Final Group Report for the Development of an Autonomous Robot

Report by

Cameron A. Craig

Aidan P.J. Gallagher

Euan W. Mutch

Andrew J. Rigg

Duncan A. Robertson

Stuart J. Thain

In Partial Fulfilment of the Requirements

for the Degree of

MEng Computing and Electronics

Heriot-Watt University

Riccarton, Edinburgh

2016

(Submitted May 4, 2016)

Authors

Cameron A. Craig

cac30@hw.ac.uk

Aidan P.J. Gallagher

apg30@hw.ac.uk

Euan W. Mutch

em232@hw.ac.uk

Andrew J. Rigg

ar339@hw.ac.uk

Duncan A. Robertson

dar30@hw.ac.uk

Stuart J. Thain

st298@hw.ac.uk

I always have a quotation for everything - it saves original thinking.

- Dorothy L. Sayers

Abstract

This report describes the development of an autonomous vehicle, Tiberius. The Tiberius development team currently comprises of six students studying MEng Computing and Electronics at Heriot-Watt University, Edinburgh.

The Tiberius robot was first envisioned in 2005 by Dr Jim Herd, and initial development work was carried out as part of a MSc project that same year. At that time the mechanical structure of Tiberius was a modular articulated vehicle with two wheels per car. Each car would be electronically detachable at the articulation point to allow each car to move independently using self-balancing control loops. This design proved too difficult to implement mechanically, given the resources available at the time.

This year, Tiberius has been re-designed from a structural and software design perspective. The starting point for this year's development iteration was a four wheeled fixed-axle structure, with incomplete integration between hardware devices, and Tiberius's Python software. We now have a completely new mechanical design, with four independently steerable wheels, and suspension.

This project has benefited greatly from the latest rapid prototyping technologies - for mechanical prototyping as well as software prototyping. 3D printing was used heavily throughout the development cycle, allowing a complete mechanical prototype to be assembled within the limited six month time frame. A continuation of software development in Python meant that existing code could be re-used and re-factored into our new software suite.

This report describes in-depth the design and implementation of Tiberius, as well providing relevant background theory. Reflections are made on the execution of the project from our point of view in our Conclusions, as well as descriptions of future areas of improvement and future work.

Contents

List of Figures

List of Tables

Chapter 1

Mechanical

1.1 Introduction and Planning

1.2 Background Theory

1.3 Design and Implementation

1.4 Verification and Validation

1.5 Conclusions and Future Work

Chapter 2

Communications

2.1 Introduction and Planning

2.2 Background Theory

2.3 Design and Implementation

2.4 Verification and Validation

2.5 Conclusions and Future Work

Chapter 3

Power Management

3.1 Introduction and Planning

3.2 Background Theory

3.3 Design and Implementation

3.4 Verification and Validation

3.5 Conclusions and Future Work

Chapter 4

Robotic Arm

4.1 Introduction and Planning

4.2 Background Theory

4.3 Design and Implementation

4.4 Verification and Validation

4.5 Conclusions and Future Work

Chapter 5

Navigation

5.1 Introduction and Planning

5.2 Background Theory

5.3 Design and Implementation

5.4 Verification and Validation

5.5 Conclusions and Future Work

Chapter 6

Web Interface

6.1 Introduction and Planning

6.2 Background Theory

6.2.1 Overview

This section is designed to present related background theory to assist in the understanding of how the web interface works. The reader will become familiarised with the key technologies behind the web interface, to allow a greater understanding of how the components interacts with Tiberius and the user.

The technologies described in this sections are kept sucinct. An overview of the technologies discussed is given below:

High Level Technologies

MVC is a the Model-View-Controller design pattern, and is most commonly used in web frameworks such as Django.

Django is a popular web framework, used for small scale application such as this one, to large commercial applications.

Laura some detail

Low Level Technologies

Kate some detail

Christina some detail

Laura some detail

6.3 Design and Implementation

6.4 Verification and Validation

6.5 Conclusions and Future Work

Appendices

Appendix A

Comparison of IMDB Implementations

A.1 Overview

Tiberius's In-memory Database is a critical area of our architecture. Two decisions regarding the architecture were which RPi to use and which database to use. A script was created to measure the speed at which different configuration could carry out operations. This script created a database, inserted arbitrary values, updated the values, queried the values and then deleted the database. All of these functions were carried out 1000 times and then an average was taken.

A.2 Test Results

A.2.1 Create

Each database must be created before any operations can be carried out on it. The creation if a one off event that occurs at the start, therefore the time delay does not have significant importance.

Table A.1: Create database

	RP1	RP2	RP3
Poly	0.00338196		
SQL	0.03206205		

A.2.2 Insert

New data received from the sensors must be inserted into the database to be stored. This action is done regularly and there the time delay is important.

Table A.2: insert into database

	RP1	RP2	RP3
Poly	0.00236365		
SQL	0.032062053		

A.2.3 Update

On certain occasions data in the table may need to be updated. This differs from inserted information as a new row of information is not appended however an existing row is searched for and it values are changed. This action is not done as often as inserting or querying although more often than creating and deleting.

Table A.3: Update the database

	RP1	RP2	RP3
Poly	0.00347629		
SQL	0.03939703		

A.2.4 Query

To retrieve the information stored in the database it has to be queried (searched). This is done regularly and the time delay is important.

Table A.4: Query into database

	RP1	RP2	RP3
Poly	0.138572986		
SQL	0.051900138		

A.2.5 Delete

The database may need to be deleted under certain circumstances. This will not happen often and therefore the delay is not important.

Table A.5: Delete the database

	RP1	RP2	RP3
Poly	0.001752495		
SQL	0.068341016		

A.3 Conclusion