Arduino based obstacle avoidance robot

Final Report for CS39440 Major Project

Author: Daniel Atkinson (daa9@aber.ac.uk)
Supervisor: Prof. Dave Barnes (dpb@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

The aim of this project is to physicaly build and write the software for a robot. This robot should be able to drive around within its environment under its own power without colliding with any obstruction. It should also be able to see areas of the environment using some form of sensor system in order to determin which direction to travel safely.

CONTENTS

1 Background & Objectives **1**

2 Development Process **3**
2.1 Introduction . . . 3
2.2 Modifications . . . 3
2.3 Version Control . . . 4

3 Design **5**
3.1 Overall Architecture . . . 5
3.2 Justifications . . . 6
3.2.1 Materials . . . 6
3.2.2 Actuators . . . 6
3.2.3 Sensors . . . 8
3.2.4 Control . . . 10
3.2.5 Power Source . . . 11
3.3 Feedback Interface . . . 12

4 Implementation **14**

5 Testing **15**
5.1 Overall Approach to Testing . . . 15
5.2 Automated Testing . . . 15
5.2.1 Unit Tests . . . 15
5.2.2 User Interface Testing . . . 15
5.2.3 Stress Testing . . . 15
5.2.4 Other types of testing . . . 15
5.3 Integration Testing . . . 15
5.4 User Testing . . . 15

6 Evaluation **16**

Appendices **17**

A Third-Party Code and Libraries **18**

B Code samples **19**
2.1 Random Number Generator . . . 19

Annotated Bibliography **22**

LIST OF FIGURES

2.1 Branch Merge Diagram 4

3.1 Basic system diagram 5
3.2 Initial design 5
3.3 Servo Motor - robotshop.com 7
3.4 DC Motor - sparkfun.com - CC BY-NC-SA 3.0 7
3.5 Stepper Motor - stepperonline.com 8
3.6 Stepper Motor Internal- robotgear.com.au 8
3.7 Light Dependant Resistor - robotics.org.za 9
3.8 Camera Module - sparkfun.com - CC BY-NC-SA 3.0 9
3.9 Infra Red Sensor - coolcomponents.co.uk 10
3.10 Ultrasonic Sensor - coolcomponents.co.uk 10
3.11 PIC - circuitstoday.com 11
3.12 Arduino arduino.cc 11
3.13 Netduino - netduino.com 12
3.14 Atom Motherboard - intel.co.uk 12
3.15 Raspberry Pi - raspberrypi.org 13

LIST OF TABLES

Chapter 1

Background & Objectives

I was first exposed to electronics in an academic enviroment in high school. This was only very basic circuitry, such as making a light flash by using simple integrated circuits. Being introduced to integrated circuits made building an electonic timer much easier, which was the first thing I produced using these small chips. This was very satisfying when it finally worked, a feeling I still get when something I make works as intended.
Fast forward to college five years later and I am still fascinated by electronics. Still using these wonderous little chips to build more interesting circuits I built an audio amplifier whereby I input a waveform into the circuit, either generated by a signal generator or my guitar, and amplify it or smooth the signal to create a new sound, then output this amplified signal to a speaker. This distorted sound is similar to those created by a guitar amplifier that has built in effects or an specific effects pedal also used by guitarists.
At college I also took a computing class in which the programming language Visial Basic was taught as part of the course. Naturally the next step would be to compine the electronics with the programming knowledge. This took the form of a small blinking light project where I use a PIC (Peripheral Interface Controller) to flash an LED (Light Emitting Diode). A PIC is a small chip (Integrated Circuit) which can run small amounts of code to read inputs and control outputs on its various pins.
In the summer between the end of College and starting University I discovered a range of open source hardware microcontrollers called Arduino. These boards made combining program code and electronic hardware much easier by doing much of the base work for me. These microcontrollers have a large community, having written all forms of libraries to interface the board with various pieces of hardware and control them with much less effort than would be needed when using a PIC. The PIC does have a large number of libraries but the Arduino ones seems to have a much wider variety of what they support and a very active community to help if you get stuck.
I have also had some experience using the pioneer research robot created by Adept MobileRobots LLC (2012) which are used by Aberystwyth University in the robotics lab. The experience with these robots was to use their ultrasonic sensors to try and avoid hitting some polystyrene boards. Due to the limited time available to use these robots the resulting code was not very effective or polished, but it has heavily influenced my ideas for designing my current project and further pricked my enthusiasm for robotics and all of the possible applications it has.

My main objective with this project is to produce a piece of hardware that can manaevere itself around an environment under it's own power without bumping into anything. This is to be built utilising the knowledge I have gained about electronics and programming from previous projects

and from the courses I have attended as part of my University degree.

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

I chose to use the iterative and incremental approach to development. This is mainly because of how modular my project is. In theory, I can add more functionality with minor adjustments to the core system, thus making iterative/incremental very suited to my needs.
Each part of the system in an incremental strategy can be developed independently and slotted together as they reach completion.
Each iteration is a review of the previous which has been reworked and improved upon.
For a well functioning system it needs good design, quality programming and a good debugging process. So, after designing the inital system, writing a simple prototype it is then time for debugging it to get an indication of what the main flaws are. Once these flaws have been clearly identified a new design has to be drawn up to correct these issues. After writing the new version following the revised design the cycle continues in the same manner, design, write then debug.

2.2 Modifications

No real modifications were made to this development process as it works for individuals and for teams without alteration.

2.3 Version Control

This is majorly useful in any project which involves managing code or documents digitally. It is even usefull as a backup tool, to be safe from accidental deletions, hard drive failure or any number of other unfortunate occurances (for instance a fire destroying your computer) as you can just re-download the files.
There are other features that version control systems offer that are of more use in this type of project. Branching and merging are two of the most used features. These enable the user to make a branch within the project in which they can work on a specific feature independantly of the main project. You may make multiple branches at the same time and work on different things all independant of each other. If you imagine a tree, where the trunk contains the current working state of a project, then if you want to change something or create something new you make a branch which shoots off from the tree but contains all the information that is in the trunk. You can then work on it independantly from the trunk, even if you or somebody else makes another branch from the trunk, the changes made in your branch will not effect it. Once these are finished you can merge them back into the main project, this is a very nice feature version control systems offer as it performs most, if not all of this for you, instead of having to manualy try and integrate each line of the branch files back into the main ones.
I have chosen to use Git for developing this project due to how powerful the merge feature is as well as a website called github **?** which will host repositories for people. The website also has nice usage statistics and offer some private repositories to students. Github repositories are normally open to the general public unless you pay a fee for having non public facing ones. Being a student enables me to have a small number of these private repositories which let me control when I am ready to release a project to public viewing.

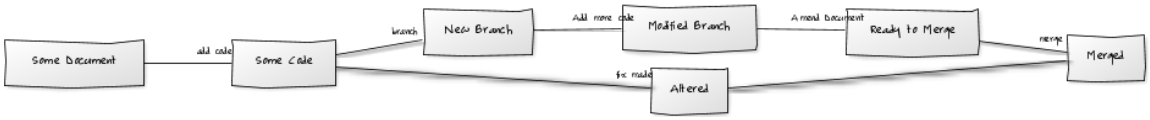

Figure 2.1: Branch Merge Diagram

Chapter 3

Design

3.1 Overall Architecture

The initial design for the robot is to produce a small wheeled vehicle with a platform for mounting the various systems. These systems should be a central control unit, motor control and the various sensors.

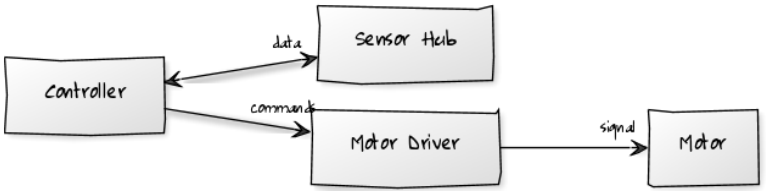

Figure 3.1: Basic system diagram

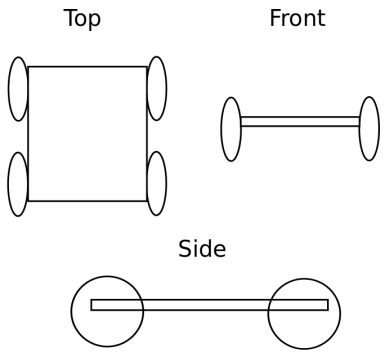

Figure 3.2: Initial design

The central control unit will be a microcontroller for ease of interfacing directly with hardware as well as keeping power consumption down. Keeping power consumption to a minimum is important so that the robot can be active for a longer period of time without needing to be recharged. This controller will interface with both a motor control system and the various sensors required to detect objects in the environment local to the robot.

3.2 Justifications

The various components that the project will need to come together into a finished product have many options.

3.2.1 Materials

I considered several materials for the robot chassis to be built of.

- Wood
  This would be the easiest material to make the chassis from as it is very cheap, easy to cut into the intended shape and easy to mount components on with either adhesive, nails or screws. Also the fact that it does not conduct electricity will help when mounting circuit boards to it.

- Plastic
  The lightest option. Good due to its low weight but may not be as strong as wood or a metal option and could bend or snap under the load of heavier components such as motors or a large power source. It can be more expensive than wood to aquire. There is a higher difficulty in cutting it into the desired shapes. It is also non-conductive, again useful to mount electronic components to. Plastic can hold a static

- Steel
  A stronger material that can withstand a much heavier load, but is itself rather heavy compared to wood or plastic. This extra base weight before adding anything else will put more strain onto the motors used to drive the robot and may even need to use more powerfull motors because of this extra weight. It is a very conductive material which means that a non-conductive mounting platform will also be needed to mount electronic components as to avoid damaging them.

- Aluminium
  A much lighter metal than steel, but still much heavier than wood or plastic or the same thickness. It can also withstand heavier loads than wood or plastic but it also much more difficult to cut. Again aluminium is a very conductive material meaning that a non-conductive mounting platform will be needed. It can also be used as a heat sink for the components that can get very hot such as the motor drivers or the motors themselves. A heatsink is a material attached to something that gets very hot and conducts that heat. It generaly has a large surface area to disipate the heat into the cooler air around it, but it may also have a fan to blow/draw the hotter air away replacing it with air/gas with a lower temperature than that of the heatsink.

Aluminium seems to be the best all round choice being strong but not as heavy as steel. It can act as a heat sink if the motors are mounted directly to it. It is also not very expensive to buy in small amounts.
In addition to the aluminium base I have decided to use plastic for mounting components to the base as it is light, inexpensive and non conductive which is suitable for electronic components.

3.2.2 Actuators

Actuators are motors used for controlling movement of a system.

- Servo
  Typical servos are a motor and a gearbox with a potentiometer, a voltage divder in this case used to determine how far a motor has turned, for feedback. These motors are great for controling such things as the direction of sensors or moving very light devices. Servos are low voltage, typically 4.8 - 6 volts, and as such do not have much strength, they are typicaly not good for driving larger equipment. Also most servos only turn up to 180 degrees or 360 degrees. In normal operation they do not turn continuously but can be modified to do so at the cost of losing the feedback of how far the motor has turned.

Figure 3.3: Servo Motor - robotshop.com

- DC Motor
  Direct current motor has a very simple operation. Apply current to one side of the motor to make it turn, reverse the direction of the current to reverse the direction the motor turns. Changing the speed of these motors is simple, either change the voltage, keeping it within the devices tolerances, or turn the current supplied to the motor on and off at high speed where how quickly it is alternated determines the speed of the motor. Typicaly these motors are attached to a gearbox to gain more torque to drive much higher loads. Optical rotary encoders can be used to determine how much the motors have turned and how fast. These encoders use a light based sensor to detect when the light changes in front of it, this can be used with a disc that has black and white lines on it. The change in color is detected, this along with how many times it changes and with what frequency this happens can determine the amount a wheel has turned and how fast it has done so.

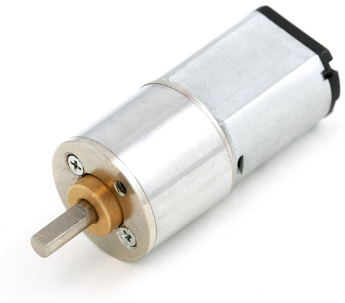

Figure 3.4: DC Motor - sparkfun.com - CC BY-NC-SA 3.0

- Steppers
  Stepper motors use an internal gear and a ring of magnets. These magnets pull the gear into position, powering the magnets in sequence which will turn the motor. Each part of this cycle is called a step. This means that a single step is a know amount of rotation. Using this type of motor ensures that you can accuratly turn whatever is attached to the motor shaft a known amount without any additional measuring equipment, although it may be used to verify that it has in fact moved the amount expected

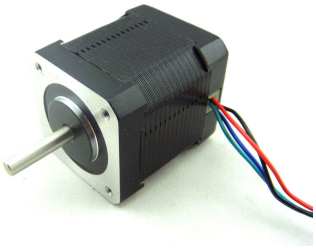

Figure 3.5: Stepper Motor - stepperonline.com

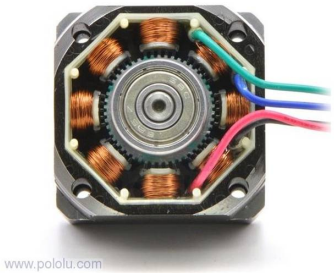

Figure 3.6: Stepper Motor Internal- robotgear.com.au

I have chosen to use stepper motors due to the ability to control the amount and speed of rotation with more accuracy than the alternatives. Stepper motors do come in high torque version which may be needed for this project as the chassis is made of metal which is a much hevaier material. A stepper motor could be using with a chassis of any of the materials mentioned, it may struggle with steel depending on how thick of a piece is used. DC motos could also be used with all materials if in conjunction with a gearbox, but the additional system needed to measure and control the exact rotation of the wheels using this method puts me off of the idea. Greater power but less accurate control.

3.2.3 Sensors

- LDR
  An LDR is a light dependant resistor. A small resistor that changes its resistance depending on how much light it is exposed to. This could be used to detect if the robot is very close to bumping into an object and avoid it as the object got closer and possibly cast a shadow onto the sensor reducing the amount of light the resistor can detect, kind of like a physical bump skirt which activates when something touches it.

- Camera
  A camera could be used to detect objects in front of it using various image processing

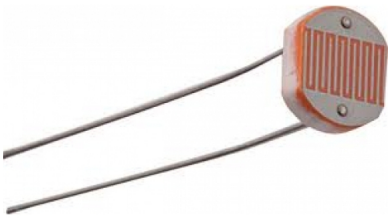

Figure 3.7: Light Dependant Resistor - robotics.org.za

techniques. This method is good because it can potentialy map a relatively large area in a single image. On the other hand it requires more processing to do, which can be slow and result in having colided into an object or be stuck in a tight space before it has finished processing. I could use a more powerful processor to overcome this but it adds complexity, cost and power consumption.

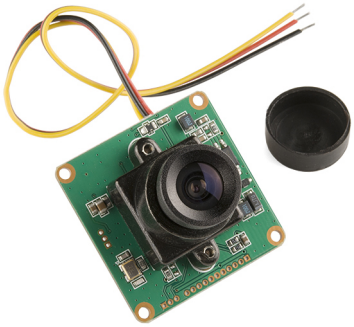

Figure 3.8: Camera Module - sparkfun.com - CC BY-NC-SA 3.0

- Infra Red
  Used to detect distance from an object. Am emiter and a reciever pair linked to work like the light dependant resistor but using infra red instead of normal visible light. Depending on the intensity of infra red picked up be the reciever it can be used to determine the distance from the source of the reflection. Ambient infra red can effect readings.

- Sonar
  Again an emiter systle approach. It emits and ultrasonic wave to bounce off of whatever surface is in front of it. The time taken from emiting the wave until recieving the wave detemrines how far away the object is. This method comes with its drawbacks. Due to how sound waves behave when they interact with the environment by bouncing off of it. If the surface is angled or curved the sound can bounce away from the reciever, either not reaching it at all giving the possible false reading that there is nothing in front of it, or it could bounce off of multiple surfaces back to the reciever giving a false reading that an object is there but further away due to the sound taking longer than it should have to reach the reciever.

A combination of both sonar and infra red logicaly seems like a good idea. One can compensate for the others weaknesses. Use the sonar to compensate for ambiant infra red and the infra red can be used to compensate for sonar bouncing around the environment. Hopefully this will reduce the number of false readings produced.

Figure 3.9: Infra Red Sensor - coolcomponents.co.uk

Figure 3.10: Ultrasonic Sensor - coolcomponents.co.uk

3.2.4 Control

The robot will need a controler, that connects the software to all the hardware.

- PIC
  Peripheral Interface Controller. Very low cost microcontroller with a small easy to learn instruction set and support serial communication/re-programming. They also come in a DIL package (dual in-line) making them easy to incorporate into through-hole printed circuit boards.

- Arduino
  An open source hardware board that is cheap but not as cheap as a PIC. These are very popular among hobbyists due to them being very easy to use and have a vast collection of community written libraries to interface with all different types of hardware.
  Arduino uses C or C++ programming language for development.

- Netduino
  This is also an open source electronics prototyping platform but instead of being based on C and C++ it is based on the .Net Micro Framework which is Microsofts version of an embedded framework.
  These boards cost more than the Arduino and PIC, and have much less community support.

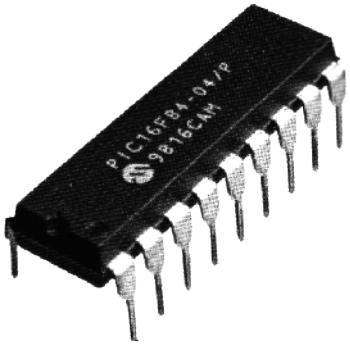

Figure 3.11: PIC - circuitstoday.com

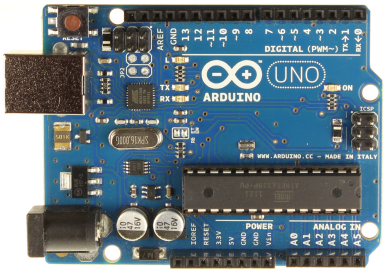

Figure 3.12: Arduino arduino.cc

- Motherboard
  A small motherboard that can be found in a home computer or a netbook/laptop. These have the widest variety of applications. It can support most operating systems and programming languages but come at the hefty price of power consumption. Compared to microcontrollers, a full motherboard draws a very large amount of power to run. Also they take far longer to power on due to running an operating system, unlike microcontrollers that have the code compiled down and run directly on the hardware itself.
  The cost of such boards is also very high as they are far more complex pieces of electronics.

- Raspberry Pi
  The Raspberry Pi is a credit card sized computer that is an embedded platform for Linux and various other operating systems. It is very cheap and runs much faster than most microcontrollers. It does also have the downside of long startup times due to running an operating system on such a compact system. Unlike normal motherboards this little board has some GPIO (general purpose input output) pins for interacting directly with various pieces of hardware like the microcontrollers do.

3.2.5 Power Source

As this robot is intended to move around freely, unhindered by power and data cables, the power-source cannot be supplied by a wall power outlet, it has to be self contained. This means it will have to be a battery. The battery will have to be several cells or a singe high output cell due to the size of motors intended.
I will need as many Amp hours as possible for longer runtimes. This could be achieved with sev-

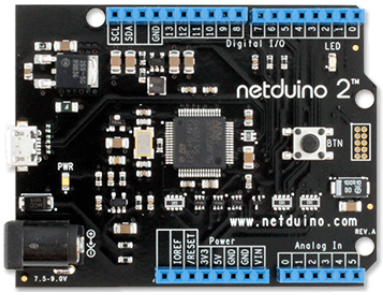

Figure 3.13: Netduino - netduino.com

Figure 3.14: Atom Motherboard - intel.co.uk

eral cells linked together ins eries (end to end) to increase voltage and/or linky more together in parrallel (side by side) to increase amp hours (runtime).

- Lithium Polymer
  LiPo batteries come in up to 11.2 volt packages, which is not quite high enough for some higher voltage motors which I may be using. These batteries are much lighter than some other alternatives and are common in embedded devices.

- Lead Acid
  A lead acid battery is a choice with a high output. A single battery can output 12 volts and can be found in high amp hour packages compares to the lithium alternatives. Due to needed a high power motor for the chassis already, a higher weight battery is not too much of an issue and provides the benefit of the higher power output.

3.3 Feedback Interface

While operating the robot, if there is any unexpected behaviour it would be nice to have some form of interface to see what the robot thinks the environment looks like. There cannot be any cables trailing from the robot to a laptop so a wireless solution would be good.
All that is needed is a small microcontroller, a wireless module and a display. An Arduino Fio is a good fit as it is small, low powered, has both lithium polymer battery socket and an xbee wireless module socket built in.
A small LCD (liquid crystal display) can be connected to the Arduino to display information it recieves.

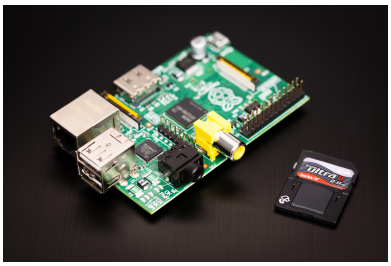

Figure 3.15: Raspberry Pi - raspberrypi.org

Chapter 4

Implementation

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 Bayes 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

Cockburn, Dr. Alistair. 2008. Using Both Incremental and Iterative Development. *Pages 27–30 of: Stsc crosstalk (usaf software technology support center)*. CrossTalk.

Detailed explanation and usefullness of the iterative development model

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.

LLC, Adept Mobilerobots. 2012. *Adept mobilerobots*. `http://www.mobilerobots.com/ResearchRobots.aspx`. Accessed October 2012.

Information various models of research robots and additional modules for them

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.