

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Integrated Sensor and Controller Framework

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

Master of Engineering

In

Information and Telecommunications Engineering

At Massey University, Palmerston North, New Zealand

Ryan David Weir

2007

Abstract

This thesis presents a software platform to integrate sensors, controllers, actuators and instrumentation within a common framework. This provides a flexible, reusable, reconfigurable and scalable system for designers to use as a base for any sensing and control platform.

The purpose of the framework is to decrease system development time, and allow more time to be spent on designing the control algorithms, rather than implementing the system.

The architecture is generic, and finds application in many areas such as home, office and factory automation, process and environmental monitoring, surveillance and robotics.

The framework uses a data driven design, which separates the data storage areas (dataslots) from the components of the framework that process the data (processors). By separating all the components of the framework in this way, it allows a flexible configuration. When a processor places data into a dataslot, the dataslot queues all the processors that use that data to run.

A system that is based on this framework is configured by a text file. All the components are defined in the file, with the interactions between them. The system can be thought of as multiple boxes, with the text file defining how these boxes are connected together. This allows rapid configuration of the system, as separate text files can be maintained for different configurations. A text file is used for the configuration instead of a graphical environment to simplify the development process, and to reduce development time.

One potential limitation of the approach of separating the computational components is an increased overhead or latency. It is acknowledged that this is an important consideration in many control applications, so the framework is designed to minimise

the latency through implementation of prioritized queues and multitasking. This prevents one slow component from degrading the performance of the rest of the system.

The operation of the framework is demonstrated through a range of different applications. These show some of the key features including: acquiring data, handling multiple dataslots that a processor reads from or writes to, controlling actuators, how the virtual instrumentation works, network communications, where controllers fit into the framework, data logging, image and video dataslots, timers and dynamically linked libraries. A number of experiments show the framework under real conditions. The framework's data passing mechanisms are demonstrated, a simple control and data logging application is shown and an image processing application is shown to demonstrate the system under load. The latency of the framework is also determined. These illustrate how the framework would operate under different hardware and software applications. Work can still be done on the framework, as extra features can be added to improve the usability.

Overall, this thesis presents a flexible system to integrate sensors, actuators, instrumentation and controllers that can be utilised in a wide range of applications.

Acknowledgements

I would like to thank my supervisors – Gourab Sen Gupta and Donald Bailey. They have put a lot of effort in directing my thesis to where it is now.

I would also like to thank friends and family for supporting me or providing assistance – especially the other postgraduate students. Your help and support is appreciated.

List of Figures

Key for Figures

Below in Figure 1 is a key for most of the figures in this thesis.

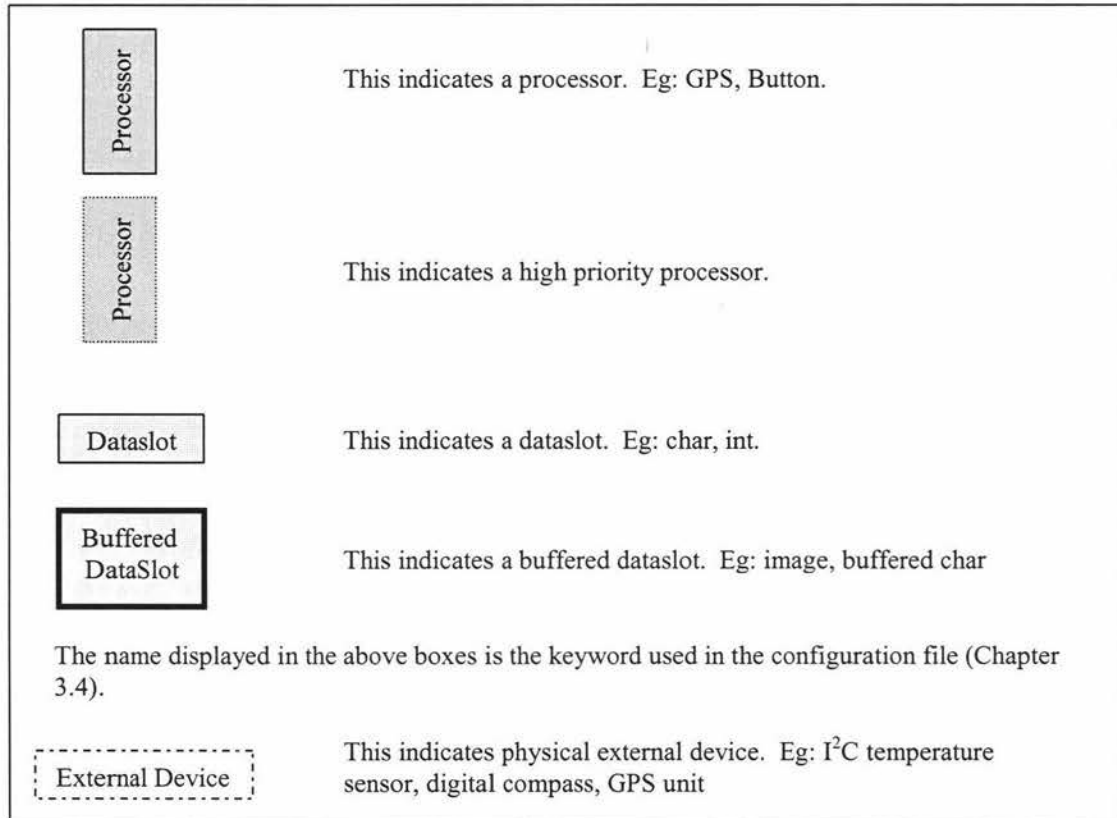


Figure 1: Key for most figures in this thesis.

Figure 1: Key for most figures in this thesis.....	v
Figure 2: Controller centric system.	17
Figure 3: Distributed control system.	18
Figure 4: Model of a framework that controls sensors and actuators showing the controllers outside the core of the system.....	18
Figure 5: Model of a framework that controls sensors and actuators showing both data storage and controllers outside of the core of the system.	19
Figure 6: Data driven design for an integrated sensing and controlling framework. This shows one complete task. One or more processors can be processing at the same time.	20
Figure 7: Class hierarchy of various Dataslot types.	24
Figure 8: Processor types grouped by type of operation that they process.....	26
Figure 9: Showing the operation of the program with multiple queues at different priority levels.	28
Figure 10: GPS processor and dataslot configuration.	29
Figure 11: Configuration file for a GPS device attached to the serial port.	31

Figure 12: Class structure of the framework. Used in the parsing of the configuration file. The keywords that each class recognises are stated.	33
Figure 13: Example dataslot to demonstrate parsing.	33
Figure 14: Initial processor with multiple dataouts.	38
Figure 15: An example of DATAOUT# being used in a configuration file.	39
Figure 16: A string from a GPS device in NMEA format.	39
Figure 17: Configuration file code to create a window.	41
Figure 18: Demonstration of a timer with an ultrasound sensor attached to the serial port.	43
Figure 19: Operations on the server and client end when a client connects to a server.	45
Figure 20: Data transference across a TCP/IP network happens through the network processors. The sending network processor acquires the data from a dataslot, while the receiving network processor places the data in a dataslot which has the same identification.	46
Figure 21: Example data logging application showing a buffered dataslot.	48
Figure 22: How to create a buffered char dataslot in the configuration file.	48
Figure 23: Security system showing frames being captured from a FireWire camera then placed into an image dataslot. The video display processor then displays this frame. The image is also taken and a image processing operation is done on the frame. From there the new image is displayed.	50
Figure 24: Screen shot of the display while the indicator is on.	52
Figure 25: Interaction of dataslots and processors for a simple button/indicator.	53
Figure 26: Configuration file for simple button/indicator.	54
Figure 27: Screen shots of the server and client windows showing the button and indicators. The indicator is on the server to check that the indicators on both computers are either on or off at the same time.	55
Figure 28: Interaction of dataslots and processors for the networked button and indicator. Both the server and client systems are shown.	56
Figure 29: The server end of the networked button/indicator application is setup using this configuration file.	57
Figure 30: Configuration file used to setup the client side of the networked button/indicator application.	58
Figure 31: How the brewery temperature controlled system was setup.	59
Figure 32: System design for the brewery temperature control application.	60
Figure 33: Screen shot of the window used in the brewery temperature control system.	61
Figure 34: Graph of vat temperature over 2 days.	62
Figure 35: Zoomed in image of the first hour and a half.	63
Figure 37: Displaying the FireWire video and colour inverted videos. The video in the top left corner is the video from the FireWire camera. The top right video is the top left video inverted. The bottom left video is the video in the top right inverted. The last video in the bottom right is the video in the bottom left inverted.	67
Figure 38: Comparison of the frame rate being stored in the first image dataslot and the frame rate coming out, with the cumulative number of dropped frames in the same dataslot.	68
Figure 39: System configuration during testing to see how fast a hard coded example can run.	70
Figure 40: System configuration during testing of latency using the framework.	70

Figure 41: Control flow for the robot soccer application (for clarity, the user interface components have been omitted).	74
Figure 42: How a PC running the framework would connect to devices it controls.....	76
Figure 43: Schematic to connect the I ² C bus to the parallel port.	82
Figure 44: Schematic to connect the heating element to the parallel port.....	83
Figure 45: Calculating resistor values and checking that parameters stay within devices limits.	84
Figure 46: A string from a GPS device in NMEA GGA format. This string is from (TELETYPE GPS 2007).....	85
Figure 47: Packet structures being transmitted over the network.....	87
Figure 48: Graph of temperature next to vat over 10 hours.....	88
Figure 49: Configuration file for the brewery temperature controller.....	92
Figure 50: Configuration file for the image processing demonstration showing what happens when the system is overloaded.	98
Figure 51: Configuration file testing the latency of the parallel port driver.....	99
Figure 52: Configuration file testing the latency of a complete sensing and controlling program.....	99

List of Tables

Table 1: A table of common software architectural styles..... 13

Table 2: Feature comparison between the proposed framework, and other platforms
for controller implementation..... 22

Table 3: Table showing what buffers are read from and written to under certain
conditions. 49

Table 4: Example GPS string broken down into its components. This table shows the
components of a GGA string. This example was taken from (TELETYPE
GPS 2007). 85

Table of Contents

Abstract.....	ii
Acknowledgements.....	iv
List of Figures.....	v
Key for Figures	v
List of Tables	viii
Table of Contents.....	ix
1 Introduction.....	1
1.1 Sensing and Control Applications	1
1.2 Framework Requirements.....	6
1.3 Previous Research.....	7
1.4 Outline of this Thesis.....	11
2 Framework for Sensors and Controllers	13
2.1 Architectural styles	13
2.2 Development of the Framework	17
2.3 Hardware and Software platforms	21
2.4 Summary.....	23
3 Component Mechanisms.....	24
3.1 Dataslots.....	24
3.1.1 Types of Dataslots	24
3.1.2 Operation of a Dataslot	25
3.2 Processes.....	25
3.2.1 Types of Processors	25
3.2.2 Operation of a Processor.....	26
3.3 Queues	27
3.3.1 Operation of a Queue.....	27
3.4 Configuration.....	28
3.4.1 Configuration File.....	29
3.4.2 System Configuration	32
3.5 Summary.....	34
4 Application Details	36
4.1 Sensors.....	36
4.1.1 Parallel port.....	37
4.2 Data Converters	38
4.3 Actuators.....	40
4.3.1 Switching a device.....	40
4.4 Controllers	40
4.5 Virtual instrumentation	40

4.5.1	Input	41
4.5.2	Output	41
4.6	Timers	42
4.7	Networking	43
4.8	Data Logging	47
4.9	Image Processing	48
4.10	Dynamically Linked Libraries	50
4.11	Summary	51
5	Demonstration Systems	52
5.1	Simple Button/Indicator	52
5.2	Networked Button/Indicator	54
5.3	Brewery Temperature Controller	58
5.4	Image Processing	63
5.5	Latency	69
5.5.1	Driver only	69
5.5.2	Driver and framework	70
5.6	Summary	71
6	Discussion	72
6.1	Other Applications	72
6.1.1	Hardware	72
6.1.2	Robot Soccer	73
6.1.3	Home Automation	74
6.2	Future Work	76
6.3	Conclusion	77
7	Authors Publications	80
8	Appendices	81
Appendix A	81
Appendix B	83
Appendix C	85
Appendix D	86
Appendix E	88
Appendix F	89
Appendix G	93
Appendix H	99
9	References	100