ANALYSIS OF FIELD-PROGRAMMABLE GATE ARRAY-BASED SAFETY SYSTEMS IN LIGHTWEIGHT APPLICATIONS

ENG470: Engineering Honours Thesis

James Hamilton
Executive Summary
The current approach to implementing safety measures for the purposes of preserving human-life and property as either hardware or software has a number of inherent limitations. This report explores and evaluates the use of the Field Programmable Gate Array (FPGA) available on the National Instruments myRIO-1900 platform as means of combining the strengths of both approaches, with the classical ball and beam control problem used as a basis of testing a number of common features of safety-systems.

Two different ball and beam configurations were used. During the prototyping phase, the beam was directly driven by a stepper motor while the final project apparatus utilised a standard servo and push-rod configuration. The change occurred due to a failure of the operational failure of the stepper motor. As a result, limited useful test data was gathered. However, feedback and analysis addressed a number of larger issues encountered during prototyping prior to the development of the final project build.

Results gathered during testing of the final prototype provided arguments for and against the use of an FPGA for the purpose of implementing safety systems. It was demonstrated that FPGA-based redundancy provides a means of reliably executing a multitude of time-dependent safety and monitoring processes at once.

However, it was also demonstrated that FPGA-based systems lack some important benefits associated with hardware and software systems. FPGA function complexity and size is quite limited and time consuming to implement, and the chip itself lacks the robustness commonly associated with hardware.
Acknowledgements

First and foremost, I would like to thank Dr Gareth Lee for his understanding and flexibility throughout the semester. I’d also like to thank Dr David Parlevliet for not only organising access to the 3D printing facilities on campus, but also providing some helpful advice to a newbie.

Next, I’d like to thank my younger brother Samuel. Without your emotional and technical support, I’m not sure this project would have ever been finished. Thanks for looking out for me.

Finally, I’d like to thank all of men and women I’ve had the pleasure of meeting and studying with over the last couple of years, particularly my friends over in control. Engineering hasn’t always been the easiest, but you guys sure made it a lot more fun.

-James Hamilton

November 2016
Contents

1.1. Safety-Instrumented Systems.................................................................15

Related Standards......................................................................................16

Components of a Safety-Instrumented System.............................................16

Software Redundancy..................................................................................16

Hardware Redundancy..................................................................................17

Limitations of Safety-Systems......................................................................17

1.2. The National Instruments myRIO-1900 Platform..................................18

Field-Programmable Gate Array .................................................................18

1.3. Stepper Motor Review...........................................................................19

Basic Operation............................................................................................19

1.4. Servo Motor Review...............................................................................20

Basic Operation and Components ...............................................................20

Common Types of Servo Motors.................................................................20

Standard Servos..........................................................................................21

Servo Motor Control....................................................................................22

1.3. Ball and Beam System.........................................................................22

2.1. System Modelling................................................................................24

Dynamic Models..........................................................................................24

Lagrangian....................................................................................................24

Newtonian Model........................................................................................25

Stepper Motor................................................................................................25
Model Simulation ............................................................................................................................................. 26

2.1. Control Development .................................................................................................................................... 27

Proposed Control Scheme ...................................................................................................................................... 27

Control Design .......................................................................................................................................................... 28

Proposed Stepper Motor Velocity Control ............................................................................................................. 28

Measured Variable Selection ..................................................................................................................................... 28

Position Control Simulation ...................................................................................................................................... 33

Closed-Loop Position Control Response .................................................................................................................. 33

3.1. Hardware Selection ......................................................................................................................................... 35

Ball Position Measurement ....................................................................................................................................... 35

Beam Angle Measurement ....................................................................................................................................... 36

Stepper Motor Selection .......................................................................................................................................... 37

Stepper Motor Interface Selection ........................................................................................................................... 38

CPU Simulation .......................................................................................................................................................... 40

3.1. Component Testing and Calibration .................................................................................................................. 41

Ball Position Sensor Testing ..................................................................................................................................... 41

Accuracy and Resolution Test Experiment ............................................................................................................. 41

Definitions ................................................................................................................................................................... 42

Experimental Setup – HC-SR04 ............................................................................................................................... 42

Experimental Setup GP2YOA21YKOF ..................................................................................................................... 44

Methodology .............................................................................................................................................................. 46

Results ......................................................................................................................................................................... 46
Deterministic Loop SubVIs .......................................................... 64
Sensor Calculation SubVIs ......................................................... 64
CPU subVIs .............................................................................. 65
Non-deterministic loop .............................................................. 66
FPGA Personality ...................................................................... 67
Majority Vote ........................................................................... 68
IO Monitor ................................................................................ 68
Stepper Indexing and Drive ...................................................... 69
Windows GUI ............................................................................ 69

3.3. Prototype Apparatus .............................................................. 70
Design and Construction of Beam Apparatus ......................... 70

3.4. Prototype Testing and Evaluation ....................................... 73
Redundancy Measure Test Design ........................................... 73
Test Results and Evaluation ..................................................... 75

3.5. Issues and Conclusions ....................................................... 76

4.1. Hardware Additions .............................................................. 77
16x2 Character LCD ................................................................. 77
Standard Servo Motor .............................................................. 78
Joystick and Pushbutton ........................................................... 79

4.1. Control Scheme Changes .................................................... 80
Control Scheme Reassessment ................................................. 80

4.2. Software Additions .............................................................. 81
Hardware Components .............................................................................................................. 81

FPGA LCD Driver ..................................................................................................................... 81

USB Data logging ..................................................................................................................... 83

4.3. Software Changes .............................................................................................................. 85

Control CPUs .......................................................................................................................... 85

CPU 1 Control .......................................................................................................................... 86

CPU 2 Control .......................................................................................................................... 87

CPU 3 Control .......................................................................................................................... 88

Redundancy Measures ............................................................................................................ 88

4.4. Final Project Design and Fabrication ................................................................................ 89

3D Designed and Printed Components ................................................................................. 89

Apparatus Construction .......................................................................................................... 92

Connection Hub ...................................................................................................................... 93

Circuit Design and Fabrication ............................................................................................... 94

4.5. Final Project Testing .......................................................................................................... 95

Experimental Design ............................................................................................................. 95

Tests Results and Evaluation ................................................................................................. 96

Modelling ................................................................................................................................ 102

Model Assumptions ................................................................................................................ 102

Parameters .............................................................................................................................. 102

Prototype Phase Closed-Loop Simulation Results .................................................................. 103

HC-SR04 Sensor Test Results ............................................................................................... 104
List of Tables

Table 1 Two-Phase On, Full Step [28] ........................................................................................................52
Table 3 Prototype Pin Mapping ..................................................................................................................73
Table 4 Majority Vote Prototype Test Results ..........................................................................................75
Table 5 Watchdog Prototype Test Results .................................................................................................75
Table 6 FPGA Redundancy Measure Performance Summary ......................................................................96
Table 7 Model Parameters ..........................................................................................................................102

List of Equations

Equation 1 Step Angle of Stepper Motor Equation .................................................................................20
Equation 4 Lagrangian of Ball and Beam System [15] .............................................................................25
Equation 5 Lagrangian Equation of Motion 1 [15] .....................................................................................25
Equation 6 Lagrangian Equation of Motion 2 [15] .....................................................................................25
Equation 7 Newtonian Beam Angle Ball Position Relationship [15] .........................................................25
Equation 8 Beam and Motor Shaft Relationship .......................................................................................25
Equation 9 Shaft Angle Change Steps Relationship ................................................................................26
Equation 10 Maximum Practical Speed of an On-line Algorithm Calculation [16] .................................31
Equation 11 Step time Reference Velocity Relationship [16] ..................................................................31
Equation 12 Stepper Motor Position Control (Continuous) .......................................................................33
Equation 13 Position Control Transfer Function ....................................................................................33
Equation 14 HC-SR04 Range Calculation .................................................................................................44
Equation 15 Single-Axis Angle Approximation .......................................................................................48
Equation 16 Dual-Axis Angle Approximation .........................................................................................48
Equation 17 Steps Per Revolution Calculation .......................................................... 53
Equation 21 lcd_move.vi Command Function ............................................................ 83

List of Figures

Figure 1 Triple Modular Redundancy [6] ......................................................................... 17
Figure 2 FPGA Structure [8] ....................................................................................... 18
Figure 3 Variable-Reluctance Principle Operation [9] ......................................................... 19
Figure 4 Standard Servo Motor Operation [12] ............................................................... 21
Figure 5 Ball and Beam System [14] ........................................................................... 23
Figure 6 Lagrangian Ball and Beam Free-Body Diagram [15] ............................................. 24
Figure 7 Ball and Beam Simulink Model ..................................................................... 26
Figure 8 Simulation Open-Loop Response .................................................................. 27
Figure 9 Proposed Closed-Loop Control Scheme ......................................................... 28
Figure 10 Velocity Control Sub-Process Fundamental Operation .................................. 29
Figure 11 General Operational Flow of On-line Algorithms Flow Chart [16]...................... 30
Figure 12 Proposed Motor Control Algorithm Process Flow [16] ...................................... 32
Figure 13 Position Control Response ......................................................................... 34
Figure 14 (Left) HC-SR04 [17] (Right) GP2Y0A21YK0F [18] ....................................... 36
Figure 15 Arduino Compatible Digital Tilt Sensor Module {Duinotech, 2013 #123} .......... 37
Figure 16 (Left) 28BYJ-48 (Right) YM2574 ................................................................ 38
Figure 17 ULN2003A Darlington Array Circuit Block Diagram [19] ................................. 39
Figure 18 XC4492 L298N Breakout Board [22] ............................................................. 40
Figure 19 HC-SR04 Test Program (FPGA Component) ................................................... 43
Figure 20 HC-SR04 Test Program (Real-time CPU Component) ................................. 43
Figure 21 HC-SR04 Component Wiring .................................................................... 44
Figure 22 GP2YOA21YKOF myRIO Main VI [25] .......................................................... 45
Figure 23 (Left) GP2YOA21YKOF Calibration subVI (Right) Voltage to Distance subVI [25] ............ 45
Figure 24 GP2YOA21YKOF Test Results .................................................................46
Figure 25 MMA7455 Resolution and Sensitivity Test Program ........................................49
Figure 26 10 Bit vs. 8 Bit X-Axis Resolution (8G) ..........................................................50
Figure 27 10 Bit vs. 8 Bit X-Axis Resolution (4G) ..........................................................50
Figure 28 10 Bit vs. 8 Bit X-Axis Resolution (2G) ..........................................................51
Figure 29 Stepper Driver Test Program .........................................................................52
Figure 30 myRIO-1900 RT CPU Program ...................................................................55
Figure 31 Digital Logic Majority Voter Test Program .......................................................57
Figure 32 Majority Voter Analogue Test Program .............................................................58
Figure 33 Analogue Sampling Test Program ....................................................................58
Figure 34 FPGA Watchdog Timer Program .....................................................................59
Figure 35 IO Watchdog LabVIEW Program ....................................................................60
Figure 36 Deterministic Loop Sensor Data .......................................................................62
Figure 37 Deterministic Loop Control Calculations .........................................................63
Figure 38 Deterministic Loop FPGA Feedback/Control Actuation ....................................64
Figure 39 MMA7455 True Case .......................................................................................65
Figure 40 Control CPU subVI .........................................................................................66
Figure 41 Non-deterministic Loop ...................................................................................67
Figure 42 FPGA Majority Vote .......................................................................................68
Figure 43 Stepper and Indexing Drive LabVIEW Program ...............................................69
Figure 44 Windows GUI Block Initial Values ...................................................................69
Figure 45 Windows GUI Main Program Loop ..................................................................70
Figure 46 L293D Mount .................................................................................................71
Figure 47 Prototype Apparatus Stepper-Axis Connection ...............................................72
Figure 48 Fail-state Control ............................................................................................74
Figure 49 Fail-state Feedback ..........................................................................................74
Introduction

Safety and Critical Control PLC’s form an integral part of Safety Instrumented Systems (SIS) in industrial processes, and are certified Safety-Integrity Level (SIL) rated, implemented and maintained in accordance to functional-safety standards IEC 61508 and 61511 [1, 2]. One common structure used by SIL systems is triple-modular redundancy, which ensures a greater degree of robustness by requiring control decisions to be made by three separately programmed entities.

However, these units are still subject to operability issues rooted in the implementation of purely hardware or software monitoring and redundancy measures. Hardware, while robust, is expensive and inflexible while software measures, although simply implemented, lack speed and are subject to common-mode failure. [3]

One solution to the aforementioned issues is the implementation of redundancy measures via field-programmable gate array (FPGA). An FPGA essentially offers a user access to a reprogrammable digital circuit. Software operations potentially gain a degree of robustness, as they become physically implemented as part of the circuit, whilst hardware redundancy measures are less inflexible, as changes can be made programmatically.

The National Instruments myRIO-1900 platform offers access to an FPGA/CPU hybrid chipset in the form of the Xilinx Zynq-7010 [4]. The advantage of using this platform is its compatibility with the LabVIEW System Design Software, that lessens the need to understand complex programming syntax by providing a graphical, block-based interface for the development and coding of engineering systems [5].

The purpose of this project was to explore the use of FPGA in triple-modular redundancy systems, by utilising the myRIO-1900 FPGA as part of a control system for a process, and testing its capacity to carry out tasks common to commercially available SIL systems.
Cursory research into an appropriate project topic indicated that, due to the hardware capabilities of the myRIO-1900 platform and the nature of systems that require the implementation of safety-systems, implementation of the unit on a large and/or dangerous process was unreasonable. Therefore, the focus shifted to demonstrating core functionality associated with triple-modular redundancy systems on a small and safe system.

After testing and partially developing a number of project-ideas, the final process that was selected was the ball-and-beam concept. Commonly used to demonstrate physical and control theories, the concept involves a ball balancing on suspended beam, with an actuator used to tilt the angle of the beam, with the intention to bring the ball to a referenced position.

The goal of this project was to design and implement redundancy features found in safety-systems onto the FPGA chip available on the myRIO-1900 platform, and to test the and evaluate the performance of an FPGA for this application using the ball and beam system as a test basis.
1. Literature Review

1.1. Safety-Instrumented Systems

Safety Instrumented Systems, or safety-related systems in general, are developed and implemented in accordance to core requirements and framework in the interest of functional safety defined by internationally recognised standard bodies, including the International Electrotechnical Commission (IEC).

It is first important to distinguish between the terms safety and functional safety. In general, Safety is defined by IEC as;

\[ \text{Freedom from unacceptable risk of physical injury or of damage to the health of people,} \]
\[ \text{either directly, or indirectly as a result of damage to property or to the environment. [2]} \]

Whilst functional safety is defined as;

\[ \text{part of the overall safety relating to the EUC (Equipment Under Control) and the EUC control system which depends on the correct functioning of the E/E/PE safety-related systems, other technology safety-related systems and external risk reduction facilities. [16]} \]

The difference between the two terms is primarily related to scope; That is, SIS isn’t developed and implemented with the intention of being solely responsible for operational safety of a particular process, as it depends on a system or piece of equipment operating correctly in response to its input. The purpose of SIS is the detection and mitigation or consequence reduction of hazardous events stemming from the process or equipment that it’s directly interfaced with, forming a larger framework of safety-related systems to ensure overall operational safety. [2]
Related Standards

Standardisation in engineering in general provides a means of unifying the approach to the development and implementation. To ensure SIS promotes functional safety, two key standards are considered; IEC 61508 (Functional safety of electrical/electronic/programmable electronic safety-related systems) and IEC 61511 (Functional safety - Safety instrumented systems for the process industry sector) [1].

The two standards are intermutually related. That is, IEC 61508 provides the generic framework and core requirements of a safety-related system and is generally associated more with product development by vendors, whilst IEC 61511 outlines good engineering practice for the implementation of functionally safe SIS in the process industry, targeted more at the party responsible for system implementation.

Components of a Safety-Instrumented System

Software Redundancy

Triple Modular Redundancy Systems

Triple Modular Redundancy (TMR) architecture is one commonly used type of N-modular, or voting, redundancy [6]. It’s commonly used in Safety PLC’s in the process industry, and promotes functional safety by assessing decisions made by a number of isolated parties prior to carrying out system changes.

The concept of TMR systems is demonstrated in Figure 1 below. In general, a TMR system consists of three circuits in parallel (in the case of Figure 1, these are labelled A, B and C) with the same input. The outputs provided by the circuits are compared by the voter, often referred to as the majority gate, which assesses these outputs and sides with the decision made by the majority of the circuits, and passes this output to the system (in the case of Figure 1, this is (0,1). [6]
Hardware Redundancy

Watchdog Timer Circuits

A watchdog timer circuit (WDT) in the case of a safety system, is an electronic timing circuit that monitors the execution of code to reset a processor if the software crashes [7]. Ideally, a WDT will operate completely independent to a CPU, in terms of both hardware and software.

The fundamental operation of a watchdog timer circuit is that, upon an event, a countdown from a preset time will begin. In the case that certain a certain trigger does not occur, and this timer is allowed to reach zero, an alarm is triggered. [7]

Limitations of Safety-Systems

Purely hardware or software based redundancy systems are historically subject to issues generally associated with either medium.

Traditionally, the most notable issue for software based safety-systems is their susceptibility to common mode failure. This occurs when multiple components of a system fail due to a single fault, such as power failure. [28]
Hardware based safety-systems are more robust, but are often more costly and difficult to change once implemented. [28]

1.2. The National Instruments myRIO-1900 Platform

Field-Programmable Gate Array

Field-programmable gate arrays (FPGAs), invented by the cofounder of Xilinx, are reconfigurable silicon chips. The chip itself is composed of a limited number of resources linked by programmable interconnects, as shown below in Figure 2 [8].

![FPGA Structure](image)

*Figure 2 FPGA Structure [8]*

The specification of the resources available on the FPGA vary. Generally, they will include;

- Configurable Logic Blocks (CLBs)
  - Composed of flip-flops and lookup tables (LUTs)
- Fixed Function Blocks (for example, multipliers)
- Size of Memory Resources (for example, embedded block RAM) [8].
An FPGA behaves a programmatically changeable digital circuit. Tasks and processes are carried out at the speed of an electrical signal, meaning it out performs traditional processors in terms of speed and power, and is even capable of performing true parallel processing. [4]

1.3. Stepper Motor Review

Basic Operation

Mechanical operation of a stepper motor occurs as the result of magnetic force induced with discrete electrical signals. The essential operation of a stepper motor is aided using an example of a typical stepper motor, a single-stack variable-reluctance motor, shown below in Figure 3 [9].

![Figure 3 Variable-Reluctance Principle Operation][9]

The example shown is composed of six salient poles and a rotor with four poles. Additionally, three sets of windings (are included in the configuration, each referred to as a phase [9].
Current is supplied from a DC source to the windings via the switches labelled I, II and III, exciting the phase. Progression through phases occurs by the excitation of phases in a particular sequence, or a step sequence. The operation relies on the principle that the rotor aligns itself in position where the magnetic flux is minimal whilst the magnetic flux produced is at a maximum. This ideal position is referred to as the equilibrium position. The direction of rotation can be altered by reversing this step sequence. [9]

The size of the rotation the stepper motor as it traverses between two equilibrium steps is referred to as step or stride angle. This value correlates to the number of stator phases and rotor teeth of the motor, as expressed below in Equation 1 [9].

**Equation 1 Step Angle of Stepper Motor Equation**

\[ \theta_s = \frac{360^\circ}{p \cdot N_r} \]

*Where:*

- \( p \) = number of phases
- \( N_r \) = number of teeth

### 1.4. Servo Motor Review

**Basic Operation and Components**

Commonly used in industrial applications, a servo motor combines a DC motor, potentiometer or positional cog and sensor, control circuit and reduction gearbox. The servo motor has three wires; positive voltage, ground and control signal. [10]

**Common Types of Servo Motors**

Servo motors for small-scale applications commonly come in two forms; standard and continuous rotation. The distinguishing feature between the two types is standard servos offer closed loop
position control over a predetermined range (usually $0^\circ$ to $180^\circ$), while continuous rotation servos offer open-loop speed control. [11]

**Standard Servos**

The basic operation of a standard servo motor is demonstrated below in Figure 4; as the shaft rotates, the positional cog and sensor (or potentiometer) provide feedback to the integrated control circuit on the position of the rotor [12].

![Figure 4 Standard Servo Motor Operation](image)

The ideal voltage characteristics of a servo motor are expressed in the equation below. Ideally, when the voltage is supplied to the motor, it will provide a constant torque to accelerate the rotor and, upon reaching the required speed, will drop to zero [13].

**Equation 2 Voltage Characteristics of Servo Motor** [13]

$$v = RI + C_m \omega_m$$
Under ideal circumstances, torque is proportional to current \( (I) \) and the motor torque constant \( (K_t) \) provided by the manufacturer [13], as shown in Equation 3 below.

**Equation 3 Servo Motor Torque Constant Equation [13]**

\[
T = K_t I
\]

**Servo Motor Control**

Modern standard servos feature digital control circuits that carry out positional control internally. Generally, control changes are communicated via a pulse-width-modulated (PWM) signal, commonly with a frequency of 50 Hz with the duty cycle varying between 5% and 10%.

Velocity control of a standard servo requires additional external circuit; as shown by Equation 2, where speed is proportional to applied voltage.

### 1.3. Ball and Beam System

The ball and beam system is a popular laboratory model used for teaching control systems engineering. The system provides a relatively simple basis for the development of controls for an open-loop unstable system. [14]

The configuration of the system varies, but the general operation is universal and demonstrated below in Figure 5 where a ball rolls along the top of a long beam. The beam is mounted to a shaft attached to a motor, and can be tilted around the centre axis. The purpose of the control in the system is to position the ball by altering the angle of the beam. [14]
Figure 5 Ball and Beam System [14]
2. Initial Development

2.1. System Modelling

Given the dynamics of the ball-and-beam system are well documented, the main concern of limited modelling with the goal of understanding the physical nature of the ball and beam system.

Dynamic Models

Lagrangian and Newtonian models describing key dynamic relationships within the system were first derived using referenced materials. For a list of parameters, symbols and assumptions, refer to Appendix A.

Lagrangian

The behaviour of interest is the relationship between the position of the ball (shown as $p(t)$ in the free-body diagram depicted in Figure 6 below) and the angle of the beam (shown as $\theta(t)$ in Figure 6). Ignoring frictional force, the ball rolls on the beam under the force of gravity, shifting position along the beam (indicated by $p(t)$ in the diagram below). External torque is applied by the stepper motor to tilt the beam (thus changing the angle) to control the position of the ball. [15]

![Figure 6 Lagrangian Ball and Beam Free-Body Diagram [15]](attachment://image.png)
The Lagrangian of a system is a quantity defined as the total kinetic energy of a system less the potential energy of a system [15]. The Lagrangian for the ball and beam system is shown below in Equation 4 (for a list of parameters, refer to Table 6 in Appendix A).

**Equation 4 Lagrangian of Ball and Beam System [15]**

\[ L = \left( \frac{1}{2} \right) \left( \frac{I_b}{r^2} + m \right) \ddot{p} + \left( \frac{1}{2} \right) (mp^2 + f) \dot{\theta} - m \cdot g \cdot p \cdot \sin \theta \]

Solving this gives two equations of motion for the ball and beam system, as shown below in Equations 5 and 6.

**Equation 5 Lagrangian Equation of Motion 1 [15]**

\[ \left( \frac{I_b}{r^2} + m \right) \ddot{p} + m \cdot g \cdot \sin \theta - m \cdot p \cdot \dot{\theta}^2 = 0 \]

**Equation 6 Lagrangian Equation of Motion 2 [15]**

\[ (m \cdot p^2 + f) \ddot{\theta} + 2 \cdot m \cdot p \cdot \dot{p} \cdot \dot{\theta} + m \cdot g \cdot p \cdot \cos \theta = \tau \]

**Newtonian Model**

The Newtonian relationship between the ball and beam angle and the position of the ball is expressed in Equation 7 below.

**Equation 7 Newtonian Beam Angle Ball Position Relationship [15]**

\[ \ddot{p} = \frac{5}{7} \cdot g \cdot \sin \theta \]

**Stepper Motor**

The relationship between shaft angle and beam angle is given by Equation 8, whilst the relationship between steps and change in beam angle is given by Equation 9.

**Equation 8 Beam and Motor Shaft Relationship**

\[ \theta_M = \theta \]
Equation 9 Shaft Angle Change Steps Relationship

\[ \theta_M = \frac{360^\circ}{S} \]

Model Simulation

Equation 5 was implemented as a subsystem in the Simulink software environment as shown in Figure 7 below.

![Figure 7 Ball and Beam Simulink Model](image)

Open-loop response, as shown below in Figure 8, was obtained using arbitrarily chosen parameters. The system exhibits the expected open-loop unstable response.
2.1. Control Development

As stepper motors are slow-actuating, at least in comparison to the ball, the control scheme needs to facilitate position changes ahead of changes in the measured position of the ball. It should be noted that the stepper motor position is directly related to the angle of the beam.

Proposed Control Scheme

The block diagram for the proposed scheme is shown below in Figure 9.
The scheme shown is purposely over-controlled and monitored with the intention of creating greater demand on the FPGA redundancy measures to be implemented. The reasoning was, by exposing the FPGA safety measures to as many variables as possible, greater amounts of experimental data would be made available for analysis.

Control Design

Once the underlying scheme had been selected, the controls were developed using the MATLAB and Simulink software packages for future implementation.

Proposed Stepper Motor Velocity Control

Measured Variable Selection

Velocity control of stepper motors for high-precision application is generally composed of two components; a velocity profile generator, which provides the blue-print outlining the required trajectory of a stepper in order to achieve desired displacement within an optimum time at a desired step size, an indexer (or velocity-to-time translator), responsible transforming the desired velocity profile into pulse-intervals that increment stepper rotor position and power drivers, responsibly for delivering these pulses as either current or voltage. [9]
In terms of this project, the myRIO-1900 platform will fill the roles of a velocity profile generator and indexer, whilst the L298N board will act as the power drivers. The fundamental relationships and a simplified version of the flow of operations to be implemented as part of the velocity controller sub process is shown in Figure 10 below.

![Figure 10 Velocity Control Sub-Process Fundamental Operation](image)

The objective of velocity control is to ensure total displacement is achieved in an acceptable time by minimising the time between steps. Velocity control schemes fall into two broad categories, on-line and off-line. Off-line schemes are carried out before stepper operation while on-line schemes are carried out during the operation of the stepper motor. [9]

On-line control was selected as the most suitable for this project. This choice was made as the implementation of off-line control would introduce the need for specialised hardware, detracting from the system responsibilities of the myRIO-1900 platform and ultimately the focus of the project.

The flow chart shown in Figure 11 below details the steps of the proposed on-line control solution. Function stems from two key components. The construction of the velocity profile ($V_r(k)$) and the calculation of the time between the current step and the next step ($\Delta t(k)$), or the indexer. For each
step, these values are recalculated, continuing until the total number of microsteps ($n$) are executed.

[9]

Figure 11 General Operational Flow of On-line Algorithms Flow Chart [16]
In general, the major draw-back of implementing an on-line algorithm is the additional demand placed on the control processor. Aside from requiring a dedicated timer per-controlled motor, the processing speed of the control imposes limitations on the actuation of the stepper motor. This relationship is expressed in Equation 10 Below, the summation of the time required to compute the algorithm \(T_c\) and the timer resolution \(T_r\) is inversely proportional to the maximum achievable speed of the stepper motor \(v_{\text{max}}\). [16]

**Equation 10 Maximum Practical Speed of an On-line Algorithm Calculation** [16]

\[
v_{\text{max}} = \frac{1}{T_c + T_r}
\]

The proposed control algorithms were formulated in direct reference to the works presented in the published article, *Novel Stepper Motor Controller Based FPGA Hardware Implementation* [16], with the following system parameters:

- \(V_r(k)\) = reference velocity at kth step
- \(\Delta t(k)\) = indexer value at kth step
- \(k\) = step

Assuming:

- \(\Delta t(k) = n_k \cdot T_c\)
- \(n_0 = 1\). [16]

The final algorithm is summarised by the process flow diagram shown in Figure 12; The algorithm incrementally tests values for variable \(n_k\) until an appropriate \(\Delta t(k)\) for the desired velocity \(V_r(k)\), whose mathematical relationship is expressed in Equation 11 below. [16]

**Equation 11 Step time Reference Velocity Relationship** [16]

\[
V_r(k) = \frac{1}{\Delta t(k)}
\]
However, as the resolution of $\Delta t(k)$ is equal to $T_c$, this relationship can’t practically exist due to the relationship demonstrated in Equation 10, instead becoming the comparison operation shown in Figure 12. If the case is true, a new step is executed and if the case is false, the $n_k$ value is incremented and the process is repeated. [16]

![Figure 12 Proposed Motor Control Algorithm Process Flow](16)
Proposed Stepper Motor Position Control

Given the relationship between the beam angle and shaft angle, proportional only control with a first order compensator was selected, with gain selected to be as large as possible without causing the system to go unstable.

The transfer function for the stepper motor control obtained using the simulated model is shown below in Equation 12.

*Equation 12 Stepper Motor Position Control (Continuous)*

\[
C_{SM}(s) = \frac{100(s + 0.01)}{s + 5}
\]

Proposed Ball Position Control

A proportional plus integral control with a lead compensator was chosen for the outside loop ball position control. The transfer function for the controller is given in Equation 13.

*Equation 13 Position Control Transfer Function*

\[
B_{BP}(s) = 65 \left( 1 + 0.6 \left( \frac{1}{s} \right) \right) \frac{(s + 0.01)}{s + 5}
\]

Position Control Simulation

Closed-Loop Position Control Response

The closed-loop response shown in Figure 13 below was obtained, with the beam angle shown in blue and the ball position shown in yellow. Analysis of the output reveals minimal offset, however, due to the speed of the stepper motor, the gain of the stepper position control would likely be greater.
Additional simulation data is available in Appendix A.
3. Initial Prototyping

The initial prototyping phase of the project included the selection and calibration of sensory and interface hardware, the selection and development of the core components of the real-time CPU software and FPGA personality and the testing and assessment of the appropriateness of each.

3.1. Hardware Selection

This section of the report explores the selection of apparatus and control components, including:

- Sensors
  - To measure beam angle and ball position

- Stepper motor and interface
  - From the minimum simulated requirements of torque and appropriate circuity to drive the motor

- Control CPUs
  - Units to provide appropriate hardware-in-loop simulation of the three control CPU’s that form part of a triple-modular redundancy safety system

Ball Position Measurement

Two sensors were considered to provide feedback on the position of the ball; the HC-SR04 ultrasonic (shown on the left in Figure 14) range finder and the Sharp GP2Y0A21YK0F IR Range Sensor (shown on the right in Figure 14).
Beam Angle Measurement

For this project, both analogue and digital sensors were considered. From these broad categories, two were selected; analogue, in the form of a rotary potentiometer attached to the axis connecting the beam and stepper motor, and digital, in the form of the MMA7455 accelerometer integrated circuit.

The MMA7455, in the form of the Duinotech XC-4484 breakout board shown in Figure 15, was selected without conducting further testing.
The potentiometer wasn’t chosen due to the nature of mechanical and analogue components: susceptibility to noise in both the field (as the electrical wiper scrapes when varying resistance) control (noise from the transmission wire and high impedance that will affect the take and hold capacitor that forms part of the ADC) and environmental changes and mechanical-ware meant, although simple to integrate, fine measurement would have been too difficult.

The shaft encoder was not chosen as it would directly provide feedback, whilst the MMA7455 would require a degree of interpretation on the control side of operation. It was reasoned this additional complexity could be used to create testable fault-states to measure the performance the FPGA redundancy measures.

Stepper Motor Selection

In order to narrow the scope in terms of motor selection, hybrid and variable reluctance motors were excluded. Hybrid stepper motors weren’t considered, as cursory research revealed, while meeting other project requirements in terms of torque and precision, they are far costlier than the other two types. Variable reluctance motors weren’t considered for the same financial reasons.

Next, the type of permanent magnet stepper motor, unipolar or bipolar, was considered. Whilst sharing the same fundamental operation, bipolar was selected. From cursory research, it was
discovered that bipolar stepper motors of comparable voltage ratings that generally provided more torque. However, it was discovered that they were also considerably more expensive.

Two options were reviewed, the Riorand 28BYJ-48 5VDC unipolar stepper motor (shown on the left in Figure 16) and the Riorand YM2754 unipolar motor (shown on the right in Figure 16).

![Figure 16 (Left) 28BYJ-48 (Right) YM2574](image)

**Stepper Motor Interface Selection**

Additional circuitry was required to translate PWM signals provided by the myRIO to movement in the motor. Given the comparatively large rated voltage of the YM2574, it would also require electrical isolation.

Two means of interfacing were considered. For the unipolar motor, the ULN2008A, in the form of the breakout board pictured below, was chosen. The IC is a high-voltage, high-current Darlington transistor array, consisting of 7 Darlington pairs, as illustrated by the simplified block diagram below in Figure 17.
A Darlington pair consists of two transistors connected together such that the amplified current signal from the first pair is amplified a second time, such that the circuit behaves like a single transistor with a gain equal to the product of the two gains [20]. This allows the comparatively small voltage signal generated by the myRIO-1900 platforms DIO to drive the much higher-current and higher-voltage unipolar windings of the Riorand 28BYJ-48 and YM2574.

For the bipolar motor, a simpler alternative to the many solutions available was considered, in the form of a H-bridge control circuit.

Two readily available components were chosen for comparison. The L298N dual full h-bridge IC and the L293D quad half h-bridge IC. The L298N was chosen, as part of the XC4492 breakout board shown below in Figure 18. The chip was far more suitable in terms of rated current output per coil, with the L293D limited to 600 mA (1.2 peak current), whilst the L298N can continuously output 2 amperes. This is important, mostly for the YM2574 motor, as the per coil operating current is approximately 1.2 A. Given that torque is proportional to the supplied current, failing to run the motor at its rated current would result in large losses of mechanical power. [21]
However, the L298N has drawbacks when compared to the L293D. Next, the L298N dissipates 25W of power (thus requiring a heatsink), whilst the L293D only dissipates 5W. Finally, the L293D offers internal circuit protection in the form of a totem-pole protection circuit, whilst the L298N requires external protection diodes. [21]

However, as power dissipation and the H-bridge configuration weren’t of primary concern for this project, and the fact that the breakout board provided protection, these issues were considered acceptable.

CPU Simulation

The relationship between the FPGA and myRIO-1900 real-time CPU (RT CPU), as described in the user manual [4] was considered a problem in terms of the project objective, as the FPGA could never operate independently.

Therefore, alternative means of simulating the control CPUs were considered as either of software-in-loop (SIL) and hardware-in-loop (HIL) simulation.
In terms of SIL, the use of an NI USB DAQ6008 card communicating with a desktop computer was considered. The DAQ would provide the means of interfacing a computer processor, which would then execute the decisions and provide the myRIO-1900 CPU component with decisions via UART.

The HIL alternative considered was a group of three Arduino Uno microcontroller units, whose CPUs would provide digital and/or analogue IO via actual output or serial (I2C or SPI) bus. The Arduino platform had a number of LabVIEW toolboxes available to either compile to or exploit the IO available on the Arduino Unos.

Minimum estimated hardware IO requirements meant the DAQ card was unsuitable, leaving the Arduinos as the next best choice, which also had a number of issues. While providing the means to physically implement a system that would more accurately reflect the multi-CPU configuration used in commercial control equipment, would require additional configuration and networking beyond the scope of this project.

3.1. Component Testing and Calibration

The purpose of carrying out testing on the components for the project was to discover the strengths and limitations of the choices and the potential effects on the performance of the project.

Additionally, early testing would indicate any special consideration needed in terms of calibration.

Ball Position Sensor Testing

Accuracy and Resolution Test Experiment

Testing was conducted in order to determine the accuracy and resolution of distance measurement provided by the HC-SR04 and GP2Y0A21YKOF modules in order to allow for more accurate dynamic simulation of the process and to provide calibration parameters that would eventually form part of the control program.
Definitions

Non-contact directional sensors geometry of the external world is determined by a measurement of angular range exclusively, within which a sensor precepts, referred to as field of view (FOV). This measurement is generally provided in the angular size of the conical space within this perceivable space as horizontal and vertical degrees. [23]

By definition, the accuracy of a measurement is defined as the closeness of a measured value to the real value, whilst precision of a measurement refers to the degree to which repeated measurements under changed conditions yield the same results. [23]

Experimental Setup – HC-SR04

The calibration test was physical setup as shown below in Figure 19; the sensor was placed a measured distance of 200 centimetres from and moved in 5 centimetre increments toward a flat surface.

The test program was composed of two parts, running separately on the FPGA and the real-time FPGA, as shown below in Figure 19, was executed on the FPGA. By utilising the FPGA, pulses and values could be sent and recorded in much finer detail. The bottom loop provides a pulse-width modulated signal to the unit, whilst the top loop measures the signal returned on the ECHO pin line and stores 10 values in a FIFO datablock.
The value returned is an unsigned 32-bit integer, on a scale from 0 to 4095 (a result of the 12-bit resolution of the DIO), which was passed to the real-time CPU program, shown below in Figure 20. The FPGA VI is opened and allowed to run, then the FIFO product of the last 10 measured values is averaged and converted to a centimetre measurement.
The user guide from Micropik [24] provides two means of calculating distance. This test program utilises the measured difference between the point at which the trigger signal is sent and the echo signal is received to make the calculation, as shown in Equation 14 below.

\[ R_{\text{Range ultrasonic}} = \frac{\mu S \text{ difference}}{58} \]

The sensor was connected according to the schematic shown below in Figure 21 below.

![Figure 21 HC-SR04 Component Wiring](image)

**Experimental Setup GP2YOA21YKOF**

The LabVIEW program adapted from the code provided as part of the myRIO Project Essentials Guide [25] materials to test the GP2YOA21YKOF module is shown below in Figure 22.
Figure 22 GP2YOA21YKOF myRIO Main VI [25]

Calibration values (shown on the left in Figure 22) and voltage to distance (shown on the right in Figure 23).

Figure 23 (Left) GP2YOA21YKOF Calibration subVI (Right) Voltage to Distance subVI [25]
Methodology

The sensors were both moved gradually further from a wall in measured increments, and given 30 seconds at each distance prior to results being recorded. This was to allow extenuating or immeasurable factors potentially influencing the reading to cease.

In order to judge the field of view of the HC-SR04 at various distances, a circular piece of cardboard with a radius of approximately 15 centimetres.

Results

The results of the GP2YOA21YKOF testing is summarized in Figure 25 below. The data indicates the most linear measurement is available between a distance of approximately 25 and 45 centimeters.

![Graph showing GP2YOA21YKOF Testing](image)

Figure 24 GP2YOA21YKOF Test Results

The recorded data (which can be found in Appendix B) provided the following corrected specification data for the HC-SR04;

- **Accuracy**
  - Absolute error = ~0.5 to -1.5 centimetres
  - Relative Error = 0 to 5%

- **Precision (average standard deviation)** = ~0.1-0.5 centimetres
• Field of View = 19.5 degrees (horizontal), 3.1 degrees (vertical)
• Spatial resolution = ~0.6-1.4º

Conclusions

The HC-SR04 provides relatively precise and accurate measurements of objects within a 60 centimetre range. However, the FOV will affect the relationship between the beam length and ball size, the relationship being approximately 4 to 1. For example, a beam of 1 meter will require a ball with a diameter of approximately 25 centimetres.

Next, the HC-SR04 provides a digital signal. The signal provided by the GP2Y0A21YK0F provides an analogue reference signal that exponentially reduces in precision as an object approaches the higher end of the available measurement range.

MMA7455 Testing and Calibration

Resolution and Sensitivity Testing

The purpose of conducting the calibration and testing experiment on the MMA7455 was to deduce the following;

• Reading offset requirement
• Appropriate resolution (8 or 10 bit)
• Appropriate sensitivity (2G, 4G or 8G)

Definitions

The tilt angle of the x-axis can be estimated using linear approximation based on the trigonometric relationship shown below in equation. [26]
Equation 15 Single-Axis Angle Approximation

\[ \theta = \sin^{-1} \left( \frac{A_{x\text{out}}}{1.0} \right) \] [26]

where:

\[ A_{x\text{out}} = x - \text{axis acceleration measurement} \]
\[ \theta = \text{degrees in radians} \]
\[ g = \text{gravity constant} \]

This approach requires a large amount of approximation. A more effective method of estimating the x-axis tilt angle is using both x and y-axis measurements as shown in the trigonometric relationship shown by Equation 16 below. [26]

This method is more effective because:

- The introduction of constant sensitivity introduced by the orthogonality of the two axes. As the sensitivity of one of the axes is reduced, such as when the acceleration of the axis reaches +1 or -1 g, the sensitivity incrementally increases for the other axis. [26]

- The reduced dependence on the alignment with the plane of gravity, as tilting on the y or z-axis when estimating with only the x-axis can cause significant error. [26]

Equation 16 Dual-Axis Angle Approximation

\[ \theta = \tan^{-1} \left( \frac{A_{x\text{out}}}{A_{y\text{out}}} \right) \] [26]

where:

\[ A_{y\text{out}} = y - \text{axis acceleration measurement} \]

Accelerometers offer varying degrees of axial sensitivity given as a scaling factor of the speed of gravity constant (for example, 2g, 4g and 8g in the case of the MMA7455). This provides a ratio of
the sensor's electrical output to mechanical input. In simple terms, the higher the value, the more sensitive the accelerometer becomes to changes in its immediate environment. [27]

Experimental Setup

The first portion of the experiment involved testing both the 8 bit and 10 bit X-axis register to determine which was most appropriate for the application. The test program, shown below in Figure 25, was written for the myRIO-1900 real-time CPU target using low level I2C blocks.

![Figure 25 MMA7455 Resolution and Sensitivity Test Program](image)

Methodology

Both resolution options were tested at each sensitivity rating for 60 seconds. During this period, the beam was rocked back and forth to its maximum tilt angle and held until the change was visibly registered on the VI graph.

These values were then exported to an excel spreadsheet, and the x-axis was linearly approximated.

Results

A summary of the results are presented below. First, the 8G comparison, summarised by the graph in Figure 26.
A comparison of 10 and 8 bit resolution at 4g is shown below in Figure 27.

Finally, the comparison of resolutions at 2G, as shown by the graph in Figure 28.
Conclusion

From the results, it was decided that the 10-bit X-axis register at 2G sensitivity was most appropriate for the application.

This was predominantly due to fact that the accelerometer data was to be used to estimate the tilt-angle of the beam; the higher resolution data at lower sensitivity allows for smaller incremental angle steps by which to estimate the tilt-angle of the beam.

Stepper Motor Testing

Testing was carried out to discover which of two selected stepper motors had electrical and mechanical characteristics most suited to the project.

The electrical and mechanical data can be found in Appendix B.
Test Program Design

In order to test the electrical and mechanical parameters of the motors, a test step program was developed for the myRIO-1900 platform using the program shown in Figure 32 below.

![Stepper Driver Test Program](image)

The user program utilises a state-machine programmatic approach. Each timed loop is executed in a predetermined order and can only be entered or exited given particular conditions.

Each motor was full-stepped; that is, following the full two phase on sequence shown by the truth table (Table 1) shown below.

### Table 1 Two-Phase On, Full Step [28]

<table>
<thead>
<tr>
<th>Step</th>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Mechanical Experiments and Results

Rotations per minute (RPM) was measured by marking a start position on both the shaft and face of the stepper motors, and recording the time taken to complete a single rotation with a stopwatch. The test program was executed at a rate of 330 PPM.

It was discovered the 28BYJ-48 was much slower, well over 1 minute to complete a revolution.

Conclusions

The 12VDC YM2574 completed revolutions much faster than the 28BYJ-48 motor. This is predominantly due to the gearing ratio of the 28BYJ-48 of 64:1; When full-stepped, the total number of steps the actual motor has to execute is much larger than the 48 steps required by the YM2574, expressed in Equation 17 below.

Equation 17 Steps Per Revolution Calculation

\[
\text{Steps per Revolution} = \frac{360^\circ}{\text{Stride Angle}} \times \text{Gear Ratio} = \frac{360^\circ}{11.625} \times 64 \cong 1982 \text{ Steps}
\]

Due to the fact that the YM2574 provides a fine enough step angle for the application with less steps per revolution, it was selected for the project.

From the modelling and component testing, a number of limitations provided the basis for the specifications of the final testing apparatus, including:

- HC-SR04 Field of View
  - The total length of the beam was limited to being approximately four times the diameter of the ball
- Total Torque Requirement
The required mechanical force to shift the beam couldn’t exceed the measured pull-in (stepping) torque of the YM2574 stepper motor (approximately 8.826 N.cm)

3.2. myRIO-1900 Development

Development of the required real-time CPU program started with the integration of the programs already developed during the testing and calibration stage of the prototyping phase, followed by the development of FPGA functionality in isolation.

Feature Identification and Initial Development

myRIO Real-Time CPU Software

Initially, the myRIO CPU program was structured as shown below in Figure 30. The original intention was to execute as much in terms of measurement and control processing on the RT CPU, with the time-imperative data processed in a timed-loop and control and actuation programming taking place in a secondary flat sequence.
It was discovered that structuring the program in this way meant changes to stepper position were slow. Additional attempts were made to separate the step actuation and indexing loop into a loop with higher priority and fast execution speed, but this approach also failed.

Therefore, it was decided that sensory measurements and time critical control process would be handled by the FPGA. The purpose of the myRIO-1900 RT CPU target would carry out control calculations provided by the FPGA, along with providing real-time feedback to the Windows UI.

**FPGA Custom Personality**

The role of the FPGA, aside from its regular duties to the myRIO-1900 CPU, was to provide safety-system like redundancy measures.

Additionally, as the HC-SR04 sensor program had already been developed for use as part of the FPGA personality, it was also included. Some attempt was made to also communicate with the MMA7455.
sensor via custom FPGA function, but the complexity of the I2C based communications proved too difficult.

**FPGA Personality Development**

Prior to developing the prototype FPGA personality, an understanding of the behaviour of the myRIO-1900 FPGA target and the relationship with the myRIO-1900 RT CPU was developed by developing smaller FPGA VIs in the same way the program for the ultrasonic sensor was developed. An FPGA personality determines the functionality and channels available on the myRIO-1900 unit [4].

Following this, two core redundancy measures were identified. The majority vote and watchdog systems. Test programs were then developed demonstrating the logic associated with these measures to ensure feasibility.

**Initial Test Programs**

A test program demonstrating the fundamental logic of the majority vote system, shown below in Figure 31, was developed to assess whether the concept was feasible.
The underlying operation, operating on DIO1 to DIO3, is as follows; If neither case is true, the value of DIO1 is passed on instead, with the case value both indicated and supplied via the Launcher IO monitor GUI. Operation was confirmed via a front panel indicator.

With the basic function of the majority vote tested, a slightly more complicated version, shown in Figure 32 was developed to assess the TMRs capacity to handle imperfect inputs (in this case, analogue voltages).
This program was then adapted to operate with analogue inputs, as shown in Figure 33. Three separate voltage sources, for the purpose of simulating independent analogue inputs were used.
The analogue inputs were varied individually across the full-range of voltage available in order to observe the ability of the majority voter to successfully carry out its function.

The majority vote required the addition of a range of operation, as even though the voltages were supplied from the same source, they were never exactly equal.

**Watchdog Timer Program**

A test FPGA program, shown in Figure 34 was developed to assess the capacity of the FPGA target to reset a failed CPU when needed. The FPGA program was designed to monitor a count value on the myRIO-1900 RT CPU VI. If the count value didn’t change within the specified period, the FPGA VI would trigger an alarm.

![Figure 34 FPGA Watchdog Timer Program](image)

The complimentary program running on the RT CPU would iterate the count at a manually changeable rate.

The acceptable count interval was altered to various values and the behaviour of the FPGA watchdog program was observed.
IO Watchdog

The purpose of the IO watchdog in safety-systems is to ensure hardware components, such as sensors or actuators, operate within predetermined parameters. The layout of the IO watchdog program is shown below in Figure 35.

![Sensor Watchdog Loop](image)

*Figure 35 IO Watchdog LabVIEW Program*

Completed Prototype Software

After the feasibility of the desired features of the prototype software was assessed, the components were combined to form single real-time CPU and FPGA personality programs. Additionally, a windows GUI was developed in order to provide network communicated feedback on the performance of the myRIO-1900 unit.
myRIO-1900 Real-Time CPU

The top level of the prototype program to execute on the myRIO platform is composed of two timed loops nested within a flat-sequence structure. This structure ensures that the FPGA personality is initialised before the real-time CPU begins executing.

The deterministic loop of the nested pair is responsible for all the major time critical operations to be carried out by the real-time CPU, including acquisition of data from the sensors, control calculation and actuation, whilst the non-deterministic loop is responsible for communicating local real-time with the Windows GUI via network variable.

Sensor monitoring and control action takes place on both the RT CPU and the FPGA. The HC-SR04s operation was left on the FPGA due to the time critical nature of its operation, whilst the indexing was also tasked to the FPGA for the same reason.

This section will summarise the operation of the program and provide detail about included subVIs not introduced prior.

Deterministic Loop Top Level Summary

The flow of the deterministic loop from left to right can be summarised as follows. Prior to entering the loop, I2C communications with the MMA7455 as introduced earlier in the report are established, followed by the acquisition of the raw HC-SR04 data. These values are then passed to subVIs responsible for mathematically translating these raw measurements to useable process variables (shown as the blue boxes labelled MMA7455 CALC and HCSR04 CALC in Figure 36 below).
These measured values are then passed to the simulated control CPUs for processing. The control parameters are provided by the windows GUI via the non-deterministic loop (shown in brown in Figure 37), whilst the control calculations are passed to both the windows GUI and FPGA (shown in blue in Figure 37) for the majority voting and watchdog processes.
Finally, feedback from the FPGA regarding the state of the system is given and passed to the windows GUI and the speed control is actuated according to the outcome of the majority vote using the myRIO PWM LabVIEW blocks, as shown below in Figure 38.
Figure 38 Deterministic Loop FPGA Feedback/Control Actuation

Deterministic Loop SubVIs

Sensor Calculation SubVIs

The MMA7455 and HC-SR04 calculation subVIs have two states determined by a case structure. The true case is designed to simulate component failure by returning a measurement result noticeably outside of the expected limits. As shown below in Figure 39, if the MMA7455 has ‘failed’ the returned tilt angle and x-axis acceleration will be -1000 and -100 respectively.
If the fail condition isn’t true, the subVIs carry out the calculations introduced in the testing section of this report.

**CPU subVIs**

The configuration of all three CPU subVIs, shown in Figure 40 below, is the same, following the control scheme introduced earlier in this section of the report, with position and speed control decisions passed back to the top level of the deterministic loop. Failure of each of the controls is simulated using the case structure approach found in the measurement subVIs.
Non-deterministic loop

The non-deterministic loop, as shown below in Figure 41, is to act as a mediator between the real-time CPU process and the windows GUI, by translating between real-time local and network published variables.
Aside from being responsible for the operation of the HC-SR04 sensor, the FPGA is also responsible for controlling and indexing the stepper motor position, monitoring sensor IO for faults and performing the majority vote operation.

Implementing the logic required by the redundancy measures and motor control required additional space to that which was available on the default personality, so functionality of the MSP connector was removed to make room. The MSP connector was chosen specifically as it doesn’t have the same complex interrupt driven relationship with the real-time CPU as the MXP connectors do, and required far less additional programming to remove.
Majority Vote

Figure 42 FPGA Majority Vote

IO Monitor
Stepper Indexing and Drive

Windows GUI

The Windows GUI VI provides an interface with the myRIO-1900 program for a user via HMI to monitor and set control values and trigger simulated failed components and controllers.

Figure 44 below shows the code inside of a stack sequence loop to execute prior to the main program loop. The purpose of this loop is to initialise the default states and values of the fail-state simulation and controls.
The main program loop, shown in Figure 45 below, displays and communicates networked variables entered by the user via a front panel.

![Figure 45 Windows GUI Main Program Loop](image)

### 3.3. Prototype Apparatus

#### Design and Construction of Beam Apparatus

The dimensions of the prototype apparatus were largely chosen to facilitate the length of the beam, as decided by the FOV of the HC-SR04 (approximately 300mm). The material used for the build was predominantly medium-density fibreboard (MDF) and treated pine. A ping-pong ball with a diameter of 40mm was used for the ball component of the build.

The L293D driving circuit was mounted to a piece of board as shown below in Figure 46, with communications to the myRIO-1900 being established via a jumper cable.
The stepper motor was attached directly to the axis of rotation, made from a threaded rod, with a tapped nut and screw, with the beam fixed in place with nuts either side, as shown below in Figure 47. Additionally, bearings were added to ensure smooth rotation.
Figure 47 Prototype Apparatus Stepper-Axis Connection
The connections between the hardware and myRIO-1900 are detailed in Table 2 below.

### Table 2 Prototype Pin Mapping

<table>
<thead>
<tr>
<th>MXP B Connections</th>
<th>L298N Connections</th>
<th>HC-SR04 Ultrasonic Sensor Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L298N Connections</strong></td>
<td><strong>MXP B Connections</strong></td>
<td><strong>HC-SR04 Pin</strong></td>
</tr>
<tr>
<td><strong>Pin</strong></td>
<td><strong>Function</strong></td>
<td><strong>L298N Pin</strong></td>
</tr>
<tr>
<td>11</td>
<td>DIO0</td>
<td>ENA</td>
</tr>
<tr>
<td>13</td>
<td>DIO1</td>
<td>IN1</td>
</tr>
<tr>
<td>15</td>
<td>DIO2</td>
<td>IN2</td>
</tr>
<tr>
<td>17</td>
<td>DIO3</td>
<td>IN3</td>
</tr>
<tr>
<td>19</td>
<td>DIO4</td>
<td>IN4</td>
</tr>
<tr>
<td>21</td>
<td>DIO5</td>
<td>ENB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MXP A Connections</th>
<th>MMA7455 3-Axis Accelerometer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pin</strong></td>
<td><strong>Function</strong></td>
</tr>
<tr>
<td>11</td>
<td>DIO0</td>
</tr>
<tr>
<td>12</td>
<td>DGND</td>
</tr>
<tr>
<td>13</td>
<td>DIO1</td>
</tr>
<tr>
<td>32</td>
<td>SCL</td>
</tr>
<tr>
<td>33</td>
<td>3.3V Supply</td>
</tr>
<tr>
<td>34</td>
<td>SDA</td>
</tr>
</tbody>
</table>

3.4. Prototype Testing and Evaluation

**Redundancy Measure Test Design**

As introduced, many of the system components include an extra case structure with the purpose of simulating a state of failure by returning unexpected values. Figure 48 below shows the control located on the front panel of the windows GUI to switch between these cases. Fail-state simulation was limited to Boolean operations during prototyping.
Feedback from the FPGA redundancies are provided by the windows GUI, as shown below in Figure 49.

During the testing of the redundancy features, results were recorded as fail or pass in an excel spreadsheet.
Test Results and Evaluation

The summary of the performance of the majority vote measure is shown below in Table 3. Results indicate that the FPGA provided correct feedback on the failure of a control at a rate of 87%.

### Table 3 Majority Vote Prototype Test Results

<table>
<thead>
<tr>
<th>Fail-State Condition</th>
<th>Test Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Stepper Motor Control 1</td>
<td>Pass</td>
</tr>
<tr>
<td>Stepper Motor Control 2</td>
<td>Pass</td>
</tr>
<tr>
<td>Stepper Motor Control 3</td>
<td>Pass</td>
</tr>
<tr>
<td>Speed Control 1</td>
<td>Pass</td>
</tr>
<tr>
<td>Speed Control 2</td>
<td>Pass</td>
</tr>
<tr>
<td>Speed Control 3</td>
<td>Pass</td>
</tr>
<tr>
<td>Ball Position Control 1</td>
<td>Pass</td>
</tr>
<tr>
<td>Ball Position Control 2</td>
<td>Pass</td>
</tr>
<tr>
<td>Ball Position Control 3</td>
<td>Pass</td>
</tr>
</tbody>
</table>

The performance of the IO watchdog measure is shown below in Table 4, indicating a success rate of 80%.

### Table 4 Watchdog Prototype Test Results

<table>
<thead>
<tr>
<th>Fail-State Condition</th>
<th>Test Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>MMA7455</td>
<td>Fail</td>
</tr>
<tr>
<td>HC-SR04</td>
<td>Pass</td>
</tr>
</tbody>
</table>

It should be noted that limited testing was conducted due to the failure of the stepper motor.
3.5. Issues and Conclusions

During the development of the prototyping apparatus, a number of issues stemming from the stepper motor presented themselves. The main issue with the choice of motor for the prototyping stage of development was the reaction speed required for the application. A number of steps were required at any given time to cope with changes in the position of the ball. Due to the method by which stepper motors rotate, the resulting current changes led to an exponential increase in the temperature of the motor coils, eventually resulting in permanent damage to the motor coils (as confirmed by ohmmeter). This led to a limited degree of control tuning during the prototyping phase.

The experimental data gathered indicates that the raw performance of the measures was quite high, but provides limited important contextual detail including the time to return a correct alarm, controller performance and ball position data, limiting the overall value of the data.

The testing of the FPGA personality and RT CPU software identified that the speed at which the program loops were processing was too fast. For example, the direction and number of steps to be taken as calculated by the controls were occurring too quickly for the hardware interface to make physical changes to the position of the motor.

Finally, the choice of materials for the prototype build were proven to be poor. The mass of the ping-pong ball used proved to be far too small to require quick control actuation. Additionally, the beam surface on which the ball rolled was uneven. On a number of occasions the ball would simply roll off the beam entirely, requiring the test to be restarted.
4. Final Project Build

After identifying the issues associated with the prototyping phase and receiving feedback, design and approach changes were made prior to the development of the final project build.

The main hardware changes included the 3D printed design and fabrication of the physical apparatus, the selection of a servo motor to replace the role of the stepper motor and the addition of an LCD screen and joystick for real-time feedback on FPGA operations.

In terms of the software, means of properly documenting experimental data and providing real-time indication of the performance of the FPGA were added.

4.1. Hardware Additions

16x2 Character LCD

The LCD breakout by Sparkfun, shown in Figure 50 below, was chosen following concern raised about the lack of visual feedback on the operation of the FPGA available on the initial prototype. The addition of an LCD in lieu of simpler visual feedback (for example, LEDs available on the myRIO unit) was made in order to give clearer and more detailed information.

The HD44780 parallel interface chipset was selected as a result of information gathered during cursory research. A set of drivers for use with the FPGAs available on a number of modules available from National Instrument were freely available through the VI Package Manager application [29].
Figure 50 HD44780 LCD [30]

Standard Servo Motor

The servo chosen was the RS-SS201-PG standard servo by Radient, shown below in Figure 50. It was chosen due to availability, desirable electrical characteristics (as it was able to operate on the 5 volt supply for the LCD) and mechanical characteristics (with a maximum rated torque of 3.5 kg-cm).
Joystick and Pushbutton

With the addition of the LCD, a method for navigating through the information provided by the FPGA was needed. After considering various packages and components, the Z6363 Joystick Module shown in Figure 51 below was selected, as it provided an integrated means of moving between and selecting objects available for display that was compatible with the voltage and IO available on the myRIO unit (utilising two analogue inputs, the 5V+ supply and a digital input).
4.1. Control Scheme Changes

Control Scheme Reassessment

The control scheme for the final project shown in Figure 53. The addition of a standard servo with on-board control makes the scheme simpler, but it was reasoned that the focus of the project is the performance of the FPGA in safety applications rather than the control development.

Additional modelling and simulation wasn’t carried out on the new process due the limited amount of time remaining for the project.
4.2. Software Additions

From feedback and issues discovered during prototyping, changes to the myRIO-1900 program were made to facilitate the new hardware, control scheme changes and log experimental data.

Hardware Components

FPGA LCD Driver

Drivers available from National Instruments [29] and developed by users [33] were used to develop the code required to communicate from the FPGA to the LCD screen without intervention from the RT CPU.

The final program is primarily based on the work of National Instruments user Ben_B [33], and is comprised of the same subVIs. The only difference is that the FIFO blocks used to store RT CPU values are instead calling values from the FPGA.

The drivers operation revolves around the `lcd_write` subVI, shown below in Figure 54. Using state-machine based programming it sequentially moves through the required steps to write characters to the LCD, which (in 4-bit mode) are;

1. Set RS Bit
2. Set the 4 digital outputs to the high half of the data byte
3. Strobe the enable (E) pin (data ready signal)
4. Set the 4 digital outputs for the low half of the data byte`
5. Strobe enable (E) pin again
6. Wait 5 milliseconds. [33]
Other subVIs take care of initialising, moving the cursor, converting characters an ASCII array and clearing the screen. [33]

In order to navigate this, an FPGA function, shown below in Figure 54, was developed to use work with the lcd_move subVI.

Figure 55 LCD Joystick Navigation FPGA Function
The purpose of this function is to increment either the row or column value when compared to the previous iteration to provide values for the command function written by `lcd_move`, as shown in Equation 18 below.

**Equation 18 `lcd_move.vi` Command Function**

\[
F = 128 + (\text{row number}) \cdot 40 + (\text{column number})
\]

Additionally, the momentary contact button that forms part of the joystick module was used to run the `clear_lcd` subVI, as shown below in Figure 56. The subVI causes the cursor to return to the home location (0,0). The FPGA function monitors the digital input at 1000 millisecond intervals and, on receiving a low logical input, initialises the subVI.

![FPGA Pushbutton Clear Function](image)

**Figure 56 FPGA Pushbutton Clear Function**

**USB Data logging**

Feedback and evaluation of the prototyping phase results indicated that some method of acquiring real-time information from the system and storing it for later analysis was required.

After reviewing the USB Flash Drive example provided in the National Instruments myRIO Project Essentials Guide [25], this feature was added. The additional code consists of three components;
first, information regarding the selected USB flash drive is received and displayed, as shown below in Figure 57.

The next component is responsible for building the file to which the data is to be recorded. In the case of Figure 58 below, process and TMR data is to be recorded in an excel (.csv) spreadsheet.

---

**Figure 57 USB Information Code**

**Figure 58 File Path Code**
The final component, shown in Figure 59 below, constructs an array from the indexed process data passed from the program loop, using the loop iteration as the data interval step which is then loaded to the Write Delimited Spreadsheet VI.

![Figure 59 Logged Data](image)

### 4.3. Software Changes

**Control CPUs**

Feedback and data gathered during the prototyping phase of the project identified the need to individually program each control CPU. The reasoning was twofold. First, it would more accurately reflect the procedure carried out on safety-systems in the field and second, it would provide a better basis of comparison when analysis test results.
Suggested methods of achieving this included;

- Programmatic approach
  - Using a combination of or purely textual/graphical based approaches to programming each control

- Control tuning
  - Take a more or less aggressive approach to tuning the control parameters of each CPU

- Control limits
  - Applying different saturation limits to control actuation

Control parameters and user implemented failure was still to be controlled from the windows GUI. In addition, measures were added so the FPGA would make some attempt to restart the failed control.

**CPU 1 Control**

The new control program for CPU 1, as shown below in Figure 60 follows a similar approach to use during prototyping by utilising built-in LabVIEW VIs, including the same Boolean case structures.

*Figure 60 CPU 1 Control Program*
This simpler control was to test the FPGAs capacity to cope with limited output ranges (the output of the inner controller was limited to less than the pulse width required to reach the maximum positional change) as well to act as a control group or basis for testing.

Additionally, lead compensation was intentionally excluded to make this control less robust than the other controllers. This was intended to introduce non-user operated control issues into the system.

**CPU 2 Control**

The CPU 2 Control subVI takes a text-based approach using formula nodes. The Ball position control is shown below in Figure 61.

```plaintext
/* BP PI Control */
if(iteration>10000)
{
    Kc=0.1;
    Ki=1000;
}
else
    err=sp-pv;
    errsum=err+errsum;
    Kc=p*err;
    Ki=errsum;
    Control=Kc*Ki;

/* Saturation Limit: */
if(sp>sp_max) sp=sp_max;
if(sp<sp_min) sp=sp_min;
```

*Figure 61 PI Formula Node Based Ball Position Control*

Included in this node is the loop-based control parameter failure a count of the loop iterations is passed from the main VI to the subVI, and after a set number of loops (in the case of Figure 61, 10000), the controller gain values revert to non-user specified values. Additionally, there are commented out options to add operational saturation limits, alter error calculations etc.
CPU 3 Control

The final control solution uses a combination of both LabVIEW formula nodes and LabVIEW control VI, as shown below in Figure 62 below.

Additionally, the pair of controls is a nested for loop that can be altered via the main program, in order to achieve the same.

Redundancy Measures

A slight change to the myRIO side of the redundancy process was made. Additional code was added to act on the majority voter to attempt to reset a control in the event it failed during the series of logic tests; if a control decision is rejected, a true signal is sent to the respective CPU.

The FPGA redundancy measures underwent minor alterations to suit the new system variables, and the IO monitoring functionality was expanded to include output ranges of the controllers.
4.4. Final Project Design and Fabrication

Due to the issues encountered in terms of the fabrication of the prototype apparatus, most of the physical apparatus was designed using the Fusion 360 software package from Autodesk [34] and printed using the 3D printing facilities provided by Murdoch University, excluding the base and servo motor push-rod.

The design was adapted from the ball-and-beam apparatus explored during the literary review detailed earlier in this report.

Additionally, a connection hub was designed and constructed using proto-board.

3D Designed and Printed Components

The 3D printed hardware was separated into two parts, the beam and the support arches. The arm component, shown below in Figure 63, was significantly shorter in terms of length to the prototype apparatus.
This was due to the limitations of the printers available (having a maximum area of 250mm by 250mm) and the performance of the HC-SR04 ultrasonic sensor over the length of the beam previously. Additionally, support bracketing for the HC-SR04 and MMA7455 units with cabling and a mounting point for the push-rod were included.

The supporting arches, shown below in Figure 64, were designed to allow the arm to sit and rotate, as well as to house the servo motor and provide a means of supporting the cabling required by the sensors. Holes in the bottom were included to provide a means of mounting the arches to the base.

![Figure 64 3D Arch Model](image)
Apparatus Construction

The 3D printed components were attached to a wooden base to form the final apparatus build, as shown below in Figure 65.

*Figure 65 Final Project Apparatus*

The HC-SR04 sensor (shown on the left of Figure 66) was fitted to the bracket designed, whilst the MMA7455 was permanently attached to the base of the arm using an acrylic adhesive (shown on the right of Figure 66).

*Figure 66 (Left) MMA7455 (Right) HC-SR04*
A push rod attached to the horn of the servo motor and a heavy-duty ball bearing was added to interface the servo and the beam, as shown in Figure 67 below.

![Figure 67 Final Project Push Rod](image)

**Connection Hub**

In order to provide a more permanent means of interfacing the myRIO with the apparatus, a connection hub was designed to connect via 34-way ribbon cable to the unit using the Proteus 8 software package and constructed using proto-board.
Circuit Design and Fabrication

The circuit, shown below in Figure 68, was first designed and tested using a solderless breadboard in such a way that all connections were localised to a single MXP header of the myRIO unit.

This arrangement provided a degree of flexibility in terms of the operation of the LCD in either 4 or 8-bit mode. Figure 69 shows the final proto-board that was fabricated.
4.5. Final Project Testing

Experimental Design

As with the testing conducted during the prototyping phase, the majority of the faults were user controlled through the windows GUI. The Boolean control shown in prototyping section had been adapted for changes, as shown below in Figure 70.

![User-Fault Control windows GUI](image)

*Figure 70 User-Fault Control windows GUI*

In addition, output range is also changeable via the panel, as shown in Figure 71 below.

![Output Range Control windows GUI](image)

*Figure 71 Output Range Control windows GUI*

IO Monitoring occurs on the LCD screen and is logged in real-time to the USB device and stored in an excel spreadsheet.
Tests Results and Evaluation

The addition of servo motor (although control parameters required retuning) added a greater degree of stability to the system, and allowed lengthier single-tests to be conducted. This provided the opportunity to test and record all the engineered faults, and observe the performance of the FPGA in terms of monitoring and control. It was also observed that the reduction in loop speed eliminated much of the data loss experienced during

A summary of the logged data regarding fail-state results is shown below in Table 5.

Table 5 FPGA Redundancy Measure Performance Summary

<table>
<thead>
<tr>
<th>Pass (%)</th>
<th>Control Communication Failure</th>
<th>Control Communication Failure</th>
<th>IO Range Failure</th>
<th>Controller Range Failure</th>
<th>Loop Dependent Control Failure</th>
<th>IO Component Failure</th>
<th>IO Calculation Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>89</td>
<td>68</td>
<td>87</td>
<td>74</td>
<td>61</td>
<td>63</td>
<td></td>
</tr>
</tbody>
</table>

The IO related values performed approximately 10% worse than each of the other redundancy measures. After examining the program and not finding any obvious issues, the reason behind this remains unclear.

Additionally, the success of the FPGA in resetting the controller was approximately 28%. This may be due to programming, as failure messages observed via LCD were observed more often.

In terms of the finite scope of this experiment, the FPGA successfully identified simulated fault conditions common to a range of processes.
5. Conclusions

The ball-and-beam model provides a useful basis for testing the designed FPGA redundancy measures as it provides the opportunity to observe the performance of safety measures and redundancies on a process with relatively complex dynamics without requiring equipment that is unfeasibly hard to acquire or manufacture.

Generally, during the testing conducted on the final ball and beam system, the FPGA performed quite well in terms of safety monitoring and control. However, the limitations of the myRIO-1900 unit meant the experiments conducted were flawed; that is, although the testing carried out on the final build was more rigorous, and included a greater variety of programmatic structures, the lack of the electrical robustness of the myRIO unit and the consequent limitation of not testing actual hardware faults means that any conclusions made about the performance of the FPGA are speculative.

The FPGA redundancy system built for this project demonstrated a number of characteristics, including the ability to be incredibly quick in reacting to a fault, the capacity to operate (reasonably) independently of the software-based controls of the system and the capacity to process and monitor many different tasks in parallel.

However, in addition to being delicate, the complexity of the programs that could be developed was quite limited and costly in terms of time. Most of the programs developed for this project for the FPGA were a combination of Boolean logic.

In conclusion, the results gathered during the course of this project indicate that, given its characteristics, an FPGA could have some role in a larger safety-system, but, due to the mentioned issues, it wouldn’t be ideal to run as a standalone unit.

Future work to be carried out in projects similar to this one could include expanding the test scenarios and apparatus to include more real elements such as power supply interruption.
References


Appendix A – Modelling and Simulation

Modelling

Model Assumptions

- Closed system
  - The system was considered bound, with every input and output known
- Frictional and slipping forces negligible
  - Both of these forces between the ball and beam surfaces and axis to be driven by the stepper motor were ignored, with the intention of minimising their real-effect during apparatus design and fabrication

Parameters

*Table 6 Model Parameters*

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>Ball position</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Beam angle</td>
</tr>
<tr>
<td>$m$</td>
<td>Mass of ball</td>
</tr>
<tr>
<td>$g$</td>
<td>Gravity constant</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Torque</td>
</tr>
<tr>
<td>$J$</td>
<td>Moment of Inertia of the Beam</td>
</tr>
<tr>
<td>$J_b$</td>
<td>Moment of Inertia of the Ball</td>
</tr>
</tbody>
</table>
Prototype Phase Closed-Loop Simulation Results

Figure 72 Simulink System Model Bode Plot
Appendix B – Additional Sensor Test Data

HC-SR04 Sensor Test Results

Figure 73 HC-SR04 Measurement Test Average Absolute Error

Figure 74 HC-SR04 Standard Deviation in Readings
Figure 75 HC-SR04 Raw Average Relative Error