****** **Faculty of Arts, Computing, Engineering & Sciences**

Department of Computing

Final Year Project 55-6727 (SEGM) - 2016

|  |  |
| --- | --- |
| **Author:** | **Martyn Edward Rushton (22015598)** |
| **Date Submitted:** | **13th April 2016** |
| **Supervisor:** | **Dr Peter Collingwood** |
| **Degree Course:** | **BSc (Hons) Computer Science** |
| **Title of Project:** | **Developing a self-driving car software framework to support a subsumption architecture.** |
| **Confidentiality Required?** | **No** |

# Abstract

This is my Abstract

# Acknowledgements

Add Acknowledgements here.

# Table of Contents

Abstract II

Acknowledgements III

Table of Contents IV

1 Introduction 1

1.1 Background 1

1.2 Aims 2

1.3 Scope 2

2 Research (10% / 800 words) 2

2.1 Previous Work 2

2.1.1 Stanford University 2

2.1.2 Google X 3

2.1.3 Baidu 4

2.1.4 Tesla 4

3 Development (15% / 1200 words) 6

3.1 Initial Design 6

3.2 Feasibility Testing 7

3.2.1 Design Revision 7

3.3 Hardware Design 8

3.3.1 Microcontroller 8

3.3.2 Other Equipment 9

3.3.2.1 Chassis 9

3.3.2.2 Actuators 9

3.3.2.3 Sensors 10

3.3.3 Fritzing Diagram 10

3.4 Software design 10

3.4.1 Hierarchical Paradigm 10

3.4.2 Subsumption Architecture 11

3.4.3 Tools 12

4 Testing (5% / 400 words ) 13

5 Conclusion (15% / 1200 words) 14

5.1 Future Work and Improvements 14

5.2 Conclusions 14

6 References 15

7 Glossary 16

# Introduction

{INTRO TO THE PROJECT}

## Background

On January 29 1886 in Mannheim, Germany, Karl Benz submitted a patent for an “automobile fuelled by gas”, widely accepted as the first modern motor vehicle. During one of its first demonstrations, the Benz Patent Motorwagen driver lost control and crashed into a wall, the first of a long list of motoring accidents of the last 130 years.

Since Benz, countless changes have been made to motor vehicles, primarily centred around speed, with safety and economy only taking centre stage fairly recently. It is amazing how far vehicles have come in such a short space of time, but unfortunately we have created a monster that has killed millions of people. The website of the Association for Safe International Road Travel (ASIRT 2015) state “Nearly 1.3 million people die in road crashes each year, on average 3,287 deaths a day”, that is the equivalent of seven Boeing 747’s falling out of the sky and killing everybody aboard each day. On top of that “An additional 20-50 million people are injured or disabled” per year, that is almost the population of Britain.

ASIRT (2015) have also stated that “Road traffic crashes rank as the 9th leading cause of death and account for 2.2% of all deaths globally” and they are the “leading cause of death among young people ages 15-29, and the second leading cause of death worldwide among young people ages 5-14”. These statistics are shocking and that is what is leading some of the biggest technology companies in the world to grant resources towards developing self-driving cars, especially for areas such as public transport and taxi services.

There is nothing new about giving control of vehicles to somebody trained. For hundreds of years, the upper classes have been avoiding taking control of their own vehicles and have paid somebody else to do it for them, from horse and carriage drivers to limo chauffeurs. The advantages behind this idea are clear, instead of doing the monotonous task of driving, you can sit back and relax or continue office work while on the move. The ideas of self-driving cars are just extensions of that, replacing the driver with a robotic mind.

Science fiction writers have been predicting self-driving cars for 50 years, most famously with Herbie from the 1968 film *The Love Bug*, KITT from the 1982 series *Knight Rider* and Johnny Cabs from the 1990 film *Total Recall*. Herbie and KITT were both Sentient beings where as Johnny Cabs was much more like what we are currently looking into with self-driving cars as he takes a destination from the user and then drives to it, if he doesn’t recognise the destination he gives error messages such as “I do not recognise that address, would you please repeat the destination”.

## Aims

The aims of the project is to

* perform research into self-driving cars and develop a microcontroller based framework to perform simple tasks to emulate a self-driving car.
* learn more about the way a subsumption architecture can be used in robotics development.
* only use hardware, software, libraries and tools that meet the Open Source Initiative (2007) requirements to be defined as open source.

## Scope

The scope of this project was defined mainly by time and cost. Firstly, the project was unfunded, meaning that mainly only equipment that the university owned could be used and anything extra would come at personal cost. Secondly, the project only had a timescale of seven months, from October 2015 to April 2016 and would only get around 500 hours of time allocated to it.

With these constraints in mind, this project will demonstrate a framework that works under a specific set of conditions and not every situation that a self-driving car would come up against.

# Research (10% / 800 words)

## Previous Work

There has been a lot of notable work on self-driving cars recently, particularly by Stanford University, Google and Tesla, but there are also other companies, such as Chinese search engine giant Baidu, pushing development forward and racing to be the first to release a fully functional product.

### Stanford University

Stanford University, led by associate professor Sebastian Thrun, are generally credited with building the first fully autonomous car while competing in the 2005 DARPA Grand Challenge, a prize competition funded by the American government to try and spur the development of automated ground vehicles. The car they built, a modified Volkswagen Touareg called Stanley, completed the 132-mile off-road course in 6 hours 54 minutes, 10 minutes ahead of Carnegie Mellon’s H1ghlander.

In 2007, Stanford placed second in the DARPA urban challenge with Junior, a Volkswagen Passat based off Stanley. The urban challenge was similar to the 2005 Grand Challenge in that it was a time based event but the vehicles had to obey traffic laws and interact with other robotic vehicles, ensuring they both acted and reacted safely. This type of interaction was not needed in the 2005 challenge and was completely new territory.

### Google X

After the success with Stanford, Sebastian Thrun moved to Google in 2011 and co-founded Google X, a semi-secret research and development facility that aims at doing ‘moonshots’ and making reality out of the technology found in science fiction.

Google X has 4 major projects, the most significant and successful of these been the self-driving car. Google had in fact started the project in 2009 but were having little success until Thrun joined with some other DARPA contestants, most importantly Chris Urmson, who is now director of Google X after Thrun’s exit in 2014 to focus on his online educational company, Udacity.

Google have made a number of vehicles, including both converting existing vehicles, mainly the Lexus RX405h SUV, as well as designing and building their own prototype vehicle. As of February 2016, there were 56 google self-driving cars on the road, made up of 23 Lexus RX450h and 33 of their own prototype. In Google’s February 2016 report, they claim to have driven “1,452,177 miles” in autonomous mode, “1,017,450 miles” in manual mode and they currently “average around 10,000 – 15,000 autonomous miles per week on public streets”.

February 14th 2016 was a bad day for the Google self-driving car as it was in a low speed collision with a bus. Although the self-driving cars had been in accidents before, it had never been found that the autonomous mode of the car was to blame, it was usually the test driver or a third party that caused the crash. The February 14th incident was without doubt the self-driving cars fault as it pulled out in front of a bus while attempting to merge from the right lane to the left due to the road been blocked by sandbags. The engineers stated that “Our car had detected the approaching bus, but predicted that it would yield to us because we were ahead of it” and this was corroborated by the test driver aboard as he claims he would have also made the same decision although Google have now tweaked the code so the self-driving car now yields to larger vehicles such as busses and trucks.

In December 2015, The Next Web (Garun 2015) reported that Google are looking to separate the self-driving car branch all together and have it as its own standalone company. The thoughts are this is because Google are “looking deeper into ways to monetize the driverless cars, directly challenging popular ride-sharing services like Uber and Lyft”.

### Baidu

Baidu is the Chinese equivalent to Google in its day job. Baidu started out as a search engine for Chinese web pages, directing around 70% of Chinese traffic in 2011. Baidu has followed a similar path to google, offering services such as maps, cloud storage, a browser, a mobile operating system and a video sharing system.

Baidu announced in 2013 that they were working on a similar project to Google but the end result would be a car that would be operated by humans and then perform autonomous actions when needed. In a The Next Web article (Hong 2014), Kai Yu, head of Baidu’s Institute of deep learning said they were not building a driverless car but instead a “highly autonomous car”. This will create a system similar to what is seen in the 2004 film ‘I, Robot’, where the driver can take control when they feel they need to.

In December 2015, the BBC reported (Baranuik 2015) that Baidu had successfully completed a 30km road test on its modified BMW 3 series driverless car, stating “The car executed a range of manoeuvres, including U-turns, lane changes and merging into traffic from ramps.”, demonstrating an autonomous car and not a driver assist system they had originally planned.

Baidu are trying something slightly different to the other large parties by using its cars and their sensors to build a 3D map of the Chinese roads and then sharing them amongst all their other vehicles. The plan is that this will enable the cars to know the roads exactly, so they can accurately know their position within a few centimetres.

The Future development plans by Baidu show that instead of focusing on getting self-driving cars out to personal users, they are planning on first getting cars out to be used as a public transport system, most likely taxi style systems and small buses to begin with, expanding to larger vehicles in time.

### Tesla

Tesla is a fairly new player in the car market but they have pushed automotive technology forwards leap and bounds, so it is unsurprising that they are also working on intelligent vehicles.

The Tesla autopilot system was released at the start of 2015 and within a year has managed to do a coast to coast drive across the United States in nine days, yet there is a slight catch. The Tesla autopilot system still requires an alert driver as the car cannot reliably complete all of the tasks of a fully automated car.

Tesla’s approach is to attempt to get its intelligent systems to learn where it is wrong, so if the driver needs to take over, the car will take note of that and learn the situation for next time. This is then uploaded into a central database so that all the vehicles on the network learn at the same time.

There are reports that Tesla are planning on having a fully autonomous car on the roads by 2017, but as it currently stands their autopilot system is just a very good driver assistance system.

# Development (15% / 1200 words)

Matarić (2007) states that “A robot is an autonomous system which exists in the physical world, can sense its environment, and can act on it to achieve some goals”, which seems to give three key requirements that the most basic robot must meet:

* Existing in the physical world, so it is not just a computer simulation.
* Senses its environment, so must collect sensor data.
* Act to sensor data to achieve some goals.

## Initial Design

The most basic version of the initial design can be seen in Figure 3-1. There are three sections to this design: sensors, data processing and actions.

The sensors section contains a mix of different sensors, such as light, temperature, ultrasonic and a reflectance array. The purpose of this section is to gather information about the world around the prototype. Referring back to the definition of a robot from Matarić, this meets the ‘Senses its environment’ requirement.

The data processing section is the next in the timeline. This will take the data gathered in the sensors section and send them to a computer in the form of a JSON object. The computer will then process the data and send back relevant actions for the computer to take, for instance, if the sensor data shows that it is dark, the processor will come up with the solution of turning on the lights and send this information back to the microcontroller.

The action section consists of real world feedback, such as motors and lights that the prototype will contain. Once the actions have been received from the computer, the microcontroller will send the relevant actions to each of the components, such as telling the telling the motors to reverse and to turn the lights red to indicate this.

The data processing and action sections mean that the “Act to sensor data” section of Matarić’s requirements. Due to the fact the plans also require a microcontroller and a computer, the requirement of “Existing in a physical world” is met and this can be classed as a valid design for a robot against that definition.



Figure ‑: Initial design of system.

## Feasibility Testing

With the initial design completed, it was decided to do some small feasibility tests and basic prototyping to ensure that the implementation is possible and there were no obvious issues.

The first test was to ensure that all of the sensors were available and could work in conjunction. This test passed with flying colours by attaching all of the sensors to an Arduino Uno at the same time and running code that was amalgamated from the examples that come with the Arduino IDE.

Another test was sending large JSON strings across the serial port, an idea that was fundamental to the project. This test raised quite a few issues, mainly that sending such large strings across a serial device was slow but also, when the Arduino was trying to both send and receive data, the board would get confused and start acting sporadically to some of the commands.

### Design Revision

Due to the message passing, a fundamental area of the project, having such large flaws, the design would have to change to compensate for this. The design in Figure 3-2 shows the proposed solution, which was to do most of the data processing on the microcontroller because although this would have a slower processing speed, the communication time would be negligible and the whole system would be faster and more reliable overall. The obvious risk that this approach would introduce is the limited program size of the Arduino, for this reasons, it was decided that a larger board such as a Mega or Due would need to be used to try and reduce the likelihood of running out of memory.

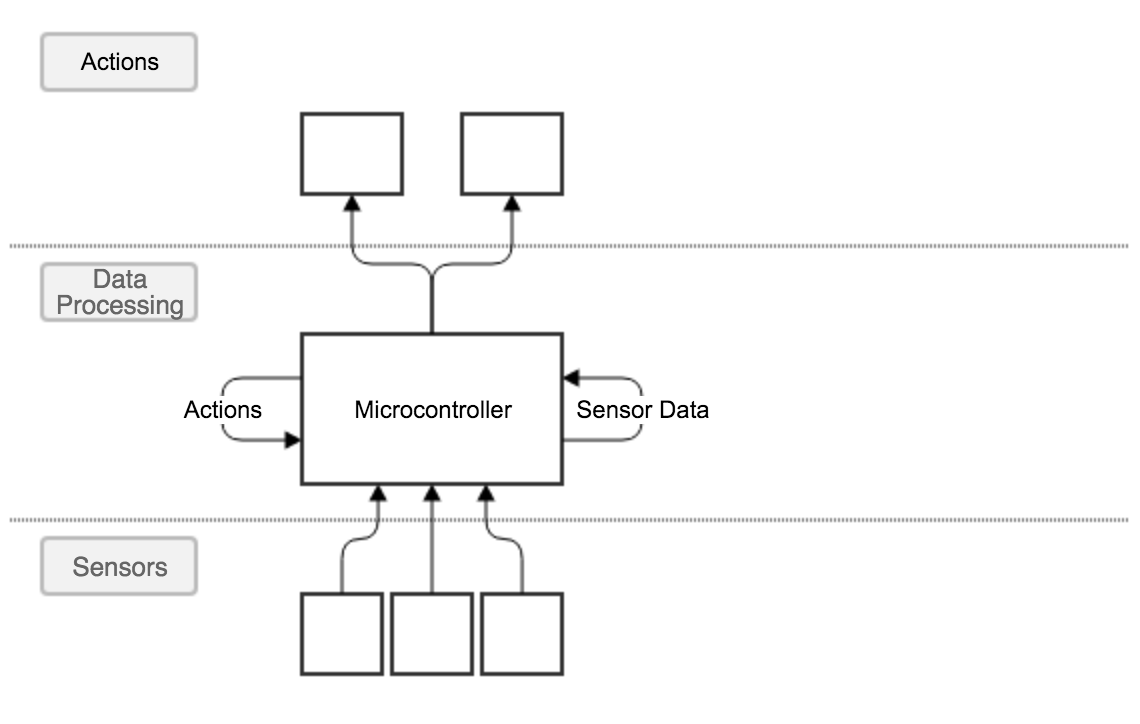


Figure ‑: Edited design after feasibility testing.

## Hardware Design

With an initial high level concept design in place, the next step was to design the hardware. The hardware in this situation contains three sections, sensors to detect the outside world, a microcontroller to reason the sensor data and actuators to receive the reasoned actions.

### Microcontroller

Although there are many microcontrollers available on the market, the obvious choice for a project such as this was something from the Arduino range. Arduino have a large range of products, from boards that can only run a very basic LED programs to boards that are powerful enough to currently be in orbit around the earth as part of test satellites.

Development of the initial prototype started out by using an Arduino Uno, the most common board due to its friendly price tag and low learning curves. The Uno has 20 I/O pins, allowing for a small amount of components to be added, which is perfect for basic prototyping and testing.

Once development had moved along, the Uno was switched out for the larger Arduino Mega. The Mega is almost identical to the Uno except it has more resources available to it, with the Mega providing 60 I/O pins and 256kb of RAM which dwarfs the 32kb provided by the Uno.

### Other Equipment

To accompany the microcontroller, both sensors and actuators would be required to get a fully working prototype.

#### Chassis

The chassis is the main base of the prototype, it connects all of the sensors and actuators together. Due to its importance, a pre-built chassis was chosen in the form of a Dagu Rover 5, pictured in figure 3-3 below. Other chassis, such as the Arduino Robot (Arduino 2016) and the Pololu Zumo (Pololu 2016) were considered, but the wheel encoders and customisability over the gearbox gave the Rover 5 the edge.

Furthermore, the Rover 5 has its own motor control board that makes interacting with its motors a very basic task instead as it is effectively plug and play instead of having to do the difficult task of integrating a third party motor control board with the Arduino Robot or Pololu Zumo.



Figure ‑: Blueprint of the Dagu Rover 5 chassis

#### Actuators

In a robot that relies on movement, the motors are the main actuator and these are included in with the Dagu Rover 5. Due to this the only actuators that needed to be included were those intended for the user, such as LED’s to show the state of the vehicle; for example, two sets of red, white and green LED’s to act as brake reverse and indication warning lights. Also, a LCD screen was added to give short pieces of information, such as warnings or problems in the system.

#### Sensors

Collecting information from the surroundings is one of the three key points that make up a robot, so it is very important to have as many relevant sensors as possible. The main sensors that were used were those that would allow the prototype to navigate its way around, this was mainly made up of ultrasonic sensors to detect other objects in the system, whether that be something as common as another moving car or something as rare as a cow in the road.

Other sensors were also included, such as a light sensor that would detect when it starts to get dark and turn the headlights on to compensate. This is a common function of cars that have been made in the last few years and would come under an action that should be automated.

### Fritzing Diagram

Once the hardware was decided, it was possible to start building up Fritzing diagrams to assist in understanding the configuration.

{INSERT FRITZING DIAGRAMS}

## Software design

{SECTION INTRO}

### Hierarchical Paradigm

Previous to the late 1980’s, robots were designed around a hierarchical system that contained a loop of sensing, reasoning and acting, such as shown in figure 3-3. The robot would first get sensor information from the physical world and pass it to a reasoning system. The reasoning system would calculate the best move and then send action data to the robot, that would then act on the physical world, thus changing it and needing to start the loop again.

The large problems with the Hierarchical paradigm came when attempting to improve one of the three systems. If the developers wanted to make any of the systems better, it would break the other two. For instance, if they wanted to improve the reasoning, it usually broke the sensing and acting systems and they would have to rewrite these to match. More often than not, it was reported that when attempting to improve these systems, they would add new functionality at the cost of losing old functionality.

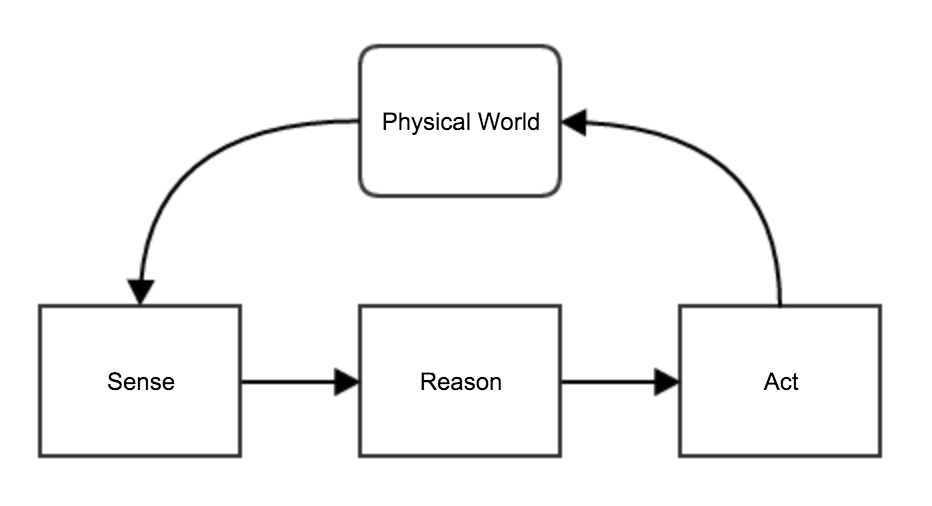


Figure ‑: A typical structure of hierarchical robot control system

### Subsumption Architecture

Due to the issues with Hierarchical paradigm, Rodney Brookes came up with a new way of designing robots, called the subsumption architecture. Simpson et al (2006, p226) state that “The subsumption architecture involves building robot control systems with increasing levels of competence.”. This means that so during development, you still have the sensing, reasoning and acting systems, but each task has its own set of sensing, reasoning and acting systems, this can be seen in figure 3-4.

One of the main features of subsumption architecture is that once a section, such as avoid, has been completed, no more changes are made to its code. Development can then start on the section above it, wander, and while this section can access the functionality of avoid, the avoid sections code should never be changed to add extra functionality used in wander. Instead, the extra functionality should be defined in wanders sense, reason and act functions.

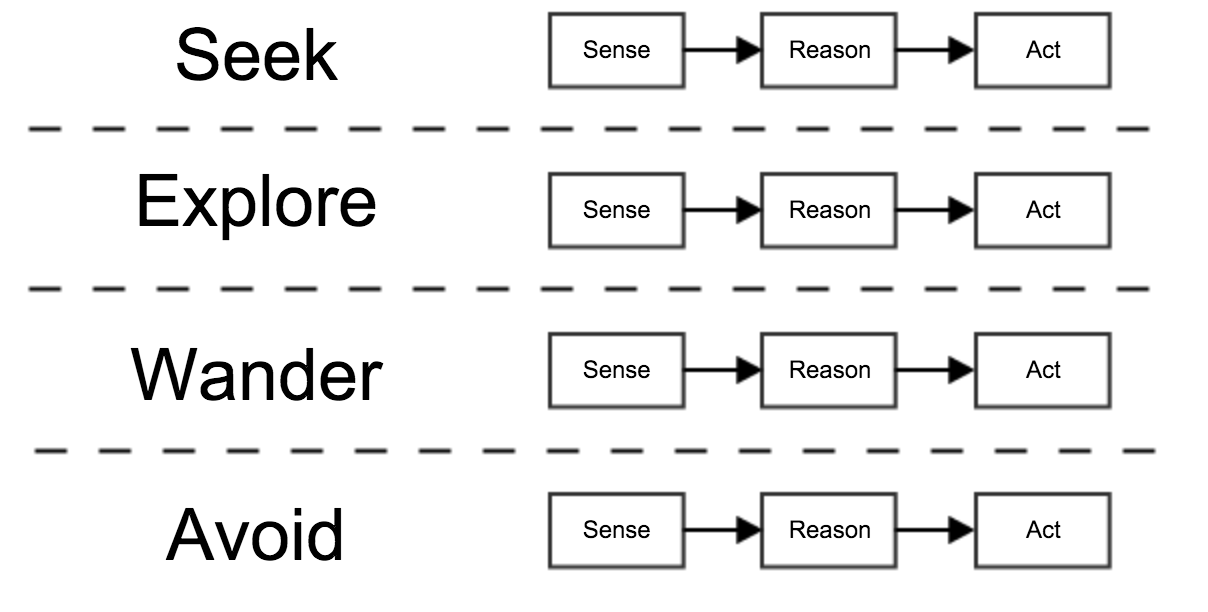


Figure ‑: A typical structure of a subsumption architecture system

The integrity of this project was held almost entirely on the software design, if the software was designed poorly then the entire project would be an uphill battle. Due to this, it was decided that the software would be designed around a subsumption architecture base.

### Tools

Due to choosing the Arduino microcontroller, the choice to develop in a C based language was easy to make as that is what the Arduino is natively written in. Therefore, C++ was the chosen language as it is a common language that has extensive documentation as well as it using the more widely used class object system instead of the struct object system that C uses.

The entire coding was done using the Atom text editor due to the fact that it is free, open-source and written by Github, one of the most successful open-source projects ever. It is also very extensible which means it has fantastic libraries for Arduino development.

Development started off using the Arturo build tool and while this was very good on the Arduino Uno, there is unfortunate bugs that means it will not work on the Arduino Due. Therefore, the project was ported over to the PlatformIO build tool as that has much better support for the Due but it is unfortunately slightly heavier to run than Arturo.

# Testing (5% / 400 words )

Write some stuff about testing the code

# Conclusion (15% / 1200 words)

## Future Work and Improvements

## Conclusions

# References

ARDUINO (2016). *Arduino Robot.* [online]. Last updated 7 March 2016. https://www.arduino.cc/en/Main/Robot

ASIRT (2015). *Road Cash Statistics*. [online]. Last updated 18 December 2015. http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics

BARANUIK, Chris (2015). *Baidu driverless car completes test on Beijing roads*. [online]. Last updated 27 December 2015. http://www.bbc.co.uk/news/technology-35059938

GARUN, Natt (2015). *Google to spin-off self-driving cars as a standalone Alphabet company next year*. [online]. Last updated 18 December 2015. http://thenextweb.com/google/2015/12/16/report-google-to-spin-off-self-driving-cars-as-a-standalone-alphabet-company-next-year

GOOGLE (2016). *Google Self-Driving Car Project Monthly Report February 2016*. [online]. Last updated 7 March 2016. https://static.googleusercontent.com/media/www.google.com/en//selfdrivingcar/files/reports/report-0216.pdf

MATARIĆ, Maja J (2007). *The Robotics Primer.* 1st ed., Cambridge, MA. The MIT Press.

OPEN SOURCE INITIATIVE (2007). *The Open Source Definition.* [online]. Last updated 7 March 2016. https://opensource.org/osd

POLOLU (2016). *Zumo Robot for Arduino*. [online]. Last updated 7 March 2016. https://www.pololu.com/product/2506

SIMPSON, Jonathan, JACOBSEN, Christian and JADUD, Matthew (2006). *Mobile Robot Control: The subsumption Architecture and occam-pi*. Communicating Process Architectures 2006, **64** (1), 225- 236.

# Glossary

|  |  |
| --- | --- |
| Glossary Term | Glossary Definition |
| Actuator | These are the output of a device such as motors, lights and buzzers. |
| I/O | I/O is shorthand for Input/Output, which is the communication between an information processing system, such as a computer or microcontroller, and the outside world. I/O assumes that if an input happens then there must also be an output. |
| Internet of Things (IoT) | The practice of connecting electrical devices to the internet (and/or to each other) so that they can send and receive data. This includes everything from coffee makers to smart watches. |
|  |  |
| Microcontroller | A small computer that takes an input, usually from a sensor, and provides output to an actuator. An example of a Microcontroller would be an Arduino Uno. |
|  |  |
| Sensor | A device that can measure a physical quantity and convert it into a signal to pass to a computer, such as a microcontroller. An example of a sensor would be an ultrasonic sensor, a temperature gauge or a camera. |
|  |  |
|  |  |
|  |  |
|  |  |