

# **Arduino based obstacle avoidance robot**

Final Report for CS39440 Major Project

*Author:* Daniel Atkinson (daa9@aber.ac.uk)

*Supervisor:* Dr./Prof. Dave Barnes (FIXME@aber.ac.uk)

1st March 2012

Version: 0.5 (Draft)

This report was submitted as partial fulfilment of a BSc degree in  
Computer Science (G401)

Department of Computer Science  
Aberystwyth University  
Aberystwyth  
Ceredigion  
SY23 3DB  
Wales, UK

## **Declaration of originality**

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

## **Consent to share this work**

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

## **Acknowledgements**

I am grateful to...

I'd like to thank...

## **Abstract**

Include an abstract for your project. This should be no more than 300 words.

# CONTENTS

<b>1</b>	<b>Background &amp; Objectives</b>	<b>1</b>
<b>2</b>	<b>Development Process</b>	<b>2</b>
2.1	Introduction . . . . .	2
2.2	Modifications . . . . .	2
<b>3</b>	<b>Design</b>	<b>3</b>
3.1	Overall Architecture . . . . .	3
3.2	Some detailed design . . . . .	3
3.2.1	Even more detail . . . . .	3
3.3	User Interface . . . . .	3
3.4	Other relevant sections . . . . .	3
<b>4</b>	<b>Implementation</b>	<b>4</b>
<b>5</b>	<b>Testing</b>	<b>5</b>
5.1	Overall Approach to Testing . . . . .	5
5.2	Automated Testing . . . . .	5
5.2.1	Unit Tests . . . . .	5
5.2.2	User Interface Testing . . . . .	5
5.2.3	Stress Testing . . . . .	5
5.2.4	Other types of testing . . . . .	5
5.3	Integration Testing . . . . .	5
5.4	User Testing . . . . .	5
<b>6</b>	<b>Evaluation</b>	<b>6</b>
	<b>Appendices</b>	<b>7</b>
<b>A</b>	<b>Third-Party Code and Libraries</b>	<b>8</b>
<b>B</b>	<b>Code samples</b>	<b>9</b>
2.1	Random Number Generator . . . . .	9
	<b>Annotated Bibliography</b>	<b>12</b>

## **LIST OF FIGURES**

## LIST OF TABLES

## Chapter 1

# Background & Objectives

This section should pick-up material from your progress report and enhance it based on the feedback and also your additional experience up to now.

**Note: All of the sections and text in this example are for illustration purposes. The main Chapters are a good starting point, but the content and actual sections that you include are likely to be different.**



## Chapter 2

# Development Process

You need to describe briefly the life cycle model that you used. Do not force your project into the waterfall model if it is better described by prototyping or some other evolutionary model. You do not need to write about all of the different process models that you are aware of. Focus on the process model that you have used. It is possible that you needed to adapt an existing process model to suit your project; clearly identify what you used and how you adapted it for your needs.

In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

You should briefly describe the design method you used and any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

### 2.1 Introduction

Introduce the specific model that you chose to use.

### 2.2 Modifications

Did you have to modify the model to suit a one-person project. If so, what did you change and why?

## Chapter 3

# Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

### 3.1 Overall Architecture

### 3.2 Some detailed design

#### 3.2.1 Even more detail

### 3.3 User Interface

### 3.4 Other relevant sections

## Chapter 4

# Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex, perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant? Your implementation might well be described in the same chapter as Problems (see below).

## Chapter 5

# Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on 'real users'? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

### 5.1 Overall Approach to Testing

### 5.2 Automated Testing

#### 5.2.1 Unit Tests

#### 5.2.2 User Interface Testing

#### 5.2.3 Stress Testing

#### 5.2.4 Other types of testing

### 5.3 Integration Testing

### 5.4 User Testing

## Chapter 6

# Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

The critical evaluation can sometimes be the weakest aspect of most project dissertations. We will discuss this in a future lecture and there are some additional points raised on the project website.

# Appendices

## **Appendix A**

# **Third-Party Code and Libraries**

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

## Appendix B

# Code samples

### 2.1 Random Number Generator

The Bays Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs Press *et al.* (1992).

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
    /*-----*/
    /* Minimum Standard Random Number Generator */
    /* Taken from Numerical recipies in C */
    /* Based on Park and Miller with Bays Durham Shuffle */
    /* Coupled Schrage methods for extra periodicity */
    /* Always call with negative number to initialise */
    /*-----*/

    int j;
    long k;
    static long idum2=123456789;
    static long iy=0;
```



```
static long iv[NTAB];
double temp;

if (*idum <=0)
{
    if (-(*idum) < 1)
    {
        *idum = 1;
    }else
    {
        *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7; j>=0; j--)
    {
        k = (*idum)/IQ1;
        *idum = IA1 *(*idum-k*IQ1) - IR1*k;
        if (*idum < 0)
        {
            *idum += IM1;
        }
        if (j < NTAB)
        {
            iv[j] = *idum;
        }
    }
    iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
    *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
    idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
    iy += IMM1;
}
if ((temp=AM*iy) > RNMX)
```

```
{  
    return RNMx;  
}else  
{  
    return temp;  
}  
}
```

# Annotated Bibliography

Dee, H. M., & Hogg, D. C. 2009. Navigational strategies in behaviour modelling. *Artificial intelligence*, **173**(2), 329–342.

This is my annotation. I should add in a description here.

Duckworth, Sylvia. 2007. *A picture of a kitten at Hellifield Peel*. <http://www.geograph.org.uk/photo/640959>. Copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

Neal, Mark, Feyereisl, Jan, Rascunà, Rosario, & Wang, Xiaolei. 2006. Don't touch me, I'm fine: Robot autonomy using an artificial innate immune system. *Pages 349–361 of: Proceedings of the 5th international conference on artificial immune systems*. Springer.

This paper...

Press, W.H., *et al.* 1992. *Numerical recipes in C*. Cambridge University Press Cambridge.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content*.

Various. 2011 (Aug.). *Fail blog*. <http://www.failblog.org/>. Accessed August 2011.

This is my annotation. I should add in a description here.