the Ziegler-Nichols tuning table (tab. 2 - 17) is used to determine proportional gain and integral and derivative time depending on controller type.

Ziegler-Nichols gain settings			
Control Type	$\boldsymbol{K_p}$	$\boldsymbol{T_i}$	$\boldsymbol{T_d}$
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
Classic PID	$0.6K_u$	$T_u/2$	$T_u/8$
Pessen Integral Rule	$0.7K_u$	$T_u/2.5$	$3T_u/20$
Some overshoot	$0.33K_u$	$T_u/2$	$T_u/3$
No overshoot	$0.2K_u$	$T_u/2$	$T_u/3$

***Table 2 - 17.* Ziegler-Nichols gain settings. (Microstar Laboratories).**

Although the Ziegler-Nichols method does provide immediate results, Åström & Hägglund (1995) states that it has been shown to often produce control that is too oscillatory and may cause a poor response time for systems with large dead time. Because of this, other avenues for PID loop tuning have been explored, including tuning parameters based on transient, steady state and frequency response. Manual tuning has also been determined to be viable depending on personnel experience.

Owing to the simple implementation and robust nature of PID controllers, they have been the most widely used controller in industry for the past fifty years (Seung-Min & Tae-Yong, 1997). Partly as a result of the increase in number of automated systems with the 1980s 'robot boom', a great deal of work has gone into PID loop tuning and PID-related control methodologies (Seung-Min & Tae-Yong, 1997).

PID-based control has been used extensively for robotic manipulator position control (Abu Talib et al., 2013, Shauri et al. 2014, Intelitek, 2008) and mechatronics systems (Zhao et al., 1992).

A paper published by Abu Talib et al. (2013) has evaluated DC motor performance under varying loads for precise PID position control of a 2-DOF robotic finger. The system utilises 2 DC-micromotors for mechanical actuation and magnetic encoders for position acquisition, a motion controller and PC are used to control the finger. According to Abu Talib et al. (2013), the system was tuned using a trial and error method, based on the performance of the transient responses of the system in terms of maximum overshoot, rise time, settle time and peak time and the steady-state error.

Abu Talib et al. (2013) transient response definitions:
Maximum overshoot*: Maximum amount system output overshoots beyond desired setting.*
Rise time*: Time required for step response to rise from 10% to 90% (underdamped).*
Settling time*: Time taken for the step response to settle within ±2% of desired output.*
Peak time*: Time required for response to reach first peak of output.*

Initially, the responses of individual motors removed from the system were evaluated. From there, gain parameters were gradually increased manually (fig. 2 - 64). Abu Talib et al. (2013) used an error formula (fig. 2 - 64) in conjunction with the transient responses of the system as a basis for comparison in determining the optimal gain settings for the desired output performance. The implemented process tuned the motors for joint and joint 2 sequentially, starting with the motor closest to the base.

$$Error(\%) = \left|\frac{output - desired}{desired}\right| \times 100$$

Gain of position controller (PP)	Proportional Term (POR)	Integral Term (I)	Derivative Term (D)	Overshoot (°)
64%	1	5	5	7.70°
64%	2	10	10	3.50°
64%	3	15	15	2.80°
64%	4	20	20	1.75°
64%	5	25	25	1.40°
64%	6	30	30	1.05°
64%	7	35	35	0.70°
64%	8	40	40	0.53°
64%	9	45	45	0.35°
64%	10	50	50	0.18°
64%	10	50	55	0.11°
64%	10	50	55	0.10°

***Figure 2 - 64*. Tuning table and error percentage calculation. (Abu Talib et al., 2013).**

Finally, the newly obtained PID values were validated for continuous motion for both motors running synchronously. Abu Talib et al. (2013) concluded that loads have a significant impact on the transient response parameters of a PID-controlled manipulator, which differ for individual joints. The influence of varying loads on a system may however

be mostly mitigated by obtaining appropriate PID gain settings, which can result in good performance.

In an extension of the aforementioned paper, several previously described robotic fingers are used to construct a 7-DOF, 3 fingered grasping hand (Shauri et al. 2014). Using the same gain settings achieved in their previous paper, the configuration was shown to provide sufficient performance to enable it to accurately grip objects.

On account of the control parameter uncertainties related to robotic arm manipulation, as well as the often non-linear responses of these systems, several advanced PID-based control methodologies have been investigated (Seung-Min & Tae-Yong, 1997, Zhao et al.,1992, Merabet & Gu, 2010).

Seung-Min & Tae-Yong (1997) have determined that while a classical PID controller applied to a DC-motor driven system may provide good control, this can only be reliably achieved if all operating conditions are exactly known. In cases where these properties are not known or change, fixed PID control may not guarantee the desired output profile (Seung-Min & Tae-Yong, 1997). Once PID controllers have been tuned, performance may degrade over time as system dynamics change. Taking this into consideration, Seung-Min & Tae-Yong (1997) have proposed an adaptive PID learning algorithm, which consists of a set of learning rules for PID gain tuning, as well as taking into account an auxiliary input. The suggested control method aims to drive an uncertain DC motor system to the desired location asymptotically, by means of position and torque feedback inputs exclusively (Seung-Min & Tae-Yong, 1997).

DC motor model as defined by Seung-Min & Tae-Yong (1997):

$$Li_a + Ri_a + K_b\theta = v_a$$
$$J\ddot{\theta} + B\dot{\theta} + K_c\theta + T_l = K_a i_a = T_a$$

v_a	applied voltage
i_a	armature current
θ	rotor displacement
T_a	output torque
T_l	unknown load torque
L	unknown armature inductance
R	unknown armature resistance
K_a	unknown torque constant
K_b	unknown back-emf constant
K_c	unknown spring constant
J	unknown rotor inertia
B	unknown vicious-friction coefficient

***Figure 2 - 65*. Dynamic model of DC motor. (Seung-Min & Tae-Yong,1997).**

This method begins by re-arranging the DC motor model:

$$\ddot{\theta} + \frac{1}{J}\left(B + \frac{K_a K_b}{R}\right)\dot{\theta} + \frac{K_c}{J}\theta + \frac{1}{J}T_l = \frac{K_a}{JR}v_a$$

$$\ddot{\theta} + a\dot{\theta} + b\theta + f = dv_a$$

a	$\frac{1}{J}\left(B + \frac{K_a K_b}{R}\right)$
b	$\frac{K_c}{J}$
f	$\frac{1}{J}T_l$
d	$\frac{K_a}{JR}$

***Figure 2 - 66*. Simplification by variable definition of dynamic DC motor model. (Seung-Min & Tae-Yong, 1997)**

From there, the learning controller is defined in terms of the applied motor voltage output and its two inputs, PID feed-back and the feed-forward learning input (Seung-Min & Tae-Yong, 1997).

$$v_a = v_{fb} + v_I$$

v_a	applied motor voltage
v_{fb}	PID feed-back input
v_I	feed-forward learning input

***Figure 2 - 67.* Adaptive learning controller formula, applied motor voltage as output. (Seung-Min & Tae-Yong, 1997).**

Where:

$$v_{fb} = K_p e + K_I \int e + K_D \dot{e}$$

e	error *value* $(e = \theta_d - \theta)$
θ_d	desired angular position
θ	actual angular position

***Figure 2 - 68.* Feedback learning input component. (Seung-Min & Tae-Yong, 1997).**

This yields an error system of (from DC motor model and learning controller):

$$\ddot{e} + (dK_D + a)\dot{e} + (dK_p + b)e + dK_I \int e = \dddot{\theta}_d + a\ddot{\theta}_d + b\dot{\theta}_d + f - d\dot{v}_I$$

$$\ddot{e} + a_0\dot{e} + b_0 e + c_0 \int e = V_d - d\dot{v}_I$$

V_d	$\dddot{\theta}_d + a\ddot{\theta}_d + b\dot{\theta}_d + f$
a_0	$dK_D + a$
b_0	$dK_p + b$
c_0	dK_I

***Figure 2 - 69*. Error system definition. (Seung-Min & Tae-Yong, 1997).**

Assuming the target characteristic equation of the error is given by Figure 2 – 69, which assumes the existence of true PID gains ($\widetilde{K_P}$, $\widetilde{K_I}$, $\widetilde{K_D}$) that satisfy matching conditions.

$$s^3 + \widetilde{a_0}s^2 + \widetilde{b_0}s + \widetilde{c_0} = 0$$

$\widetilde{a_0} = d\widetilde{K_D} + a$
$\widetilde{b_0} = d\widetilde{K_p} + b$
$\widetilde{c_0} = d\widetilde{K_I}$

Figure 2 - 70. **Target characteristic equation and PID gain definitions. (Seung-Min & Tae-Yong, 1997).**

Using $\{\widetilde{a_0}, \widetilde{b_0}, \widetilde{c_0}\}$ from the target characteristic equation, the error system can be written as:

$$\ddot{e} + \widetilde{a_0}\dot{e} + \widetilde{b_0}e + \widetilde{c_0}\int e = V_d - dv_I + (\widetilde{a_0} - a_0)\dot{e} + (\widetilde{b_0} - b_0)e + (\widetilde{c_0} - c_0)\int e$$

Figure 2 - 71. **Error system from target characteristic equation. (Seung-Min & Tae-Yong, 1997).**

From definitions of $\{a_0, b_0, c_0\}$ and $\{\widetilde{a_0}, \widetilde{b_0}, \widetilde{c_0}\}$ we obtain:

$$\ddot{e} + \widetilde{a_0}\dot{e} + \widetilde{b_0}e + \widetilde{c_0}\int e = d\widetilde{\widetilde{K_D}}\dot{e} + d\widetilde{\widetilde{K_P}}e + d\widetilde{\widetilde{K_I}}\int e + d\tilde{\tilde{v}}_I$$

$\widetilde{\widetilde{K_D}}$	$\widetilde{K_D} - K_D$
$\widetilde{\widetilde{K_P}}$	$\widetilde{K_P} - K_P$
$\widetilde{\widetilde{K_I}}$	$\widetilde{K_I} - K_I$
$\tilde{\tilde{v}}_I$	$v_d - v_I$ for $v_d = \frac{1}{d}V_d$

Figure 2 - 72. **Final error system. (Seung-Min & Tae-Yong, 1997).**

As stated by Seung-Min & Tae-Yong (1997), the target characteristic equation is driven by the PID gain error terms and learning error input. In this way, the learning rules will update the PID gains and learning input so that they converge to their ideal values $\{\widetilde{K_D}, \widetilde{K_P}, \widetilde{K_I}\}$ and $\{v_d\}$ asymptotically, while maintaining stability in the error system.

Learning Rules:

$$K(t) = pr[K(t-T)] + A_1 xy(t)$$
$$v_I(t) = pr[v_I(t-T)] + A_2 y(t)$$

$pr[.]$	projection into bounded set
A_1, A_2	positive learning gain matrices
$t - T$	update interval

Figure 2 - 73. **Learning rules as defined by Seung-Min & Tae-Yong, 1997.**

Once the learning rules have been defined depending on the performance characteristics desired by the user, Seung-Min & Tae-Yong (1997) have shown through simulation and experimentation that the adaptive algorithm can provide both better set point convergence and disturbance rejection (fig. 2 - 74). To compare traditional PID with the

adaptive learning algorithm, the parameters of a DC motor were provided and the fixed PID gains were acquired through the target characteristic equation: $(s+1)^2(s+100) = 0$. Constrains for the adaptive learning PID gains were implemented, and Seung-Min & Tae-Yong (1997) calculated the gain matrices by use of the SPR lemma function. Throughout the simulation, the motor parameters are known to the fixed-PID controller, while they are not known by the adaptive PID learning controller. The external torque disturbance applied throughout Figure 2 - 74 is $10sin(\pi t)$.

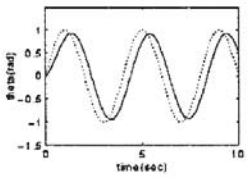


Figure 5. Response of the classical PID controller to a sinusoidal input without load torque(simulation result)

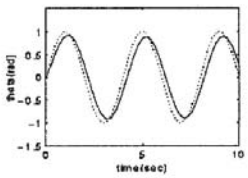


Figure 6. Response of the PID learning controller to a sinusoidal input without load torque(simulation result)

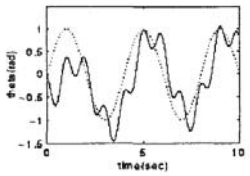


Figure 7. Response of the classical PID controller to a sinusoidal input with sinusoidal load torque variation(simulation result)

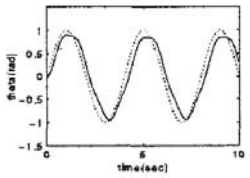


Figure 8. Response of the PID learning controller to a sinusoidal input with sinusoidal load torque variation(simulation result)

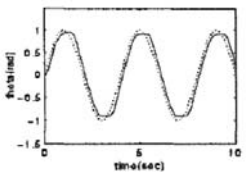


Figure 11. Response of the classical PID controller to a sinusoidal input without disturbance(experimental result)

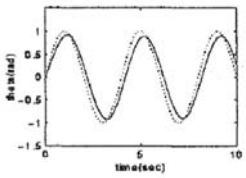


Figure 12. Response of the PID learning controller to a sinusoidal input without disturbance(experimental result)

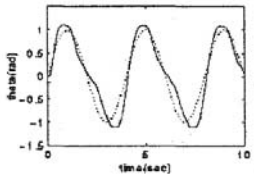


Figure 13. Response of the classical PID controller to a sinusoidal input with sinusoidal disturbance variation(experimental result)

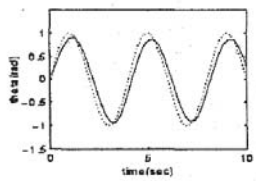


Figure 14. Response of the PID learning controller to a sinusoidal input with sinusoidal disturbance variation(experimental result)

Figure 2 - 74. **Various control responses. (Seung-Min & Tae-Yong, 1997).**

Seung-Min & Tae-Yong (1997) claim that this algorithm may be easily implemented on a 'cheap microprocessor' as the proposed learning rules do not require a great deal of processing power due to their simplicity once set-up.

2.7.2 Cascade PID control

Because of the inherent characteristics of robotic manipulator control, the system may not always respond linearly, depending on loads and position. The nature of classical PID only allows for good control if the system response is linear or approximately linear. As such, several other avenues of control have been explored to deal with non-linear systems (Åström & Hägglund, 1995, Novák & Perfilieva, 2001, Merabet & Gu, 2010).

A cascade controller (fig. 2 - 75) comprises of two interacting PID loops, where the outer-loop determines the set point of the inner-loop. Each loop has its own distinct input (Åström & Hägglund, 1995). This arrangement can provide improved dynamic and disturbance rejection performance compared to traditional PID.

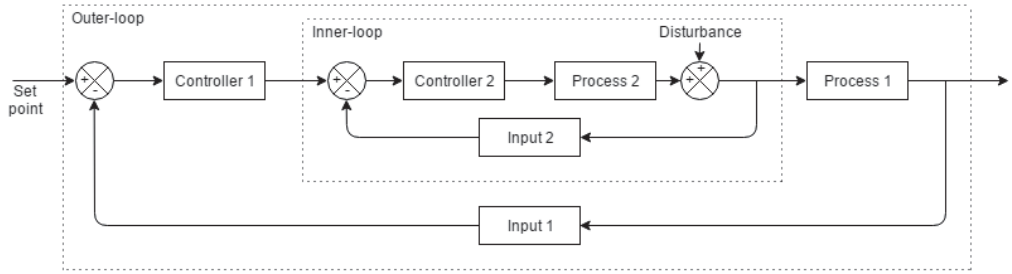


Figure 2 - 75. **Cascade PID control loop diagram.**

The inner-loop usually handles a much faster changing parameter relative to the outer-loop; in this way the inner-loop may control the disturbance before it can impact the outer-loop, which is where the improved control presents itself (Åström & Hägglund, 1995). The outer-loop controls the main process . According to Åström & Hägglund (1995) the inner-loop must run at least 5-10 times faster than the outer-loop to avoid any interaction between the two. Åström & Hägglund (1995) also recommend using a proportional-only or PI controller for the inner-loop and tuning the inner-loop and outer-loop sequentially, beginning with the former.

2.7.3 Fuzzy Logic Control

Boolean logic as first introduced by George Boole in *The Mathematical Analysis of Logic* (Boole, 1847) is a mathematical concept in which a variable must be true or false exclusively, usually denoted 1 or 0 respectively. Fuzzy logic is based on this notion, except that it employs a degree of truth, which allows for a logical spectrum ranging from 1 to 0.

A classic example of this is given by Novák & Perfilieva (2001), in which one examines a heap of wheat. Traditional logic will allow for either, *a heap* or *not a heap*. This does not take vagueness into account, as one grain of wheat does not make a heap, neither does two, three etc. hence, there are no heaps (Novák & Perfilieva, 2001).

Through this type of logic, a membership function may be established. A membership function is a map which assigns non-numeric output linguistic value terms to express a numeric input fuzzy value on a set scale (Passino & Yurkovich, 1992). Figure 2 - 77 is an example of this in which the fuzzy input variable *age* is used to determine the output linguistic value, in this case, *young*, *medium age* or *old*.

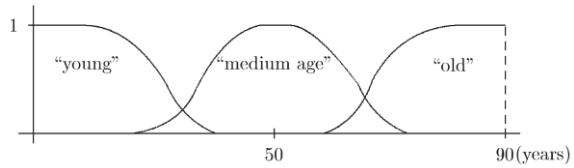


***Figure 2 - 76*. Fuzzy logic linguistic membership function. (Novák & Perfilieva, 2001).**

These principles are utilised in conjunction with an output fuzzy rule set to fashion a fuzzy logic controller (fig. 2 - 79), which is to say that the output will be dependent on the input in a specified way. Usually, the error and derivative of the error are used in this type of control if there is a single input (Cerecero-Natale et al., 2012, Passino & Yurkovich, 1992). There are several types of membership functions, most commonly, Gaussian and triangular.

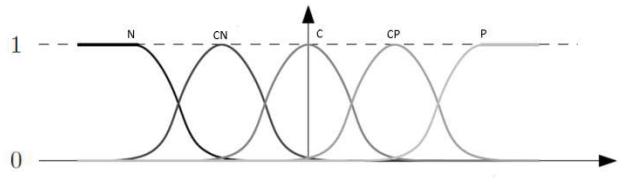


***Figure 2 - 77*. Gaussian membership function.**

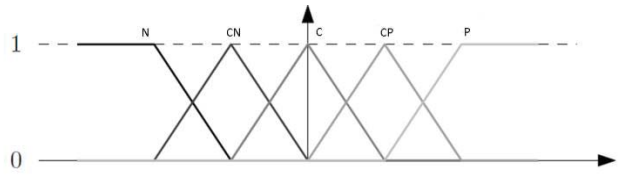


***Figure 2 - 78*. Triangular membership function.**

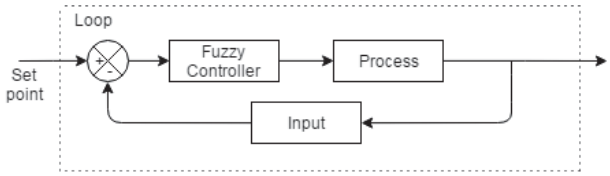


Figure 2 - 79. **Fuzzy logic controller diagram.**

As stated by Passino & Yurkovich (1992), electronic fuzzy controllers usually consist of 5 linguistic values, *negLarge*, *negSmall*, *zero*, *posSmall* and *posLarge*, which may be converted to *negative*, *centre-negative*, *centre*, *centre-positive* and *positive* as per Figure 2 – 77 and 2 - 78 (Cerecero-Natale et al., 2012). Rule sets define the relationship between linguistic values and the controller output.

Electronic controllers begin by fuzzifying all inputs values according to their corresponding membership function (Passino & Yurkovich, 1992), assigning linguistic value to the numeric input. Once all the inputs have been classified, the fuzzy rule set is executed to compute the applicable numeric outputs. From there, centroid de-fuzzification is applied using a max-min inference computation such as seen in Figure 2 - 80, providing a 'crisp' controller output. The maximum and minimum inference is based on OR (+) and AND (×) logic respectively, as prescribed in the fuzzy rule set.

$$Output(crisp) = \frac{mu_1 \times output_1 + mu_2 \times output_2 + mu_3 \times output_3 + \cdots + mu_n \times output_n}{mu_1 + mu_2 + mu_3 + \cdots + mu_n}$$

$Output(crisp)$	controller output
mu	result from membership function
$output$	rule set output

Figure 2 - 80. **Centroid de-fuzzification equation. (Passino & Yurkovich, 1992).**

2.7.4 Fuzzy PID Control

As explored by Cerecero-Natale et al. (2012), the past three decades have been prolific in the field of fuzzy logic controllers; not only has a great deal of work gone into these types of controller, but their applications have broadened and come to include PID fuzzy gain scheduling. Classical continuous gain scheduling is the notion of switching between PID control parameters, in an effort to more accurately control a system with a non-linear response. In essence, this establishes a non-linear controller through the splicing of a number of linear controllers (Naus, 2009).

Traditional gain scheduling implementation as described by Naus, (2009) consists of four main steps. The first step involves deriving a number of linear approximations from a non-linear system at constant operating points, characterised by measureable and changeable variables. From there, the second step calls for the construction of multiple controllers that satisfy performance and stability requirements for the previously acquired set of models at their respective operating points. Step three then implements continuous scheduling of the gain coefficients of the PID controller, corresponding to the previously found linear controllers, based on the current operating point of the system. The final step is simply performance assessment; the system is tested for local and global performance and robustness analytically. Criteria for good system behaviour may be established by the users' desired control outcome.

It should be noted that Naus, (2009) has identified a *'chattering behaviour'* which may occur when implementing this type of control incorrectly. This refers to observed jerky performance as the controller switches between control regions, which may be a result of a large jump in coefficient size. Naus, (2009) states that this can be avoided through linearization of the schedule; additionally, increasing the number of regions will provide smoother control region transitions.

The difficulty with this type of control arises when deciding on how to schedule the varying gains and to what degree. One approach is fuzzy logic.

A paper published by Cerecero-Natale et al. (2012), has evaluated the use of fuzzy logic for gain scheduling on a PI controller for position control of a mechatronic system. Implementation and robustness of this type of control was assessed through both simulation and experimentation. A Two Input Two Output (TITO), 5 linguistic term (fig. 2 - 81) fuzzy controller was used where error and error derivative serve as inputs and proportional gain (K_p) and integral gain (K_i) as outputs. Cerecero-Natale et al. (2012) defined the fuzzy rule set as per Figure 2 - 82, comprising of 25 AND logic operator rules.

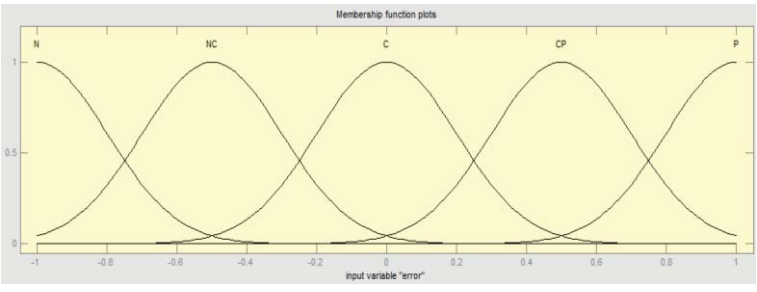


***Figure 2 - 81*. Membership function. (Cerecero-Natale et al., 2012).**

Fuzzy partitions		Singleton outputs	
error	error derivate	Gains	
e	$\dot{e}$	K_p	K_i
N	N	1	0.01
N	NC	0.96	0.04
N	C	0.92	0.08
N	CP	0.88	0.12
N	P	0.84	0.16
NC	N	0.80	0.2
NC	NC	0.76	0.24
NC	C	0.72	0.28
NC	CP	0.68	0.32
NC	P	0.62	0.38
C	N	0.58	0.42
C	NC	0.54	0.46
C	C	0.5	0.5
C	CP	0.46	0.54
C	P	0.42	0.58
CP	N	0.38	0.62
CP	NC	0.32	0.68
CP	C	0.28	0.72
CP	CP	0.24	0.76
CP	P	0.2	0.8
P	N	0.16	0.84
P	NC	0.12	0.88
P	C	0.08	0.92
P	CP	0.04	0.96
P	P	0.01	1

Figure 2 - 82. **Fuzzy rule set. (Cerecero-Natale et al., 2012).**

For convenience, error and error derivative were normalised by Cerecero-Natale et al. (2012) as follows:

$$e = \frac{e - e_{min}}{e_{max} - e_{min}}$$

$$\dot{e} = \frac{\dot{e} - \dot{e}_{min}}{\dot{e}_{max} - \dot{e}_{min}}$$

Figure 2 - 83. **Error and error derivative equations. (Cerecero-Natale et al., 2012).**

The membership function plot (fig. 2 - 81) and fuzzy rule set (fig. 2 - 82) result in non-linear control surfaces for outputs K_p (fig. 2 - 84) and output K_i (fig 2 - 84) after applying a centroid defuzzifier (fig 2 - 80).

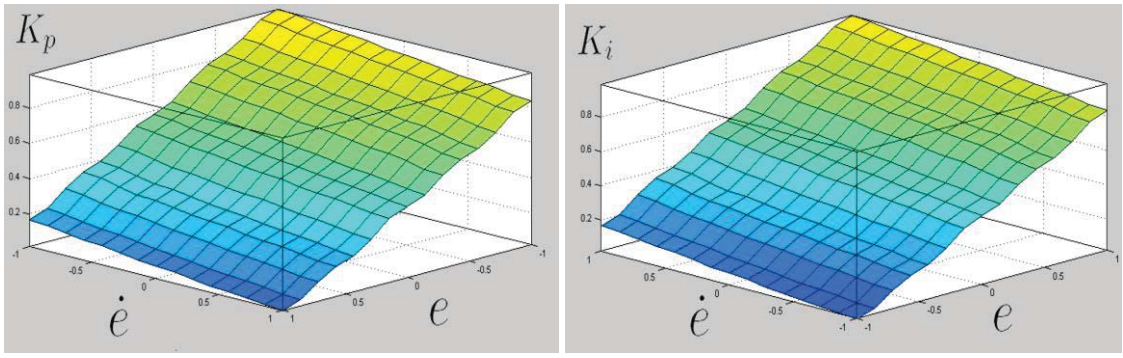


Figure 2 - 84. **Resulting control surfaces. (Cerecero-Natale et al., 2012).**

This yields the following output response (fig. 2 - 85) and gain performance (fig 2 - 86).

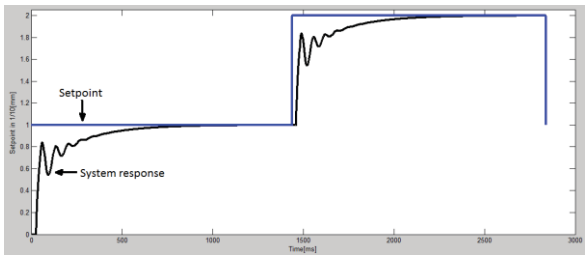


Figure 2 - 85. **Fuzzy PI control set-point response. (Cerecero-Natale et al., 2012).**

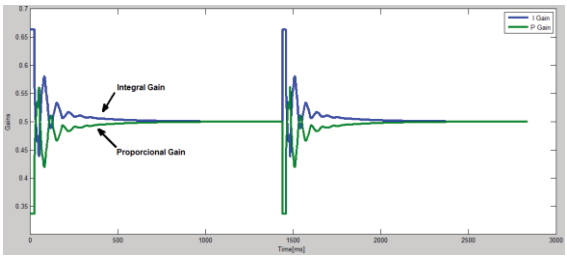


Figure 2 - 86. **Proportional and integral gain responses. (Cerecero-Natale et al., 2012).**

Once all parameters needed for the proposed fuzzy logic PI controller were established, Cerecero-Natale et al. (2012) used a first order mathematical system for comparing the simulated behaviour of a traditional PI controller and the proposed controller (fig. 2 - 87). The Ziegler-Nichols method was used to determine fixed controller gains K_p and K_i.

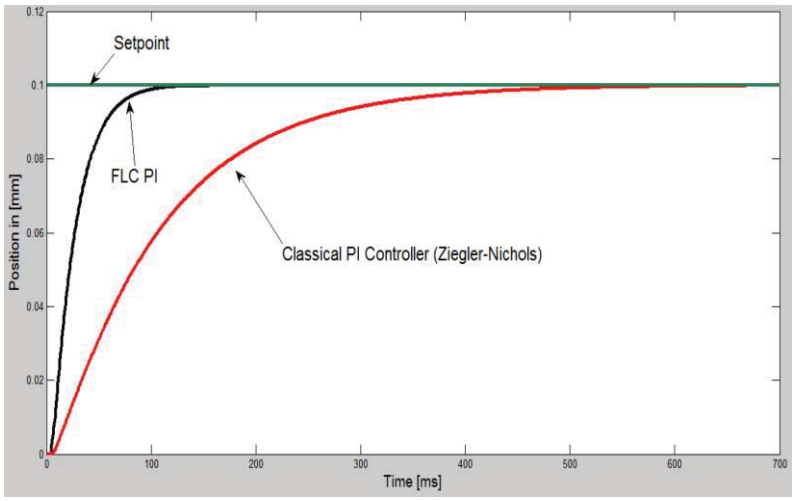


Figure 2 - 87. **FLC PI vs. Classical ZM PI set-point response. (Cerecero-Natale et al., 2012).**

Although this paper claims to have implemented this system on a light-weight, 8 bit PIC 18F4550 microcontroller (Cerecero-Natale et al., 2012), no significant amount of experimental results are provided. Cerecero-Natale et al. (2012) concluded that the main point of response control improvement associated with the fuzzy logic gain scheduling PI controller is its ability to reach the desired set point in less time (fig. 2 - 87), while reliably avoiding overshoot. It is recommended that a Gaussian membership such as Figure 2 - 81 is used, as this will provide smoother control in comparison to triangular memberships (fig. 2 - 78).

In a similar paper produced by Zhao et al. (1992), fuzzy logic gain scheduling of PID controllers are described in a broader sense. Zhao et al. (1992) recommend using a discrete-time equivalent expression for the PID algorithm, referred to as the standard

form in the PID section 2.7.1 (fig. 2 - 62). For simulation, this paper has decided to use a 7-term linguistic membership function, with error and error derivative as inputs, yielding two outputs, K_p and K_d. K_i is calculated with reference to the derivative time constant (fig. 2 - 88).

$$T_i = \alpha T_d$$

$$K_i = \frac{K_p}{(\alpha T_d)} = \frac{K_p^2}{(\alpha K_d)}$$

***Figure 2 - 88*. Integral term calculation. (Zhao et al., 1992).**

The fuzzy rule set comprises of 49 rules, similarly defined by Cerecero-Natale et al. (2012), producing similar membership function output plots. Inputs and outputs are also normalised in a similar fashion. Although full PID is used in this fuzzy gain scheduling scheme by Zhao et al. (1992), in this case simulation does not seem to suggest a significant improvement over traditional, Ziegler-Nichols tuned PID control (fig. 2 - 89) when dealing with a 3rd order process.

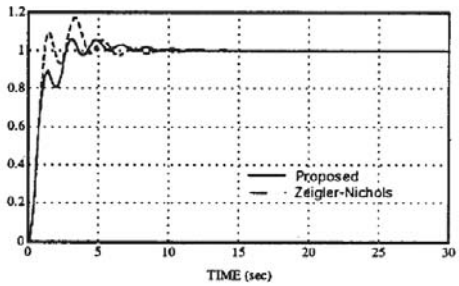


***Figure 2 - 89*. Controller 3rd order response. (Zhao et al., 1992).**

Zhao et al. (1992) briefly touch on the concept of a 'hybrid controller'. It suggests that in some situations it may be viable to use dynamic fuzzy controlled PID in tandem with traditional fixed gain PID. For large set point changes, fuzzy PID is employed to utilise its quick transient response, as error reduces and the system begins to settle, the controller is then switched to fixed gain PID. Zhao et al. (1992) states that sometimes set PID parameters may be excellent for good set point response, but lack in appropriate performance for load disturbance rejection.

2.7.5 Model Based Predictive Control

The notion of Model based Predictive Control (MPC) is one that bases its control strategy on a known model of the output of the controlled system. By pre-emptively calculating the expected behaviour of a system, a controller is able to take appropriate control action based on that prediction (Åström & Hägglund, 1995). According to Merabet & Gu (2010), MPC is considered a very effective control method for non-linear processes and disturbances.

As stated by Åström & Hägglund (1995), there are three main steps to implementing an MPC controller, once a model of the system has been specified. The first step simply acquires information about the system, including the measure of process inputs, outputs and disturbances. From there, the current output and predicted future state of the system is calculated through use of the output model. Step two then calculates the control changes necessary to minimise error between the desired and actual trajectories, subject to prescribed constraints. The final step simply executes this control action and restarts the process.

There are several major advantages associated with MPC over traditional control methodologies according to Åström & Hägglund (1995):

- Multivariable
- Permits hard constraints
- Is predictive
- Deals with trajectories rather than single points

In a paper published by Merabet & Gu (2010) in 2010, the fundamentals and implementation of a comprehensive MPC controller is evaluated for advanced non-linear robotic manipulator control.

The use of MPC to control a rigid robotic arm is somewhat problematic when compared to applying the same methodology on other systems. The increased control difficulty arises from un-modelled dynamics such as variable payloads, friction torques, torque disturbances, parameter variations and measurement noise (Merabet & Gu, 2010). Merabet & Gu (2010) suggest that this may be further emphasised by inaccuracies present in the robot model.

In order to implement MPC, first a mathematical model of the arm must be constructed with respect to angular joint positions. Merabet & Gu (2010) uses the Euler-Lagrange equation for a dynamic rigid robot manipulator of n-links as a base for further development:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u$$

Where:

$q(t) \in \Re^n$	angular joint position vector
$u(t) \in \Re^n$	driving torque vector (control input)
$D(q) \in \Re^{n\times n}, \quad D(q) = D(q)^T > 0$	link inertia matrix
$C(q,\dot{q})\dot{q}\Re^n$	Coriolis & centripetal torques vector
$G(q) \in \Re^n$	gravitational torques vector

Figure 2 - 90. **Mathematical model of manipulator arm. (Merabet & Gu, 2010).**

When implementing an arm, there are practical uncertainties that must be considered and compensated for to achieve robust and accurate control. These may include factors such as modelling errors, unknown loads and computational errors (Merabet & Gu, 2010).

Uncertainty added to matrices:

$$D(q) = D_0(q) + \Delta D$$
$$C(q, \dot{q}) = C_0(q, \dot{q}) + \Delta C$$
$$G(q) = G_0(q) + \Delta G$$

$\Delta(.)$	system error

***Figure 2 - 91*. Uncertainty definitions. (Merabet & Gu, 2010).**

Adding uncertainty and other unmodelled quantities to robot equation:

$$(D_0(q) + \Delta D)\ddot{q} + (C_0(q, \dot{q}) + \Delta C)\dot{q} + G_0(q) + \Delta G + F_r = u + b$$

$F_r(t) \in \Re^n$	friction vector
$b(t) \in \Re^n$	external disturbance vector

***Figure 2 - 92.* Manipulator arm equation with added uncertainty. (Merabet & Gu, 2010).**

Simplification:

$$D_0(q)\ddot{q} + C_0(q, \dot{q})\dot{q} + G_0(q) = u + \eta(\ddot{q}, \dot{q}, q, b)$$

$\eta\ (uncertainty)$	$-\{\Delta D\ddot{q} + \Delta C q + \Delta G \dot{+} F_r - B\}$

***Figure 2 - 93*. Uncertainty definition for simplification. (Merabet & Gu, 2010).**

Robust control law:

$$u(t) = -D_0(x_1)\{K_1(y - y_r) + K_2(\dot{y} - \dot{y_r}) - D_0(x_1)^{-1}\big(C(x_1, x_2)x_2 + G(x_1)\big) - \ddot{y}_r\} - \eta_{est}(t)$$

K_1	$\left(\frac{10}{3\tau_r^2}\right) * I_{n\times n}$
K_2	$\left(\frac{5}{2\tau_r}\right) * I_{n\times n}$

***Figure 2 - 94.* Robust control law defined by Merabet & Gu (2010).**

This produces the final dynamic nonlinear mathematical model of a robotic arm as prescribed by Merabet & Gu (2010) in terms of the state vector x:

$$\dot{x} = f(x) + g(x)u + g(x)\eta$$

$f(x)$	$\begin{bmatrix} x_2 \\ -D_0(x_1)^{-1}\big(C_0(x_1, x_2)x_2 + G_0(x_1)\big) \end{bmatrix}$
$g(x)$	$\begin{bmatrix} 0_{n\times n} \\ D_0(x_1)^{-1} \end{bmatrix}$

***Figure 2 - 95*. Final dynamic nonlinear model of robotic manipulator. (Merabet & Gu, 2010).**

Finally, the controlled angular position output vector may be defined as follows:

$$y = h(x) = Cx$$

Cx	$[I_{n\times n} \quad 0_{n\times n}]$

Figure 2 - 96. **Output vector definition. (Merabet & Gu, 2010).**

According to Merabet & Gu (2010), the associated control law must be derived from an undisturbed system where uncertainties are excluded. When the appropriate model has been achieved, robust control law follows and is mathematically implemented by compensating for any error through estimation.

Once a robust mathematical model of the manipulator arm to be controlled has been constructed in terms of measureable and controllable parameters, MPC may be employed. The supposed control system as defined by Merabet & Gu (2010), consists of an optimisation algorithm to solve for the future control trajectory through the use of the dynamic process model (fig. 2 - 95). It aims to do this through the minimisation of a cost function.

General form of cost function:

$$\Im = f\big(e_y(t+\tau), x, u\big)$$

$e_y(t+\tau)$	predicted error
$y(t+\tau)$	τ-step ahead prediction
$\tau > 0$	prediction time horizon

Figure 2 - 97. **General form cost function definition. (Merabet & Gu, 2010).**

Merabet & Gu (2010) then uses a mathematical tool based on Taylor series expansion carried out by Lie derivatives to develop the prediction model, by use of the process model. Yielding a final prediction model of:

$$y(t+\tau) = T(\tau)Y(t)$$

$T(\tau)$	$\begin{bmatrix} I_{n\times n} \\ \tau * I_{n\times n} \\ \left(\frac{\tau^2}{2}\right) * I_{n\times n} \end{bmatrix}$
$Y(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \ddot{y}(t) \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_1 \\ -D_0(x_1)^{-1}\big(C_0(x_1,x_2)x_2 + G_0(x_1)\big) \end{bmatrix} + \begin{bmatrix} 0_{n\times 1} \\ 0_{n\times 1} \\ D_0(x_1)^{-1}u(t) \end{bmatrix}$

Figure 2 - 98. **Final predictive model. (Merabet & Gu, 2010).**

Similarly, a reference trajectory analysis may be constructed as follows:

$$y_r(t+\tau) = T(\tau)Y_r(t)$$

$Y_r(t)$	$[y_r \quad \dot{y}_r \quad \ddot{y}_r]^T$

Figure 2 - 99. **Reference trajectory analysis equation definition. (Merabet & Gu, 2010).**

Allowing for the predicted error calculation required by the cost function:

$$e_y(t+\tau) = y(t+\tau) - y_r(t+\tau) = T(\tau)(Y(t) - Y_r(t))$$

***Figure 2 - 100*. Predicted error calculation. (Merabet & Gu, 2010).**

It is now possible to carry out optimal control law, through the minimisation of the cost function in relation to the control input (Merabet & Gu, 2010). At this point, the definition of the cost function is crucial. Merabet & Gu (2010) has provided two approaches:

$$\Im = \frac{1}{2}\int_0^{\tau_r} e_y(t+\tau)^T Q e_y(t+\tau) d\tau + \frac{1}{2}\int_0^{\tau_u} u(t+\tau)^T R u(t+\tau) d\tau$$

***Figure 2 - 101*. Expanded general form of cost function. (Merabet & Gu, 2010).**

$$\Im = \frac{1}{2}\int_0^{\tau_r} \big(y(t+\tau) - y_r(t+\tau)\big)^T \big(y(t+\tau) - y_r(t+\tau)\big) d\tau$$

***Figure 2 - 102*. Tracking error-based cost function. (Merabet & Gu, 2010).**

To evaluate the transient response and robustness of the suggested control, Merabet & Gu (2010) mathematically simulate the previously described system on a rigid robot manipulator of two links. The controller is of form fig. 2 - 103.

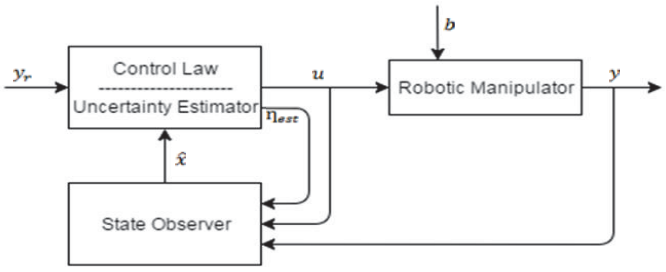


***Figure 2 - 103*. Model based predictive control diagram. (Merabet & Gu, 2010).**

Model simulation was implemented with a sample time of 100μ seconds (Merabet & Gu, 2010), with the manipulator initially stationary. An external disturbance was applied periodically. Merabet & Gu (2010) suggests robust control of the simulated manipulator has been achieved, providing the resultant angular positions and tracking errors for both controlled joints as evidence (fig. 2 - 104). As a side note, it is mentioned that the manipulator presents periodic steady state errors, brought on by external disturbances. It is also stated that this may be mitigated if this information was made available to the control law. Further control improvement may be implemented if load parameters are known (Merabet & Gu, 2010); in this case, it is regarded as part of the second link, with the system accommodating the load through a change in link properties such as mass, length, centre of mass and moment of inertia.

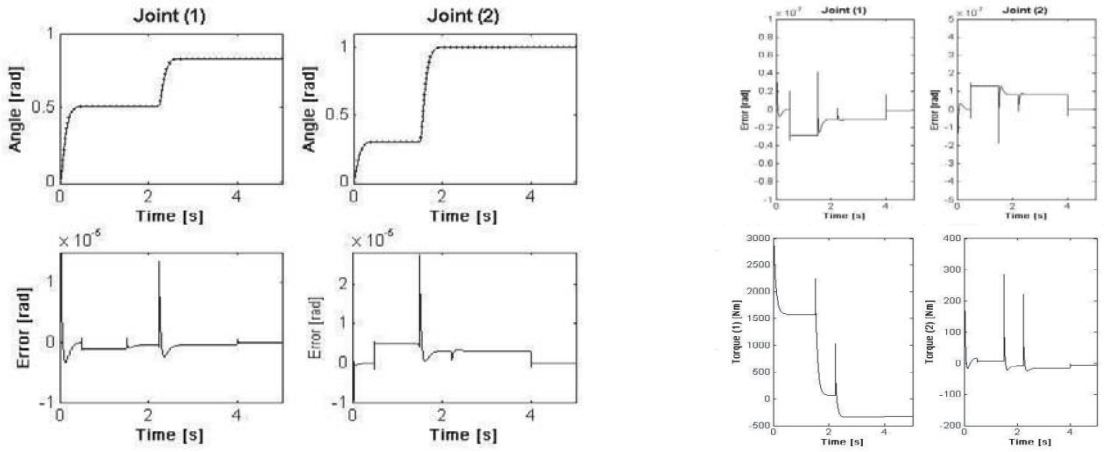


Figure 2 - 104. **Controller joint set-point response diagrams. (Merabet & Gu, 2010).**

Merabet & Gu (2010) have concluded that this control system through investigation has proved able to provide robust control with respect to modelling errors, effective disturbance rejection and no parameter uncertainty induced steady state error. However, Merabet & Gu (2010) state that it is well known that a theoretical model of a systems behaviour may not always translate well into reality. The uncertainty compensator is one approach of dealing with this issue. Merabet & Gu (2010) suggest that a Fuzzy logic controller may be another viable alternative.

2.7.6 Trajectory Planning

Robotic arm industrial applications often require a high degree of fine motion control to achieve the required operations at the desired efficiency. Depending on the task, control of the end-effector may range from point-to-point movements to fine collision-avoiding path control with varying velocity profiles. Three useful classifications relating to robotic motion have been defined by Lehtla (2008):

Path planning: *Complete planned motion from point A to B, made up of trajectories and other operations*
Trajectory planning: *Single continuous motion trajectory*
Path finding: *If final planned path is variable, subject to information, the optimum path must be found*

In determining the final path of the robotic manipulator, generally obstacle and path constrains must be considered (Lehtla, 2008). Obstacle constrains are introduced when the planned motion of the manipulator passes through space which is currently occupied by something other than the arm. On the other hand, path constraints arise from any fixed motion requirements, which are generally associated with operation.

Lehtla (2008) has noted that often there may be manipulator capability constraints imposed on determining a final path. Depending on the robotic system, it may sometimes be difficult to provide the end-effector with rotational action while the arm is in motion. Hardware degree of freedom may also affect the output. In this case, Lehtla (2008) has suggested it could be useful to consider other part rotating alternatives (fig. 2 - 105).

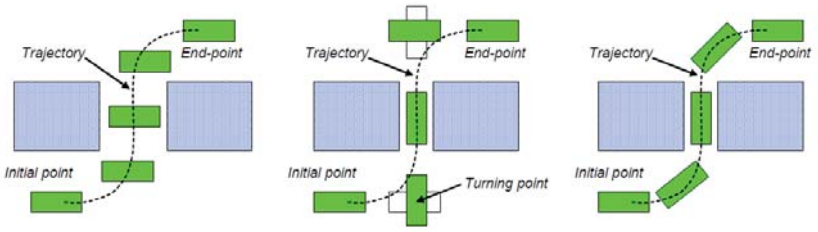


***Figure 2 - 105*. Various part collision avoidance methods. (Lehtla, 2008). Copyright 2008 by Tõnu Lehtla.**

A distinction between local and global collision avoidance (fig. 2 - 106) has been made by Lehtla (2008), adding further that collision detection is the single most important factor of path planning. Excluding active collision detection, a manipulator must work in a safe environment, specifically catered to isolate the system from external influences, alternatively, the manipulator may include off-line collision avoidance if information about the work space is known and does not change. Local collision avoidance is accommodated through the use of sensors. In comparison, global collision avoidance utilises prior knowledge of the environment to steer clear of obstacles.

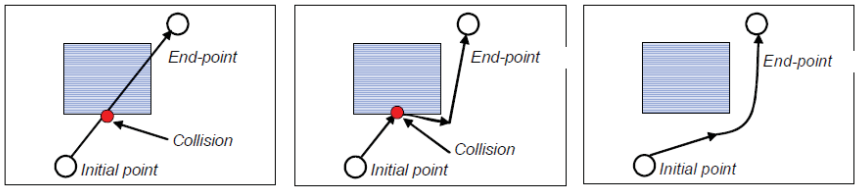


***Figure 2 - 106.* Local and global collision avoidance diagram. (Lehtla, 2008). Copyright 2008 by Tõnu Lehtla.**

The actual intended path for the manipulator to follow is subject to operation. It may be pre-determined according to the task, while only varying from the path to avoid collisions, or the manipulator may need to find an optimal path to navigate through obstacles (fig. 2 - 107) (Lehtla, 2008). A planned path within close proximity to a potential collision must consider its load dimensions, while maintaining a safety margin. Lehtla (2008) gives an example in which image analysis was used in determining obstacle locations, followed by the Laplacian of Gaussian to determine the exit path of the end-effector (Lehtla, 2008).

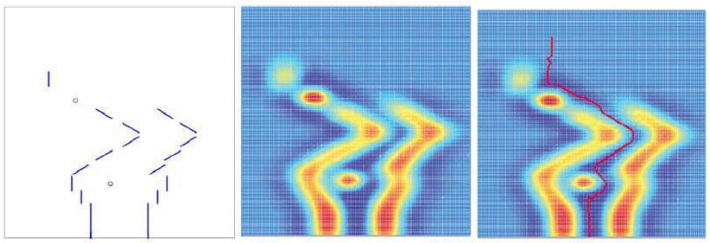


***Figure 2 - 107*. Object detection image analysis output. (Lehtla, 2008). Copyright 2008 by Tõnu Lehtla.**

Lehtla (2008) has suggested the construction of a dynamic path planning loop as shown in Figure 2 – 108, which may provide robust and reliable collision avoidance and path execution. The robustness of the proposed controller stems from its continually-updated consideration of environmental constraints and feedback from sensors.

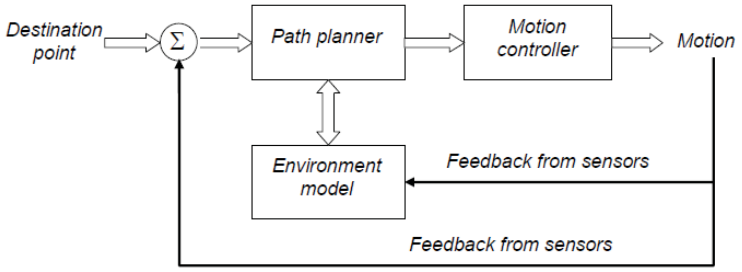


Figure 2 - 108. **Dynamic path planning loop diagram. (Lehtla, 2008). Copyright 2008 by Tõnu Lehtla.**

In determining a final path for a robotic manipulator, Lehtla (2008) states it is very important to consider position, velocity, acceleration and jerk for each robotic joint as the arm traverses through a planned path. Jerk is defined as the derivative of acceleration. Lehtla (2008) has mathematically defined each parameter by use of a polynomial function.

$$s(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$
$$v(t) = \frac{ds}{dt} = c_1 + 2c_2 t + 3c_3 t^3$$
$$a(t) = \frac{dv}{dt} = 2c_2 + 6c_3 t$$
$$j(t) = \frac{da}{dt} = 6c_3$$

c_0, c_1, c_2, c_3	Defining motion characteristics

Figure 2 - 109. **Motion parameter definitions. (Lehtla, 2008). Copyright 2008 by Tõnu Lehtla.**

2.7.7 Kinematics

Kinematics in the context of robotics is a branch of mathematical mechanics that describe a system of points relating to the chain of links and actuators that make up a robotic manipulator. Geometric relationships are employed in determining relative positions and velocities of robotic joints and actuators in space. As stated by Hydzik (2004) there are two types of actuations assumed:

Revolute: *Provides rotational actuation in a single plane. Generally denoted as* θ_i
Prismatic: *Provides linear actuation in a single plane. Generally denoted as* d_i

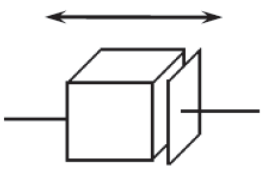
Revolute (θ_i)	Prismatic (d_i)
	

Table 2 - 18. **Revolute and Pismatic graphical representation. (Hydzik, 2004).**

Inverse kinematic equations are concerned with end-effector location, realising chain mathematical relationships with respect to varying coordinate frames, that lead up to this point. This allows for specification of the end-effector and computes the associated joint angles needed to achieve this. In contrast, forward kinematics allows for the entry of joint angles and computes the configuration of the chain.

Forward Kinematics

In constructing forward kinematic relationships, a reference coordinate frame must first be defined. Hydzik (2004) has denoted this as $O_0x_0y_0z_0$, referred to as the inertial frame. Furthermore, Hydzik (2004) suggests attaching this frame to the base of the manipulator as per Figure 2 - 110. As can be seen in Figure 2 - 110, a unique reference frame for each joint is given by the notationO_n.

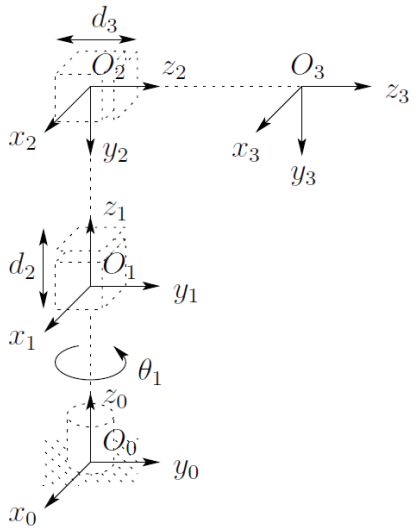


Figure 2 - 110. **Forward kinematic reference coordinate frame definitions for a cylindrical robot. (Hydzik, 2004).**

The convention for assigning coordinate frames begins with choosing the orientation of the z_0 axis, along the actuation line of the 1st link. According to Hydzik (2004), it follows that all $z_{(i-1)}$ axes are defined along the actuation lines of i^{th} links. Once the z axis has been defined, next the x axis is implemented so that two Denavit-Hartenberg conventions hold:

DH1*: The axis x_{i+1} is perpendicular to z_i*
DH2*: The axis x_{i+1} intersects the axis z_i*

If both z_i, x_i vectors have been chosen, y_i is assigned by cross-product multiplication:

$$\overrightarrow{y_i} = \overrightarrow{z_i} \times \overrightarrow{x_i}$$

Basic set of homogeneous transformations (Hydzik, 2004):

$$Trans_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Rot_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_\alpha & -S_\alpha & 0 \\ 0 & S_\alpha & C_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Rot_{y,\beta} = \begin{bmatrix} C_\beta & 0 & S_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -S_\beta & 0 & C_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trans_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Rot_{z,\gamma} = \begin{bmatrix} C_\gamma & -S_\gamma & 0 & 0 \\ S_\gamma & C_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous transformation associated with $Rot, Trans$.

$$H = \begin{bmatrix} Rot & Trans \\ 0_{1\times3} & 1 \end{bmatrix}$$

Denavit-Hartenberg Convention (single frame):

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

θ_i	joint angle (revolute action)
α_i	link twist
d_i	link offset (prismatic action)
a_i	link length
$S\ or\ s$	sine function
$C\ or\ c$	cosine function

***Figure 2 - 111*. Denavit-Hartenberg Convention and associated definitions. (Hydzik, 2004).**

Given by Hydzik (2004), each homogeneous transformation A_i is of form:

$$A_i = \begin{bmatrix} R_i^{i-1} & O_i^{i-1} \\ 0 & 1 \end{bmatrix}$$

Matrix R_j^i expresses the orientation of $O_j x_j y_j z_j$ with respect to $O_i x_i y_i z_i$:

$$R_j^i = R_{i+1}^i \cdots R_j^{j-1}$$

The coordinate vectors O_j^i are given by:

$$O_j^i = O_{j-1}^i + R_{j-1}^i O_j^{j-1},$$

Transformation matrix:

$$T_j^i = A_i A_{i+1} \cdots A_{j-1} A_j = \begin{bmatrix} R_j^i & O_j^i \\ 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} Rot.Matrix & Trans.Vector \\ Perspect.Trans & Scaling\ Factor \end{bmatrix}$$

At the completion of an inertial frame and relative coordinate frames attached to each respective joint (fig. 2 - 112), it is now possible to complete a homogeneous matrix for each coordinate frame. In keeping with the cylindrical manipulator example provided (fig. 2 - 10), forward kinematics will be computed for this type of arm to express the mathematical process.

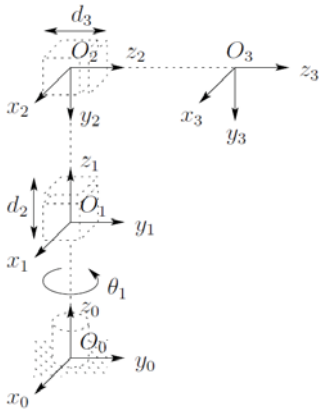


***Figure 2 - 112*. Relative coordinate frame definitions. (Hydzik, 2004).**

Link	a_i	α_i	d_i	θ_i
1	0	0	d_1	θ_1^*
2	0	-90	d_2^*	0
3	0	0	d_3^*	0

* variable

Link 1 - $A_i = Rot_{z,\theta_i} Trans_{z,d_i}$
Link 2 - $A_i = Trans_{z,d_i} Rot_{x,\alpha_i}$
Link 3 - $A_i = Trans_{z,d_i}$

Producing homogeneous transformations for each frame (Hydzik, 2004):

$$A_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A transformation matrix then follows to allow for the end-effector coordinates and orientation to be computed with respect to the inertial frame:

$$T_3^0 = A_1A_2A_3 = \begin{bmatrix} c_1 & 0 & -s_1 & -s_1d_3 \\ s_1 & 0 & c_1 & c_1d_3 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

x, y, z End-effector coordinates relative to the inertial frame:

$$x = -s_1d_3$$
$$y = c_1d_3$$
$$z = d_1 + d_2$$

End-effector orientation relative to inertial frame:

$$\theta_x = \begin{bmatrix} c_1 \\ s_1 \\ 0 \end{bmatrix}$$

$$\theta_y = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

$$\theta_z = \begin{bmatrix} -s_1 \\ c_1 \\ 0 \end{bmatrix}$$

Inverse Kinematics

So far, forward kinematics have allowed for the computation of end-effector position and orientation, given joint angles and extension. In contrast, inverse kinematics will compute the joint angles and translations required by the actuators to satisfy specified end-effector orientation and location parameters.

Firstly, it must be noted that it is possible to calculate inverse kinematic properties from a transformation matrix acquired through the forward kinematic method, however, this generally produces a system containing 12 simultaneous equations. As stated by Hydzik (2004), realistically that is much too difficult to solve directly. Because of this, it is necessary to assume a heuristic, geometric approach in defining the inverse positions of a manipulator.

This approach generally projects the arm onto various planes, producing a number of single plane projections which may be used in tandem to provide unique joint angles. An example given by Hydzik (2004), evaluates this process for the articulated configuration.

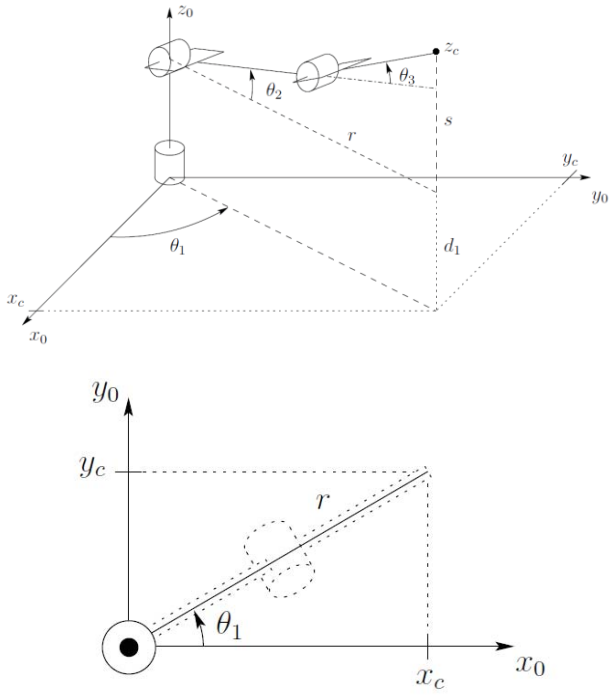


***Figure 2 - 113*. Inverse kinematic projection. (Hydzik, 2004).**

Hydzik (2004) begins by projecting the end-effector location onto the x_c - y_c plane (fig. 2 - 113). From the projection, it can be shown that:

$$\theta_1 = Atan(x_c, y_c)$$

Where $Atan(x_c, y_c)$ is the arctangent function in which all $(x, y) \neq (0,0)$ and provides a unique angle such that:

$$cos\theta = \frac{x}{(x^2+y^2)^{\frac{1}{2}}} \qquad sin\theta = \frac{y}{(x^2+y^2)^{\frac{1}{2}}}$$

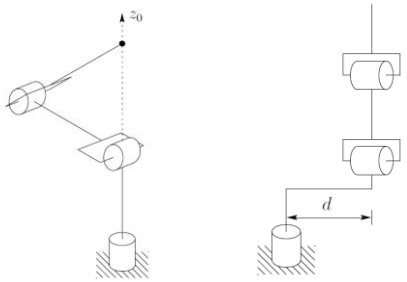


***Figure 2 - 114*. Singularity position. (Hydzik, 2004).**

It should be noted that in a configuration as seen on the left of Figure 2 - 114, the solution for θ_1 is always valid unless $x_c = y_c = 0$. In this case, the kinematic equation results in an infinite number of solutions for θ_1. This is what is known as a singularity position (Hydzik, 2004). Generally, this may be solved by the addition of an offset (right of fig. 2 - 114), so that the end-effector position may never intersect z_0.

This however, results in two solutions for θ_1, where the end-effector position achieves the same location with two different arm configurations, as may be seen in Figure 2 - 117.

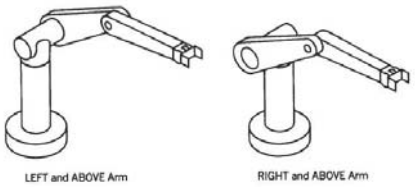


Figure 2 - 115. **Left above arm, right above arm configurations. (Hydzik, 2004).**

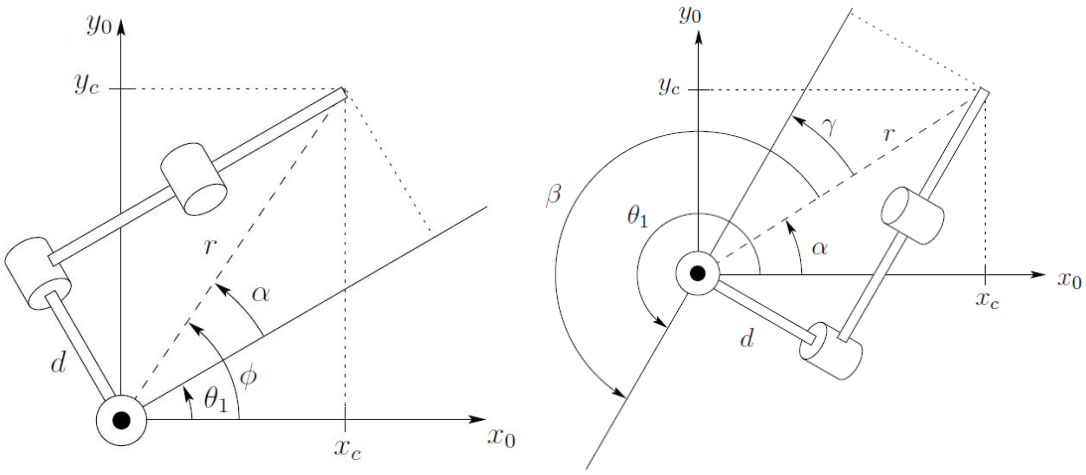


Figure 2 - 116. **Left and right arm configuration mathematical definitions. (Hydzik, 2004).**

In computing the left arm configuration according to parameters defined in Figure 2 - 116, it may be shown that (Hydzik, 2004):

$$\theta_1 = \emptyset - \alpha$$

Where:

$$\emptyset = Atan(x_c, y_c)$$

$$\alpha = Atan\left(\sqrt{r^2 - d^2}, d\right) = Atan\left(\sqrt{x_c^2 + y_c^2 - d^2}, d\right)$$

Alternatively, the right arm configuration (fig. 2 - 116) may be used:

$$\theta_1 = \alpha + \beta$$

Where:

$$\alpha = Atan(x_c, y_c)$$

$$\beta = \gamma + \pi$$

$$\gamma = Atan\left(\sqrt{r^2 - d^2}, d\right)$$

Therefore:

$$\beta = Atan\left(-\sqrt{r^2 - d^2}, -d\right)$$

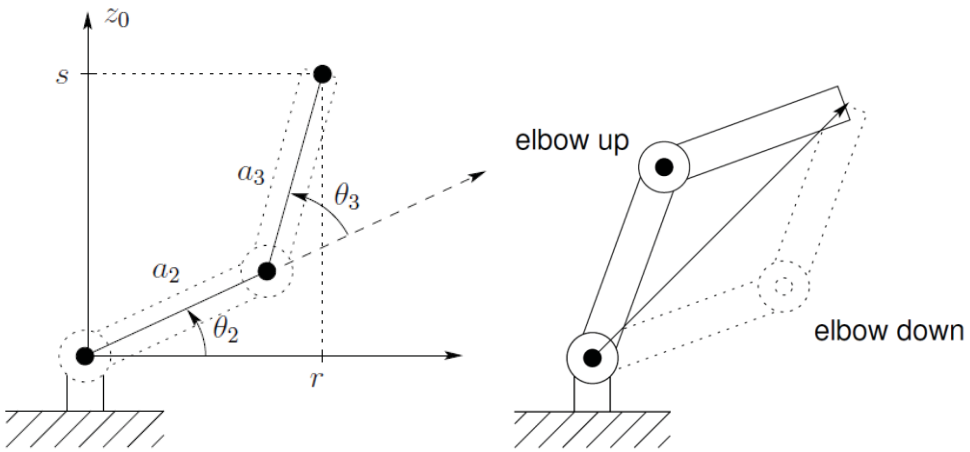


Figure 2 - 117. **Elbow up, elbow down configurations and mathematical definitions. (Hydzik, 2004).**

In determining θ_2 and θ_3, Figure 2 - 117 may be used. Through the application of the cosine law (fig. 2 - 119), θ_3 can be show to be of the form:

$$cos\theta_3 = \frac{x_c^2 + y_c^2 - d^2 + z_c^2 - a_2^2 - a_3^3}{2a_2a_3} := D$$

Therefore:

$$\theta_3 = Atan\left(D, \pm\sqrt{a - D^2}\right)$$

Similarly, Hydzik (2004) has shown that θ_2 is:

$$\theta_2 = Atan\left(\sqrt{x_c^2 + y_c^2 - d^2}, z_c\right) - Atan(a_2 + a_3c_3 + a_3s_3)$$

As a side note, analogous to the solution of θ_1, θ_2 and θ_3 will be capable of achieving the same end-effector coordinates with two different arm configurations, shown in Figure 2 - 118.

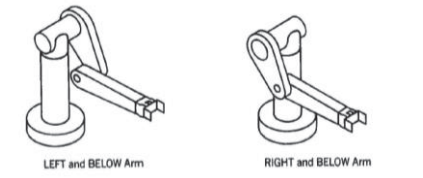
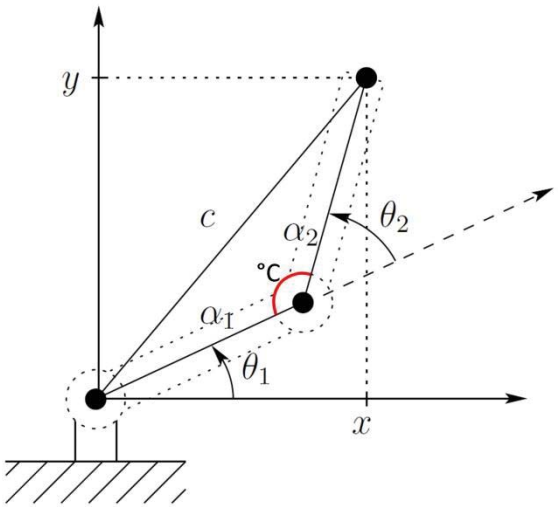


Figure 2 - 118. **Left below arm, right below arm configurations. (Hydzik, 2004).**



$$Cos(°C) = \frac{\alpha_2^2 + \alpha_1^2 - c^2}{2\alpha_2\alpha_1}$$

***Figure 2 - 119*. Cosine law and corresponding coordinate definitions. (Hydzik, 2004).**

Chapter 3

Design Considerations and the Proposed System

3.1 Shell-Core Concept

Whilst reviewing literature pertaining to current, non-industrial robotics (section 2.3 - 2.5), it appeared that there was a major lack of variability of educational robotic manipulators on the market, aimed at the secondary education level, seemingly indicating a potential market gap. Fixed arm configurations like DexTAR, Mover6 and Scorbot can be very useful at teaching robotic theory; however, these systems are limited to only provide educational content directly relating to that specific type of robotic arm and therefore cannot educate a user about the numerous types of arms popular for industrial use.

Throughout the mainstream industrial robotics industry, the type of robotic arm in use generally falls into one of the following categories: Cartesian, SCARA, cylindrical, spherical or articulated (tab. 3 - 1). The Shell-Core structured manipulator concept presented in this research aims to provide a platform in which all mainstream arms may be constructed. This is achieved through the use of various linkages (shells) and two types of mechanical actuators, rotational and linear (cores) as shown in Table 3 - 2. The design and development of these unique shells and cores provide a good educational study basis relating to mainstream industrial robotic configurations. Unique relating features include their compact design, interchangeable nature, simple configuration and reconfiguration, plug and play accessibility and their ability to construct commonly used mainstream industrial robotic manipulators. Tables 3 - 3 and 3 - 4 present some of the industrial robotic arms constructed using the Shell-Core concept.

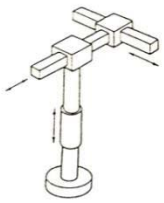
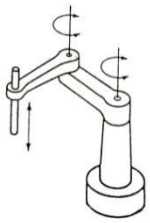
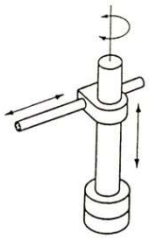
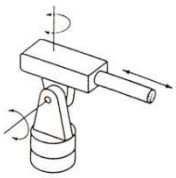
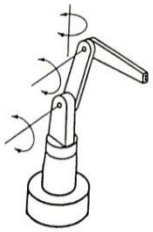
Cartesian	SCARA	Cylindrical	Spherical (Polar)	Articulated
				
3 linear	1 linear 2 rotational	2 linear 1 rotational	1 linear 2 rotational	3+ rotational

***Table 3 - 1*. Basic manipulators and the actuators needed to achieve configurations. (Meier, 2016).**

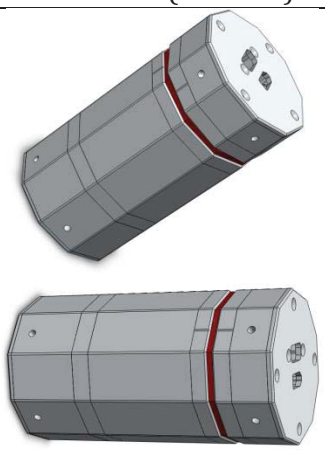
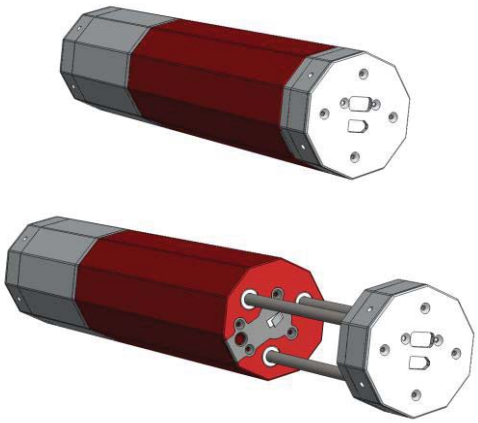
Rotational (revolute)	Linear (prismatic)
	

***Table 3 - 2*. Basic types of actuator.**

Cartesian	SCARA	Cylindrical	Spherical (Polar)	Articulated
				

***Table 3 - 3*. Configurations realised with the proposed system (no end-effector).**

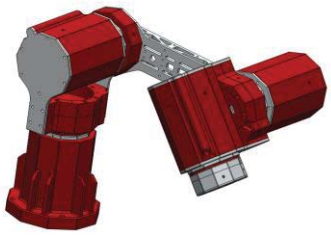
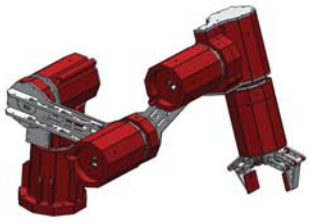
PUMA (no gripper, no final axis)	SCARA (no linear, no gripper)	Articulated (with gripper)
		

***Table 3 - 4*. Other configurations.**

3.1.1 Actuators Design

As defined in the kinematics section (tab. 2 - 18), each core unit will enable either rotational actuation or linear actuation in a single plane (fig. 3 - 1). One end of each core will be stationary with respect to the connected structural element, with both ends held in place by locating mounts. Due to the nature of inter-connecting, modular design, each core unit has its own on-board control electronics as well as pass along both data and power to the next core unit in the queue.

Each core unit has a unique ID and will be able to detect its position in the queue constructing the robotic arm. The queue position information, added with structural link type information, allows for complete knowledge of the assembled manipulator.

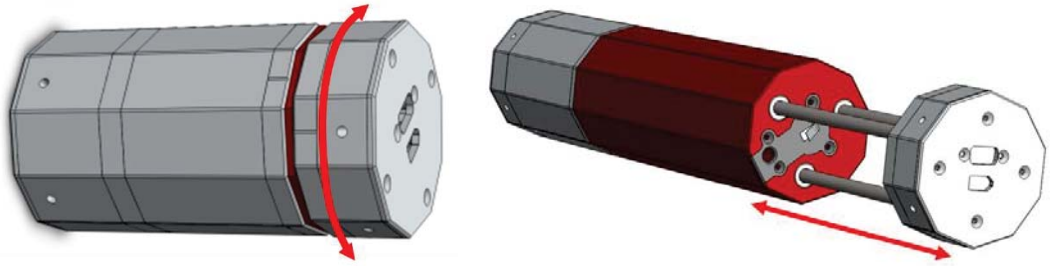


Figure 3 - 1. **Actuator planes of motion.**

Mechanical stops are implemented for the rotational actuator in a cam design, which limits its rotational range to less than 360°. This allows for a simplified homing methodology as well as preventing infinite rotation, reducing the overall complexity needed for power and data transmission.

The amount of actuation required from each core unit making up the manipulator arm may not be equal. Because of this, more power is needed of the rotational actuator at the base of the robot, resulting in components with varying power outputs. Hence two types of actuators are included, self-locking and non-self-locking.

Passive building blocks are designed to the same size and dimensions as a rotational or linear actuation unit, however, this unit will not allow for any movement whatsoever. Power and data lines are supported. The purpose of this component is to increase the variety of useable robotic building blocks, which in turn will accommodate for different assembly of robotic arms, such as the cylindrical configuration (fig. 3 - 2).

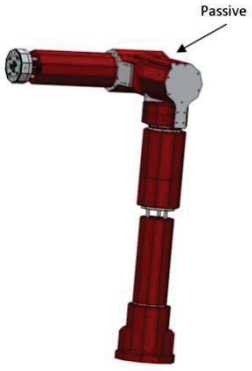


Figure 3 - 2. **Cylindrical configuration with passive block.**

3.1.2 Shell Structural Units

Shell structural units as shown in Figure 3-5 are specifically designed to make the assembly of a robotic arm possible. The goals of these shells are to inter-connect actuators and provide appropriate offsets for mechanical motion in various planes in different robot arm configurations.

Each shell structural unit has a unique ID attached to its specific dimensions, which are passed along and recognised by a pre-programmed microcontroller, allowing it to act accordingly. This information is also sent to the master control unit which may in turn adjust system kinematics to suit the specific arrangement. Upon installation the *'Structure type'* is configured to correspond with the correct robot arm structure.

Power and data will be conveyed through the structural piece itself.

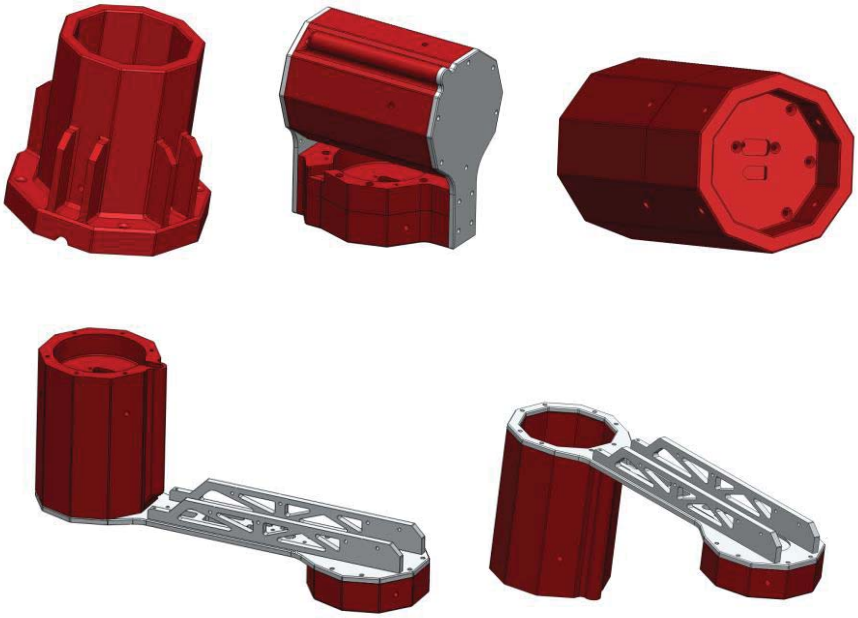


Figure 3 - 3. **Shell Structural Units.**

3.1.3 End-Effectors

As the nature of the system is modular, the end-of-arm tooling platform is flexible and permits any appropriate design requiring power and data exclusively. Varying end-effectors will also be included depending on application, eg: electro-magnetic, 2-fingered gripper, 3-fingered gripper, drawing utensil attachment.

The 2-fingered electronic gripper will be the first addition to the eventual range of grippers accompanying this system. The unit will be a separate component, supporting the plug-and-play feature as the other modules in the system. The 2-fingered gripper may also introduce another rotational axis, providing actuation directly to the handled part; in addition, this will simplify the configuration of some robotic manipulators, such as the PUMA (fig. 3 - 4).

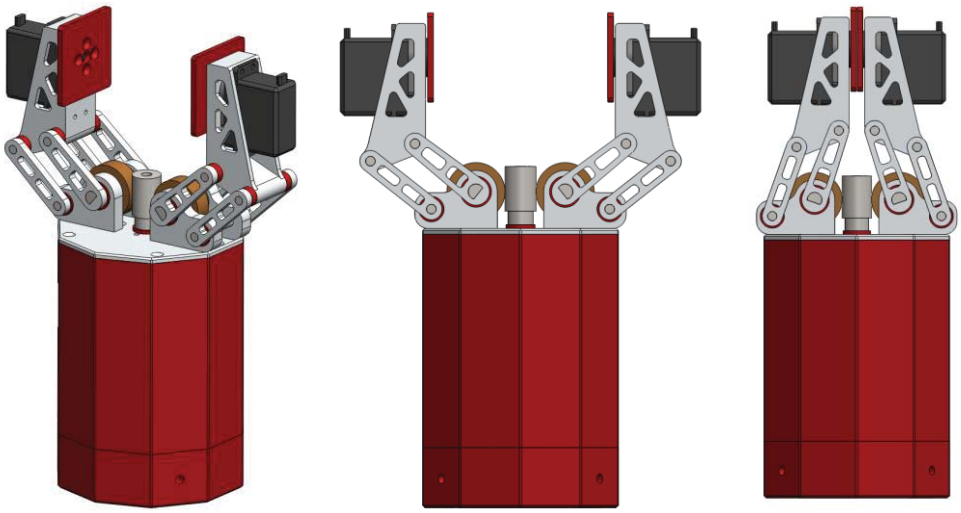


Figure 3 - 4. **2-Fingered gripper pictures.**

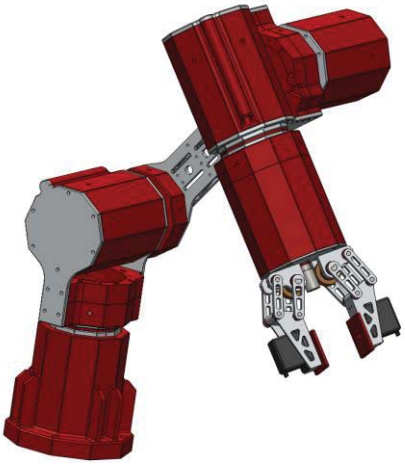


Figure 3 - 5. **PUMA realised with gripper addition.**

Chapter 4

Joint Actuator Design

4.1 Rotational Actuator System Design

A rotational actuator is the rotational core unit that realises rotational movement. Based on literature study, a DC motor is favoured for driving the mobile part of the rotational actuator. Power from the motor is transmitted through an internal gear and pinion arrangement as shown in Figure 4 - 1. The steel internal gear is press-fitted into the driving mount, held in place by the drive shaft, which is in turn held in place by two bearings. A plastic pad is added (fig. 4 - 1) between moving parts to reduce friction; additionally, this thickness may be varied to increase or decrease pressure and degree of friction between each part as desired.

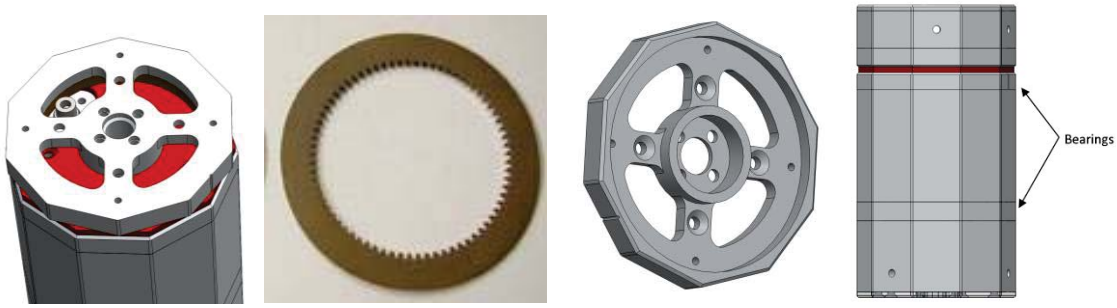


Figure 4 - 1. **Various rotational actuator drive pictures.**

Control componentry is fitted to a plastic stand, mounted to the bottom bearing holder, as seen in Figure 4 - 2. Electric wiring is then run past the motor into the drive shaft inlet and through to the other side of the actuator. Wiring slack is provided in the space between the plastic electronics stand and the drive shaft as to avoid any tension during rotational motion.

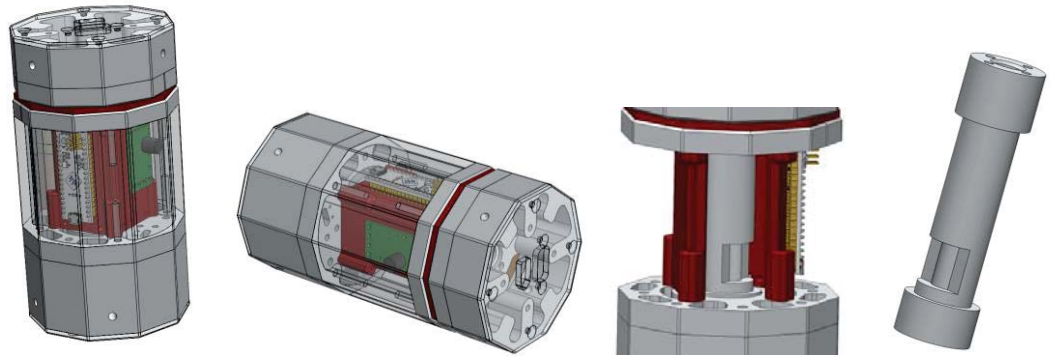


Figure 4 - 2. **Rotational actuator electronics and drive shaft pictures.**

The implemented rotational actuator is capable of approximately 318° degrees of rotational movement. It is desirable to limit this to less than 360° to avoid excessive twist in through wiring; furthermore, this simplifies the design by eliminating the need for complex full-rotation designs that may limit current. This range is achieved by use of a cam (fig. 4 - 3), fixed to the drive shaft (fig. 4 - 2). The minimum limit is imposed on the system through the homing limit switch (tab. 4 - 1); in contrast, maximum rotation is halted by a mechanical stop. This range is known by the microcontroller that will attempt to avoid these limits. In the case that an external torque is applied to the actuator in excess of motor output, the drive shaft and mechanical limits will bear this excess.

Currently, two different motors are used to provide rotation to the rotational core unit, depending on application. A self-locking, 100 rpm high-power motor is used and a non-self-locking, 78 rpm medium-power motor is used. With gearing, this results in effective rotational speeds of ~13 and ~9.8 rpm respectively, taking the actuator ~4.2 and ~5.4 second to traverse its full 318° range.

It should be noted that in implementation, the homing switch will be toggled before minimum actuation is achieved, as can be seen in Table 4 - 1.

Figure 4 - 3. **Rotational actuator homing cam.**

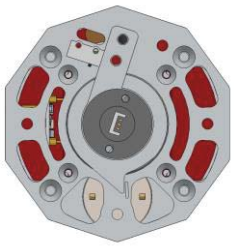
Home position	Minimum	Maximum	Mid-point
			

Table 4 - 1. **Rotational actuator rotational positions.**

In an attempt to aid with the configuration process, a small mark was added to both an actuated and non-actuated part (fig. 4 - 4) of the assembly so that it may be easy to roughly recognise where the actuator is in its range of motion. The indentation was added at the mid-point of rotation, where the component is capable of rotating clockwise and anti-clockwise to the same degree.

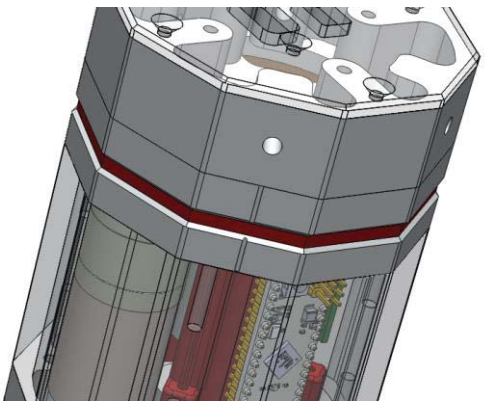


Figure 4 - 4. **Mid-point position (Corresponds to Table 4 – 1).**

For rotational position control, an encoder is implemented. The encoder is fitted to a small bracket, just above the mechanical stop (fig. 4 - 5), while the encoder magnet is fixed directly to the actuated shaft so that a measurement may be obtained that is true to the output motion. As with the other side of the actuator, a plastic pad has been introduced.

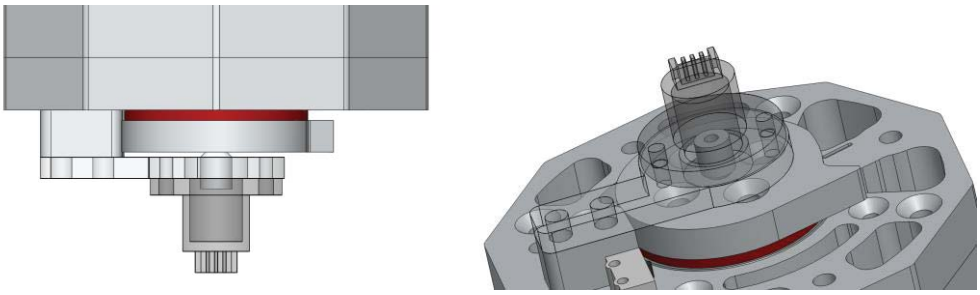


Figure 4 - 5. **Encoder position.**

End covers are of such design that mounting screws sit flush (fig. 4 - 6).

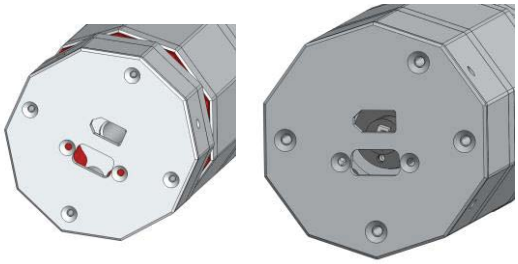


Figure 4 - 6. **Rotational actuator end covers.**

An opening was added to the plastic friction pad (fig. 4 - 7) directly opposite the motor opening to minimise the number of parts that need to be removed to program the microcontroller. Once finalised, there should be no need to reprogram the controller.

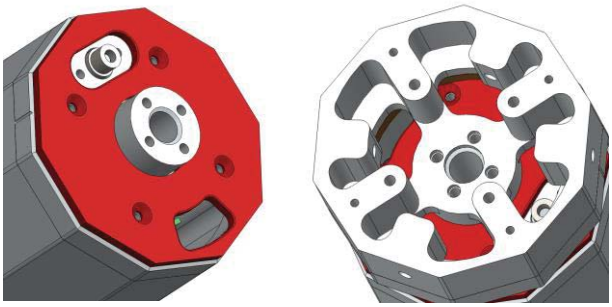


Figure 4 - 7. **Programming openings.**

Weight	580-670 g
Height	156 mm
Width	85 mm
Power Consumption (idle)	<0.1 W
Power Consumption (average)	12 W
Power Consumption (max)	60 W

Table 4 - 2. **Rotational actuator specifications.**

4.2 FEA Analysis

As a means of testing design robustness, several simulations were run on the assembly. The Solidworks 2014 Simulation tool was used. To observe how the actuator handles weight acting upon it, the plastic slider was removed and a force of 50N was applied vertically (fig. 4 - 8). 50N is a maximum operational weight estimation, with the articulated configuration (tab. 3 - 4) containing the most parts, weighing approximately 4.8 kg. The actuator was fixed at the outer cover to find the displacement of the drive shaft arrangement exclusively, as this will be one of the main sources of displacement imposed on the system by the rotational component, affecting repeatability.

Standard Mesh:
5mm global size
0.25mm tolerance

Elastic Modulus	6.9e+010 N/m^2
Poissons Ratio	0.33
Shear Modulus	2.6e+010 N/m^2
Density	2,700 kg/m^3
Tensile Strength	124,084,000 N/m^2
Yield Strength	55,148,500 N/m^2

Table 4 - 3. **Material: 6061 Aluminium alloy properties.**

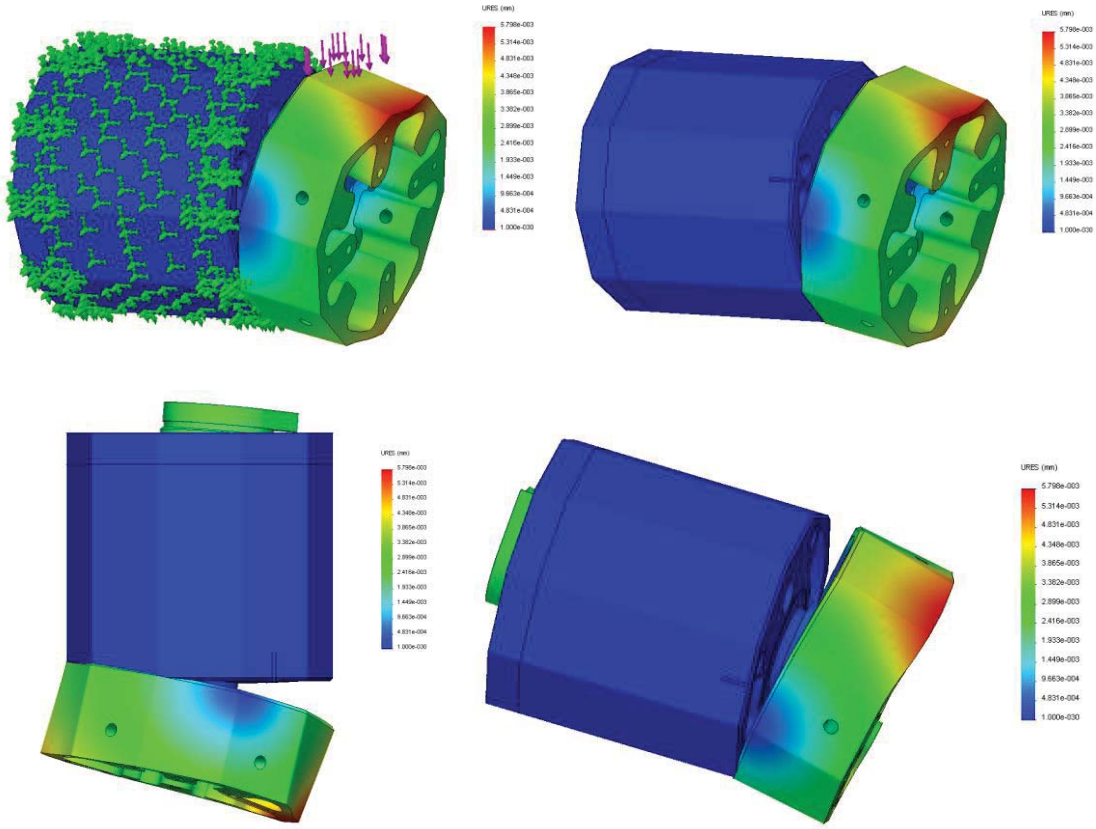


Figure 4 - 8. **Resultant exaggerated displacement diagrams (0.0058 mm max).**

To further investigate the design robustness, a torque of 50 Nm was applied in simulation to the drive shaft (fig. 4 - 9) to observe the resultant stress concentration when the actuator has reached its maximum range of motion. The maximum torque output of the self-locking motor used is 16.5 Nm after gearing. A value of 50 Nm was used to accommodate for a heavier payload at the end-effector and in cases of impulse.

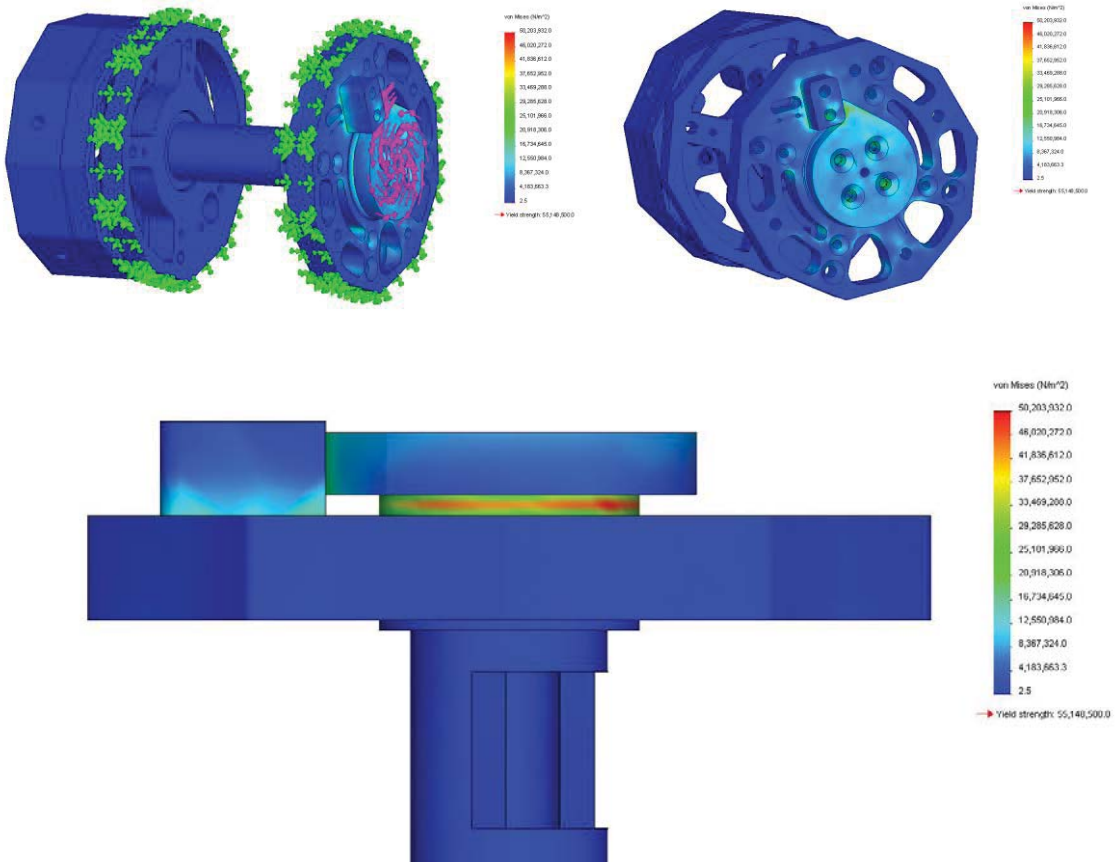


Figure 4 - 9. **Resultant exaggerated stress diagrams (50.2 MNm^2 max).**

The resulting analyses show good material performance in both displacement and stress scenarios. A displacement resulting from an added force is unavoidable in any material, however, this value has been minimised to a manageable 5.8 microns. The result determined in Figure 4 - 8 is negligible, especially when considering the uncertainty related to the control methodology. As expected, a sheer stress concentration has occurred in the area shown in Figure 4 - 9 as a consequence of the mechanical stop. This concentration will not affect the arrangement provided that the maximum torque stays below 50Nm, at which point the material may begin to yield. It should be noted that 50Nm is a large overestimate of applied torque.

4.3 Control Componentry (Rotational Actuator)

4.3.1 Non-Self-Locking Motor

Size and power output was a large aspect of motor selection. Polulu has a range of geared motors of varying properties. The 99:1, medium-power Gearmotor was selected to provide a non-self-locking, low power option for the rotational actuator.

It was found that, when unpowered, this motor required just enough torque to hinder manipulator movement under its own weight, but provided free movement when exerting an external torque. In contrast, the manipulator becomes self-locking when powered. Final rotational speed was required to be of a magnitude that allows the actuator to traverse its full range in no more than 7 seconds. Output torque needed to exceed 6.5 Nm in order to actuate a link with two rotational actuators, a 90-deg bend and gripper attached.

The non-self-locking nature exhibited by this motor is useful to the final system, allowing the user to manually move the manipulator to a desired position, so that an arm location may be captured in software for later use.

Gearing:

Motor pinion no. teeth	15
Internal gear no. teeth	120

$$12 \times \left(\frac{120}{15}\right) = 96\ kgcm = 9.4\ Nm$$

Rotational Velocity:

$$76 \times \left(\frac{15}{120}\right) = 9.8\ rpm = 6.2\ sec/revolution$$

Figure 4 - 10. **Polulu 99:1 metal gearmotor MP 12V.**

Weight	91g
Gear ratio	98.78:1
Free-run speed	76 rpm
Free-run current 12V	200 mA
Stall current 12V	2100 mA
Stall torque	12 kg-cm

Table 4 - 4. **Polulu 99:1 MP gearmotor specifications.**

4.3.2 Self-Locking Motor

Although non-self-locking is useful in adding to the usability of the final system, none of the available medium-power motors provide enough torque for larger configurations, without severely impacting rotational velocity. The motor, placed at the base of the articulated configuration, would need a calculated maximum torque of 15 Nm to move the manipulator. As a consequence, self-locking, high-power motors were investigated as a means of arm actuation.

The 25 mm diameter, high-power geared-motor package does not allow for any rotation when unpowered. This proved to be very useful in terms of power usage, as the control signal may be cut from these motors once they are at their desired position. In contrast, the medium-power motors require a constant signal to stay self-locking.

Gearing:

$$21 \times \left(\frac{120}{15}\right) = 168\ kgcm = 16.5\ Nm$$

Rotational Velocity:

$$100 \times \left(\frac{15}{120}\right) = 13\ rpm = 4.8\ sec/revolution$$

***Figure 4 - 11*. Polulu 99:1 metal gearmotor HP 12V.**

Weight	91g
Gear ratio	98.78:1
Free-run speed	100 rpm
Free-run current 12V	300 mA
Stall current 12V	5600 mA
Stall torque	21 kg-cm

***Table 4 - 5*. Polulu 99:1 HP gearmotor specifications.**

4.3.3 Motor Driver

The Polulu G2 high-power motor driver was found to be ideal for the proposed system in terms of size, power output and signal control. This small, discrete MOSFET H-bridge package is capable of delivering a continuous 13 A and includes basic current-sensing and limiting functionality. The current sense output voltage is stated to be 40 mV/A, with an offset of 50 mV.

Unfortunately the Arduino Micro analogue pins have a read range that cuts 5V into 1024 segments. This results in a maximum voltage read sensitivity of 49 mV, which is too large to provide an accurate motor current calculation. A precise current reading may have resulted in the ability to calculate rough motor output torque estimations, which may have been very useful. Nevertheless, this sensitivity is still of the magnitude to inform the microcontroller if current usage spikes excessively.

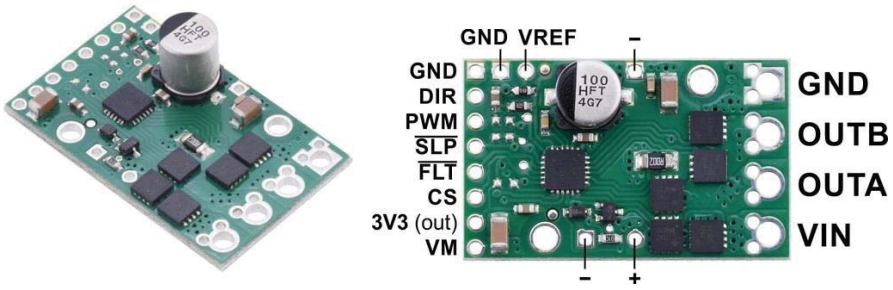


Figure 4 - 12. **Polulu G2 High-Power motor driver 24v13.**

A current limiting reference resistor was added to the driver, corresponding to power of the type of motor it was to drive, as per Figure 4 - 13, offering extra over-current protection.

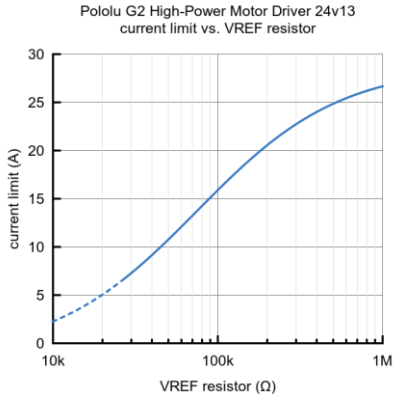


Figure 4 - 13. **Polulu G2 24v13 driver current limit reference.**

Weight	3.3g
Min operating voltage	6.5 V
Max operating voltage	40 V
Continuous output current	13 A
Current Sense	0.04 V/A
Maximum PWM frequency	100 kHz
Logic operating voltage	5V

Table 4 - 6. **Polulu G2 24v13 driver specifications.**

4.3.4 Encoder

Since the rotational actuator design calls for an encoder mounted directly to the actuated shaft, it was desirable for the resolution to be of a higher magnitude. As such, an absolute 12-bit magnetic encoder was used with 4,096 positions per revolution, translating into a resolution of ~0.09 degrees. With an offset link length of 259 mm, the resulting calculated maximum end accuracy resolution per link is 0.4 mm.

US Digital provides two, electronically identical versions of the encoder, the MAE3 shaft encoder and the MAE3 encoder kit (fig. 4 - 14). The encoder kit was favoured as it is documented to tolerate shaft axial play of up to ± 0.635 mm.

Figure 4 - 14. **MAE3 encoder pictures.**

Resolution	4096 positions per revolution
Operating voltage	5V
Typical current usage	16 mA
Sampling rate	250 Hz

Table 4 - 7. **MAE3 encoder specifications.**

Encoder position is determined by its output duty cycle 'ON' / 'OFF' ratio (fig. 4 - 15).

12-bit duty cycle position calculation:

$$x = \left(\frac{t_{on} \times 4098}{t_{on} + t_{off}}\right) - 1$$

$$if\ x\ \leq 4094, position = x$$
$$if\ x = 4096, position = 4095$$

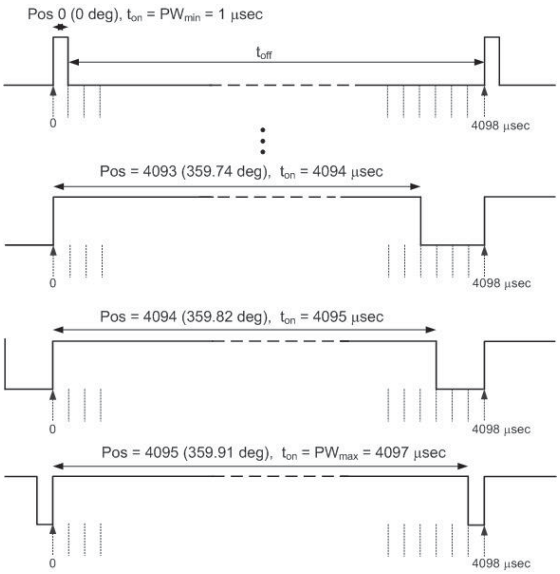


Figure 4 - 15. **Encoder output.**

4.3.5 Micro-Controller

The Arduino Micro is a powerful microcontroller, consisting of some basic control componentry to better serve its integrated ATmega32U4 chip. This chip is available by itself, however it was chosen to go for the entire board, as this may be easily replaced in the event of catastrophic failure. The on-board voltage regulator also increased the ease-of-use for this package.

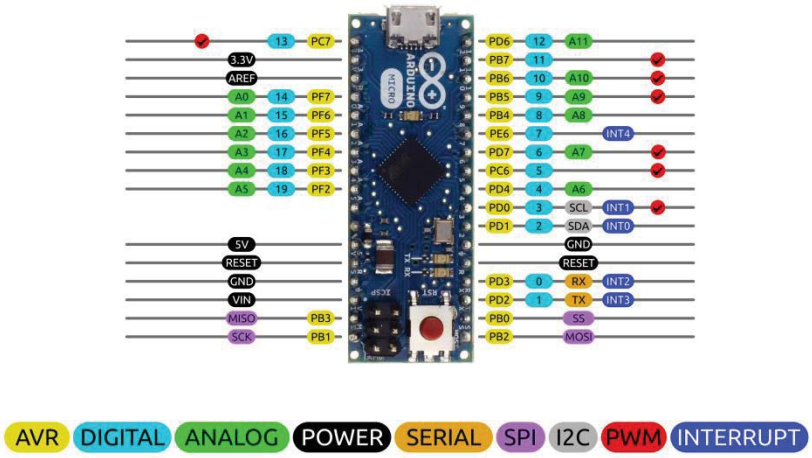


Figure 4 - 16. **Microcontroller pinouts.**

It was decided to go for Arduino as there is significant documentation available for both the hardware and the proprietary software accompanying the system. The number and type of I/O pins provided by the Micro were also found to be adequate.

Microcontroller	ATmega32U4
Operating voltage	5V
Input voltage range	7-12V
Digital I/O pins	20
PWM channels	7
Analogue input channels	12
Clock speed	16 MHz
Weight	13g

Table 4 - 8. **Arduino micro specifications.**

4.4 Control Methodologies and Programming (Rotational Actuator)

4.4.1 Position Acquisition

The *'void loop()'* function (fig. 4 - 17, line 145) is called in an infinite loop by the microcontroller, which will attempt to cycle it as fast as possible. The I²C communications library used will interrupt this loop, execute any contents within the interrupt, and then continue where it left off.

To better control the actuator, a system of toggle functions was devised (fig. 4 – 17, line 159-171). As soon as a command is given to the controller, rather than execute that command directly, a variable is toggled *'true'*. This toggled variable will then be picked up in the main loop, the corresponding function will be executed and finally this variable will be toggled *'false'*. This was done to give full priority to communications, so that in the event a command is received during a process, this process will be interrupted, rather than lose the command. This method also allows for functions to be constantly cycled if their specified conditions are met (fig. 4 – 17, line 157).

```
145 void loop(){
146   readPos();
147
148   if(movementEnabled==false){
149     reachedPOS=false;
150     armStartTime=millis();
151     iTerm=0;
152     Output=0;
153     smoothedVal=0;
154     digitalWrite(SLEEP,LOW);}
155
156   if(errorID==0){
157     if(initTrue==true && movementEnabled==true){locate();}
158
159     if(homeFlag==true){
160        initialize();
161        homeFlag=false;}
162
163     if(newID ==true){
164        transmitID();
165        delay(100);
166        AssignNextID();
167        newID = false;}
168
169     if(IDresetFlag==true){
170       ID=2;
171       IDresetFlag=false;}}
```

Figure 4 - 17. **Main loop function.**

The position acquisition function *'readPos()'* is cycled in the main loop. This function uses the logic as shown in Figure 4 - 18 to calculate the encoder's current position, but prior to that, it waits for a full *'HIGH/LOW'* cycle to pass before beginning a measurement (fig. 4 – 18, line 331-336).

```
324 void readPos(){
325     commIntFlag=false;
326     long calcPOS;
327     long rawPOS;
328     unsigned long timeChange;
329     unsigned long initTime=millis();
330 
331     if(commIntFlag==false){
332     while(digitalRead(PWMOUT)== HIGH && errorID==0){
333       if((timeChange=millis()-initTime)>12){errorID=2;}}
334 
335     while(digitalRead(PWMOUT)== LOW && errorID==0){
336       if((timeChange=millis()-initTime)>12){errorID=2;}}
337 
338     triggerON = micros();
339 
340     while(digitalRead(PWMOUT)== HIGH && errorID==0){
341       if((timeChange=millis()-initTime)>12){errorID=2;}}
342 
343     tON = micros() - triggerON;
344     tOFF = 4098-tON;
```

Figure 4 - 18. **readPos() function.**

A timer (fig. 4 – 18, line 328-329) has been implemented to keep track of how long the measurement process is taking. If this passes 12 milliseconds, an error is tripped, the function is exited and the microcontroller awaits an action.

```
347     if(errorID==0){
348       calcPOS = ((tON*4098)/(tON+tOFF))-1;
349       if(calcPOS<=4094){rawPOS=calcPOS;}
350       if(calcPOS==4096){rawPOS=4095;}
351 
352       for(int i=0; i<30; i++){
353         encoderArray[i-1]=encoderArray[i];}
354       encoderArray[29]=rawPOS;
355 
356       rollAVG=0;
357       for(int i=0; i<30;i++){rollAVG = rollAVG+encoderArray[i];}
358       rollAVG = rollAVG/30;}
359     }
```

Figure 4 - 19. **Position calculation code snippet.**

If there are no errors, a raw encoder position is calculated (fig. 4 – 19, line 348-350). This measurement is then fed onto the end of a 1x30 array, which is then tallied and divided by 30, adding a rolling average filter onto the raw measurement.

To accommodate for an interruption which may occur from communications, a *'commIntFlag'* toggle has been included (fig. 4 – 18, line 325). In the event this toggle is switched, the rolling average filter takes the last known measurement and discards the current measurement. Since position is determined by time and the communications interrupt will add more time, if the measurement process is interrupted, the resulting calculation will be incorrect.

```
361     if(commIntFlag==true){
362       encoderArray[29]=interruptPos;
363       rollAVG=interruptPos;}
364     }
```

Figure 4 - 20. **Communication interrupt code snippet.**

4.4.2 Communications

The *'receiveEvent()'* interrupt function is called any time a message is registered on the I²C communication line. To accommodate for any interruptions caused by this function, the *'commIntFlag'* variable is toggled and the current encoder position is logged (fig. 4 – 21, 203-204). *'errorPrint'* is also toggled so that any active errors will be printed as soon as the function is exited.

Received information is put into a string variable to be used later in the same function. It should be noted that communications are expected to end with *','* and in the event they do not, the communication protocol will simply wait for a small amount of time before accepting this information regardless.

```
201 void receiveEvent(int howMany){
202   errorPrint=true;
203   commIntFlag=true;
204   interruptPos=rollAVG;
205
206   while (Wire.available()){
207     String c = Wire.readStringUntil(',');
```

Figure 4 - 21. **Receive communications code snippet.**

4.4.3 Velocity Profiling

Communication Protocol:

$$s0544,$$
$$s - 05 - 44,$$
$$command - ID - profile\ number$$

In the event that a command is sent of the above structure, the communications function will de-construct this information (fig. 4 – 22, line 214-216) and assess its intended address. If this address corresponds to the microcontrollers address (fig. 4 – 22, line 218), it will decide on which velocity profile to use and toggle a variable to then activate the function which sets those parameters.

```
213     if(c[0]=='s'){
214       int tens = c[1] -'0';
215       int ones = c[2] -'0';
216       int rID =(10*tens)+ones;
217
218       if(rID==ID){
219         int sTens =c[3] - '0';
220         int sOnes = c[4] - '0';
221
222         vProfile= (10*sTens) + sOnes;
223         vpFlag=true;
224       }
225     }
```

```
192   if(vpFlag==true){
193      setVelProfile();
194      vpFlag=false;}
```

Figure 4 - 22. **Velocity profiling code snippets.**

The *'setVelProfile()'* simply contains the control parameters used by the *'locate()'* function to set how the microcontroller behaves (fig. 4 - 23). It does this by changing global variables involved in actuator control processes.

```
656 void setVelProfile(void){
657
658   if(vProfile==1){         //SINGLE ACTUATOR
659     if(armType='r'){
660       deadBandSize=3;
661       stayEnergised=false;
662       intActDist=40;
663       outMinSet=14;
664       filterVal=0.93;
665       iMax =80;
666       Kp = 0.38;     //0.38
667       Ki = 0.012;    //012
668     }
669
670     if(armType=='R'){     //SINGLE ACTUATOR SELF-LOCKING
671       deadBandSize=4;
672       stayEnergised=false;
673       intActDist=40;
674       outMinSet=24;
675       filterVal=0.93;
676       iMax =80;
677       Kp = 0.37;     //0.38
678       Ki = 0.012;    //012
679     }
680
681   }
```

Figure 4 - 23. **setVelProfile() function.**

4.4.4 Movement Switch

A *'movementEnabled'* Boolean flag is toggled through communication (fig. 4 - 24) and cycled in the main loop to determine whether or not the *'locate()'* function is permitted to run.

```
227     if(c=="1"){movementEnabled=true;}
228     if(c=="0"){movementEnabled=false;}

157   if(initTrue==true && movementEnabled==true){locate();}
```

Figure 4 - 24. **Movement switch code snippet.**

Once '*movementEnabled*' has been toggled *'false'*, it is important to reset some of the control parameters, so that the function is aware of its stationary disposition (fig. 4 - 25).

```
148   if(movementEnabled==false){
149     reachedPOS=false;
150     armStartTime=millis();
151     iTerm=0;
152     Output=0;
153     smoothedVal=0;
154     digitalWrite(SLEEP,LOW);}
```

Figure 4 - 25. **Disable movement code snippet.**

4.4.5 Request Position

Communication protocol:

$$i05,$$
$$i - 05,$$
$$command - ID$$

When an encoder position request is sent via the communications channel, the first thing the controller does is check whether or not the sent ID conforms to the controllers ID (fig. 4 - 26 line 231-235). If this is the case, *'requestPos'* is toggled *'true'* and the corresponding function may be picked up and executed in the main loop.

```
230     if(c[0]=='i'){
231       int tens = c[1] -'0';
232       int ones = c[2] -'0';
233       int rID =(10*tens)+ones;
234
235       if(rID==ID){requestPos=true;} }
```

```
184   if(requestPos==true){
185     communicatePos();
186     requestPos=false;}
```

Figure 4 - 26. **Request position code snippets.**

```
622 void communicatePos(void){
623   int tens = ID/10;
624   int ones = ID-(tens*10);
625
626   int mTens = mountType/10;
627   int mOnes = mountType-(mTens*10);
628
629
630   int pThousands = (rollAVG-zeroPOS)/1000;
631   int pHundreds = ((rollAVG-zeroPOS) -(pThousands*1000))/100;
632   int pTens = ((rollAVG-zeroPOS) -(pThousands*1000) - (pHundreds*100))/10;
633   int pOnes = (rollAVG-zeroPOS)-(pThousands*1000) - (pHundreds*100) -(pTens*10);
634
635   Wire.beginTransmission(1);
636
637   Wire.write(mTens+'0');
638   Wire.write(mOnes+'0');
639   Wire.write(armType);
640   Wire.write(tens+'0');
641   Wire.write(ones+'0');
642
643   Wire.write("-");
644   Wire.write(pThousands + '0');
645   Wire.write(pHundreds + '0');
646   Wire.write(pTens + '0');
647   Wire.write(pOnes + '0');
648
649   Wire.write("\n");
650   Wire.endTransmission();}
```

Figure 4 - 27. **communicatePos() function.**

This function gathers information about the condition of the current system and conveys it over the communications channel, producing an output as structured below. The Arduino I²C library used, only allows for integer data types to be sent, resulting in majority of this function simply converting information into a library friendly format.

$$02r05 - 1840$$
$$02 - r - 05 - 1840$$
$$mount\ type - actuator\ type - ID - position$$

4.4.6 Clear Error

Error triggers are scattered throughout the rotational actuator control code. These may relate to timers which overflow, or any other aspect relating to some undesired behaviour. An error value of 0 is the default for *'no error'*, with any *'errorID'* of more than 0, halting the system until it is resolved. If an *'e'* is sent over the communications line, the *'errorID'* will be set to its default value (fig. 4 - 28), essentially clearing any active errors.

```
237     if(c[0]=='e'){errorID=0;}

174   if(errorID>0){
175       digitalWrite(SLEEP, LOW);
176       homeFlag=false;
177       initTrue=false;
178       if(errorPrint==true){
179         errorHandler();
180         errorPrint=false;}}
```

Figure 4 - 28. **Clear error code snippets.**

The above figure shows a snippet of code that prohibits the motor from running, resets the *'homeFlag'* flag and the *'initTrue'* flag when an error has been detected. Additionally, this will call the *'errorHandler()'* function to print any errors, only once. It is important to note that the communications interrupt will set the *'errorPrint'* toggle to *'true'*, so that no further functions may be called until the error has been resolved, or cleared. This also results in a communication of *'errorID'* and an error description at all communication attempts.

```
580 void errorHandler(){
581   int tens = ID/10;
582   int ones = ID-(tens*10);
583
584   int mTens = mountType/10;
585   int mOnes = mountType-(mTens*10);
586
587   Wire.beginTransmission(1);
588
589   Wire.write(mTens+'0');
590   Wire.write(mOnes+'0');
591   Wire.write(armType);
592   Wire.write(tens+'0');
593   Wire.write(ones+'0');
594   Wire.write("-Err:");
595
596   if(errorID==0){Wire.write("0-No error");}
597   if(errorID==1){Wire.write("1-Config error");}
598   if(errorID==2){Wire.write("2-Encoder read error");}
599   if(errorID==3){Wire.write("3-Homing timeout");}
600   if(errorID==4){Wire.write("4-Arm movement timeout");}
601   if(errorID==5){Wire.write("5-Motor Direction Fault");}
602   if(errorID==6){Wire.write("6-Home position error");}
603   if(errorID==7){Wire.write("7-Homing collision");}
604
605   Wire.write("\n");
606   Wire.endTransmission();}
```

Figure 4 - 29. **errorHandler() function.**

Similar to the *'communicatePos'* function, a significant portion of this content deals with data type conversations. A database of errors and their corresponding IDs are stored here (fig 4 – 29, line 596-603) and called when an error occurs to identify that error, of the form:

$$02r05 - Err: 1 - Config\ error$$
$$02 - r - 05 - Err: \ -1 - Config\ error$$
$$mount\ type - actuator\ type - ID - ' Err' - error\ ID\ and\ discription$$

4.4.7 Reset ID

Once an ID has been set by the *'assignNextID'* function, it is useful to have a function to reset this ID to its default value of 2 in cases of manipulator reconfiguration without a disruption to power. As with most functions, the *'IDresetFlag'* is toggled through communications, and executed via the main loop (fig. 4 - 30).

```
239     if(c[0]=='x'){IDresetFlag=true;}
```

```
169     if(IDresetFlag==true){
170       ID=2;
171       IDresetFlag=false;}}
```

Figure 4 - 30. **Reset ID code snippets.**

4.4.8 Program ID

In order to program each microcontroller in the queue sequentially, a serial select line has been added as both an input and an output. A schematic and detailed description is available in section 7.1. When receiving a new ID, the microcontroller must receive a *'P'* and simultaneously be reading a *'HIGH'* logic value from the select input line (fig. 4 – 31, line 241). If both these conditions have been met, the new ID value is assigned based on received information. Generally, when assigning a new ID, the assigning controller must send a protocol of construction:

$$P05$$
$$P - 05$$
$$command - assigned\ ID$$

```
241   if(digitalRead(SL)==HIGH && c[0] =='P'){
242     initTrue=false;
243     digitalWrite(SLEEP, LOW);
244     int tens = c[1] -'0';
245     int ones = c[2] -'0';
246
247     getMountType();
248
249     ID =(10*tens)+ones;
250
251     newID = true;}
```

Figure 4 - 31. **Program ID code snippets.**

Once a new ID value has been received, the *'getMountType()'* is first run to acquire information about the structure type connected to the actuator. This is a simple function that adds to a counter if a pin registers a logic *'HIGH'* value. Binary is used to allow for a total number of 16 unique combinations, 0 is reserved for a default error value in cases where the controller cannot detect any *'HIGH'* values.

```
610 void getMountType(void){
611   mountType=0;
612      if(digitalRead(c1)==HIGH){mountType+=8;}
613      if(digitalRead(c2)==HIGH){mountType+=4;}
614      if(digitalRead(c3)==HIGH){mountType+=2;}
615      if(digitalRead(c4)==HIGH){mountType+=1;}
616 
617      if(mountType==0){
618       errorID=1;
619       errorPrint=true;}
620 }
```

Figure 4 - 32. **getMountType function.**

The *'newID'* variable toggled by receiving a new ID request corresponds to a small piece of code that calls two functions sequentially. First the *'transmitID()'* function is called, followed by a small delay, after which the *'AssignNextID()'* function.

```
163     if(newID ==true){
164        transmitID();
165        delay(100);
166        AssignNextID();
167        newID = false;}
```

Figure 4 - 33. **New ID code snippet.**

The *'transmitID()'* function simply conveys the controllers' current ID and connected link type over the communication line of form:

$$02r05$$
$$02 - r - 05$$
$$mount\ type - actuator\ type - ID$$

```
304 void transmitID(){
305   int tens = ID/10;
306   int ones = ID-(tens*10);
307 
308   int mTens = mountType/10;
309   int mOnes = mountType-(mTens*10);
310 
311   Wire.beginTransmission(1);
312   Wire.write(mTens+'0');
313   Wire.write(mOnes+'0');
314   Wire.write(armType);
315   Wire.write(tens+'0');
316   Wire.write(ones+'0');
317   Wire.write("\n");
318   Wire.endTransmission();}
```

Figure 4 - 34. **transmitID() function.**

In order to assign the next actuator with a unique ID, the current controller must already have an ID. The initial ID comes from the master controller, which is fixed. Each unit will

assign the next controllers' ID by writing *'HIGH'* to the select write line (fig. 4 – 35, line 293), and sending the next calculated ID, an increment of its own (fig. 4 – 35, line 284). The string to be sent is concatenated by the code snippet Figure 4 – 35, line 288-291.

```
283 void AssignNextID(){
284   int nextID = ID + 1;
285   int newTens = nextID/10;
286   int newOnes = nextID-(10*newTens);
287
288   String toSend ="P";
289   toSend +=newTens;
290   toSend +=newOnes;
291   toSend +=",";
292
293   digitalWrite(WL,HIGH);
294   delay(100);
295
296   Wire.beginTransmission(2);
297   Wire.write(toSend.c_str());
298   Wire.endTransmission();
299
300   delay(100);
301   digitalWrite(WL,LOW);}
```

Figure 4 - 35. **AssignNextID function.**

A small delay of 100 milliseconds was added either side of communicating the string to avoid any hysteresis.

4.4.9 Move to Location

The *'locate()'* function is continually cycled by the microcontroller in the main loop, given that *'initTrue'* and *'movementEnabled'* are both of logic *'true'* values (fig. 4 – 36, line 157). This function controls the direction and PWM signals sent to the motor driver. When a communications command of *'t'* is registered, the microcontroller, as most other commands, checks the intended ID first (fig. 4 – 36, 255-259), followed by a deconstruction of the command into its relevant parts. A set-point change command is generally of the following arrangement:

$$t052410,$$
$$t - 05 - 2410,$$
$$command - ID - position$$

```
156   if(errorID==0){
157     if(initTrue==true && movementEnabled==true){locate();}
```

```
254     if(c[0]=='t' && (reachedPOS==true || movementEnabled==false)){
255       int tens = c[1] -'0';
256       int ones = c[2] -'0';
257       int rID =(10*tens)+ones;
258
259       if(rID==ID){
260         armStartTime=millis();
261         iTerm=0;
262
263         int rThousands =c[3]-'0';
264         int rHundred = c[4] -'0';
265         int rTens = c[5] -'0';
266         int rOnes = c[6] -'0';
267
268         prevSetpoint=Setpoint;
269         Setpoint=(1000*rThousands)+(100*rHundred)+(10*rTens)+rOnes;
270
271         if(Setpoint < lowerLimit){Setpoint = lowerLimit;}
272         if(Setpoint > upperLimit){Setpoint = upperLimit;}}
273     }
```

Figure 4 - 36. **Receive movement command code snippets.**

Once the new set-point has been extracted from the message (fig. 4 – 36, line 263-266), the old set-point is saved and the new set-point is set (fig. 4 – 36, line 269). Before exiting the interrupt, the set-point value is clamped to ensure it is within a useable range (fig. 4 - 36 line 271-272). Since the *'locate'* function cycles continuously, the actuator will try to reach its specified set-point continuously, which may be updated by the communications interrupt live.

The methodology used to determine how the actuator will respond to set-point changes and external disturbances employs a hybrid fuzzy logic PI control loop, with encoder position as the process variable and PWM and direction as the control variables.

Before any control is employed, it is first made sure that the loop runs at a fixed rate according to the '*SampleTime*' variable, set to 20 milliseconds.

```
379 void locate(){
380   unsigned long now = millis();
381   long timeChange = (now - lastTime);
382
383   if(timeChange>=SampleTime && errorID==0){
```

Figure 4 - 37. **locate() function code snippet.**

Firstly, the set-point is scaled depending on where the homing position was found to be (fig. 4 – 38, line 385). This has been done to assure that the coordinate systems of each actuator are consistent with one another. Next, an error is calculated between actuator current position and intended position (fig. 4 – 38, line 386), the P-term, also fed to the I-term. Value of the *'iTerm'* may only be calculated if the actuator is within a specific range to its target, or a timer overflows. Resulting in the controller acting as a P-only controller until it is close enough to the set-point to allow the integral term to act on the process. The strength of the integral term is clamped between *'iMax'* and *'-iMax'* (fig. 4 – 38, line 391-392). Integral action was added to utilise its disturbance rejection response. As the P-term was found to be sufficient to reach the intended set-point within an appropriate amount of time, the I-term mainly helps with small set-point changes and steady-state errors.

```
385     scaledPOS = Setpoint+zeroPOS;
386     float error = scaledPOS - rollAVG;                              //P
387
388     if(abs(error)<=intActDist || (abs(now-armStartTime) > 10000)){iTerm+=(Ki*error);}    //I
389     else{iTerm=0;}
390
391     if(iTerm > iMax){iTerm = iMax;}
392     else if(iTerm < -iMax){iTerm = -iMax;}
```

Figure 4 - 38. **Scaled position calculation code snippet.**

Because a gear-and-pinion system was used, a small amount of backlash was present in the system; hence a dead-band was implemented. As soon as the calculated error is of the correct magnitude (fig. 4 – 39, line 394), it is hidden from the controller (fig. 4 - 39 line 397), the I-term was set to 0, and *'reachedPos'* is toggled '*true*' (fig 4 – 39, line 398). Depending on controller settings, motor power may be cut. This is contingent on which type of motor is used or their intended function, with self-locking (*'stayEnergised==false'*) or non-self-locking (*'stayEnergised==true'* - under load).

```
394     if(abs(error) <=deadBandSize){                                          //DEADBAND
395       if(stayEnergised==false){digitalWrite(SLEEP, LOW);}
396       iTerm=0;
397       error=0;            //DEADBAND
398       reachedPOS=true;}
399     else{digitalWrite(SLEEP, HIGH);}
```

Figure 4 - 39. **Deadband calculation code snippet.**

A minimum PWM value may also be set (fig. 4 - 40). Similar to the I-term, the minimum controller setting is only activate under certain conditions. This has been added to avoid overshoot, as a reduction in PWM tends to brake the motors momentum, which is very useful in achieving low set-point errors at low PWM values.

```
402     if(abs(Setpoint-prevSetpoint)<500 || reachedPOS==true){outMin=outMinSet;}
403     else{outMin=0;}
```

Figure 4 - 40. **Output minimum set code snippet.**

The raw output PWM value is calculated in the following code snippet (fig. 4 – 41, line 447). Direction of the actuator is determined based on the sign of the *'Output'* value (fig. 4 – 41, line 448-449). The absolute value of *'Output'* is then taken and clamped between its specified minimum and maximum, determined by the controller's velocity profile. Before this value may be sent to the motor driver, it is first run through a first order filter (fig 4 – 41, line 455). The degree of filtering this value is also determined by the current velocity profile.

```
447     Output = (Kp*error) + iTerm;
448     if(Output<=0){digitalWrite(DIR, LOW);}//CW
449     else{digitalWrite(DIR, HIGH);}//ACW
450
451     Output = abs(Output);
452     if(Output > outMax){Output = outMax;}
453     else if(Output < outMin){Output = outMin;}
454
455     smoothedVal = smooth(Output,filterVal, smoothedVal);  //good value = 0.93
456
457
458
459
460
461     if(smoothedVal > outMax){smoothedVal = outMax;}
462     else if(smoothedVal < outMin){smoothedVal = outMin;}
463     smoothedVal=(unsigned int)smoothedVal;
```

```
374 int smooth(int data, float filterVal, float smoothedVal){
375   smoothedVal = (data * (1 - filterVal)) + (smoothedVal  *  filterVal);
376   return (int)smoothedVal;}
```

Figure 4 - 41. **PWM output and smoothed output calculation code snippets.**

Finally, the filtered value may be sent to the motor driver (fig. 4 – 42, line 482). A time stamp value *'lastTime'* is taken prior to exiting the function, allowing for the sample time calculation.

```
482     analogWrite(PWM,smoothedVal);
483     lastTime = now;}}
```

Figure 4 - 42. **Write PWM value to motor code snippet.**

4.4.10 Homing Function

When a *'h'* command is received through the communications line, followed by an ID corresponding to the controller specific ID, the *'homeFlag'* variable will be toggled *'true'*. This is also the case if *'hALL'* is received. Homing protocol:

$$h05,$$
$$hALL,$$
$$h-05,$$
$$command-ID$$

```
275     if(c[0]=='h'){
276       int tens = c[1] -'0';
277       int ones = c[2] -'0';
278       int rID =(10*tens)+ones;
279
280       if(rID==ID){homeFlag=true;}
281       if(c=="hALL"){homeFlag=true;}
```

```
159     if(homeFlag==true){
160        initialize();
161        homeFlag=false;}
```

***Figure 4 - 43*. Homing function code snippets.**

The *'initialize'* function homes the actuator, finding the encoder position corresponding to the toggling of the home switch through moving the actuator clock-wise. Sampling and error timers are set-up in Figure 4 – 44, line 492-495. The homing function will run continuously until either the home switch has been activated, or the homing timer overflows.

A small function (fig. 4 – 44, line 502-508) has been added to check motor direction through the encoder and throw an error if it is incorrect.

```
489 void initialize(){
490   readPos();
491   int acc=20;
492   unsigned long initTime = millis();
493   unsigned long timeChange;
494   unsigned long lastInitTime;
495   long lastInitPos=rollAVG;
496
497   if(errorID==0){
498   while(digitalRead(homeSwitch)==LOW && errorID==0){
499     timeChange =  millis()-initTime;
500     readPos();
501
502     if((millis()-lastInitTime)>100 && errorID==0){
503       if((rollAVG-lastInitPos)>5){
504         errorID=5;
505         errorPrint=true;}
506
507       lastInitPos=rollAVG;
508       lastInitTime=millis();}
```

***Figure 4 - 44*. initialize() function.**

The homing process speed is determined by 4 fuzzy rules (fig. 4 - 45). These determine the maximum rotational speed of the actuator, depending on location. It was found that this methodology produced more consistent homing positions by approaching the home position at a lower speed, compared to a fixed homing velocity.

```
510     if(rollAVG>800){
511       if(acc<150){acc++;}}
512
513     if(rollAVG<=800 && rollAVG>500){
514       if(acc>80){acc--;}
515       else{acc++;}}
516
517     if(rollAVG<=500 && rollAVG>200){
518       if(acc>50){acc--;}
519       else{acc++;}}
520
521     if(rollAVG<=200){
522       if(acc>35){acc--;}
523       else{acc++;}}
```

Figure 4 - 45. **Homing fuzzy rules code snippet.**

The following code snippet (fig. 4 – 46) simply assures a logic value of *'HIGH'* is sent to the motor driver, direction is set to clock-wise and the previously calculated *'acc'* term is written to the motor driver PWM pin. Figure 4 – 46 also contains the homing process overflow timer, calling for an error at overflow.

```
528     digitalWrite(SLEEP, HIGH);
529     digitalWrite(DIR, LOW); //CW
530     analogWrite(PWM,acc);
531
532     if(timeChange>20500){
533       errorID=3;
534       errorPrint=true;}}
```

Figure 4 - 46. **Homing error tester code snippet.**

Once the homing process is complete and no errors have been found, the *'initTrue'* function is toggled *'true'*. After some delay, several control parameters are set (fig. 4 – 47, line 550-557), followed by a set-point change. Once the *'initialize'* function exits, the *'locate()'* function is free to run continuously. A set-point value of 1745 was calculated to be at the centre the actuators range of motion, corresponding to Figure 4 – 48.

```
544     if(errorID==0){
545       zeroPOS= rollAVG;
546       initTrue=true;
547
548       delay(1000);
549
550       armStartTime= millis();
551       reachedPOS=false;
552
553       iTerm=0;
554       lastPos=rollAVG;
555       smoothedDER=0;
556       Output=0;
557       smoothedVal=0;
558       Setpoint=1745;}
559   }
560 }
```

Figure 4 - 47. **Post homing code snippet.**

Figure 4 - 48. **Physical home position.**

4.5 Linear Actuator Design

Figure 4-49 shows the design of the linear actuator. The control electronics are mounted at the back end to an electronics stand similar to that of the rotational actuator. The principle behind providing actuation for this component makes use of a lead-screw and lead-screw nut (fig. 4 - 50). The proposed lead screw is of 10 mm diameter and 2mm lead size. By use of a 500 rpm (~ 8 rps) motor, the lead screw will be capable of linear speeds up to ~ 16 mms^{-1}, meaning it would take ~ 9 seconds to traverse the actuators entire 150 mm of available travel.

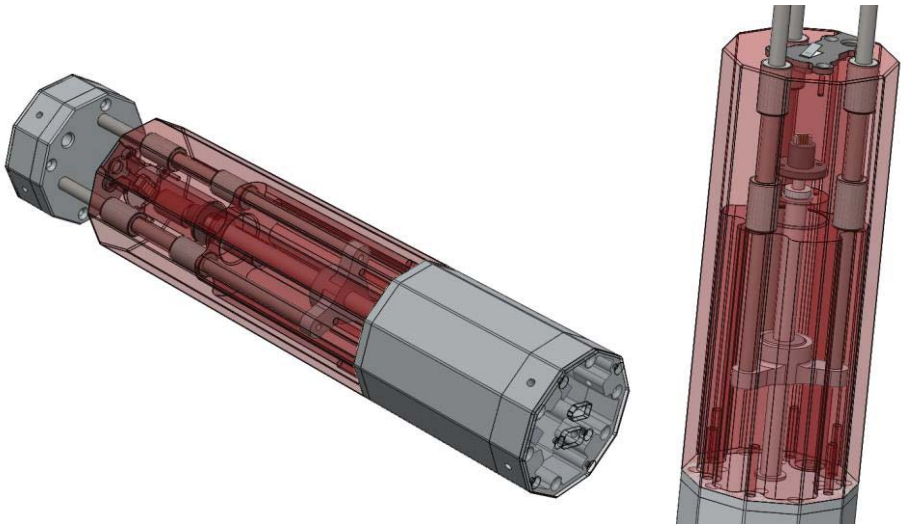


***Figure 4 - 49*. Linear actuator design.**

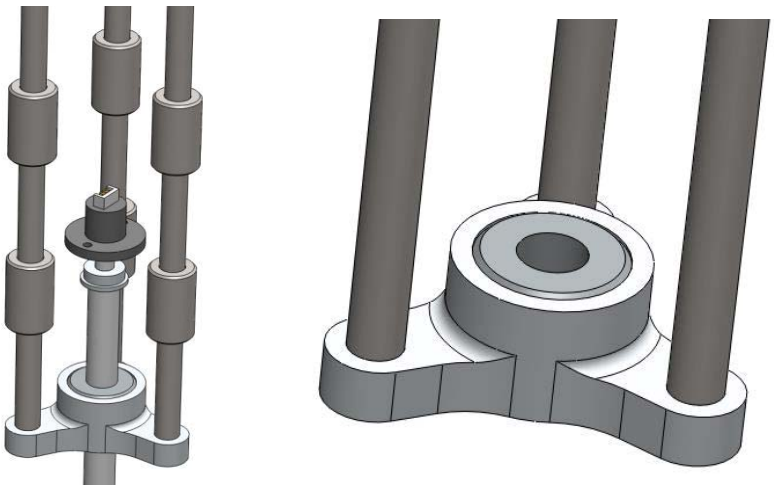


***Figure 4 - 50*. Liner actuator support bearings and lead nut.**

The three support beams were chosen (fig. 4 - 50) to provide a robust base for resisting displacement, which may arise as a result of an applied force in any plane.

The diameter of each linear bearing placed relative to the centre is 51 mm and the distance at which each corresponding bearing on the same beam is placed from the other is 83 mm. This allows for a distance/diameter contact ratio of ~1.6, which should be of acceptable magnitude to avoid any potential self-locking of the system caused by misalignment.

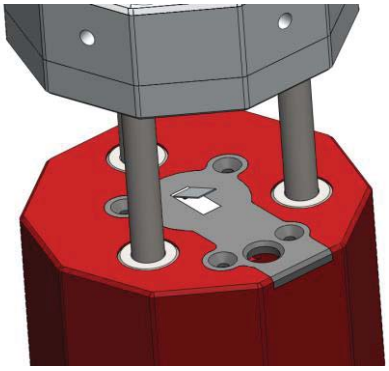


Figure 4 - 51. **Linear actuator home switch.**

Position of the actuator is calculated with respect to the home toggle switch position as seen in Figure 4 - 51 via an encoder, mounted directly to the shaft rotated by the motor. This allows for a measurement that is true to the motion of the lead shaft, resulting in a reading that may not perfectly correspond to the actual output motion. Subsequently, accuracy of the linear core is contingent on how well the lead screw and lead shaft mesh.

Chapter 5

Robot Gripper Design

5.1 2-Fingered Gripper Design

The 2-fingered gripper is the first of many grippers and end-effectors to be implemented with the proposed robotic arms. It operates through a parallel mechanism that actuates one of the two extensions on either side of the device (fig. 5 - 1). The second extension is allowed to rotate freely about its fixed axis. Both extensions have a unique effective radius that their ends must travel through (fig. 5 - 1), determined by their fixture to the base of the gripper.

By replicating the base offset at the head of the gripper, the ends of the extensions are able to maintain this offset throughout, resulting in actuation that does not affect gripper head angle. This is an important design aspect as it always allows for a perpendicular point of contact with the handled part, maximising the acting component of friction.

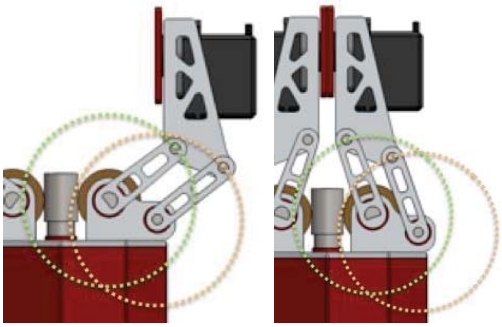


Figure 5 - 1. **Fingered gripper extension travel.**

Each gripping pad is fitted with a force sensitive resistor, yielding a gripper capable of determining gripping strength applied to an object, which may be useful when handling sensitive payloads. This results in a simplified homing methodology, where the very right image of Figure 5 - 2 may be considered the zero position, based on this reading. The implemented drive mechanism consists of a worm drive assembly, where the worm is in the centre of the gripper, meshing with two worm gears (fig. 5 - 2). Rotation at the worm will result in the same degree of displacement at both worm gears. Additionally, the worm drive delivers a very low gear ratio and self-locking characteristics.

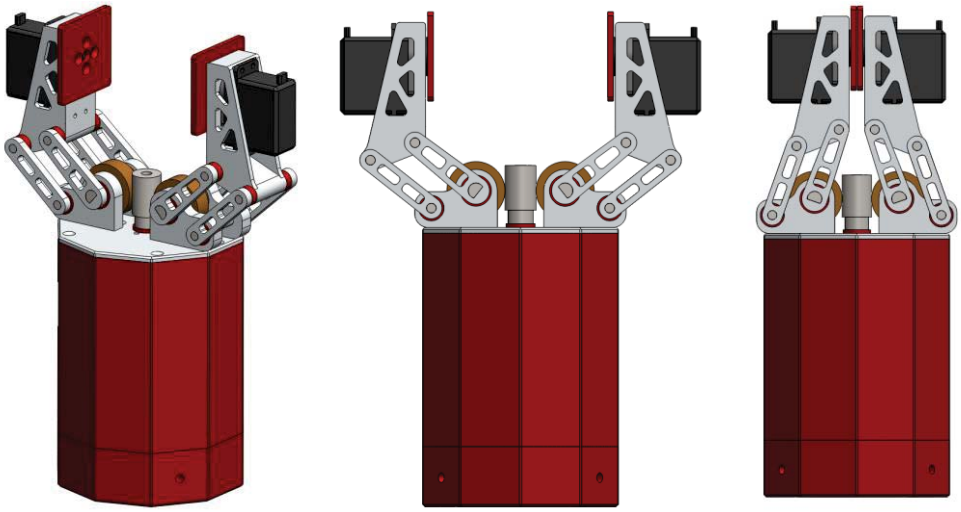


Figure 5 - 2. **2-Fingered gripper pictures.**

A cover will be fitted at the base of the gripper with the goal of hiding exposed gearing. This provides an excellent base for the attachment of additional sensors. An infrared distance sensor has been investigated, capable of providing the system with means of measuring distances from the end-effector position.

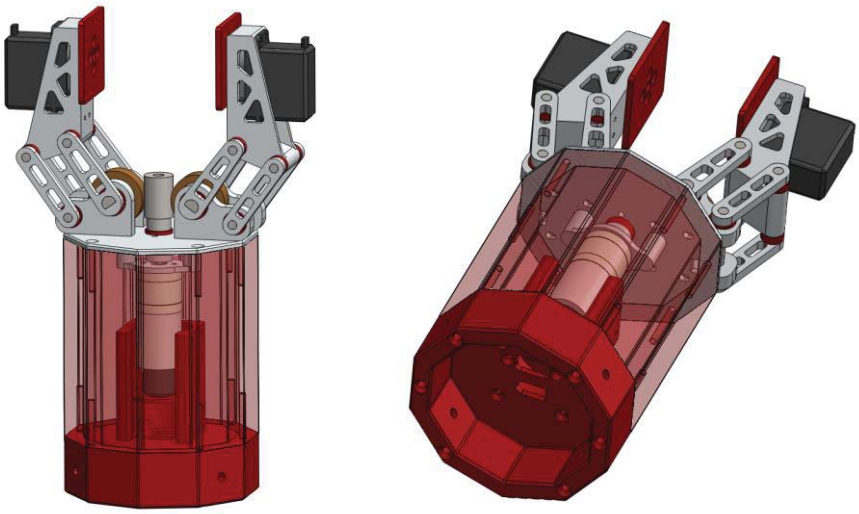


Figure 5 - 3. **2-Fingered gripper section.**

In the event rotational actuation is required to complete a configuration near or at the end-effector location, with a standard grip-only end-of-arm tool (fig. 5 - 3), a 90-degree bend must be added in conjunction with a flipped offset structural element. As such, an additional rotational axis has been added to the 2-fingered-gripper unit at the part level (fig. 5 - 2).

Take the following two arms for example (fig. 5 - 4). Both left and right assemblies have a similar capacity for payload manipulation, however the left arm relies on the offsets added by the gripper itself in comparison to the right arm, where an additional offset and rotational actuator have been included. By addition of the extra rotational axis at part level to the 2-fingered-gripper, significant simplification is achievable, subsequently lowering power requirements of the base rotational actuator and overall stress on the system.

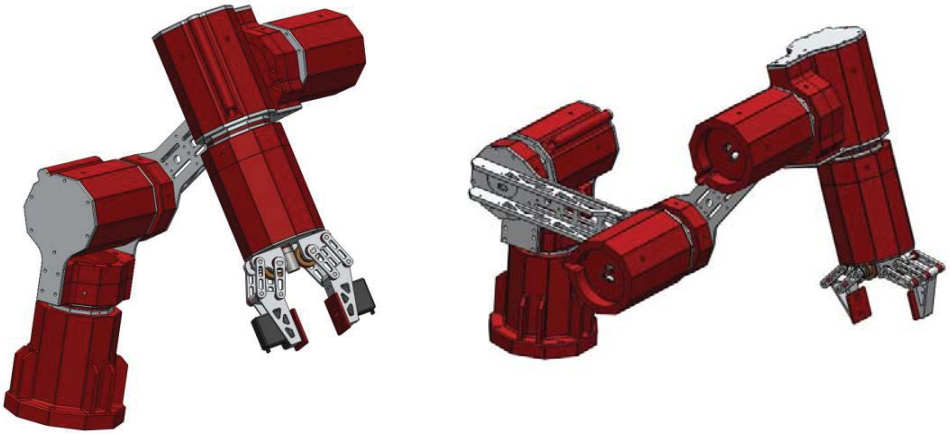


Figure 5 - 4. **PUMA configuration with gripper, articulated configuration with gripper.**

Final revolute motion is added to the gripper by means of two servomotors acting 180° out of phase (fig. 5 - 2). The plastic gripper pad sits flush to the gripper structure, so that the servomotor does not bear any force from gripping an object. Spacers are used to align the gripping structures appropriately.

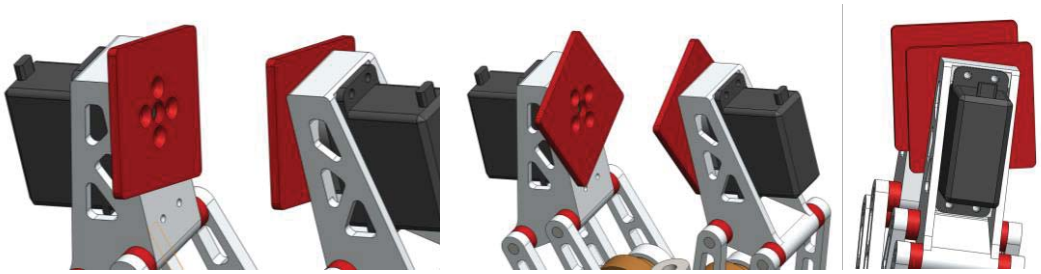


***Figure 5 - 5*. 2-Fingered gripper end rotational axis.**

5.2 Control Componentry (2-Fingered Gripper)

5.2.1 Force-Sensing Resistor Square

Polulu offers a variety of Interlink force-sensing resistors (FSR). This passive component exhibits a decrease in resistance proportional to the magnitude of applied force (fig. 5 - 6). The 40x40mm square FSR was chosen because of its appropriate size, low cost and sensitivity, with a range of 0.2N to 20N.

This component is ideal for use with an Arduino microcontroller. Implementation makes use of a voltage divider of appropriate resistance, so that the range shown in Figure 5 - 6 may correspond well with the 0-5 V analogue read range available to the Arduino. It should be noted that the relating datasheet has specified a - 15% drop in resistance for *'Hot Operation'* and in contrast, a - 5% drop in resistance from *'Cold Operation'*. A repeatability of ± 2-6% has also been specified depending on operation with an accuracy range of 5-25%. As a result of this large potential uncertainty in measurement, Polulu has suggested that this FSR is not suitable for precise force measurements and may be more useful for qualitative purposes.

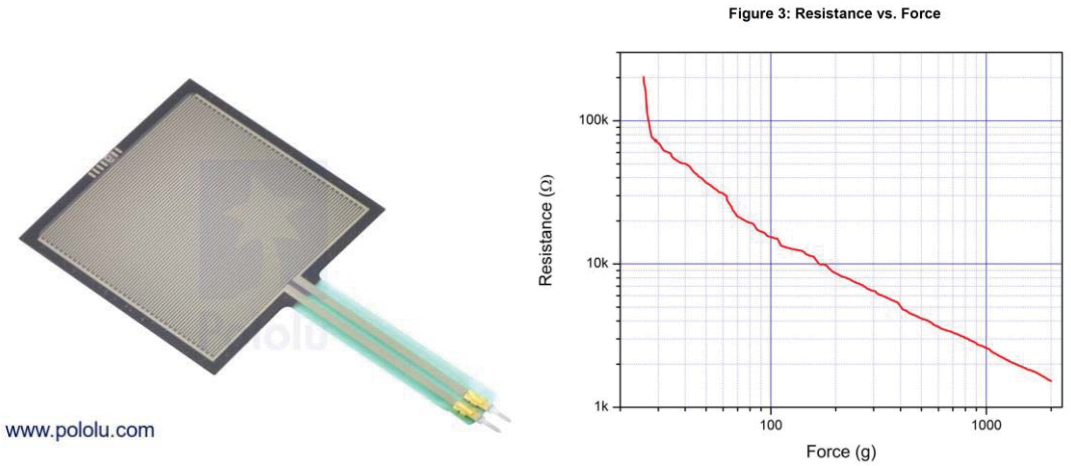


Figure 5 - 6. **Square FSR picture, FSR resistance vs. force diagram.**

Size	40 x 40 mm
Weight	1.2 g
Minimum force measurement	0.2 N
Maximum force measurement	20 N
Maximum resolution	0.5% of full range
Hysteresis	+ 10% average

Table 5 - 1. **Square FSR specifications**

5.2.2 Analogue Infrared Distance Sensor

Determining a relative distance between the gripper unit and a body may provide a very useful measurement when manipulating an object. A distance sensor may work well in conjunction with other object-detection systems, including image analysis. As such, a small infrared distance sensor was investigated as means of achieving this.

The Sharp analogue distance sensor provides a lightweight, low-powered package capable of measuring a distance to a surface within a range of 40 - 300 mm. As the distance increases above the minimum range, a voltage drop may be observed at the data pin line relative to ground (fig. 5 - 7). The graph shown in Figure 5 - 7 also suggests that an incorrect measurement may be taken when attempting to measure a distance below the minimum range.

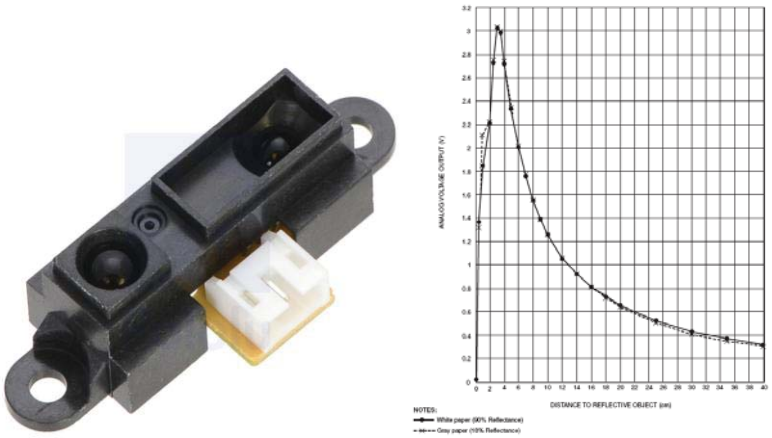


Figure 5 - 7. **analogue infrared picture, sensor voltage vs. distance diagram.**

The sensor makes use of an infrared LED and photosensitive diode in determining a distance. Although this may result in a relatively low-cost component compared to sonar distance sensors, the measurement may be left somewhat vulnerable to the reflectivity of the measured surface.

Weight	3.5 g
Minimum range	40 mm
Maximum range	300 mm
Sampling rate	60 Hz
Operating Voltage	4.5 - 5.5 V
Output Type	Analogue voltage

Table 5 - 2. **Analogue infrared sensor specifications.**

5.2.3 Geared Motor and Encoder

The motor used for the 2-fingered gripper unit is part of the Polulu Metal Gearmotor series. A similar motor was used to provide revolute motion for the rotational actuator. In comparison, this motor is in the low-power *'Motor Type'* category and does not exclude the shaft-mounted encoder. This specific motor was chosen for its high RPM and low power usage. As the gripper unit will by-design be the final unit to draw power from the system, current requirements must be a consideration in motor selection. Additionally, the gripping actuation worm mechanism (fig. 5 - 8) does not require a low speed, high torque motor, as the worm gear will provide a low gearing ratio. A low-current motor may also be less prone to overheat when operating for longer periods of time, which may be a requirement for the gripping unit when applying a constant gripping pressure to the payload.

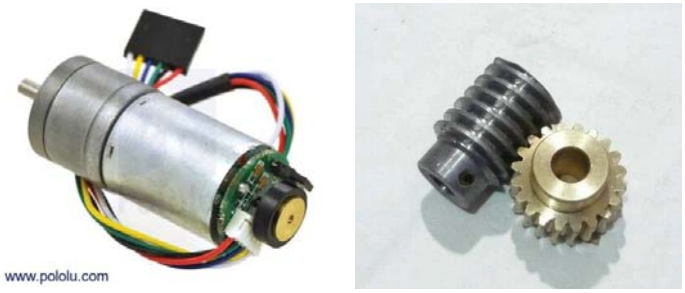
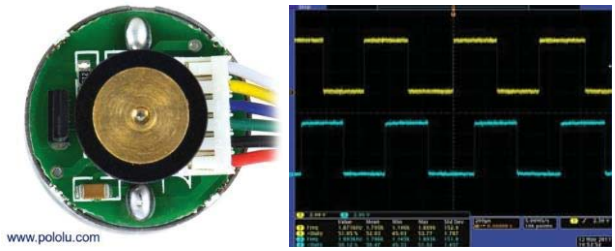


***Figure 5 - 8*. 9.7:1LP Gearmotor, worm gear mechanism.**

Weight	95 g
Gear ratio	9.68:1
Free-run speed	560 rpm
Free-run current 12V	100 mA
Stall current 12V	1,100 mA
Stall torque 12V	1 kg-cm

***Table 5 - 3*. 9.7:1 Gearmotor specifications.**

The Polulu standard encoder (fig. 5 - 9) included in this package is relatively low resolution, registering 48 counts per revolution at the motor shaft. An encoder count is taken prior to gearing, resulting in a final accuracy of 465 counts per revolution at the gearbox output. This accuracy might be affected by the play in the gearbox. It should be noted that an encoder resolution of 12 counts per revolution may be attained if only a single edge of one channel is used.



Color	Function
Red	motor power (connects to one motor terminal)
Black	motor power (connects to the other motor terminal)
Green	encoder GND
Blue	encoder Vcc (3.5 – 20 V)
Yellow	encoder A output
White	encoder B output

***Figure 5 - 9*. Gearmotor encoder, encoder output, encoder colour functions.**

5.2.4 Servomotors

Two Hobbytech YM-2763 servomotors were used in conjunction to implement an additional rotational axis at the end of the 2-fingered gripper. Servomotors provide a 180° range of motion, with a resolution of 1°. Arduino operation consists of dedicating a single PWM pin per servomotor. A library may then be used to manipulate the frequency of the PWM signal sent to the servomotor, which determines the output set-point.

The YM-2763 model transmits motor power to the output shaft through metal gearing, made rigid by its dual ball-bearing design.

Figure 5 - 10. **YM-2763 picture.**

Weight	55 g
Operating voltage	4.8-7.2 V
Stall torque	13 kg/cm
Operating speed	0.17sec/60 degrees (6V)
Dimensions	40.7x19.7x42.9 mm

Table 5 - 4. **YM-2763 servomotor specifications.**

5.3 Control Methodologies and Programming (2-Fingered Gripper)

5.3.1 Main Loop and Position Acquisition

The same control loop methodology described in the rotational section (3.2.3) has been applied throughout the 2-fingered gripper control code. Main differences include a slight change in communications protocol, control basis and a lack of mount type and velocity profiles.

As with the rotational control foundation, the main loop will continually test certain variables and allow or disallow functions to be executed accordingly (fig. 5 - 11).

```
93  void loop(){
94    readEncoder();
95
96    if(enableMovement==false){
97      Output =0;
98      digitalWrite(SLEEP,LOW);}
99
100
101   if(errorID==0){
102     if(initTrue == true && enableMovement==true){locateGripper();}
103
104     if(homeFlag==true){
105        initialize();
106        homeFlag=false;}
107
108     if(newID ==true){
109        transmitID();
110        delay(100);
111        newID = false;}
112
113     if(IDresetFlag==true){
114       ID=2;
115       IDresetFlag=false;}
116   }
117
118   if(requestPos==true){
119     communicatePos();
120     requestPos=false;}
```

Figure 5 - 11. **Main loop function.**

The main loop will cycle as fast as the microprocessor will allow, calling the *'readEncoder()'* function each time. The purpose of this function is to test the logic value of both encoder channels. Each time the *'Ain'* channel registers a logic state change from *'LOW'* to *'HIGH'* (fig. 5 – 12, line 136), the encoder count is incremented. Whether the *'encoderPosition'* variable is increased or decreased is dependent on the logic state of the *'Bin'* channel. The code snippet Figure 5 - 12 is designed to interpret an oscilloscope reading such as shown in Figure 5 - 12. Resultant encoder frequency will correspond to twelve times motor shaft rotation frequency, running at a maximum of 67,200 Hz. The rated motor shaft rotational speed is 5600 RPM.

```
133 void readEncoder(){
134   currentState=digitalRead(Ain);
135
136   if((previousState==LOW) && (currentState==HIGH)){
137     if(digitalRead(Bin)==LOW){encoderPosition++;}
138     else{encoderPosition--;}}
139   previousState=currentState;
140 }
```

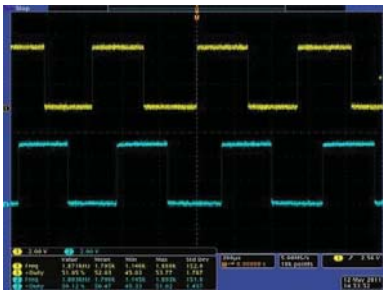


Figure 5 - 12. **readEncoder() function code snippet and corresponding PWM signal.**

5.3.2 Move to Location

As the main loop cycles it will continually attempt to call the *'locateGripper()'* function. Whether or not this function executes depends on system error (fig. 5 – 13, line 101) and the state of Boolean variables *'initTrue'* and *'enableMovement'* (fig. 5 – 13, line 102). The goal of this function is to match both gripper actuator and servomotor positions with the specified set-point. Actuator and servo positions are set through a communication of the below form. Unlike the rotational *'locate()'* control function, gripper methodology does not at this point incorporate any means to control gripper actuator velocity, nor servomotor velocity.

$$t050855090,$$
$$t - 05 - 0855 - 090,$$
$$command - ID - actuator\ position - servo\ position$$

```
101   if(errorID==0){
102     if(initTrue == true && enableMovement==true){locateGripper();}
```

Figure 5 - 13. **locateGripper() function condition code snippet.**

Synonymous with the rotational communication methodology, a message that begins with *'t'* signifies a movement command from the master controller. Desired actuator and servo positions are extracted from a command of the above form. Both set-points are capped to their relative maximum or minimum values respectively (fig. 5 – 14, line 306 and 314).

```
295     if(c[0]=='t' && enableMovement==true){
296       int tens = c[1] -'0';
297       int ones = c[2] -'0';
298       int rID =(10*tens)+ones;
299
300       if(rID==ID){
301         int rThousands =c[3]-'0';
302         int rHundred = c[4] -'0';
303         int rTens = c[5] -'0';
304         int rOnes = c[6] -'0';
305
306         Setpoint=(1000*rThousands)+(100*rHundred)+(10*rTens)+rOnes;
307         if(Setpoint < lowerLimit){Setpoint = lowerLimit;}
308         if(Setpoint > upperLimit){Setpoint = upperLimit;}}
309
310         int servHundred = c[7] -'0';
311         int servTens = c[8] -'0';
312         int servOnes = c[9] -'0';
313
314         servoPos = (100*servHundred)+(10*servTens)+servOnes;
315         if(servoPos < 0){servoPos = 0;}
316         if(servoPos > 180){servoPos = 180;}
317
318         startTime=millis();
319     }
```

Figure 5 - 14. **Movement communication set-point change code snippet.**

The *'locateGripper()'* function handles control for both the gripping actuator and servomotor. Gripper actuator methodology makes use of a time-dependent P-only control algorithm, in which a proportional term is calculated (fig. 5 – 15, line 165) and employed to provide direction and magnitude of the PWM value sent to the motor controller.

Once a set-point change command is registered, the controller is given 2 seconds to reach this position, at which point the *'Setpoint'* variable will be set to whatever the current *'encoderPosition'* value is at that point (fig. 5 – 15, line 156). When gripping an object, it is desirable to have the controller attempt to reach a set-point corresponding to a smaller gripping pad to gripping pad distance than the actual object point of contact encoder position. In this way the motor will exert its full gripping force on the object, at which point motor power may be cut, as the gearing is self-locking. It is important to avoid long periods of high motor current to prolong its expected life.
The Arduino servo library handles servomotor control, which may be manipulated through a *'write'* command consisting of a single, 3-digit coordinate (fig. 5 – 15, line 152-153). To limit speed of the each servomotor, a coordinate change of 1 is incremented each sample time until the desired set-point is reached (fig. 5 – 15, line 150-151). Servomotor coordinates are calculated with a 180° relative offset (fig. 5 – 15, line 153) so that they may move in the same direction in implementation.

```
145 void locateGripper(){
146   unsigned long now = millis();
147   long timeChange = (now-lastTime);
148
149   if(timeChange>=SampleTime){
150       if(servoPos>sTemp){sTemp++;}       //SERVO
151       else{sTemp--;}
152       serv1.write(sTemp);
153       serv2.write(180-sTemp);
154
155
156     if((now-startTime)>2000){Setpoint=encoderPosition;}
157
158     float error = Setpoint - encoderPosition;
159
160     if(abs(error)<10){digitalWrite(SLEEP,LOW);}
161     else{digitalWrite(SLEEP,HIGH);}
162
163     Kp=3;
164
165     Output= (Kp*error);
166     if(Output<=0){digitalWrite(DIR, HIGH);} //CLOSE
167     else{digitalWrite(DIR, LOW);}           //OPEN
168
169     Output=abs(Output);
170     if(Output > outMax){Output = outMax;}
171     else if(Output < outMin){Output = outMin;}
172
173     analogWrite(PWM,Output);
174     lastTime = now;
175   }
```

Figure 5 - 15. **locateGripper() function**

5.3.3 Homing Function

It is assumed that the gripper will always be the last effector in the manipulator configuration queue, connected straight to either a rotational or linear actuator. As such, the gripper cannot by-design attach to a structural element and therefore no mount information is required from this module.

Gripper homing functionality is somewhat less convoluted than that of the rotational actuator. Firstly, servomotor positions are set to their home positions (fig line 184-185). After a small delay, each gripping arm is moved inward until a force is registered from either one of the FSRs (fig line 189). At this point, encoder position is set to 0, which may be considered home position. The set-point is then set to 300, corresponding to approximately a third of total gripper freedom.

$$h05,$$
$$hALL,$$
$$h-05,$$
$$command-ID$$

```
180 void initialize(){
181   unsigned long initTime = millis();
182   unsigned long timeChange;
183 
184   serv1.write(90);
185   serv2.write(90);
186   sTemp=90;
187   delay(2000);
188 
189   while(analogRead(FSR1) <50 && analogRead(FSR2)<50 && errorID==0){
190     timeChange =  millis()-initTime;
191 
192     digitalWrite(DIR,HIGH); //CLOSE
193     digitalWrite(SLEEP,HIGH);
194     analogWrite(PWM,255);
195 
196     if(timeChange>1800){
197       errorID=1;
198       errorPrint=true;}
199     }
200 
201   digitalWrite(SLEEP,LOW);
202   delay(1000);
203   Setpoint = 300;
204   encoderPosition=0;
205   initTrue=true;
206 }
```

Figure 5 - 16. **initialize() function.**

A small over-time error monitoring function has been included (fig line 196-198) that calls for an error if the homing process exceeds 1.8 seconds.

Chapter 6

Shell Structural Unit Design

6.1 Shell Structural Design

Relative to the previously described actuators, structural unit functionality and design is comparatively basic. The goal of each element is to provide a load bearing offset that resists displacement and conveys power / data to the next actuator in the configuration queue. Only the structural element with the highest degree of stress and displacement has been included in this study.

When a structure type element is completed, power and data lines are connected at both ends of actuator contact. At this point, the mount type *'config'* lines are configured with a unique static binary ID. More information on this may be attained from section 7.1. The main structure of the unit is of such design to accommodate the attachment of a large female actuator adaptor in both a direct and flipped orientation. Unfortunately this lead to a small area that does not benefit from the rigidity a vertical rail may provide, resulting in a localised stress concentration and displacement (fig. 6 – 1, 6 - 2). This was mitigated through the use of a more rigid material (tab. 6 - 1).

6.2 FEA Analysis

In operation, the shell unit may be used either vertically orientated or horizontally orientated depending on the type of manipulator arm constructed. When in vertical operation (as may be observed in a SCARA configuration, Table 3 - 3), all force is applied to the structure vertically as shown in Figures 6 - 1 and 6 – 2. Alternatively, when used horizontally (as is the case for the Articulated and PUMA configurations, Tables 3 - 3 and 3 - 4), a torque is introduced as may be seen in Figures 6 – 3 and 6 – 4.

As with the rotational unit FEA analysis, the Solidworks 2014 Simulation tool was used. Elements were fixed at a single end by their mountings only (fig. 6 - 1). A force of 50N was applied to the opposite end vertically or horizontally, depending on the study, to simulate an estimated maximum operational weight of 4.8 kg. The main support beam was simulated as aluminium of material properties shown in Table 6 - 1. Shells were given the properties of ABS (Table 6 - 2).

The purpose of the FEA analysis was to test the robustness and rigidity of the structural units. This is important as any deflection introduced by this part will be detrimental to the final locational accuracy of the manipulator system.

Elastic Modulus	6.9e+010 N/m^2
Poissons Ratio	0.33
Shear Modulus	2.6e+010 N/m^2
Density	2,700 kg/m^3
Tensile Strength	124,084,000 N/m^2
Yield Strength	55,148,500 N/m^2

Table 6 - 1. **Material: 6061 Aluminium alloy properties.**

Elastic Modulus	2.0e+09 N/m^2
Poissons Ratio	0.394
Shear Modulus	3.2e+08 N/m^2
Density	1,020 kg/m^3
Tensile Strength	30,000,000 N/m^2

***Table 6 - 2*. Material: ABS properties.**

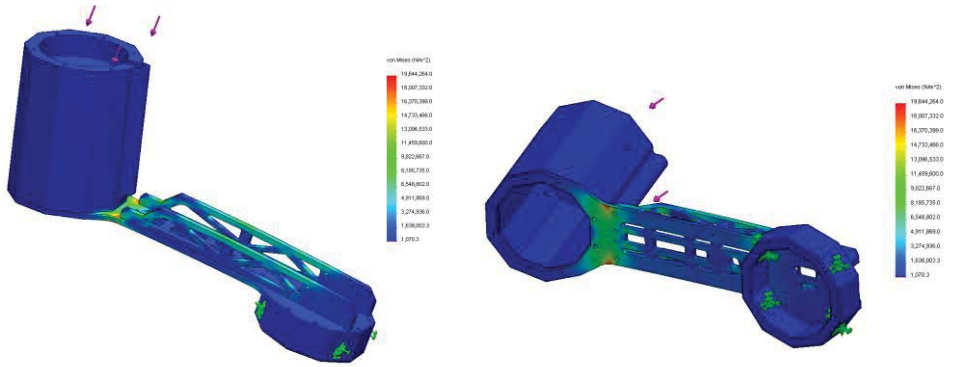


***Figure 6 - 1*. Resultant exaggerated vertical stress diagrams (20 MNm^2 max).**

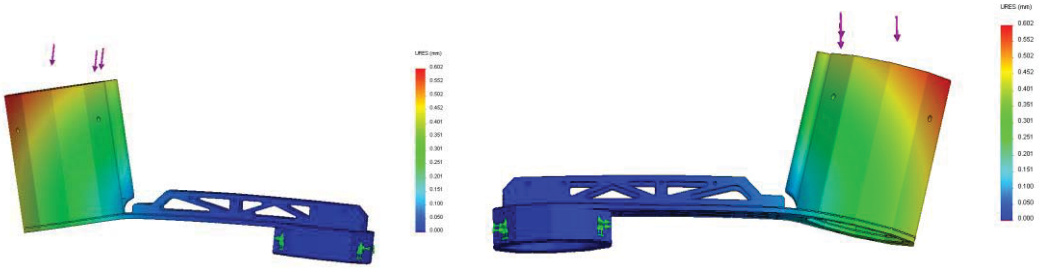
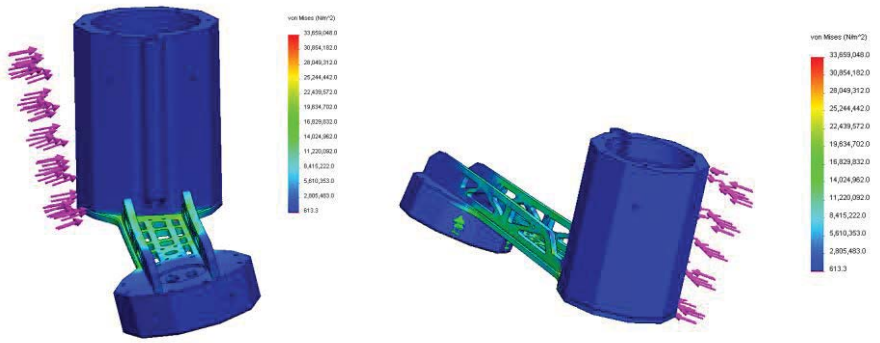


***Figure 6 - 2*. Resultant exaggerated vertical displacement diagrams (0.6 mm max).**



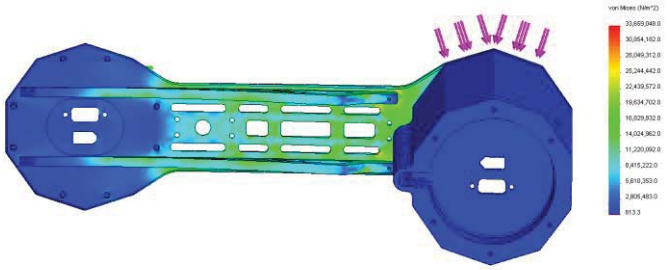


***Figure 6 - 3*. Resultant exaggerated twist stress diagrams (34 MNm² max).**

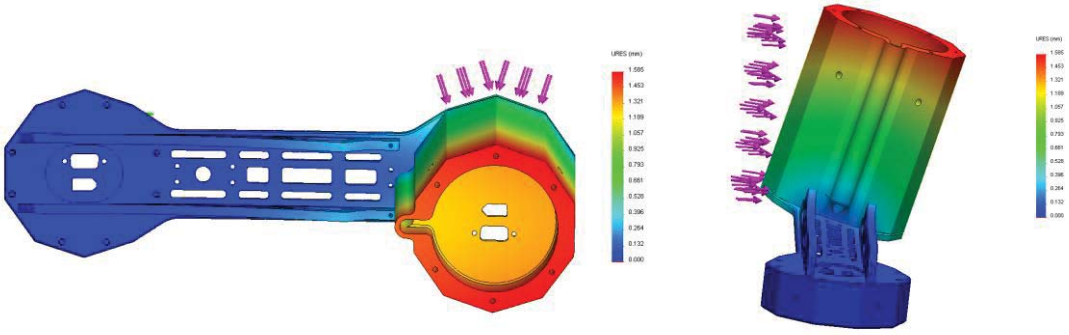


***Figure 6 - 4*. Resultant exaggerated twist displacement diagrams (1.6 mm max top, 0.4 mm max bottom).**

The overall outcomes of the varied simulations seem to suggest good all-round performance for the straight offset structural unit. Both the resultant vertical displacement and stress diagrams show the largest drawback of this design. Ideally the support rail would extend to the ABS core shell as to not rely solely on a horizontal piece of material to resist vertical displacement. Although this is the case, the maximum operational stress as determined by the FEA analysis is still significantly below the yield stress of the material. A displacement of 0.6 mm may also seem extreme, however it should be taken into account that this is measured at an extreme point and will produce an insignificant offset angle when compared to the control methodology.

Chapter 7

Communication, Power Supply and PCB Board Design

7.1 Communications

Electronic communication is a very important aspect of robotic systems, as this determines characteristics such as system response time, which may in turn affect the overall utility of such a system. The amount of information handled by a communication network should also be considered, as well as flexibility.

The Inter-Integrated Circuit (I^2C) communications protocol is a robust serial computer bus capable of supporting multiple masters and multiple slaves. I^2C employs two bi-directional, open drain lines, the Serial Data Line (SDA) and the Serial Clock Line (SCL), each connected to the Vcc line in a pull-up arrangement as per Figure 7 - 1. Both Micro and Uno Arduino controllers have built-in support for the protocol, with an accompanying library. The benefit of I^2C communication in comparison to conventional serial communication is its ability to send information in parallel, so that all connected I^2C compatible devices register a command simultaneously (fig. 7 - 1).

In conjunction with the I^2C communication lines, a select line was added in series (fig. 7 - 1) so that each microcontroller may know its position in the queue. This is achieved by sending a logic *'HIGH'* value from a controller that already knows its ID, to the next controller in the queue only. Concurrently the new ID is communicated to all microcontrollers. At any point in time, only one controller will register a *'HIGH'* logic value from the select line and may at this point accept the communicated ID.

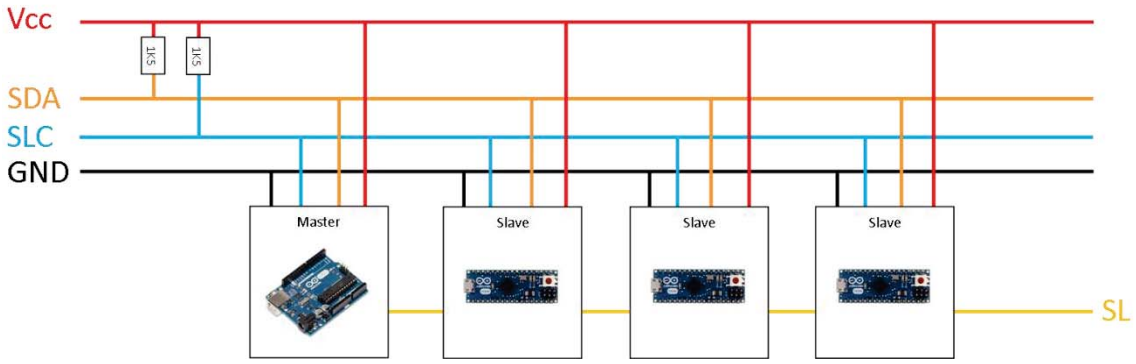
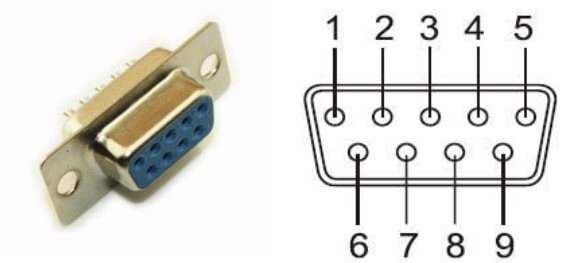


Figure 7 - 1. **Communications diagram.**

Vcc	Positive (+5V)
SDA	Serial data line
SLC	Serial clock line
GND	Ground
SL	Select line

Table 7 - 1. **Communications definitions.**

A DB9 connector, also known as a D-sub connector, was used to convey data and provide a good basis for other low current, logic electrical signals coupled to the microcontroller. Actuators are fitted with male connectors and structural elements are fitted with female connectors. Each channel is colour coded according to Table 7 - 2.



Pin	Designation
1	SCL (blue)
2	SDA (orange)
3	GND
4	Select line (yellow (in),green(out))
5	+5V (red)
6	Config 4 (grey)
7	Config 3 (grey)
8	Config 2 (grey)
9	Config 1 (red)

***Table 7 - 2*. DB9 pinouts.**

Lines 5-9 are used for structural element mount type determination. The *'Config'* lines (6-9) act as a four digit binary number, resulting in a total of 16 unique IDs (fig. 7 - 3). From the microcontroller perspective, all input *'Config'* lines by default will register a *'LOW'* logic value, unless the *'+5V'* line is wired to any of *'Config'* lines at the structural element end. In connecting the *'+5V'* line to specific *'Config'* lines, it is possible to assign an ID to a structural unit.

$$total\ numer\ IDs = 2^4 = 16$$

ID	Structural Element
01	Base
03	Straight offset connection
04	Straight offset connection flipped
02	90 degree bend

***Table 7 - 3*. Total number of IDs and their corresponding usage.**

Several communication standards have been implemented across the entire system:

Actuator Type	Designator
Rotational non-self-locking	'r'
Rotation self-locking	'R'
Linear	'L'
2-fingered gripper	'g'

***Table 7 - 4*. Actuator type designators.**

Communication	Function
't'	Movement manipulation
'h'	Homing function

‘i’	Request position
‘e’	Clear error
‘P’	Program ID
‘s’	Velocity profiling
‘1’	Movement enable
‘0’	Movement disable
‘x’	ID reset

***Table 7 - 5*. General communications protocols.**

To achieve system control from a PC, the user communicates with the master via serial, through the built-in USB port. The master then broadcasts this data over the I^2C channel. All communications are delimited with a ‘,’ character to signify the end of a message. The current control code supports a 5-99 range of actuator IDs, with 0-4 reserved.

7.2 Power Supply

The XT60 nylon plugs have been documented capable of supporting in excess of 60A for extended periods of time. Because of the nature of the reconfigurable manipulator design, actuators and structural elements closest to the base of the robot will experience a current stacking effect. At this point, the component must be able to handle the sum of power used by the manipulator. For an articulated configuration, theoretical maximum current usage may be as high as 15.1A (fig. 7-2). It should be noted that this is an unrealistic figure, as the motor driver limits this current and at no point in articulated operation will all motors be stalled simultaneously without prompting a microcontroller error.

***Figure 7 - 2*. XT60 power connector.**

$$articulated\ power\ chain\ (gripper) = 2.1A + 5.6A + 2.1A + 2.1A + 2.1A + 1.1A$$

7.3 Control Box

In an effort to simplify the reconfigurable manipulator system, a control box was fashioned which includes means to power and communicate with the robotic arm. A Jaycar *‘Pro Quality’*, vented instrument case (fig. 7 - 3), featuring internal mounts and integrated

feet was used to house componentry. A 150W, 12V switch mode power supply of specifications shown in Table 7 - 6 provided adequate power for the system.

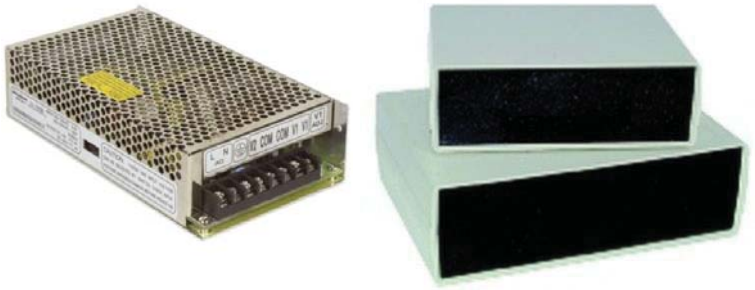


***Figure 7 - 3*. Control box components.**

Weight	650 g
Output voltage	12 V (DC)
Tolerance	±10 %
Ripple & noise	180 mV
Output current	12.5 A
Efficiency	82 %

***Table 7 - 6*. 150 W power supply specifications.**

An Arduino Uno was used as the sole master on the I^2C line, mounted inside the control box, connected to a PC via USB. In most cases, Uno operation is identical to Micro operation. The main differences are physical size and number / location of pins. The Uno also features more robust supplementary electronics.

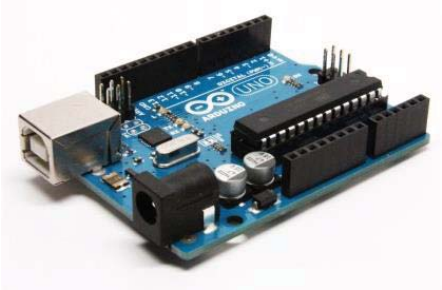


***Figure 7 - 4*. Arduino Uno.**

Microcontroller	ATmega328P
Operating voltage	5 V
Input voltage range	7-12 V
Digital I/O pins	14
PWM channels	6
Analogue input channels	6
Clock speed	16 MHz
Weight	25 g

***Table 7 - 7*. Arduino Uno specifications.**

7.4 Circuit-Board Design

7.4.1 PCBs in Rotational Actuator

Circuit boards have been designed to function in groups of three, dictated by the electronics mount (fig. 7 - 5). Regardless of actuator type, single boards will generally contain circuitry to serve a dedicated component or function.

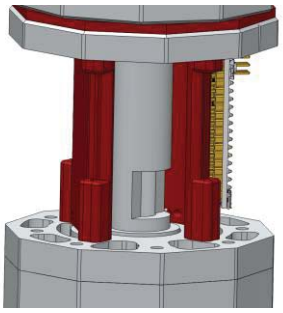


Figure 7 - 5. **Rotational actuator rotational electronics mount.**

The rotational actuator electronics group has 2 power line inputs and 8 data line inputs. In contrast, only 3 data lines and 2 power lines are outputs. Each circuit board has four mounting holes that correspond to the four mounting positions on the 3D-printed electronics mount, so that the final electronics unit consists of three inter-connected circuit boards.

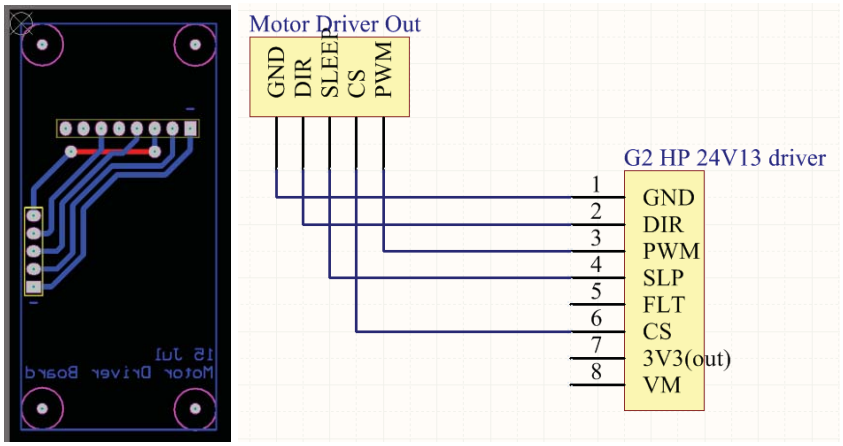


Figure 7 - 6. **Motor driver board PCB and schematic.**

A 220-microfarad capacitor has been added to the input power line to better accommodate for the high instantaneous current draw related to the motor driver. Documentation specifies that a capacitor of minimum of 100-microfarad should be added in parallel with the motor driver for optimal performance. This also provides a small amount of protection against disruptions in the power supply. The Arduino is supplied with 12 V directly, so that the on-board voltage regulator may step-down this voltage to 5 V.

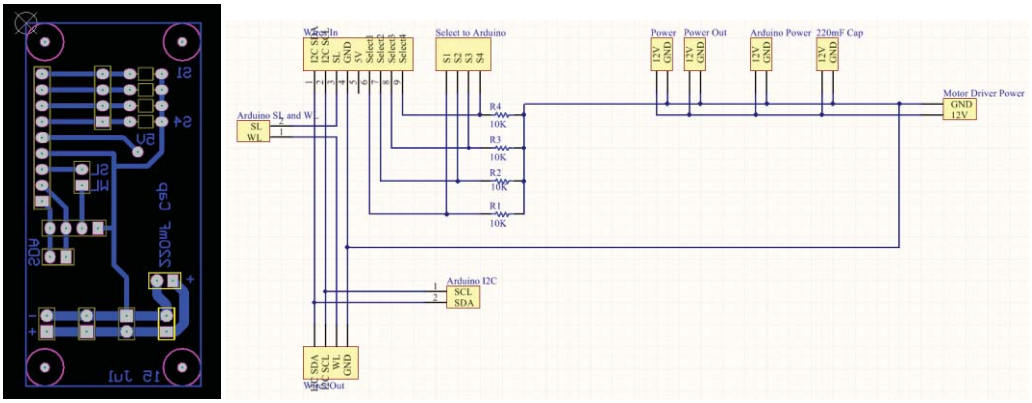


Figure 7 - 7. **Power board PCB and schematic.**

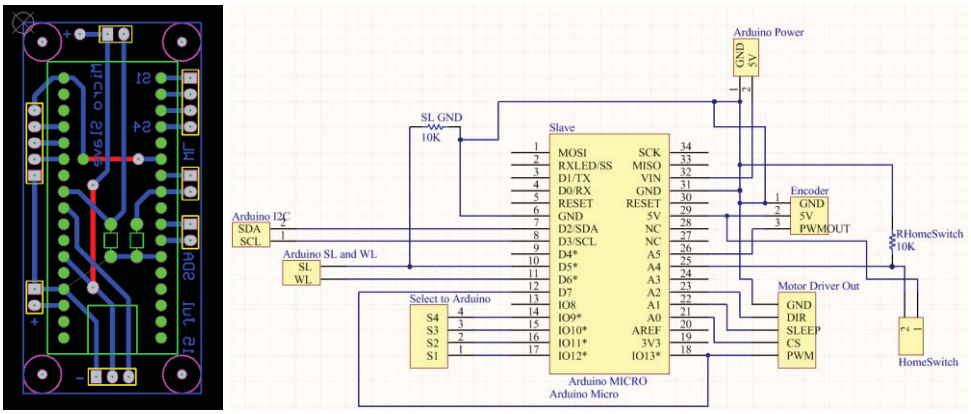


Figure 7 - 8. **Slave wiring board PCB and schematic.**

7.4.2 PCBs in 2-Fingered Gripper

The 2-fingered gripper PCB group incorporates less componentry than its rotational counterpart, mainly due to the lack of output lines and mount configuration lines.

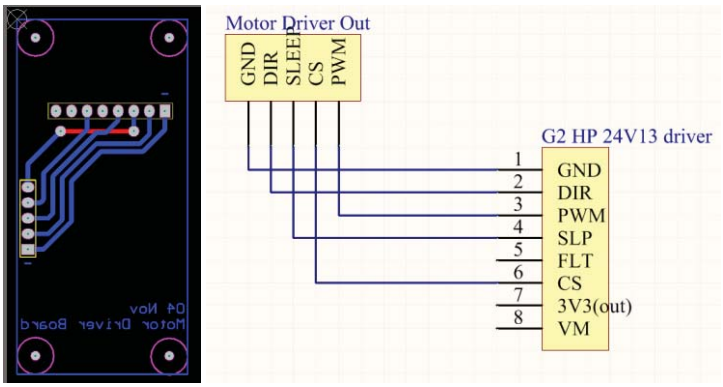


Figure 7 - 9. **Slave wiring board PCB and schematic**

One 220-microfarad capacitor was used per servomotor output channel to improve performance.

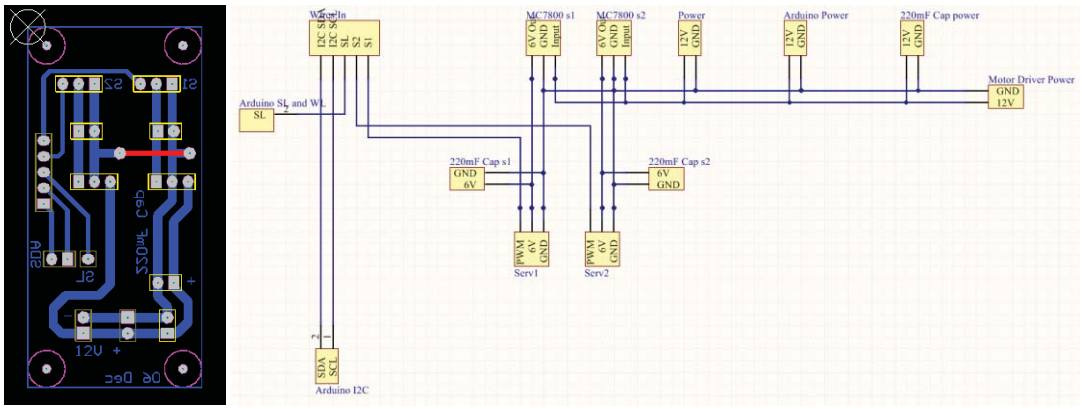


Figure 7 - 10. **Gripper power board PCB and schematic.**

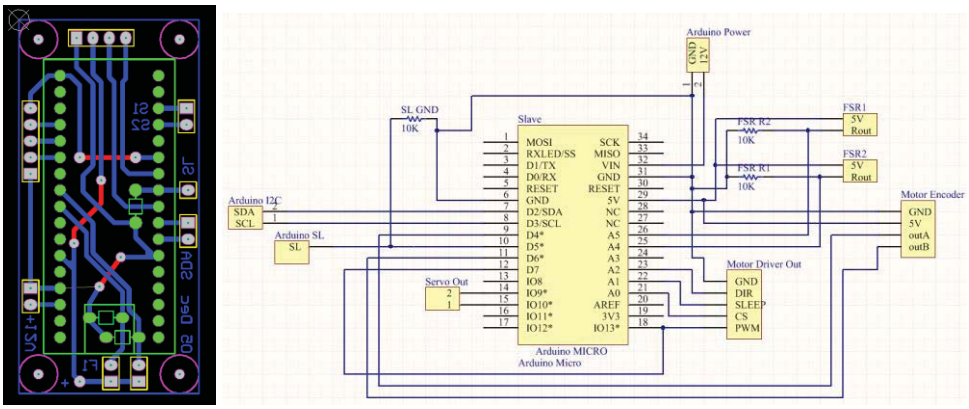


Figure 7 - 11. **Gripper slave wiring board PCB and schematic.**

Chapter 8

Implementation, Testing and Results

8.1 System Implementation

Time and resources put into the realisation of the aforementioned concepts and designs yielded several realised prototype structural elements, rotational actuators and one 2-fingered gripper shown in Figure 8 - 2. An accompanying control box was also implemented. General control of the system may be summed up and simplified by the below flowchart (fig. 8 – 1).

Figure 8 - 1. **System function diagram.**

It was found that shell-type, female elements benefit from a 0.2 mm increase in decagon size when compared to core-type actuators. This specific increase in size reduced the force needed to insert an actuator into a female structural unit, while still providing a good, tight fit. Power and data connectors also added further robustness to the fit. Due to the contact area between a large female shell unit and the corresponding back-end of a core component, the resulting friction was generally found to be sufficient for operation without the use of the mounting screws, which was not the case for the smaller female shell elements.

Figure 8 - 2. **Final prototype parts.**

A combination of machined Aluminium and ABS 3-D printed parts were used in the construction of effectors and structural elements, resulting in component weights of:

Rotational actuator	580-670 g
Test rig	1,006 g
2-Fingered gripper	957 g
90-deg bend	582 g
Straight offset connector	686 g
Straight offset connector flipped	718 g

Table 8 - 1. **Prototype element weights**

The SCARA configuration was tested for deformation by adding a weight of 1954 g at end-effector location and measuring the resulting change in structural element distances relative to a fixed structure with a resolution of 0.5 mm. Maximum displacement at the tool location was found to be 2 mm (tab. 8 - 2), relative to no load. From observation it was very noticeable that this displacement was mainly the result of deformation in the 3D-printed base structure and the non-aluminium parts in the rotational actuator between links 1 and 2 (fig. 8 - 3). A mainly aluminium rotational actuator was used at the base of the SCARA (fig. 8 - 3), which was not subject to any measureable deformation.

Tested Component	Displacement	Displacement (2 kg)	Difference
Link 1	3.5 mm	5.5 mm	2 mm
Link 2	5 mm	7 mm	2 mm

Table 8 - 2. **Measured SCARA displacement**

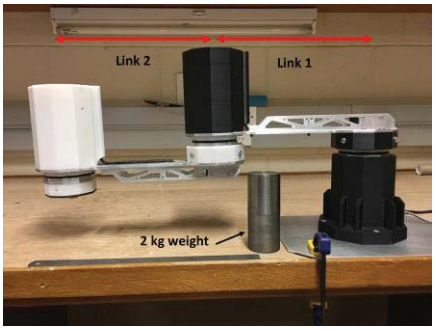


Figure 8 - 3. **SCARA displacement test definitions.**

With the limited number and type of elements realised, it was possible to achieve a few of the proposed popular industrial-type configurations, shown in Figure 8 - 4. These included the SCARA configuration and the articulated configuration. The combination of available building blocks also allowed for the construction of a complete PUMA configuration.

Figure 8 - 4. **Realised configurations.**

Transition Type	Time taken
Loose components to SCARA (no gripper)	33 sec
Loose components to Articulated (no gripper)	31 sec
Loose components to PUMA (with gripper)	54 sec
SCARA to Articulated	48 sec
Articulated to SCARA	33 sec

***Table 8 - 3*. Tested configurations and transition times.**

One rotational actuator was left incomplete to serve as a *'test rig'* of sorts (fig. 8 - 5). As the control electronics were external to the actuator, the programming process was simplified by having the ability to communicate on two separate channels and reprogram on-the-fly. This also made it possible to attain voltage, PWM and current measurements during actuator operation. All load-bearing parts of the *'test rig'* were constructed of aluminium, providing a gauge of the physical properties of the proposed final actuator.

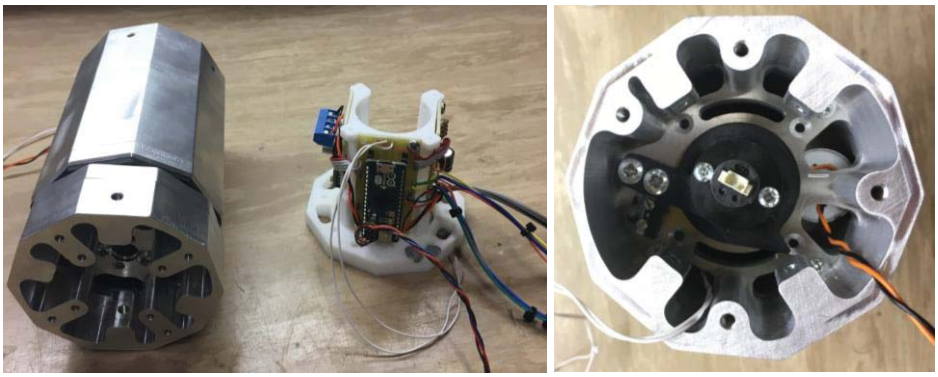


***Figure 8 - 5*. Aluminium test rig.**

8.1.1 Control Basis

The generalised communication protocols as described in section 7.1 proved very useful in providing a reliable, simplistic and real-time control basis for the system. Varying effector type units (rotational, linear, gripper) could be easily distinguished from one another by a single designator (fig. 8 - 6), pre-programmed into the microcontroller upon commission. The parallel nature of I²C communication in conjunction with a select line made it possible for each actuator to determine its own position in the manipulator queue (fig 8 - 6). Each actuator was also able to attain information about its connection to a structural element. If the master controller is made aware of actuator queue position and the corresponding structural element position and type, it is able to build a complete picture of the current configuration of the manipulator arm.

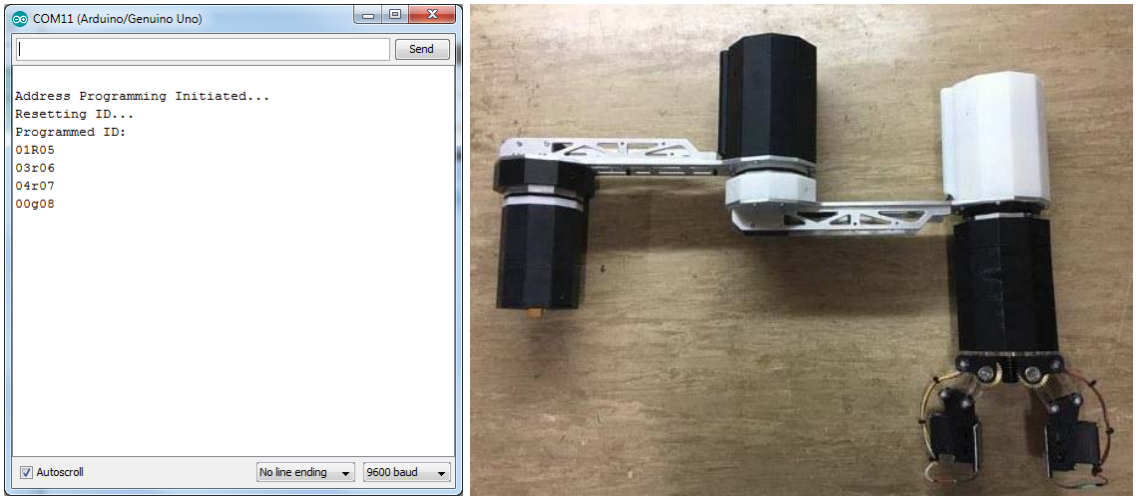


***Figure 8 - 6*. Sending 'P,' to master controller (SCARA configuration removed from base with gripper).**

Communication bandwidth and speed was found to be more than adequate; so much so that it was possible for an actuator to send a constant stream of detailed control variable information during operation. It was also possible to send a single long communication on the I²C line made up of several messages, separated by ‘,’ so that multiple actuators may receive a unique command simultaneously. The Arduino master controller communicated with a PC via USB, acting as a slave to the PC. As this is the case, it is possible to develop finer, GUI-based control methods of the robotic manipulator arm through the use of software such as LabView, Matlab, C#, etc. Serial commands sent to the master generally do not exceed 11 characters.

8.1.2 2-Fingered Gripper

Flexibility of the power and communications platform made the addition of a 2-fingered gripper unproblematic. A broad range of attachments may be coupled with the system, provided the attachment conforms to the decagon design aspect, is of an appropriate size, can be powered by 12V and communicates via I^2C.

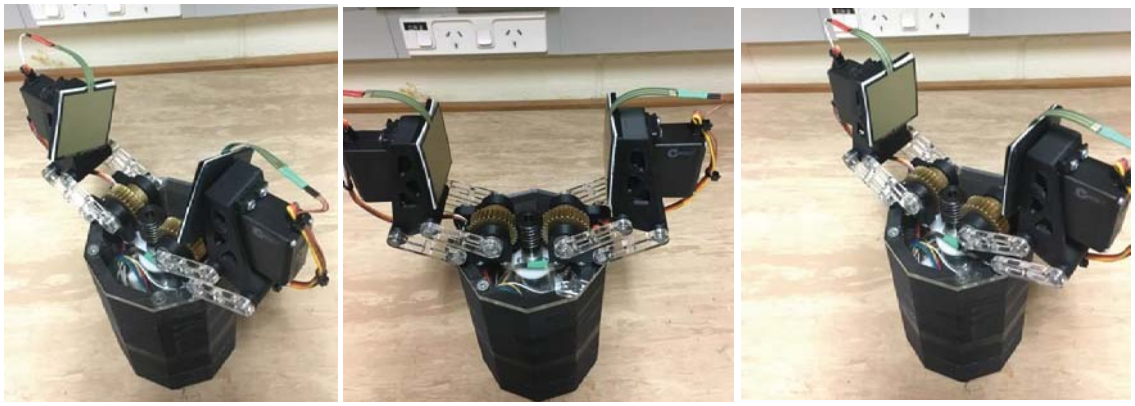


Figure 8 - 7. **Implemented 2-Fingered gripper.**

The parallel gripping mechanism as described in section 5.1 proved to be very robust in implementation. Pressure may be initially exerted onto a part with the low-powered motor, followed by a cut in power. The self-locking nature of the worm-gear meant that there was no significant loss in gripping strength after removing the gear driving force (tab. 8 - 4). The selected low-powered motor was found to be an ideal candidate for the gripper unit as it provided fast mechanism tracking speed (tab. 8 - 5) and was not prone to overheating issues.

Gripping type	Strength
Motor on (255 PWM @ max 1 A)	603 g
Motor off	584 g

Table 8 - 4. **Approximate gripping strength.**

Distance (encoder steps)	Distance	Time taken
300	42 mm	0.5 s
600	82 mm	0.8 s
900	105 mm	1.2 s

Table 8 - 5. **Approximate gripper tracking speed.**

8.2 Control Outcome

8.2.1 Rotational Actuator

In implementation, the US Digital MAE3 absolute magnetic encoder kit (section 4.3) was found to produce a very noisy position reading over time (fig. 8 - 8). The magnitude of the noise made it very difficult to implement an accurate control methodology which was not subject to *'jitters'*. As such, a rolling average filter was explored to remedy this problem.

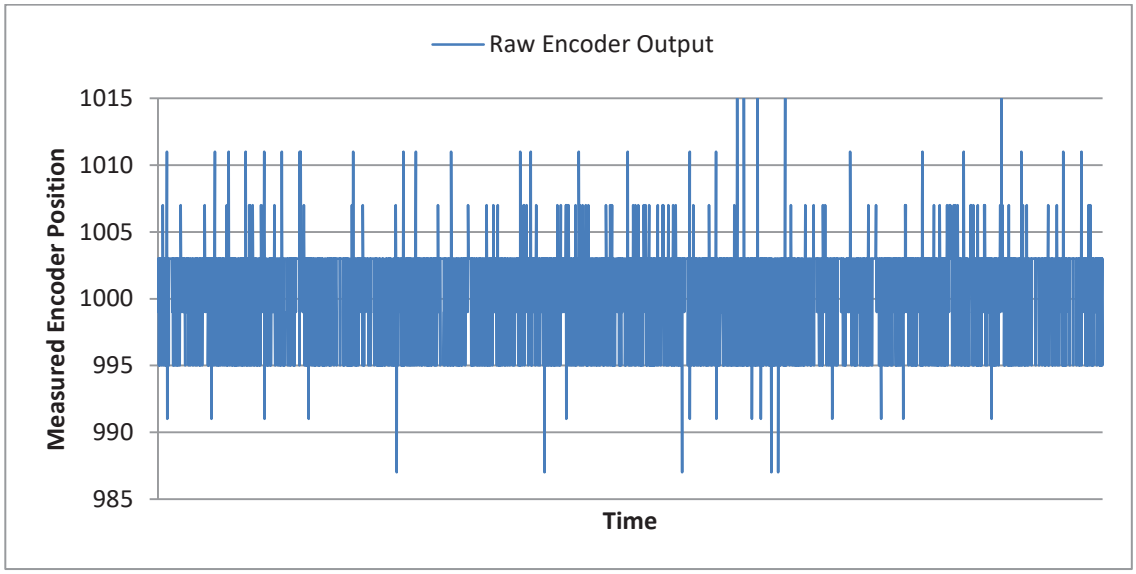


Figure 8 - 8. **Raw output (2000 samples).**

The number of measurements to include in the rolling average filter was varied until an ideal size could be found that did not compromise loop speed or introduce significant system lag to the point of affecting control. This number was 30. As can be seen when comparing the unfiltered data (fig. 8 - 8) with the filtered data (fig. 8 - 9), the rolling average applied to the position acquisition process provided a significant reduction in measured noise. It should be noted that each measurement is documented to take approximately 4 milliseconds.

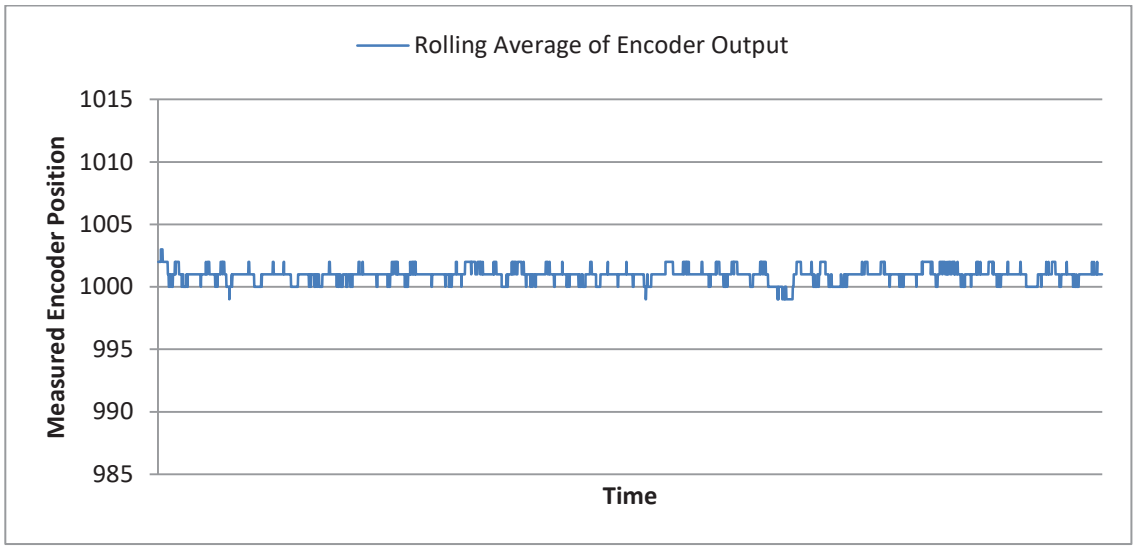


Figure 8 - 9. **Rolling average of 30 output (2000 samples).**

Position of the actuator was not changed between Figures 8 - 8 and 8 - 9. Additionally, both figures were plotted on the same scale for comparison and the number of samples kept consistent. For further comparison between filtered and unfiltered methodologies, a sample of 4000 measurements was taken for each. As shown in Table 8 - 6, the rolling average filter reduced positional standard deviation significantly when compared to the unfiltered data.

Position	Rolling average output		Raw output	
	Standard deviation	Range	Standard deviation	Range
220	0.595418398	3	2.82565162	28
1000	0.594248029	4	3.462788143	28
2000	0.50344748	3	2.96760386	24
3000	0.640058088	3	3.351942702	32
3600	0.630346797	3	3.38981898	28

***Table 8 - 6*. Rolling average vs. raw output comparison.**

From the above sample, it seems that generally, the worst-case noise interval from the filtered measurement is 4 encoder steps. This corresponds to approximately 0.4°.

$$\text{Noise Interval} = \frac{4 \times 360}{4095} = 0.3516°$$

The straight offset structural element has a length between effectors of 259 mm. With an angular measurement that is within ± 0.1758° of the desired location, the maximum theoretical end-effector accuracy of a single arm may be calculated to be ± 0.7948 mm. It was found that the prototype had a typical single arm position repeatability of ± 0.5 mm (non-self-locking) as control tended to locate the arm to the noise midpoint.

The implemented smart control algorithm consists of a filtered PI controller, with fuzzy gain and output constraints. Control profiles have also been implemented that determine PI gains and other control properties depending on robot configuration. Further information may be attained in section 4.4. The second order filter applied to the calculated PWM output provided smooth actuator movements that were not prone to fast changes in momentum.

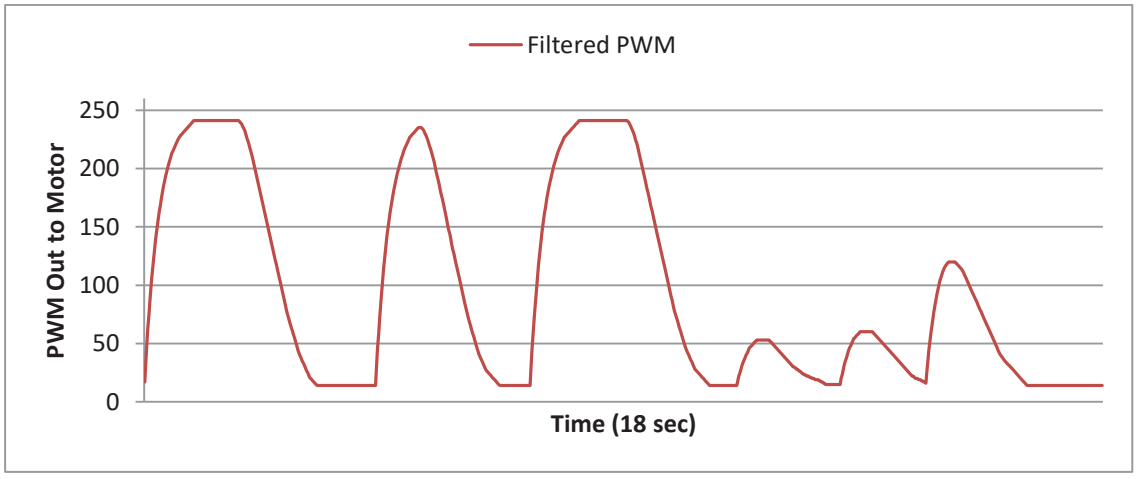


***Figure 8 - 10*. Typical PWM response in response to set-point change (918 samples).**

The following sets of graphs have been attained from the *'test rig'*, with either a self-locking or non-self-locking motor installed, with an attached straight offset connector and no additional load. Each sample is taken at a rate of 20 milliseconds.

Self-Locking

The following graphs were gathered from the *'test rig'* and straight offset connector assembly directly after the homing sequence, with a high-power (HP), self-locking motor installed. It should be noted that when compared to a medium-power (MP), non-self-locking motor, there is no initial dip (fig. 8 - 11). This is a result of the self-locking nature of the HP motor, as all motion is ceased when power is removed.

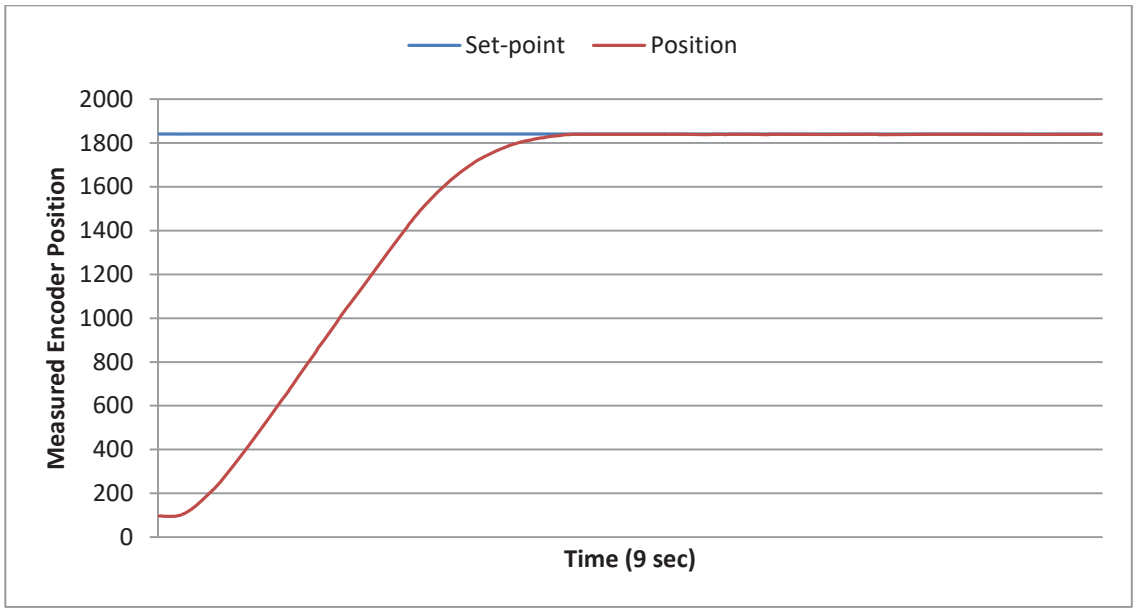


Figure 8 - 11. **Self-locking (R VP1) - moving to home location (437 samples).**

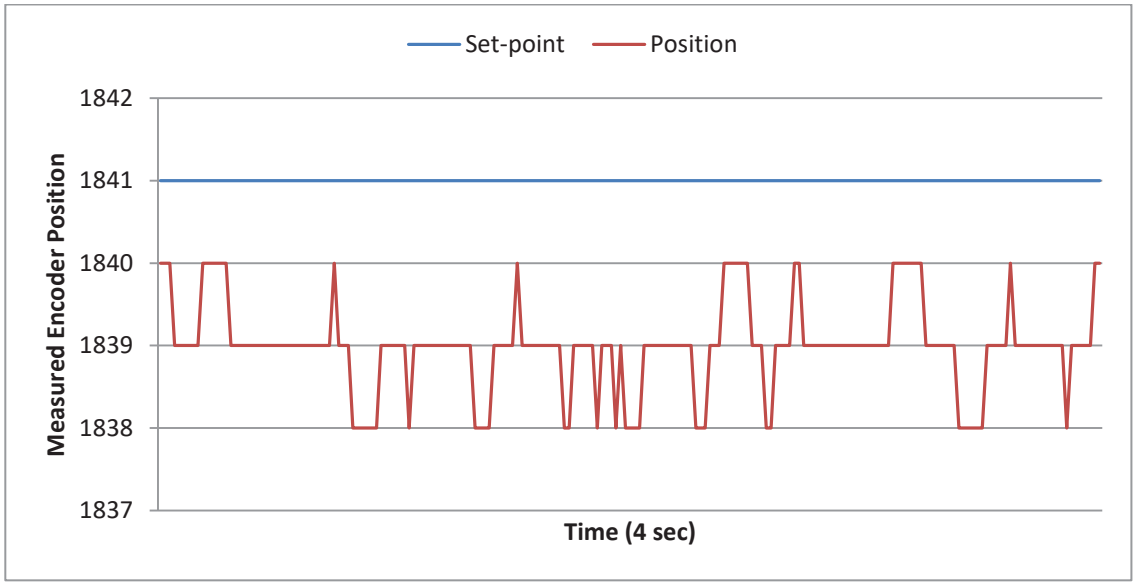


Figure 8 - 12. **Steady-state error (200 samples).**

As shown in Figure 8 - 12, the rotational actuator settles to within 3 encoder steps of the desired location. The desired position may be slightly offset to other actuators, as it is determined by the physical zero location of the home switch.

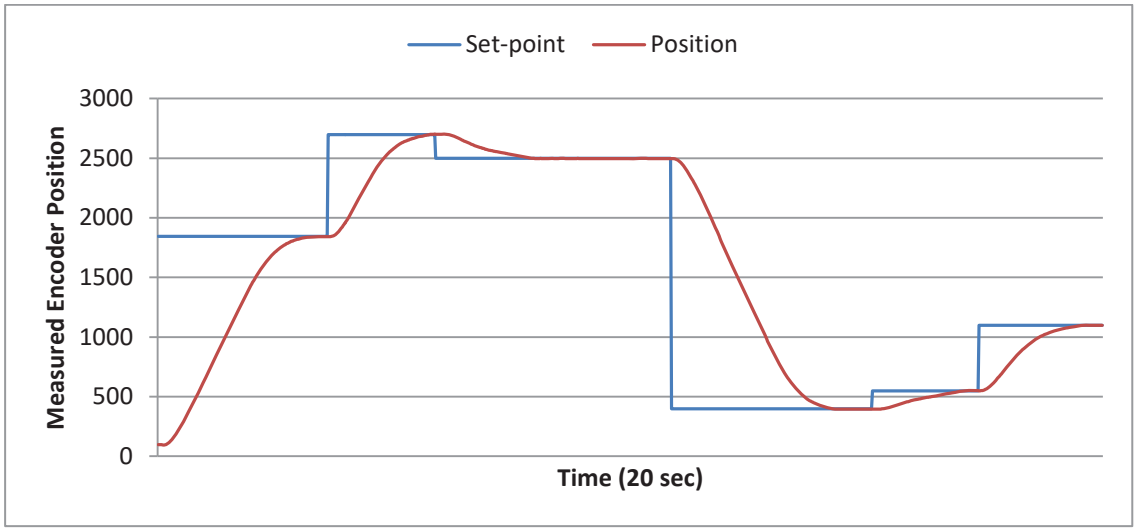


Figure 8 - 13. **Set-point change response (1011 samples).**

The HP tuning parameters seemed to respond very well to set-point changes (fig. 8 - 13). In comparison to MP control, hysteresis was a major influencer in control inaccuracies, evidently resulting in a larger final positional offset (fig. 8 - 12).

Momentum of the motor stator could not be fully anticipated by the smart controller, as the motor would tend to remain stationary until a certain PWM duty cycle is reached (24), at which point a lower PWM frequency may be sent to the motor to maintain motion. This resulted in a relatively high minimum motor speed.

Non-Self-Locking

The following graphs were gathered from the *'test rig'* and straight offset connector assembly directly after the homing sequence, with a medium-power, non-self-locking motor installed. It should be noted that when compared to the home position tracking of HP control (fig. 8 - 11), there is an initial dip (fig. 8 - 14), where position decreases before increasing. As the MP motor homes the actuator, power is cut once the home switch has been triggered. However, the MP motor does not resist motion when unpowered, resulting in momentum carrying the actuator past the home switch location until it is mechanically stopped by the cam design. It should be noted that this does not affect the zero position.

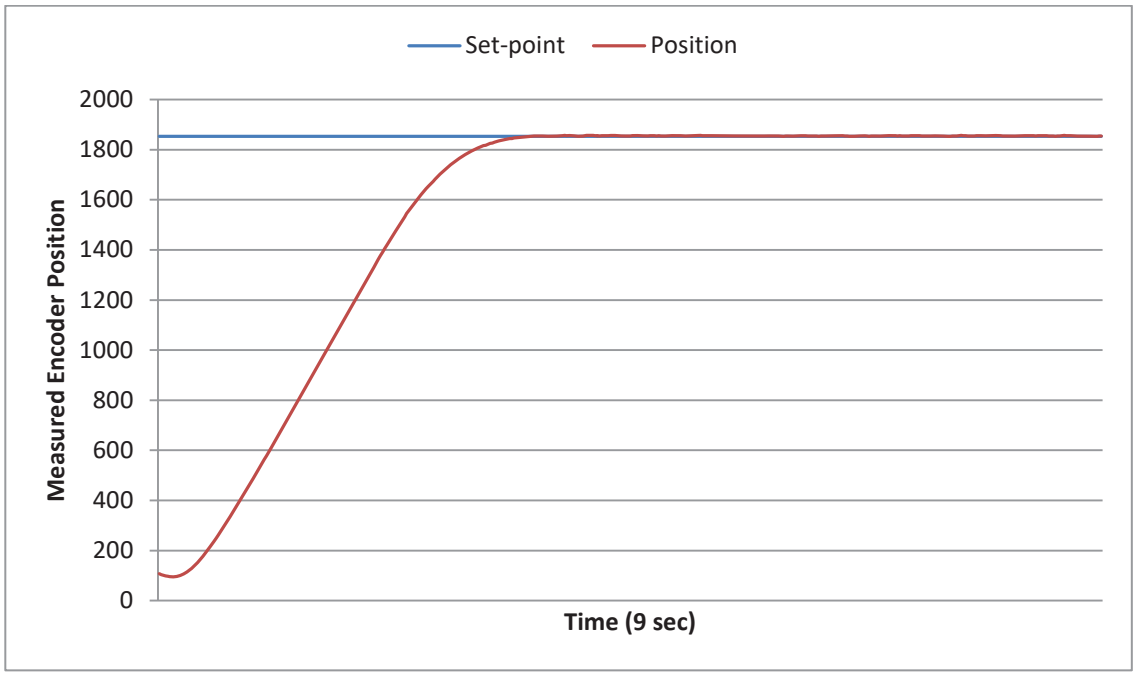


Figure 8 - 14. **Non-self-locking (r VP1) (431 samples).**

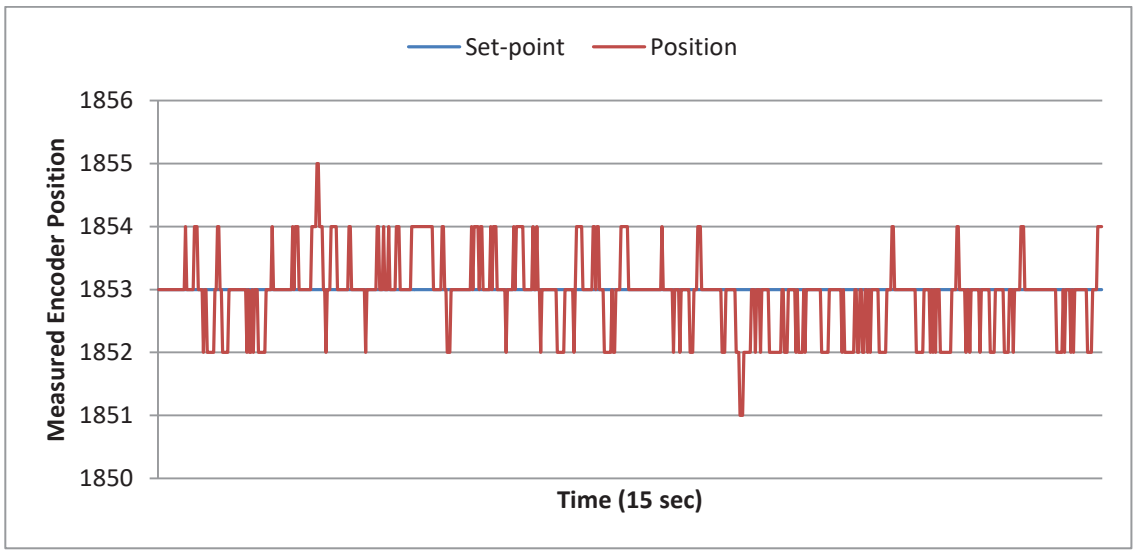


Figure 8 - 15. **Steady-state error (748 samples).**

In comparison to HP control (fig. 8 - 12), MP control parameters tended to locate the actuator much more accurately (fig. 8 - 15), generally settling to the midpoint of encoder noise. This was mainly due to the finer motor control available to the smart controller, as the motor resisted movement only until a PWM duty cycle of 14, resulting in a significantly lower minimum speed than the HP motor. The dead-band size was also reduced to 3, in comparison to 4 used in HP control.

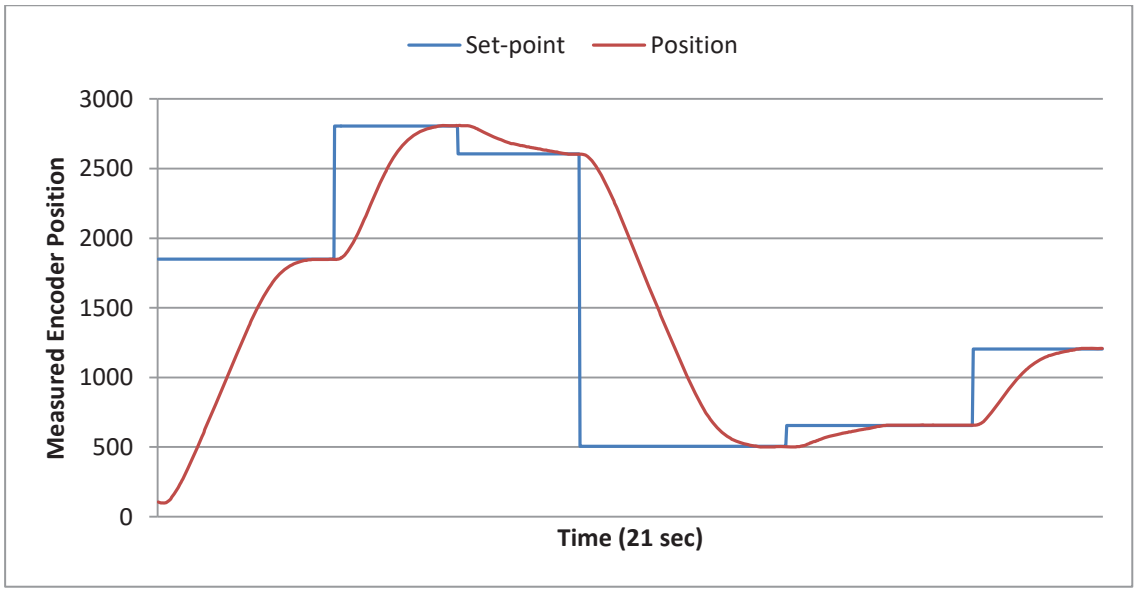


Figure 8 - 16. **Set-point change response (1033 samples).**

The smart control algorithm yielded good set-point change response for the MP motor, tracking to the desired location relatively quickly (fig. 8 - 16) and generally resulting in a steady state similar to Figure 8 - 15 regardless of previous location and/or size of set-point change.

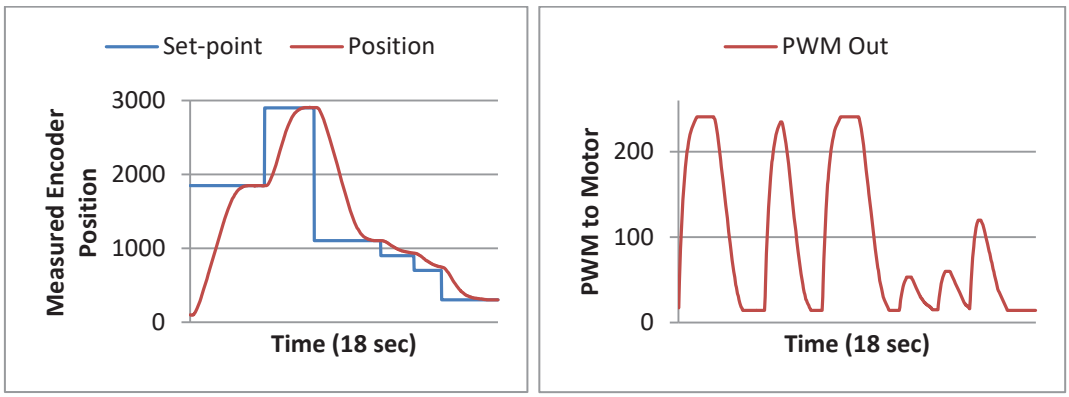


Figure 8 - 17. **Set-point change PWM response (918 samples).**

The resulting controller output motor PWM for set-point changes are generally smoothed (fig. 8 - 17), providing control that does not experience extreme changes in direction or momentum, regardless of communicated desired position.

As shown in (fig. 8 - 18), the smart controller is capable of rejecting a large disturbance within approximately 3 seconds. A large disturbance was defined as a torque on the actuator resulting in a controller PWM duty cycle response that exceeds the fuzzy set minimum of 14. For smaller disturbances (some parts of Figure 8 - 18 and 8 - 20) the *'PWM Out'* signal does not change, but the encoder position does. This is a result of the clamped PWM output enforced on the controller; power may be removed from the motor, but the minimum output does not change.

It should be noted that when driven at a certain PWM duty cycle, the medium-powered geared motor became self-locking, the effects of which may be observed around the 10 second mark of Figure 8 - 18, where the disturbance, regardless of magnitude is not allowed to further move the actuator.

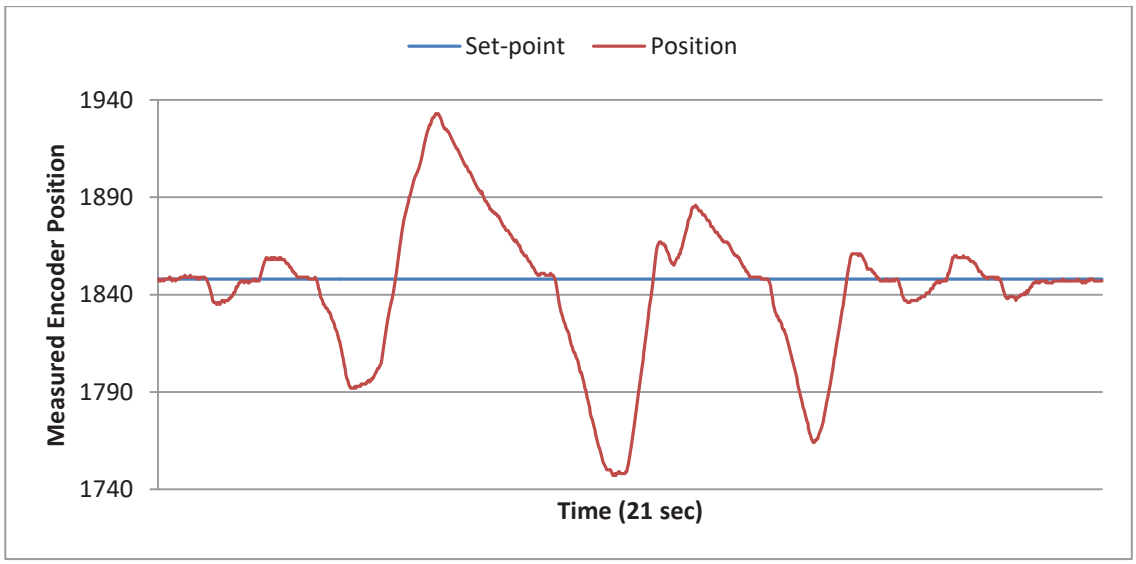


Figure 8 - 18. **Large random disturbances, little time to settle (1043 samples).**

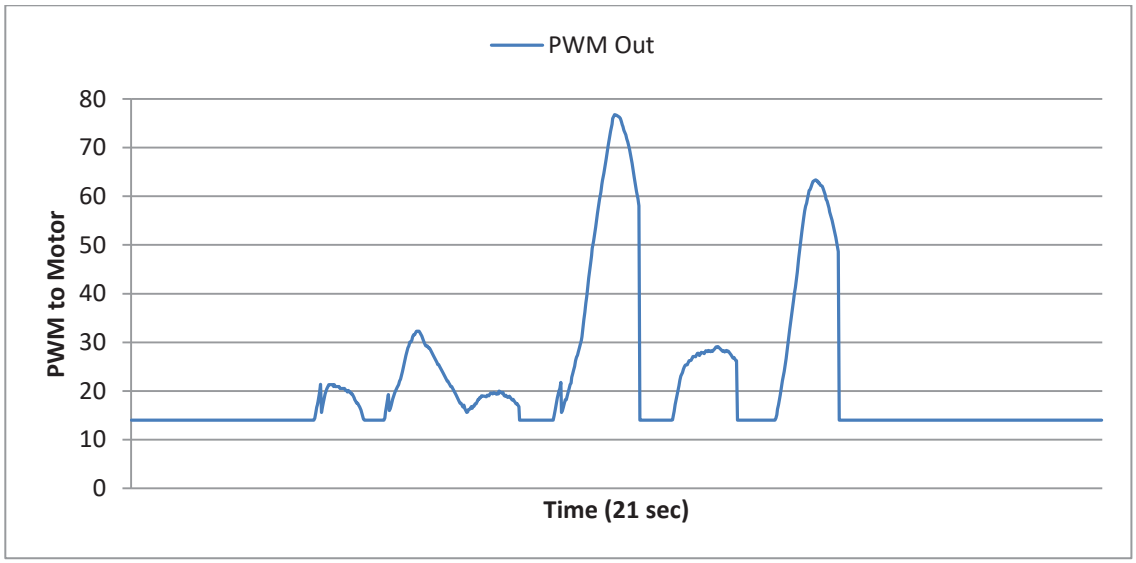


Figure 8 - 19. **Large disturbances PWM response (corresponds to fig).**

Because of the physical properties of the medium-powered motor and the lash introduced by the gear-drive mechanism, the final rigidity of the arm was somewhat compromised if the controller was not active. To save power in this configuration, no motors would be driven once their desired set-points were reached, with the controller monitoring position and re-powering the motors if position exceeds the specific dead-band. As a result, small disturbances may be applied to the actuator before the controller can compensate (fig. 8 - 20), generally in the order of 10 encoder counts (0.88°). Despite gear lash, the smart controller small disturbance response was adequate. For small disturbances of magnitude, the controller does not generally exceed a PWM duty cycle of 14.

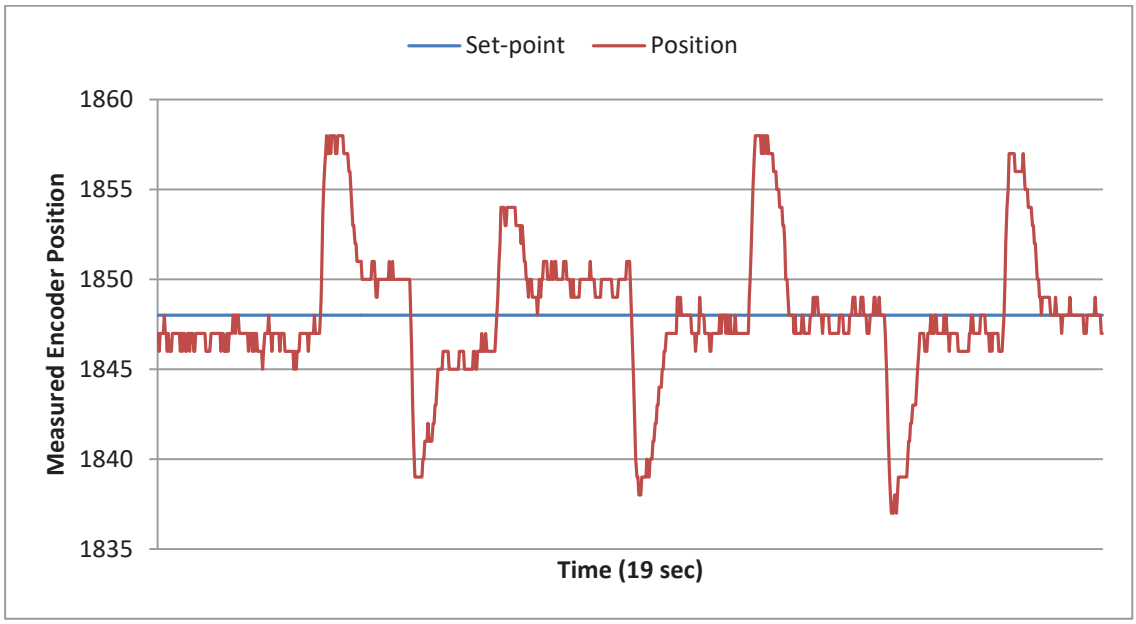


***Figure 8 - 20*. Small random disturbances, little time to settle (932 samples).**

8.2.2 2-Fingered Gripper

The 2-fingered gripper controller, despite being significantly less complex than the rotational smart controller, provided sufficient control, reaching the desired position quickly (fig. 8 - 21). Although there may be some overshoot noticeable by the rotational sensor in the below graph, this was not the case in implementation.

The gripping mechanism was found to be robust, with a significant amount of lash introduced by the meshing of the worm gear mechanism, resulting in a typical gripping pad to gripping pad play of 4 mm (2mm per extension), hence precise control was not needed. The nature of a gripper of this sort does not need accurate position control to sufficiently grip an object. Furthermore, final control of the 2-fingered gripper will be based on the gripping force as provided by the two force sensing resistors, which may result in two simple *'open'* and *'close'* gripper positions.

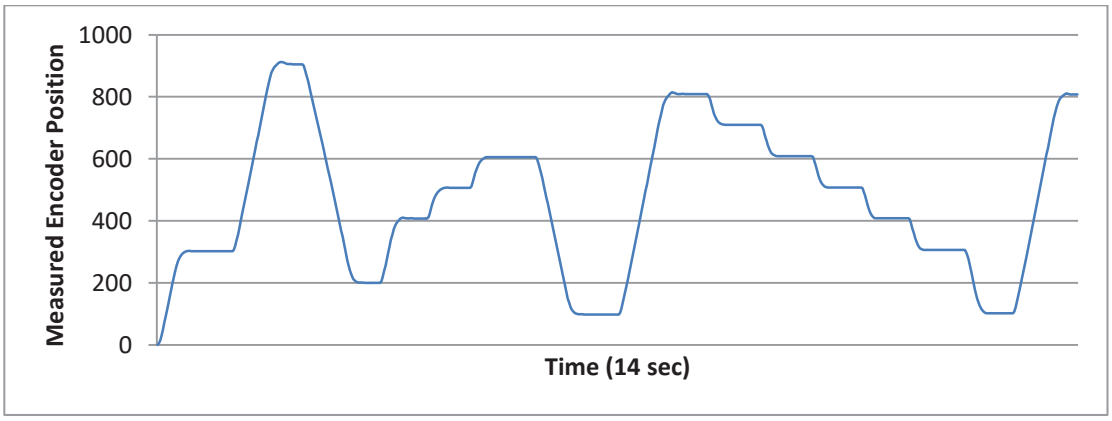


Figure 8 - 21. **Gripper set-point change (688 samples).**

The proposed distance sensor was tuned to a white piece of printer paper, producing the below readings (fig. 8 - 22). Measurements were graphed and a line of best fit was added so that the microcontroller could calculate a rough distance to an object based on a voltage reading. It should be added that the infrared sensor is documented to respond differently depending on the reflectivity of the measured object; hence the produced distance measurement is only an approximation.

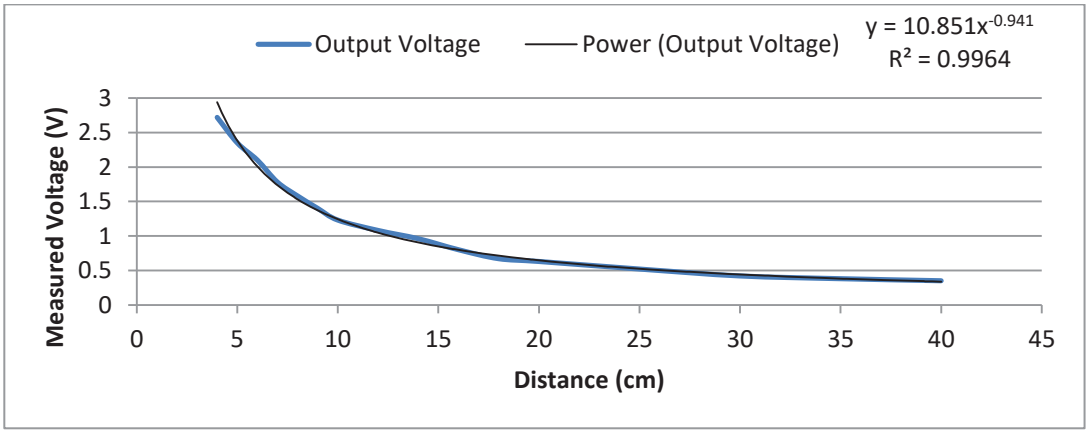


Figure 8 - 22. **Distance sensor measurement**

Chapter 9

Discussion, Conclusion and Recommendations

9.1 Discussion

Through the investigation of the pedagogical robotic products currently on the market it seems that there are no mainstream systems that deal with the specific reconfigurable nature described in this thesis. The notion that this may be known conclusively however is not one that may survive further investigation. As such, the Massey R&D and IP teams were employed to further examine novelty of the proposed concept, and at the time this thesis was written, the patent search confirmed the originality of the Shell and Core robotic manipulator concept.

The conducted research achieved the set goals as defined in Section 1 by developing a Shell-Core reconfigurable robotic manipulator concept and producing a physical prototype that includes the following features:

- Compact joint control system.
- Quick-change modules.
- Robotic manipulator configuration auto-identification.
- Plug-and-Play.
- Position tracking.
- Neat hidden wiring.

Testing of the prototype proved the proposed concept and demonstrated that the system could change from one robot arm configuration into another in less than three minutes to perform new tasks.

While this research produced promising outcomes, there are several aspects that may benefit from improvement and further research, such as the implementation of artificial intelligence for job identification, part holding and path planning. Based on the development of the prototype, the considered control theories relating to the implementation of the proposed system provided a significant insight into the range and type of control methodologies appropriate to this area of discourse. Although a smart PI controller with some fuzzy elements was used as the main type of controller for the rotational actuator, it is not unreasonable to assume that a larger dependence may be put on fuzzy logic when implementing control for an assembly where response is affected by gravity in a non-uniform fashion. Nonlinearities are introduced into the response of the system as soon as a link is added vertically (fig. 9 - 1).

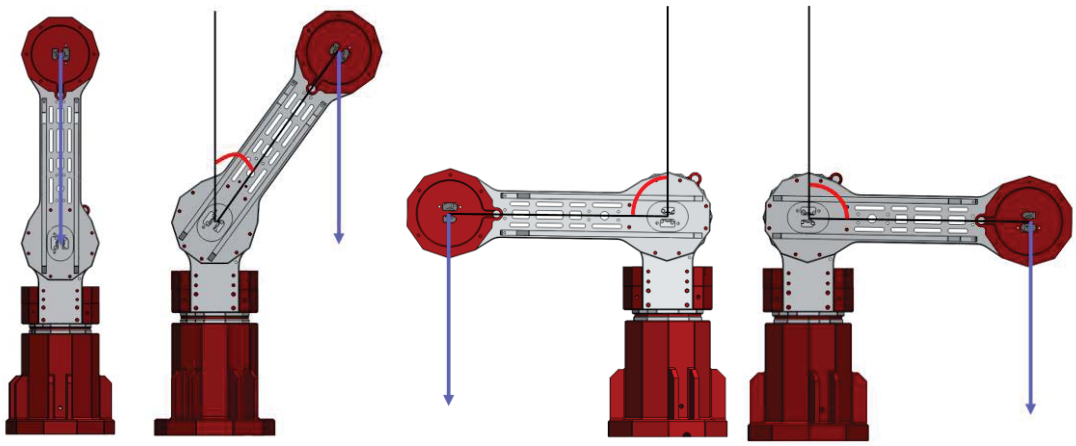


Figure 9 - 1. **Introduced nonlinearities.**

From briefly testing the smart PI controller in all of the above configuration (fig. 9 - 1), it is clear that the controller in its current state is not capable of automatically handling the non-linear response of the system to a precise degree, often overshooting the desired set-point and settling to a position within 30 encoder counts for HP control and 70 for MP control.

This was mitigated by manually selecting the correct velocity profile. Initially, this was a consideration and it was planned to implement a sliding fuzzy logic element to the controller that would automatically determine both proportional and integral gains depending on distance from the vertical (fig. 9 - 1), but due to time constraints, this can only be implemented in the next stage of this research project.

A drawback in terms of control for the reconfigurable robotic manipulator prototype was the backlash introduced by the motor driven gear and pinion mechanism (fig. 9 - 2). Although final steady-state accuracy in the SCARA configuration was not affected, steady-state rigidity was somewhat compromised.

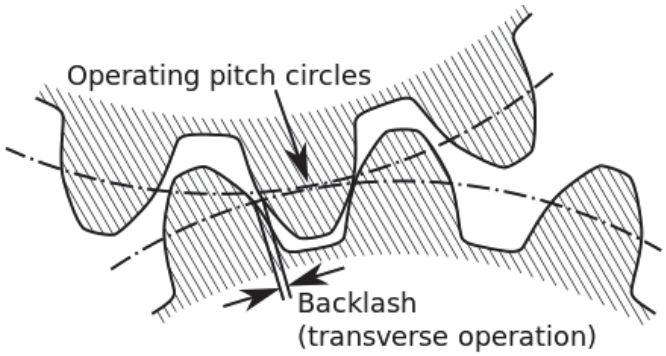


***Figure 9 - 2*. Gear backlash diagram.**

Play from the gear alone was found to be approximately 10 encoder steps (fig. 8 - 20). Although this backlash could be prompted by the smallest of disturbances during steady-state, it did not seem to be an issue during normal operation, nor did an added load affect control accuracy. If the actuator was disturbed during position tracking, final steady-state position was not affected. It seems that a rotational actuator's lack of rigidity is fairly unimportant when the current manipulator operates in two planes only, such as the SCARA configuration, and is not disturbed when stationary. Gear backlash is almost removed completely when dealing with a configuration of the form shown in Figure 9 - 1, as there is constant force acting on the actuator. However, this is not the case when the arm is perfectly vertical. Of course, these singularity positions can be avoided through control methodologies.

The resulting system platform concept was found to produce very encouraging results, capable of achieving all mainstream industrial robotic manipulator configurations as defined in Section 3.1, given enough building elements. The end prototype is very promising in terms of proof-of-concept. Communication and power infrastructure is very flexible, providing a base system prototype capable of accommodating a range of future effectors, add-ons, supplementary software and advanced GUI control without a significant amount of additional work.

This basis already allows for ease of control through a GUI capable, PC based program such as Matlab through serial commands, setting the precedent for numerous advanced

control concepts of the manipulator for various applications. Without further improvement or development to the prototype system, it is possible to implement proprietary software that may calculate the reverse kinematics of the arm, based on information known about specific dimensions of the configuration. It is also possible to implement trajectory control of the manipulator. The notion of *'teaching'* the manipulator arm what to do is already within the capability of the prototype. If all non-self-locking actuators are used, the user may move the arm to a desired location. Software may then request the current position of each arm and so a sequence of locations and tasks may be established.

The prototype's repeatability was tested to be within ± 0.5 mm for the SCARA configuration. The intended purpose of this prototype did not incorporate use within a high accuracy industrial context; therefore this manipulator accuracy is acceptable. Taking this into consideration, it may be said that the prototype provided proof that the concept is viable and realistically achievable. Furthermore, accuracy, rigidity and manipulator strength may all be improved given a larger budget.

9.2 Conclusion

In conclusion, this thesis has briefly explored the current educational robotic manipulator arm market and in conjunction with the Massey IP team, found confidence in the novelty of the Shell-Core structured robotic manipulator as defined in Section 3. Relevant control methodologies have been explored and implemented in the prototype, with consideration of future advanced control.

This research proposed and developed a Shell-Core reconfigurable robotic manipulator arm concept, capable of achieving all mainstream industrial type robotic arm configurations (section 3.1). The system is of a robust nature, with promise of providing enough compatibility and ease of use for further development, for both software and add-on hardware. At the completion and implementation of several linear effectors, the prototype in its current state seems capable of handling the new type of actuator without a change to the concept.

9.3 Recommendations

The current rotational core unit uses a gear and pinion mechanism, driven by a geared DC motor for actuation. Future iterations of the prototype could consider the investigation of switching to a stepper motor and a drive mechanism with lower backlash. For self-locking actuators, a worm-gear may be a good alternative.

Ideally, a harmonic drive assembly such as Figure 9 - 3 should be utilised to provide motion to the rotational actuator. The location of the encoder in Figure 9 - 3 will provide significantly improved positional repeatability and precision, given that there is no backlash introduced by the harmonic drive. A stepper motor may also provide improved holding torque.

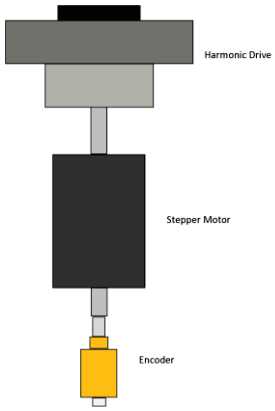


Figure 9 - 3. **Proposed harmonic drive assembly.**

A harmonic drive may only be an option with a large budget, as these products tend to be very expensive to manufacture. However, their design makes them ideal candidates as a driving mechanism for the rotational actuator, as they incorporate a *'through bore'* for wiring, very low gear ratios and no backlash. The FD harmonic drive series by HarmonicDrive provide a selection of units with gear reduction ratios that range from 1:78 to 1:320. Taking the 1:78 drive as an example; in the above configuration, a Polulu 9 kg-cm 1,200 rpm stepper motor could provide a holding torque of 702 kg-cm and achieve a full rotation in roughly 4 seconds. With a stepping angle of 1.8°, each full-step may actuate the output shaft by 0.0056°. If an MAE3 shaft encoder is used as per Figure 9 - 3, one could theoretically achieve an accuracy of $7.6e^{-7}$ degrees per encoder count at the output shaft.

If a stepper motor is chosen for future developments, it would be prudent to adopt a control methodology such as Figure 9 - 4. A velocity curve may be pre-calculated and compared with the actual actuator position, which may then take action to correct any variation from the desired curve.

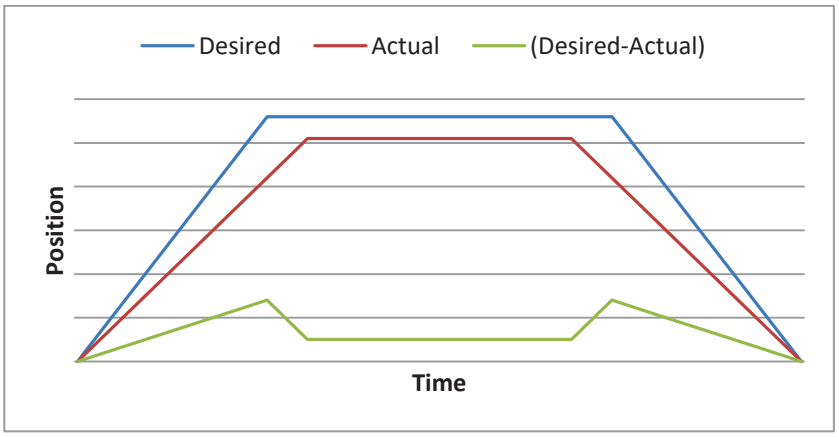


Figure 9 - 4. **Proposed stepper control curve.**

Future improvement of the prototype should focus on the introduction of fuzzy logic scheduling of proportional and integral gains in relation to the vertical position (fig. 9 - 1). This will significantly improve control algorithm performance for configurations that do not respond linearly. Several sets of fuzzy scheduling may be established and utilised depending on the specific configuration. It may also be worth investigating auto-tuning methodologies.

In the future, once an advanced GUI program has been implemented to work with the reconfigurable robotic manipulator, modular sliding kinematics should be developed. This could be done by calculating the reverse kinematic equations of each possible configuration of the arm. The master controller may then communicate the current configuration of the manipulator to software, which may then select the appropriate kinematic control to use.

Bibliography

Abu Talib, M.A., Khamarazaman, A.S., Salimun, M.Y., Jaafar, J., & Shauri, R. L. A. (2013). *PID position control for 2-DOF robotic finger* (pp. 152-157). Paper presented at the 2013 IEEE 4th Control and System Graduate Research Colloquium.

Åström, K. & T.Hägglund. (1995). *PID Controllers: Theory, Design, and Tuning* (2nd Ed). Durham, NC: The International Society for Measurement and Control.

Bennett, S. (1993). Development of the PID controller. *IEEE Control Systems*, 13(6), 58-62. doi:10.1109/37.248006

Bestic AB. (2015). *Manual For Bestic - Eating Assistive Device.* Retrieved from http://www.besticinc.com/wp-content/uploads/2015/12/PD2023_Manual-for-Bestic-2.4_EN_ver3.pdf

Bonev, IIian. (2001). *Delta Parallel Robot – the Story of Success*. Retrieved from http://www.parallemic.org/Reviews/Review002.html

Boole, G. (1847). *The mathematical analysis of LOGIC.* Retrieved from http://www.gutenberg.org/files/36884/36884-pdf.pdf?session_id=900fcfc45f3e3977c088e6142fa24c13856000c2

Cerecero-Natale, L. F., Ramos Fernández, J. C., Ramos-Velasco, L. E., Pedraza Vera, V. E. (2012). *Fuzzy Gain Scheduling PI Controller for a Mechatronic System* (Report No. 1569535951). Pachuca, Mexico: Centro de Investigación en Tecnologías.

Commonplace Robotics. (2014). *Robot Arm Mover6 - User Guide*. Retrieved from http://www.cpr-robots.com/ressourcen/UserGuide_Mover6.pdf

Commonplace Robotics. (2016). *Robot Arm Mover - Product Sheet*. Retrieved from http://www.cpr-robots.com/ressourcen/ProductSheet_Mover6.pdf

Hydzik, T. (2004). *Robotics315*. Perth, Australia: School of Mechanical Engineering, The University of Western Australia. Retrieved from http://thydzik.com/academic/robotics-315/index.php

Intelitek. (1996). *Scorbot-ER IX User's Manual 2nd Edition*. Retrieved from http://www.theoldrobots.com/book45/scorbot-ERIX.pdf

Intelitek. (2008). *Controller USB-Pro User Manual (Catalog #200029 Rev. B)*. Retrieved from http://www.intelitekdownloads.com/Manuals/Robots/ER-9%20PRO/200029B%20-%20Controller%20USB%20PRO.pdf

Intelitek. (2008). *Scorbot-ER 9PRO User Manual (Catalog #200034 Rev. A)*. Retrieved from ftp://ftp.robotec.co.il/Techsup/er9pro/Books/200034%20Rev%20A%20Scorbot%20ER%209Pro.pdf

Intelitek. (2009). *Scorbase Version 6.1 and Higher User Manual.* Retrieved from http://www.intelitekdownloads.com/Manuals/Robots/ER-14%20PRO/200030%20Rev%20B%20Scorbase-USBPRO_Print.pdf

International Federation of Robotics. (2012). History of industrial robotics. *IFR Germany*. Retrieved from http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

Kawasaki Heavy Industries, Ltd. Robot Division. (2015). *Kawasaki Robot Y Series High-speed picking robot.* Retrieved from https://robotics.kawasaki.com/userAssets1/productPDF/Y-Series.pdf

Kawasaki Heavy Industries, Ltd. Robot Division. (2015). *Kawasaki Robot R Series Small-to-medium payload robots up to 80 kg.* Retrieved from https://robotics.kawasaki.com/userAssets1/productPDF/Asia-Oceania/Kawasaki_R-Series.pdf

Kawasaki Heavy Industries, Ltd. Robot Division.(2015). *Kawasaki Robot Palletizing Robots.* Retrieved from https://robotics.kawasaki.com/userAssets1/productPDF/Europe-Africa/Kawasaki_Palletizing-Series.pdf

Kinovarobotics. (2014). *Jaco² User Guide.* Retrieved from http://kinovarobotics.com/wp-content/uploads/2015/04/JACO%C2%B2-User-Guide-Assistive-Robotics-2-0-2.pdf

Kinovarobotics. (2014). *Kinova Robotic Arms | Jacosoft User Guide.* Retrieved from http://www.kinovarobotics.com/wp-content/uploads/2015/04/Jacosoft-User-Guide.pdf

Kinovarobotics. (2014). *K-Series Actuators Technical Specifications.* Retrieved from http://kinovarobotics.com/wp-content/uploads/2015/02/K-Series_Technical-specifications-v1.0.21.pdf

Kinovarobotics. (2015). *JACO Assistive Robotic Arm e-Brochure.* Retrieved from http://www.kinovarobotics.com/wp-content/uploads/2015/11/JACO_Assistive_Seb_EN_CANADA_Final.pdf

Kinovarobotics. (2015). *KG-Series Grippers Technical Specifications.* Retrieved from Grippers - http://kinovarobotics.com/wp-content/uploads/2015/02/KG-Series_Technical-specifications-v1.0.pdf

Knowles-Cutler, A., Frey, C. B., Osborne, M. A. (2014). London Futures, Agiletown: the relentless march of technology and London's response. *Deloitte.* Retrieved from https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/uk-futures/london-futures-agiletown.pdf

Knowles-Cutler, A., Sproul, D., Lewis, H. (2015). From brawn to brains, the impact of technology on jobs in the UK. *Deloitte.* Retrieved from https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/Growth/deloitte-uk-insights-from-brawns-to-brain.pdf

Lehtla, T. (2008). 3. Path and Trajectory Planning. In Laane, M. A., *Introduction to Robotics* (pp. 69-96). Retrieved from http://www.ene.ttu.ee/elektriajamid/oppeinfo/materjal/AAR0040/03_Robotics.pdf

Mecademic. (2014). *Parallel robot for education and research Brochure.* Retrieved from http://www.mecademic.com/DexTAR_Brochure-EN.pdf

Mecademic. (2015). *Dual-arm SCARA educational robot*. Retrieved from http://www.mecademic.com/DexTAR.html

Meccano Ltd. (1938, March). An automatic block-setting crane. *Meccano Magazine, 23 (3),* 172. Retrieved from: http://www.mecademic.com/references/MeccanoMagazine1938.pdf

Medademic. (2015). *DexTAR User's Manual Version 1.1*. Retrieved from http://www.mecademic.com/downloads/UsersManual.pdf

Meier, M. (2016, January 11). *Robots*. Retrieved from http://mkmra2.blogspot.co.nz/2016/01/robots.html

Merabet, A. & Gu, J. (2010). *Advanced Nonlinear Control of Robot Manipulators.* In Lazinica, A. & Kawai, H. (Eds.), *Robot manipulators, New Achievements.* ISBN: 978-953-307-090-2.

Microstar Laboratories. (n.d.). *Ziegler-Nichols Tuning Rules for PID*. Retrieved from http://www.mstarlabs.com/control/znrule.html

Minorsky, N. (1922). Directional stability of automatically steered bodies. In J. Amer. Sov. Naval Eng. 34 (2), *Naval Engineers Journal* (pp. 280-309). doi:10.1111/j.1559-3584.1922.tb04958.x

Modrobotics. (2012). *Getting Started - Building Modular Robotics with Cublets.* Retrieved from http://www.modrobotics.com/cubelets/cubelets-getting-started/#16

Modrobotics. (2014). *Modular Robotics Press Kit*. Retrieved from http://www.modrobotics.com/press-kit/

Modrobotics. *Cublets Education*. Retrieved from http://www.modrobotics.com/education/?utm_source=SiteNav&utm_medium=NavClick&utm_content=Education&utm_campaign=Education

Modrobotics. *Cublets Studio*. Retrieved from https://www.modrobotics.com/cubelets/apps/cubelets-studio/

Naus, G. J. L. (2009). *Gain Scheduling Robust Design and Automated Tuning of Automotive Controllers*. Eindhoven, Netherlands: Department of Advanced Chassis and Transport Systems.

Novák, V., Perfilieva, I. (2001). *The principles of Fuzzy Logic: Its mathematical and computational aspects.* Ostrava, Czech Republic: Institute for Research and Applications of Fuzzy Modelling.

Passino, K. M., & Yurkovich, S. (1998). *Fuzzy Control*. Boston, MA: Addison-Wesley.

Robot Test Kitchen. (2014). *Review: Cublets*. Retrieved from http://robottestkitchen.com/2014/10/19/cubelets/

Seung-Min, B. & Tae-Yong, K. (1997). *An adaptive PID learning control of DC motors*. Paper presented at the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation.

Shauri, R. L. A., Salleh, N. M., & Hadi, A. K. A. (2014). *PID position control of 7-DOF three-fingered robotic hand for grasping task*. Paper presented at the 2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014).

Tabuchi, H. (2008, March 2). Japan looks to robot future. *NBC News.* Retrieved from http://www.nbcnews.com/id/23438322/ns/technology_and_science-innovation/t/japan-looks-robot-future/#.WHgBR1N96Ul

Zhao, Z. Y., Tomizuka, M., & Isaka, S. (1992). *Fuzzy gain scheduling of PID controllers*. Paper presented at the [Proceedings 1992] The First IEEE Conference on Control Applications.

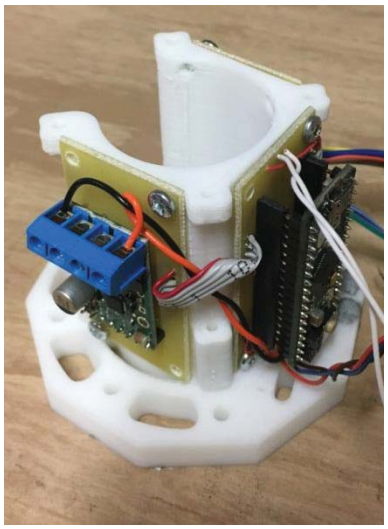
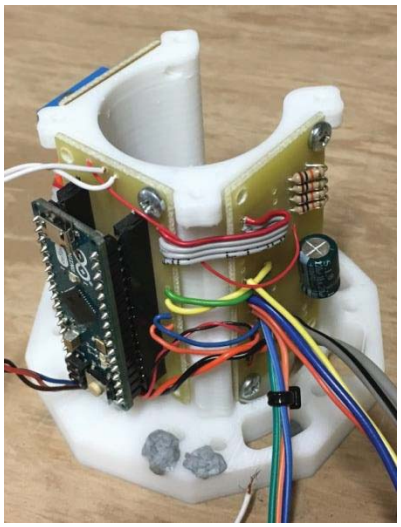
Ziegler, J.G., Nichols, N. B. (1942). Optimum Settings for Automatic Controllers. *Transactions of the A.S.M.E.* (pp. 759-765). Retrieved from http://abe-research.illinois.edu/Faculty/grift/ABE463_2016/Resources/ZieglerNichols.pdf

Appendix A Miscellaneous Pictures

A.1 Control Box




A.2 Electronics Stand

A.3 SCARA


A.4 PUMA