

Obstacle avoidance system in vehicles through Arduino UNO

Author: Mohammed Sadiq Abuwala

Student ID: 23920203

Supervisor: Dr. Masoud Shakiba

Table of Contents

Abstract.....	3
Chapter 1:	4
1.1 Background	4
1.2 Project objectives	4
1.3 Functional requirements	5
1.4 Nonfunctional requirements	6
1.5 Project assumptions	7
1.6 Project constraints.....	7
Chapter 2:	7
2.1 Resource requirements	7
2.2 literature review	8
2.3 Risk analysis	8
Chapter 3:	9
3.1 Resource information.....	10
3.2 Software architecture.....	11
3.3 Implementation	13
3.3.1 Stage 1	13
3.3.2 Stage 2.....	15
Chapter 4:	21
4.1 Unit test	21
4.2 Black box test	22
4.3 Performance.....	23
Chapter 5:	23
5.1 Further discussion	23
5.2 Recommendations.....	24
5.3 Future work	24
Chapter 6: Conclusion	25
Chapter 7: Appendices	25
7.1 Workbook	25
7.2 Production & deployment.....	30
7.3Test report.....	34

ABSTRACT

The aim of the project is to modify and extend a vehicle using an Arduino UNO board to develop an autonomous car. In addition, the vehicle will be able to detect obstacles such as other road vehicles, road debris, etc. Once an obstacle is detected, brakes should automatically be applied and corrective steering actions should be taken. The vehicle will consist of sensors which will monitor the position of other objects and invoke a reduction of vehicle speed or by restricting vehicle movement to control its position. Obstacle avoidance systems can now be observed in most modern transportation services such as vehicles and airplanes. The invention of devices such as smartphones and music players have caused a steady increase in distracted driving and consumers are increasingly looking for better safe guards to protect them and their vehicle in the case of an imminent collision.

The current safety technology used in most modern vehicles consists of seat belts and airbags. Systems such as these can only help reduce damage in the event of an accident. They cannot protect the vehicle and only provide minimum protection to the passenger in the event of a collision. In contrast, obstacles avoidance systems are also known as “pre-crash systems” (i.e. they prevent an accident altogether). By pairing an autonomous car with an Obstacle Avoidance System (OAS), the aim is to create a smart car which will be much safer in operation than traditional machines.

CHAPTER 1 - INTRODUCTION

1.1. Background

The current safety systems used in most traditional vehicles include features such as seat belts and airbags. Such features do not prevent collisions but help minimize damage in the event of a collision and are hence known as Passive Safety Systems. As automotive technology evolves, newer safety features such as the OAS is being developed that are aimed at preventing an accident altogether.

With distracted driving on the rise due to reasons such as increased usage of modern inventions such as smart phones and music players while on the wheel, consumers are demanding more advanced safety systems that can help prevent accidents before they happen. 80% of collisions and 65% of near crashes have some form of driver inattention as contributing factors (National Highway Traffic Safety Administration, 2009). Sometimes, even though the driver is focused, they may not be aware of blind spots in their field of view (e.g. a person about to cross the road, etc.).

OAS are known to have other uses other than its application in road vehicles. For example, Obstacle Avoidance Systems can also be applied to unmanned vehicles which are sent to unknown environments such as disaster zones to autonomously drive around while avoiding obstacles to search for human survivors or simply map an area.

Based on the discussion above, the study states the problems as following:

- A steady number of people are suffering avoidable fatal crashes and injuries every year as there are no crash prevention features such as the OAS in traditional vehicles.
- Distracted driving has increased in recent times. Certain measures need to be taken to prevent or at least counter this problem.
- Drivers may sometimes be unable to detect blind spots in their field of view.

1.2. Project Objectives

To solve the problems stated above, an OAS will be implemented on a scaled version of a real world vehicle using an Arduino UNO board. The solution stated above can be broken down further into the following objectives:

- The vehicle should be able to maneuver itself (i.e. drive autonomously).
- The OAS being developed for the vehicle should be able to detect obstacles in front of it.

FINAL REPORT

- Additionally, the OAS should cause the car to brake in the event of an imminent collision and furthermore, find the best alternative route to drive towards.

1.3. Functional Requirements

The functional requirements consist of all the activities that it is required to perform (Satzinger, Jackson & Burd, 2000). They section below documents these activities as follows:

<u>Function</u>	<u>Description</u>
Power on the vehicle	The car can be powered on by plugging a power pack to the motor shield which is stacked on top of the Arduino UNO board. Once turned on, the Arduino will load the program and the car will start to move autonomously.
The car should be able to go in the left direction if required.	The vehicle can be made to go in the left direction by reducing power to the left wheel while the right wheel moves at the same power. This will cause the vehicle to move in the left direction.
The car should be able to go in the right direction if it is required.	The vehicle can be made to go in the right direction by reducing power to the right wheel while the left wheel moves at the same power. This will cause the vehicle to move in the right direction.
Brake the vehicle if it is required.	The vehicle can be made to brake by reducing the power going to each wheel from the motors to zero.
The vehicle should be able to detect obstacles	Once the vehicle is turned on, the sensors on the system will automatically and repeatedly keep scanning the area in front of the car to detect an obstacle.
Turn the sensors periodically to scan field of view for obstacles.	The sensor is attached to a servo motor which can move at specific angles and in the process turn the sensor around to scan its field of view.
The vehicle should be able to perform evasive maneuvers to avoid the detected obstacle.	When the OAS determines that a collision is imminent, the vehicle should automatically brake to avoid the collision. Furthermore, it should find the best route to avoid the detected obstacle and move in that direction. In the event that an obstacle is too close and the vehicle is unable to turn, the vehicle should reverse.

1.4. Non-functional Requirements

The non-functional requirements consist of all the characteristics other than its functionalities, such as the activities it must perform or support (Satzinger, Jackson & Burd, 2000). They are stated in the table below:

<u>Function</u>	<u>Description</u>
Usability	The vehicle should be easy to start up and well documented manuals should be available
Performance	The system should perform without any lag. The motors should not stall and the system should not hang or reset in the middle of operation and perform smoothly.
Security	Access to the files throughout development will only be available to the developer and project supervisor. An constant up-to-date backup of the program will also be taken from time to time to ensure system integrity is not let down in the event of a malfunction or malicious damage.
Reliability	The system should be available for use at all times. We also need to ensure that the failure rate for the service delivery expected by the end user is below the acceptable threshold.
Safety	Most semiconductor parts in electronic devices such as the one in the Arduino UNO board can generally withstand a temperature range of 0 to 70 degrees Celsius. Operating the vehicle in an environment outside this temperature range will cause the system to malfunction and thus should be avoided.
Design constraints	No sensors will be installed at the back of the vehicle and hence the vehicle will not be able to detect obstacles behind it.
Supportability	As the vehicle is small, it will be portable and hence can be transferred from one location to another easily. The system will not be designed to deal with other international metrics such as languages or number formats.

1.5. Project Assumption

A number of assumptions made while compiling the project objectives have been outlined below:

- The OAS will be designed to detect and avoid obstacles in front of the vehicle.
- An obstacle in this case is defined as an object which rises at least 10 cm above the ground.
- It is also assumed that the ground is flat. As the chosen chassis and motors are inexpensive but not durable enough to handle rough terrain, it is assumed that the vehicle will be operated on flat ground.

1.6. Project Constraints

A number of constraints were identified and have been stated below:

- While the OAS will continue to work in case a driver becomes inattentive or unresponsive, the system will not be designed to overcome mechanical failure. For example, in the event of a brake failure, the OAS will not be able to stop the car in the event of an imminent collision. The system also not be able to alert the driver in the event that the OAS fails.
- The system will be designed to only detect obstacles in front of it and not behind it.
- The system will only be able to detect obstacles which are above ground. The vehicle will not be able to detect obstacles below the ground such as potholes.

CHAPTER 2

2.1. Resource Requirement

The software requirements only consists of the Arduino UI which will be used to program the Arduino UNO board. The hardware requirements are listed below and further explanation is provided in the later sections:

HARDWARE NAME	DESCRIPTION
Arduino UNO kit	An micro-controller based board used to create interactive devices.
Ultrasonic sensor	Sensors which can detect how far an object is from its origin.
Vehicle chassis	Will be built and modified to implement an OAS.

Portable power pack	Used to power on the Arduino and hence run the vehicle.
---------------------	---

Further research on the project objectives provided knowledge that helped directly in making some key decisions. These are described below:

2.2. Literature Review

The sensors chosen for detecting obstacles are known as ultrasonic sensors. Existing studies note that ultrasonic range-finders, referred also as sonars, are known as robust and cheap distance measurement devices suitable for various applications, gathering of information from environment for real-world modeling as well as for navigating in mobile robotics. As they rely on sound waves to detect obstacles, the sensors are also capable of working just as accurately in dark conditions. In fact, it is reported that pedestrians are most likely to be killed in collisions between 3am and 6am, suggesting that drivers fail to recognize pedestrians walking on or crossing the road under dark conditions (NHTSA, 2008). Conventional “general-purpose” obstacle avoidance systems usually surround the robot with a ring of ultrasonic sensors (Cui, Wei, & Guo, 2011). In such a case, if one sensor malfunctions, the entire system fails. So many sensors on one vehicle will also take up too much space and power. As our vehicle is already small, there is not enough storage to fit so many sensors.

Therefore, it was decided to use a device called a servo motor which would be attached to a single ultrasonic sensor and would move the sensor periodically in different directions to scan its field of view for obstacles. Using this process, a singular ultrasonic sensor can cover the field of view that would usually be only possible with multiple ultrasonic sensors. If multiple sensors were used, in the event of a malfunction, we would first have to test every sensor individually to find which sensor is malfunctioning and then install a replacement. In contrast, if we use a single sensor as described above, structural design, maintenance, and replacement of the system would be much easier to conduct.

2.3. Risk Analysis

RISK	PROBABILITY	RISK REDUCTION STRATEGY
SCHEDULE RISKS		
Incorrect project time estimation	LOW	Project objectives should be clearly identified and understood. Make sure that there are no known scheduling constraints and that all resources are identified and committed.
Unexpected project scope	LOW	Project objectives are already clearly identified and project scope expansion

FINAL REPORT

expansions.		will only take place once primary objectives are fulfilled.
BUDGET RISKS		
Cost overruns	HIGH	Communicate with stakeholders about the budget requirements. Detailed and validated estimates should be identified and confirmed before submitting a budget plan. As a strict budget of RM300 needs to be followed, the risk of cost overruns are high.
Expansion of project scope	LOW	Stakeholders have agreed to communicate about budget extensions in the event of a project scope expansion.
OPERATIONAL RISKS		
Development team is unfamiliar with programming environment	HIGH	A large number of resources on Arduino Programming exists. The developer should use it to familiarize itself with the system. The developer can also communicate with the project supervisor regarding programming issues.
Integration of project modules	MEDIUM	
More than 1 facility involved in project production	LOW	Only one facility will be generally used for production which will help reduce transportation issues.
Sensor is very delicate and any mishaps during installation will cause a malfunctioning sensor to be installed	HIGH	Additional sensors should be kept in storage to ensure there are sufficient replacements.
Technological risks	LOW	Conventional off-the-shelf software will be used for the project. This will reduce

FINAL REPORT

		risks as such software have extensive resources available and are thoroughly tested.
EXTERNAL RISKS		
Project manager has no direct control of the groups supplying the resources	MEDIUM	Resources required will be gathered before project development begins. This way, the programmers will not have to wait for the resources to arrive during the development phase.

CHAPTER 3: METHODOLOGY

In this section, we will discuss more about the technical aspects of the OAS. Firstly, a brief explanation is provided to get ourselves familiar with the resources that are required. Since we have already explained about how we are using the ultrasonic sensor to measure how far obstacles are, in this next section, we will provide more information about the equipment being used.

3.1. Resource Information

- Arduino UNO kit: An Arduino UNO is a microcontroller based circuit board consisting of pins which can be used for both inputting and outputting data. The microcontroller can be connected to a computer via a USB port and is easily programmable through its own IDE. While the Arduino kit itself is inexpensive, the IDE can be downloaded for free from their website. The Arduino IDE uses a simplified version of the C++ programming language. The entire Arduino kit should at least consist of some jumper cables to make connections between the microcontroller and other equipment.

- Vehicle chassis: The chassis chosen was a 2-wheel drive (2WD) vehicle. The vehicle consists of two wheels and each wheel is connected by one DC motor. The chassis was chosen as it was durable enough to hold the weight of all the equipment needed to implement the OAS and also because the chassis chosen is easily available and inexpensive.

- Portable power pack: While the USB is disconnected from the computer, the Arduino board needs a portable power source to power itself. Each DC motor takes about 3V of power. The servo motor and the sensors also need considerable power. Therefore, it was calculated that at least 9V of power would be needed to run the vehicle. It was decided that 6AA batteries

providing 1.5V of power each would be used to power the vehicle. Assuming that there is enough space on the chassis to fit the power source, this was chosen to be a good solution.

3.2 Software architecture

The program consists of a single main class consisting of various methods to run the vehicle. It is shown in the UML class diagram below:

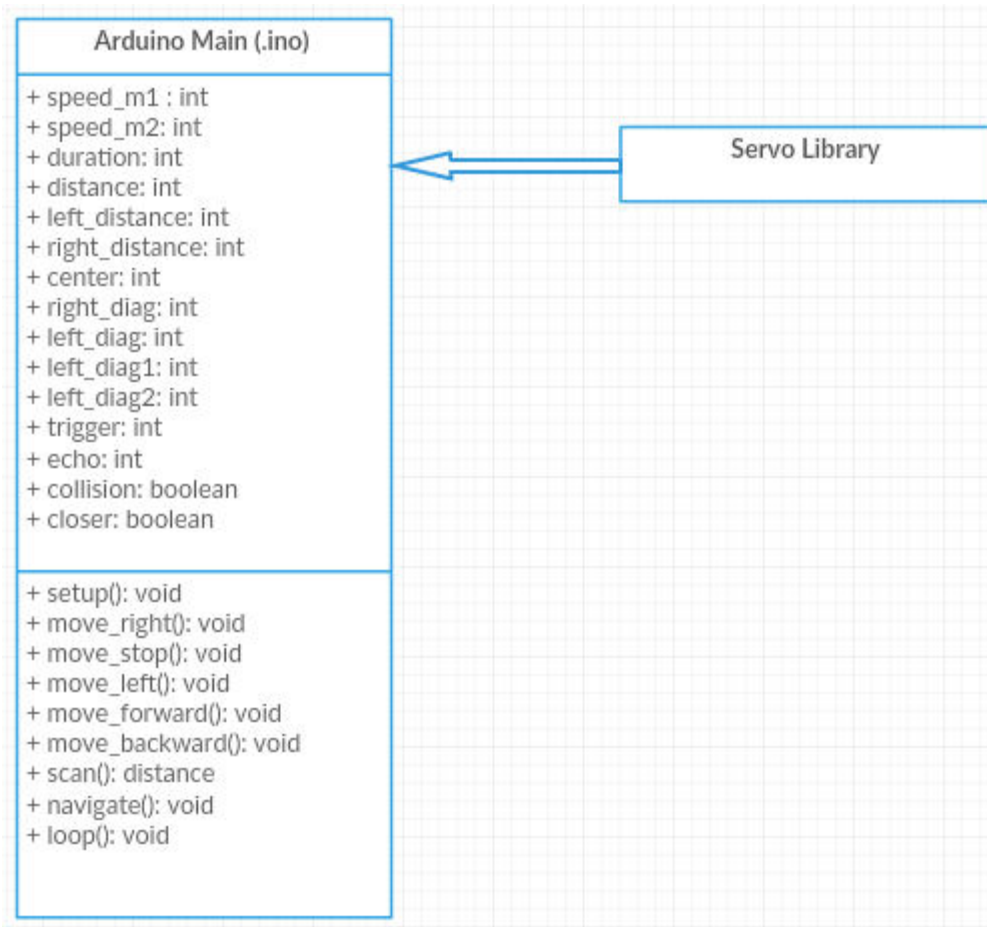


Figure 1 (UML Class diagram)

FINAL REPORT

The following use case diagram details the dynamic aspects of the systems. In other words, the diagram below illustrates all the usage requirements of the system:

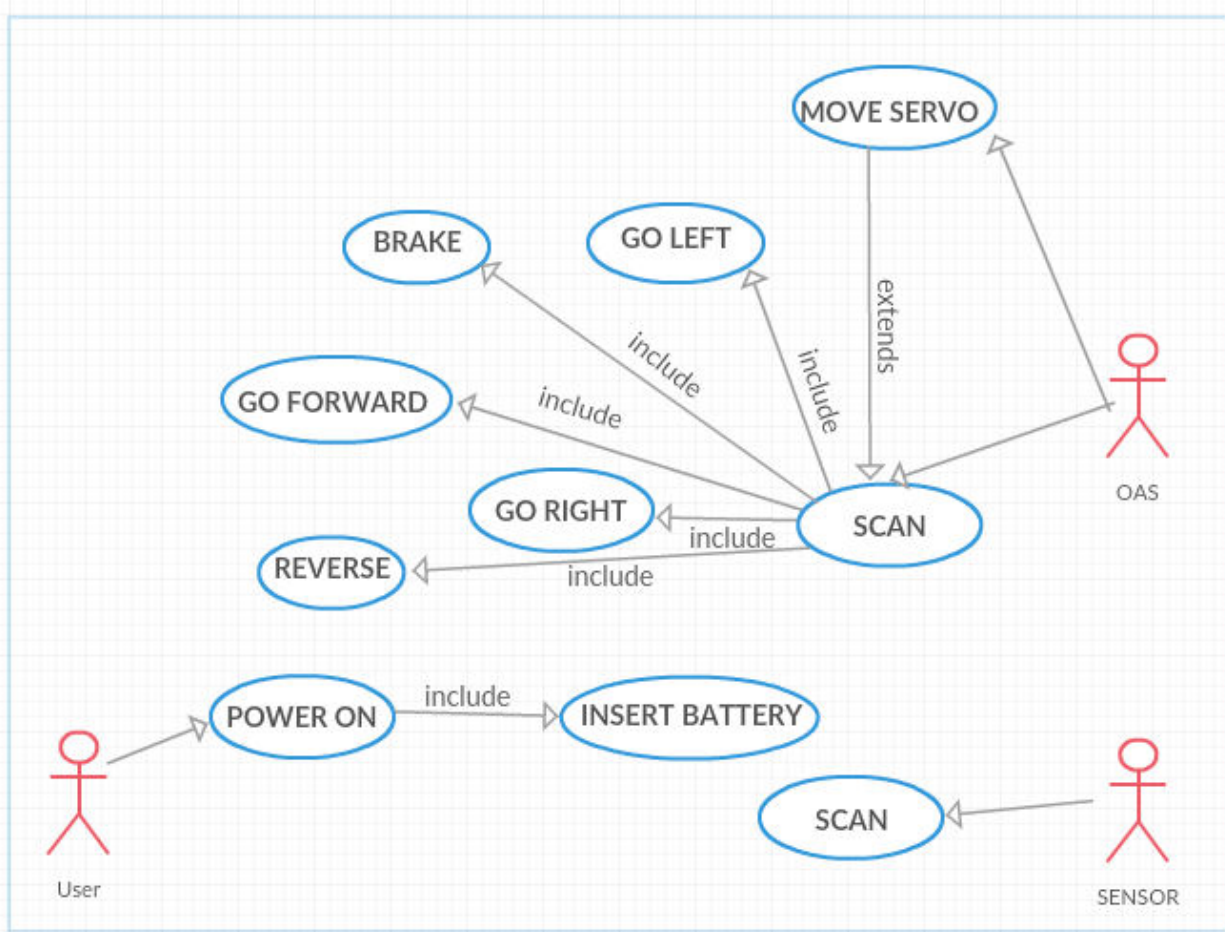


Figure 2 (Use case diagram)

Finally, an activity diagram below details the sequence of activities that takes place when the system is in operation:

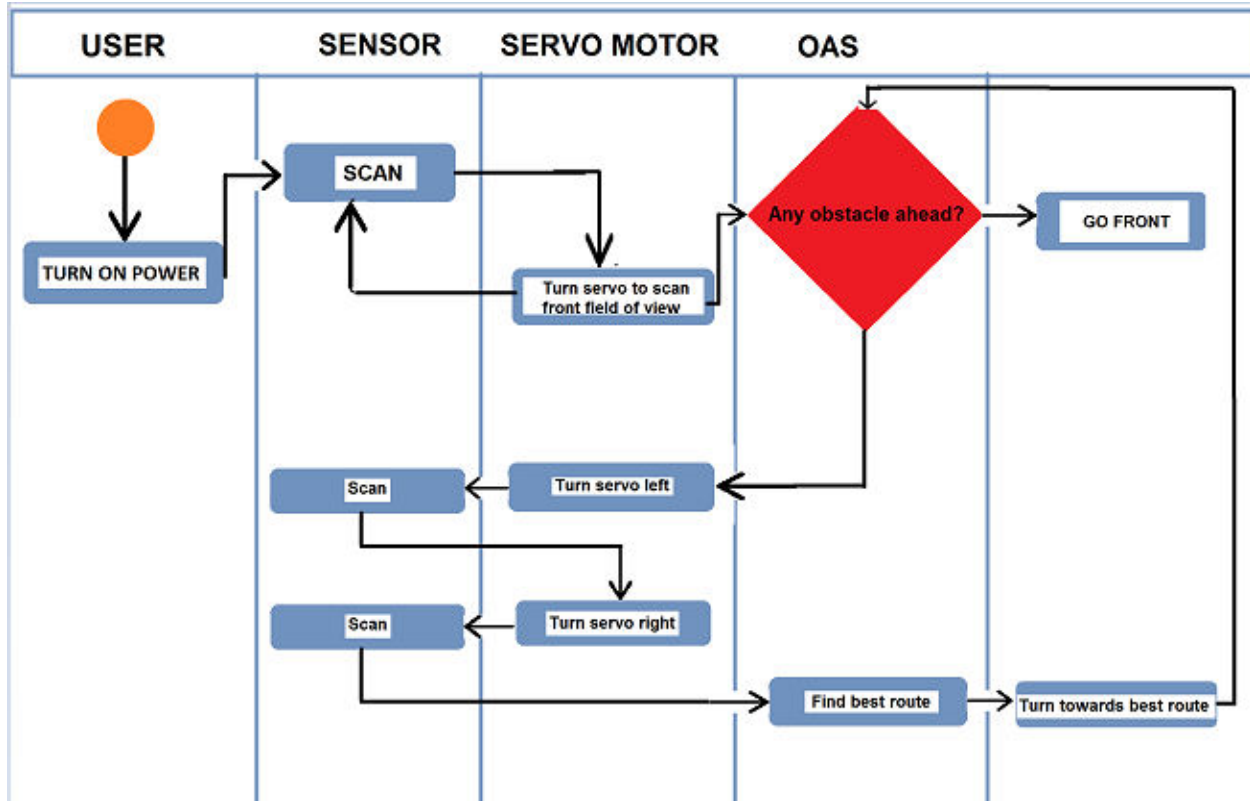


Figure 3 (Activity diagram)

3.3. Implementation

3.3.1. Stage 1 – Building the vehicle

The chassis was not ready out of the box and had to be built first. Detailed build instructions are provided in Appendix A. The wheels had to be connected to the chassis and the motors needed to be attached to the wheels. A portable holder which can hold 6 AA batteries was attached on top of the chassis with screws.

Once the chassis was built, a servo motor was attached at the center on the front end of the vehicle. An ultrasonic sensor was attached on top of the servo motor. The motor shield was stacked on top of the Arduino UNO board and the board itself was then screwed to the top of the chassis. Once all the motors and sensors needed for the OAS is installed onto the chassis, we can now connect them to the motor shield which is stacked on top of the Arduino UNO board.

FINAL REPORT

The connections are detailed in the following table:

ULTRASONIC SENSOR	
Trigger pin	8
Echo pin	9
Gnd	GND pin
Vcc	Vcc pin
DC MOTOR 1	
Pin 1	M1
Pin 2	M1

SERVO MOTOR	
Output pin	11
Gnd	GND pin
Vcc	3.3V pin
DC MOTOR 2	
Pin 1	M2
Pin 2	M2
POWER SUPPLY	
VCC	Motor shield VCC
GND	Motor shield GND

Figure 4 (Connections)

The images below show the vehicle after it was built and the sensors and motors are installed:

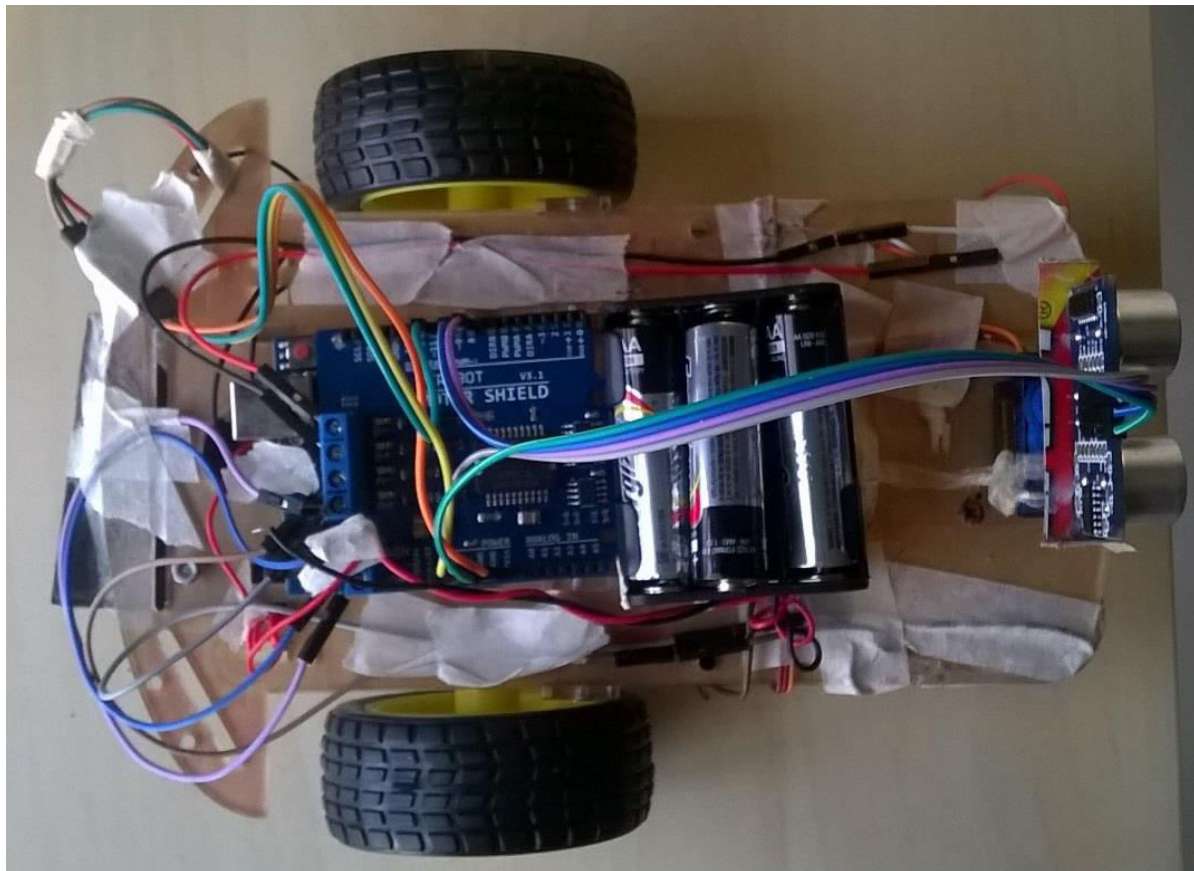


Figure 5 (Actual vehicle image from above)



Figure 5 (Actual vehicle seen from front)

3.3.2. Stage 2 – Code Implementation

Explanation about the code can be found within the program itself. Hence, in this section we will discuss some key functions which entail how some of the equipment installed on the vehicle actually operate:

- 1) Ultrasonic sensor: Each ultrasonic sensor may require a different calculation to measure distances as they each emit sound waves of different frequencies. The HCSR04 ultrasonic sensor works as follows:


```

66 int scan(){
67     digitalWrite(trigPin, LOW);
68     delayMicroseconds(2);
69     digitalWrite(trigPin, HIGH);
70     delayMicroseconds(10);
71     digitalWrite(trigPin, LOW);
72     duration = pulseIn(echoPin, HIGH);
73     distance = (duration / 2) / 29.1;
74     return distance;
75 }

```

Figure 6 (Ultrasonic sensor test stub)

By setting the trigger pin to low, we can clear data from the trigger pin to get it ready for the next calculation.

By setting the trigger pin to high, we can emit a soundwave.

Setting a delay of 10 micro seconds for the wave to echo back to the echo pin.

The trigger pin when set to low stops the trigger pin from emitting any longer.

The wave is echoed back at the echo pin and collected at the 'duration' variable and determines how long it took for the sound wave to echo back.

The distance calculated is divided by 2 as the soundwave has to travel and return and thus the distance first achieved is twice the actual distance.

2) Making the motor to run forward:

```

41 void move_forward(){
42
43     analogWrite(Speed_M1, 110);
44     analogWrite(Speed_M2, 110);
45     digitalWrite(M1, HIGH);
46     digitalWrite(M2, HIGH);
47 }

```

Figure 7 (DC motor test stub)

Sets the speed of motor 1 to 110

Sets the speed of motor 2 to 110

Setting the motor on HIGH makes it run in the forward direction (Setting it on LOW causes it to run in the opposite direction).

Sets the speed of motor 1 to 110

Sets the speed of motor 2 to 0

By setting the speed of motor 2 to 0, the left wheel stops running and only the right wheel will run. Hence, this will cause the vehicle to turn in the left direction. The same can be done

FINAL REPORT

3) Making the motor to turn left:

```
34 void move_left() {  
35     analogWrite(Speed_M1, 110);  
36     analogWrite(Speed_M2, 0);  
37     digitalWrite(M1, HIGH);  
38  
39 }
```

to turn the car to the right.

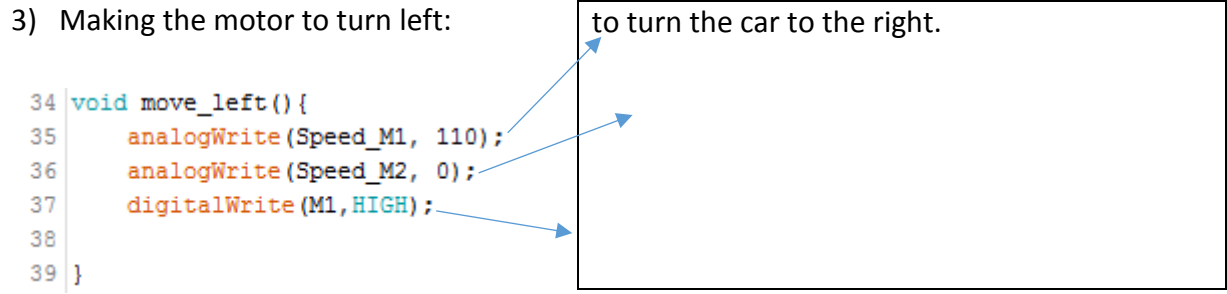
A diagram showing a code block for turning left and a rectangular box for turning right. Three blue arrows point from the code to the box: one from line 35 to the top-left corner, one from line 36 to the top-right corner, and one from line 37 to the bottom-left corner.

Figure 8 (DC motor test stub)

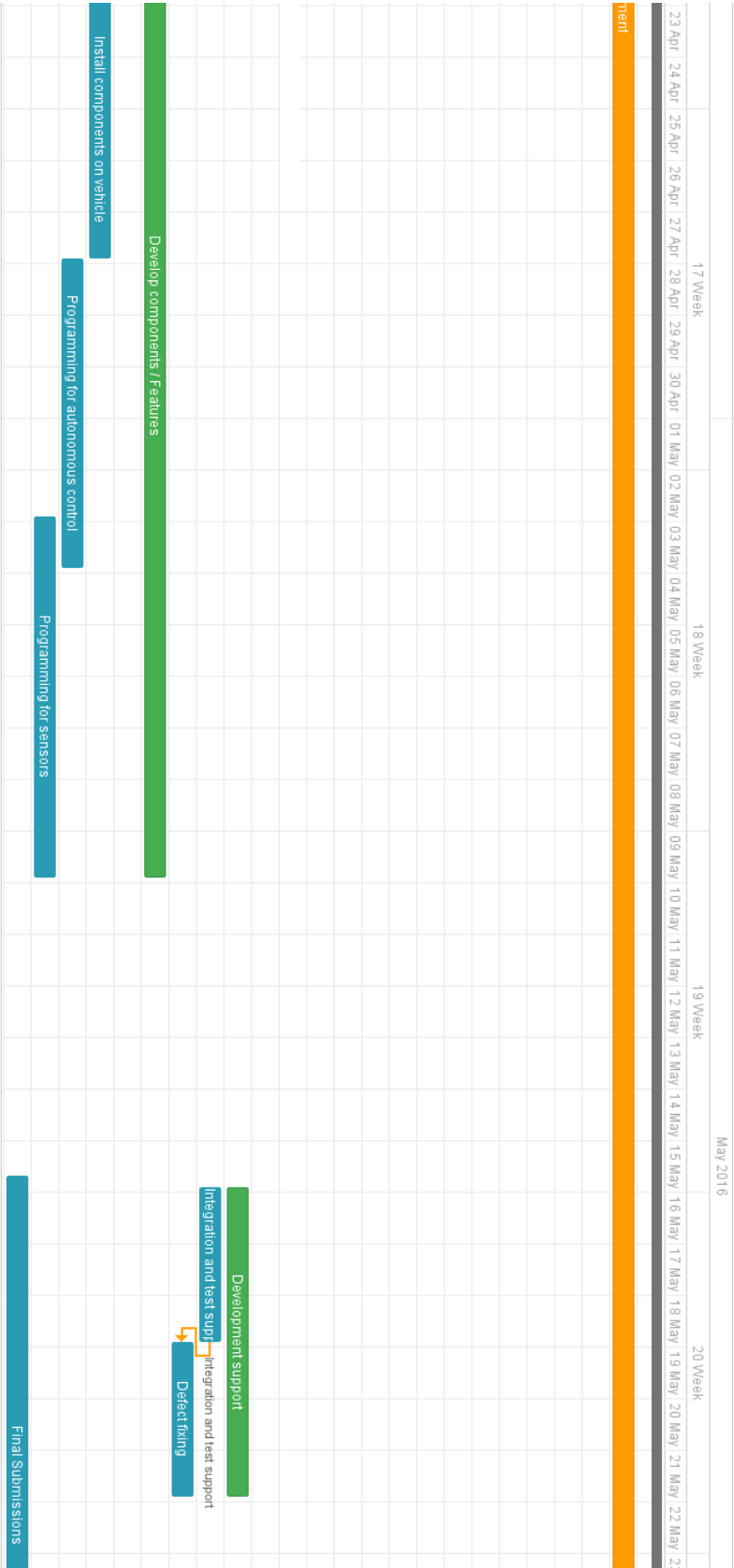
FINAL REPORT

The timeline of the project is shown below in both a tabular format and as a Gantt chart:

Task name	Start date	Duration day	Progress	Assigned
▼ Total Estimate	17/03/16 14:0	71.08		
▼ Software Development	17/03/16 14:0	71.08	100%	
▼ Project management	17/03/16 14:0	19.00	100%	
Managing / Monitoring	17/03/16 14:0	4.00	100%	Developer
Project Research	21/03/16 14:0	4.00	100%	Developer
Getting used to programming in /	25/03/16 14:0	11.00	100%	Developer
▼ Requirements	02/04/16 14:0	20.33	100%	
Gather resource requirements	02/04/16 14:0	3.00	100%	Developer
Initial Report	05/04/16 14:0	3.00	100%	Developer
▼ 3MT	08/04/16 21:2	14.00	100%	
3MT Presentation	08/04/16 21:2	6.00	100%	Developer
Project Specification	14/04/16 21:2	8.00	100%	Developer
▼ Architectural Definition	18/04/16 21:2	4.00	100%	
Define candidate architecture	18/04/16 21:2	4.00	100%	Developer
Refine architecture	18/04/16 21:2	3.00	100%	Developer
▼ Development support	15/05/16 21:2	6.00	100%	
Integration and test support	15/05/16 21:2	3.00	100%	Developer
Defect fixing	18/05/16 21:2	3.00	100%	Developer
▼ Develop components / Features	18/04/16 21:2	21.00	100%	
Build vehicle chassis	18/04/16 21:2	4.00	100%	Developer
Install components on vehicle	22/04/16 21:2	5.00	100%	Developer
Programming for autonomous co	27/04/16 21:2	6.00	100%	Developer
Programming for sensors	02/05/16 21:2	7.00	100%	Developer
Final Submissions	15/05/16 16:0	12.00	100%	Developer

Figure 9 (Time schedule for project)

FINAL REPORT



FINAL REPORT

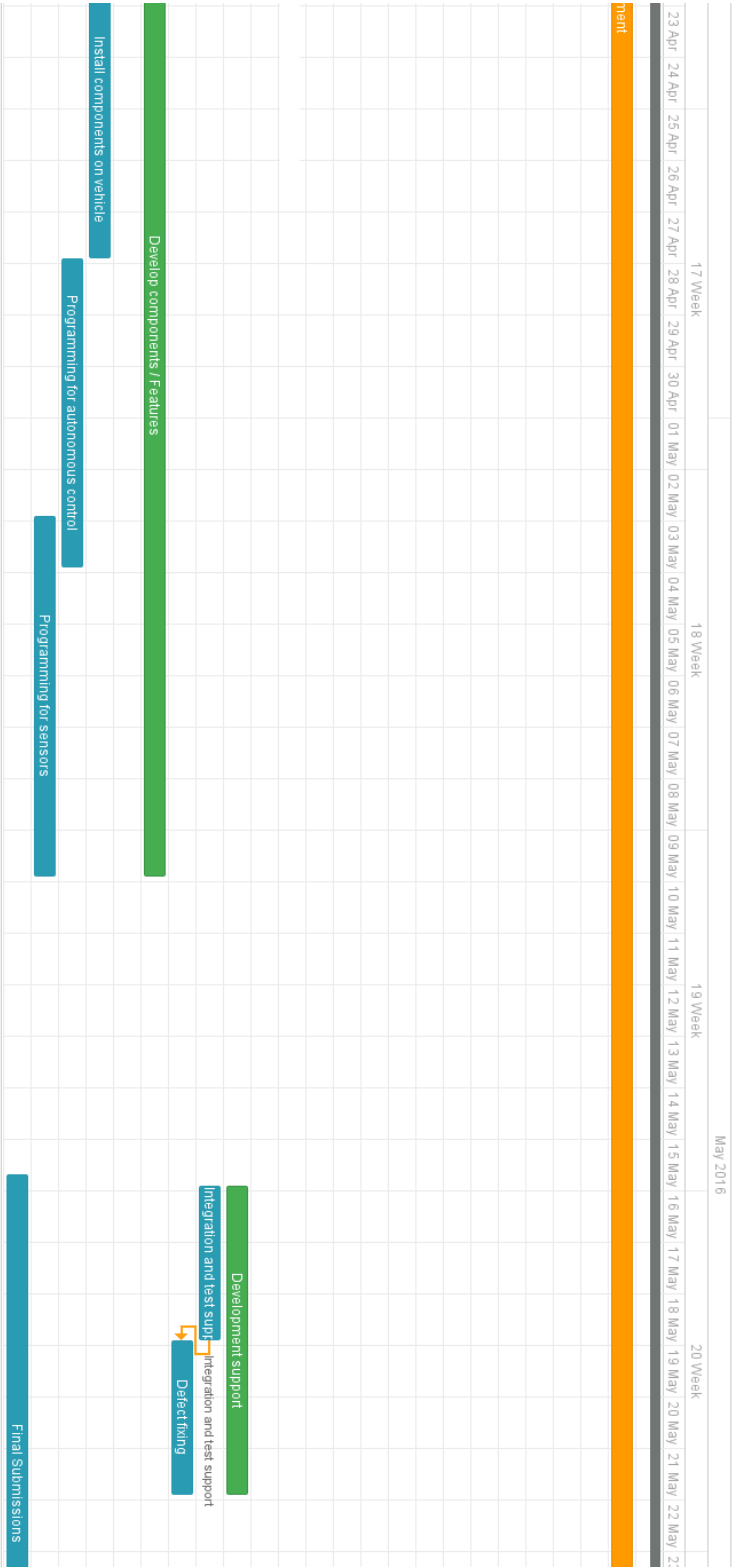


Figure 10 (Gantt chart)

4. RESULTS

4.1. Unit test

Unit testing was done to ensure that each component worked as required. In unit testing, each unit is tested separately before integrating them into modules to test the interfaces between modules. In this case, the operation of the servo motor, the DC motor, and the ultrasonic sensor are each separated into units and tested separately. Stubs are written for each unit and tested separately. The test stubs are shown below:

1) Test Stub for ultrasonic sensor

```
int trigger = 7;
int echo = 9;
int time_taken, distance;
void setup() {
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  Serial.begin(250000);
}
void loop() {
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  time_taken = pulseIn(echo, HIGH);
  distance = (time_taken / 2) / 29.1;
  Serial.println(distance);
}
```

EXPECTED RESULT	ACTUAL RESULT
The ultrasonic sensor should accurately measure how far an object is. As we have mentioned earlier that ultrasonic sensors are not very accurate, we consider an error of 5cms as reasonable.	The sensor accurately measures how far an object is from the sensor up to an accuracy of 5cms. Hence, the test stub was successful and is ready for module integration.

2) Test stub for servo motor

```
#include <Servo.h>
Servo servo;
void setup()
{
  Serial.begin(250000);
  servo.attach(11);
}
void loop()
{
  servo.write(90);
  delay(300);
  servo.write(110);
  delay(300);
  servo.write(130);
  delay(300);
  servo.write(73);
  delay(300);
  servo.write(53);
  delay(300);
}
```

EXPECTED RESULT	ACTUAL RESULT
In order to scan the field of view in front of the vehicle, the servo motor needs to move the ultrasonic sensor in 5 different angles to scan the entire field of view in front. Hence, the servo motor needs to turn in five different angles constantly.	The sensor accurately moves in five different angles accurately.

EXPECTED RESULT	ACTUAL RESULT
-----------------	---------------

FINAL REPORT

3) Test stub for DC motors

```
int speed_m1 = 5;
int m1 = 4;
int speed_m2 = 6;
int m2 = 7;
void setup()
{
    Serial.begin(250000);
    pinMode(m1, 5);
    pinMode(m2, 6);
}

void loop()
{
    analogWrite(speed_m1, 110);
    analogWrite(speed_m2, 110);
    digitalWrite(m1, HIGH);
    digitalWrite(m2, HIGH);
}
```

The motor should run in the forward direction.

The motors respond as expected by running in the forward direction.

4.2. Black-box test

We also need to ensure that the implemented system meets the functionalities agreed to at the beginning of the project. Black-box testing is a method of software testing that examines the functionalities of an application based on the specifications. It is also known as specification based testing (). The table below lists the objectives, the expected result and actual result of the black-box test:

TEST DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT
Power on the vehicle by installing AA batteries in the power source.	The lights on the Arduino should turn on indicating that the vehicle is now turned on.	The lights on the Arduino board turns on.
The vehicle starts moving automatically.	Once the batteries are involved, the vehicle should start scanning field of view ahead and find best route.	The vehicle starts moving as expected.
The servo motor moves the ultrasonic sensor automatically at 5 different angles to scan field of view.	To scan field of view ahead of vehicle, the servo motor should turn the ultrasonic sensor in programmed angles constantly.	The servo motor turns the sensor as expected.
If there is no obstacle ahead, the vehicle moves forward.	After scanning field of view, if there is no obstacle ahead, the vehicle will move forward.	The vehicle moves forward as expected.
Detect obstacles in field of view.	If an object is in front of the vehicle, the sensor should detect it and brake the vehicle to avoid a collision with the	The sensor does detect the obstacle and brake.

FINAL REPORT

	obstacle.	
Brake when an obstacle is detected.	As mentioned above, once an obstacle is detected, the vehicle should brake.	The vehicle does brake and avoid a collision.
Turn the sensor at 0 degrees and 180 degrees to find best route to avoid obstacle	The vehicle should turn at 0 degrees and 180 degrees to find best route around an obstacle once an obstacle is detected (i.e. there is an object in front of the vehicle).	The servo successfully turns the sensor at the programmed angles to scan field of view around the car to find best route.
Turn the car towards best route.	The vehicle should reduce power of the motor in the direction which the car wants to turn.	Power is reduced on the left motor if the vehicle wants to go in the left direction. To go in the right direction, the system reduces power to the right motor.
If vehicle is too close to an obstacle and is unable to turn, it should reverse.	If the vehicle brakes too close to an obstacle and is unable to turn without a collision, it should reverse.	The vehicle does reverse as expected.

4.3. Performance

Time complexity characteristics have been explained in the source code itself but have been briefly mentioned in the following section. The 'loop()' method acts like the "main" method in Python. It runs repeatedly as long as the Arduino is powered on. The method consists of rules which are compared and related methods are called accordingly. The time complexity of the program runs at $O(n)$. Additionally, the program also takes little space. The Arduino board has 32,256 bytes of storage memory available out of which approximately 5730 bytes are used (i.e. approximately 17% of storage).

CHAPTER 5: RECOMMENDATION

5.1 Further discussion

Based on the product evaluation, it was devised that the vehicle was not only capable of maneuvering through obstacles, it was also able to maneuver itself through more complex obstacles such as narrow paths. It is important to note that there are no sensors installed at the back of the vehicle. This means that the vehicle cannot detect obstacles behind it. This was not particularly a problem for the current project as the vehicle itself has the ability to turn at extremely steep angles and hence rarely requires to reverse.

Some additional issues were also encountered. It was found that for the circuit to operate accurately, it needed a more constant source of power which stays stable throughout operation. As the motors consume a low of power, the six AA batteries can only provide a running time of approximately ten minutes. Hence, using AA batteries for the operation of the vehicle is cumbersome as they have to be repeatedly changed and also expensive. In addition, the AA batteries are not capable of providing a continuous stable source of power to the vehicle.

Currently, the servo motor is simply glued to the sensor to allow it to rotate the sensor. This highly increases the probability of the sensor being damaged during installation.

Installing only one sensor instead of many around the vehicle also has its own drawbacks. The servo motor takes approximately 1.5 seconds to turn the sensor around and scan a field of view. This means that during that time, the vehicle cannot move and needs to brake. Hence, every time an obstacle is encountered, the vehicle needs to brake first so that the scanning of obstacles around can take place.

5.2. Recommendations

Based on the problems discussed above, the recommendations made are as following:

- Use a more stable power source such as a 10V LIPO or acid battery pack. These battery packs can be recharged once they are out of charge and are hence much cheaper to use in the long run.
- Sensor mounts can be used to hold the sensor safely in place. The mounts are built in such a way that they can be screwed to the servo motor and then the sensor can be screwed to the mount.

5.3. Future work

The OAS system can be set up in a number of ways to avoid obstacles. It would be interesting to implement a ring of ultrasonic sensors around the vehicle to measure how the scenario would compare in relation to using a singular sensor which is capable of turning at different angles (i.e. our current scenario).

The OAS system could be paired with other safety systems to make the vehicle safer to operate such as a 'cliff avoidance system'. The system would be able to detect obstacles below ground such as potholes and cliffs. A lane tracking system would also allow the vehicle to ensure that the vehicle never drifts off a lane.

CHAPTER 6. CONCLUSION

Obstacle avoidance systems have been integrated in most modern transportation services in recent times. As manufacturing costs decrease further in the near future, installation prices of such systems will also decrease. Additionally, such systems are getting increasingly popular as people realize that the safety benefits of such a system far outweighs its costs. Furthermore, as mentioned earlier, such systems are now being integrated on both small vehicles such as unmanned vehicles being sent to map an unknown environment or a disaster stricken zone where humans cannot be present but a portable autonomous vehicle as such could be used to search for survivors to larger vehicles such as airplanes which used OAS to look for other airplanes nearby.

The Arduino platform was extremely suitable for this project due to its open source nature. A large number of resources could be found detailing the use of every piece of equipment and the tools required are also inexpensive and readily available. All the stated goals specified in the project specification document were met except for the implementation of manual control of the vehicle due to time constraints. Furthermore, the manual control feature implementation was primarily considered an optional objective and it was hence decided to skip the feature. We look forward to installing additional features such as manual control, cliff avoidance, and lane tracking system in the future which would allow the vehicle to be even safer on the road.

CHAPTER 7: APPENDICES

7.1. Appendix A: Workbook

Week 3:

Tasks completed:

- Get familiar with Arduino UNO.
- Get familiar with C++ programming.

Problems encountered:

- Without any prior knowledge of the C++ programming language and Arduino platform, it took a considerable amount of time to familiarize with both platforms.

Upcoming tasks:

- Submit project proposal.

FINAL REPORT

- Submit resources required.

Week 4:

Tasks completed:

- Project proposal submitted.
- Resource requirements are gathered.
- Resources are ordered online.

Problems encountered:

- Strict budget requirements had to be met (Approximately: RM300).
- Resources needed to be bought online which took a considerable amount of time to be delivered (Approximately: 4 days).

Upcoming tasks:

- Build vehicle chassis.
- Decide on the position and the number of sensors that are required.
- Install power source to the chassis.

Week 5:

Tasks completed:

- Vehicle chassis is built.
- It is decided that one sensor is fitted at the center of the front end of the vehicle. The sensor installation is completed.
- A holder able to contain 6 AA batteries is installed on top of the chassis. The batteries will be able to provide approximately 9V of power which will be enough to power the vehicle.

Problems encountered:

- There are too many blind spots as one sensor attached at the front cannot detect obstacles diagonal to the vehicle or on the side of the vehicle. The position and the number of sensors to be installed on the vehicle need to be reconsidered.

Problems encountered:

- As mentioned above, the sensor is not able to cover all blind spots and the car crashes periodically. We need to rethink on the installation of the sensors.

Upcoming tasks:

- Decide on new implementation of the ultrasonic sensors.

FINAL REPORT

- Program the DC motors to run.
- Program the vehicle to turn.
- Program the vehicle to go forward.
- Program the vehicle to reverse.

Week 6:

Tasks completed:

- It was decided that a servo motor would be used that would be coupled with one ultrasonic sensor. The servo motor would rotate the sensor and allow it to scan entire field of view ahead of at its sides.
- Servo motor was ordered online.
- Dc motors are connected to the motor shield and can now successfully turn the vehicle forward and backwards.
- The vehicle is also able to turn in both directions.

Problems encountered:

- The DC motors turned out to be highly inaccurate. One provides less power than the others. In order to make the vehicle go in a straight direction, we need to increase the power sent to the motor which produces the lower power. Additionally, both motors do not seem to provide a stable power supply to the wheel making it extremely difficult for the vehicle to go in a straight direction.

Upcoming tasks:

- Install the servo motor on the chassis and attach the sensor to the motor.
- Familiarize with the servo motor and the ultrasonic sensor.
- Program each to work separately.

Week 7:

Tasks completed:

- Servo motor installed on the front end of the chassis at the center position.
- The sensor is glued on the rotary fan which is attached to the motor.
- Servo motor works successfully when programmed individually.
- Ultrasonic sensor works successfully when programmed to work individually.

Problems encountered:

- Sticking the sensor to the motor via glue was an inaccurate decision as the glue destroyed at least one sensor. That had to be subsequently replaced by a new sensor.

FINAL REPORT

Upcoming tasks:

- Integrate the servo motor with the ultrasonic sensor. The sensor should be turned at a sufficient amount of angles to capture the entire field of view in front of the vehicle.

Week 8:

Tasks completed:

- Ultrasonic sensor is successfully integrated with the servo motor. The servo motor now successfully turns the sensor at specific angles consistently to measure field of view.

Problems encountered:

- Initial number of scans were not enough. The number of angles at which measurements were taken were increased from 3 to 5 to eliminate blind spots. Now, although the system takes a longer time to scan the field of view ahead as the servo motor moves in two additional angles, it is more accurate as there are no blind spots in field of view.

Upcoming tasks:

- Integrate the servo motor and the sensor with the DC motors to run the vehicle based on if any obstacles are detected.
- Program the rules for the vehicle to run.

Week 9:

Tasks completed:

- Servo motor and sensor is integrated with the DC motors and the entire system can now run together.

Problems encountered:

- More refinements need to be made on how fast the vehicle turns, goes forward, and reverses.
- For reasons yet unidentified, the vehicle keeps driving in a loop periodically.
- The servo motor takes approximately 1.5 seconds to scan field of view. Hence, during that 1.5 seconds, obstacles will not be detected and the vehicle could crash.

Upcoming tasks:

- Refine the program so that the vehicle turns at approximately 30 degree intervals.

FINAL REPORT

- Tune the vehicle so that it does not run erratically.

Week 10:

Tasks completed:

- Vehicle turns more swiftly at approximately 30 degree intervals.
- The program is modified so that the vehicle can only cover a distance of approximately 30cms at once and then brakes to scan the area ahead again. This is done as scanning the field of view takes approximately 1.5 seconds during which the vehicle is not looking for obstacles at real time and hence could crash. By making the vehicle brake after travelling every 30cms, it can stop and look for obstacles ahead and then decide whether it should go any further. This solves the lag that we suffered last week during operation of the vehicle.

Problems encountered:

- As the vehicle does not run continuously and has to brake periodically to scan its field of view, it is more accurate but also slower. Even then, this scenario was implemented as the chosen option as it is very accurate at avoiding obstacles.

Upcoming tasks:

- Product testing.

Week 11:

Tasks completed:

- Unit testing was done to ensure each component (servo motor, DC motor, ultrasonic sensor) works accurately.
- Further black box testing was done to ensure vehicle runs as required and fulfills all its objectives.

Problems encountered:

- More refinements were made to ensure the vehicle runs at desired speed.

Upcoming tasks:

- Prepare for final presentation.
- Compile final project report.

Week 12:

Tasks completed:

- Preparation for final presentation completed.
- Final report completed.

Problems encountered:

- None.

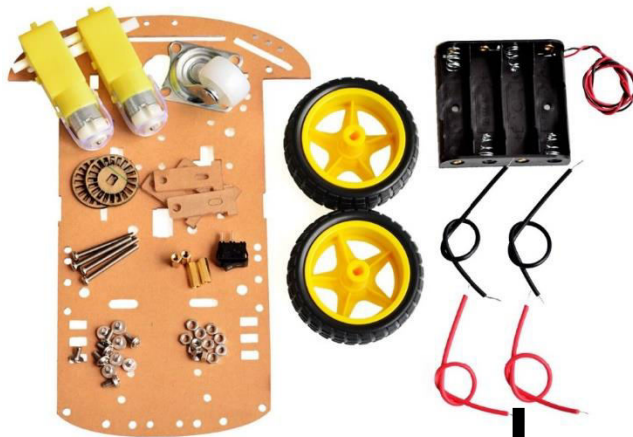
Upcoming tasks: None. Project has been completed.

7.2. Production and Deployment

7.2.1. Building the chassis

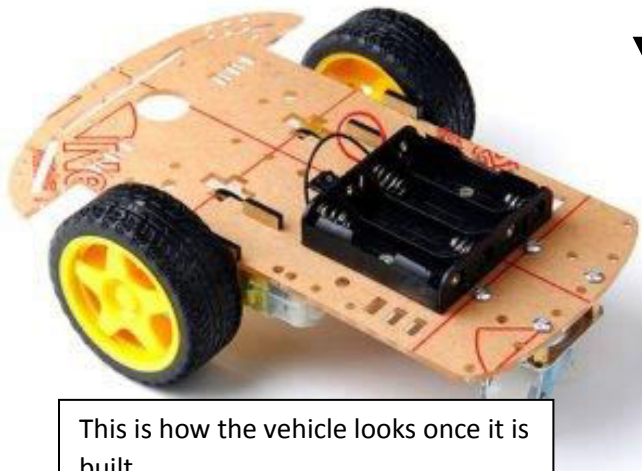
The chassis is readily available and can be inexpensively ordered online. The chassis kit comes with its own manual which is resourceful enough in building the chassis. Hence, build instructions of the chassis is not provided here and can be found in the manual that comes with the chassis.

Even then, the following images illustrate the start and the end product after the chassis is built. Although the images are not of the actual chassis, it is of the same chassis model and is for illustration purposes only.

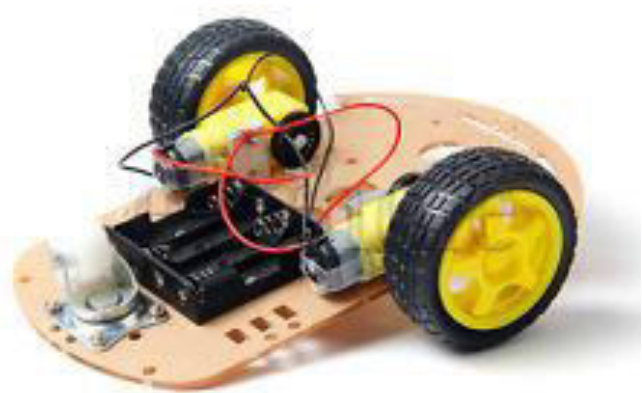


The kit consists of all the tools needed to build the chassis and are listed as following:

- 2 wheels
- Power container which can only hold 4 batteries. This need to be replaced so that 6 AA batteries can be held
- Screws to attach the required equipment.
- Caster wheel at the front which helps the vehicle turns.
- 2 DC motors to power the wheels



This is how the vehicle looks once it is built.



This is how the vehicle looks underneath.

7.2.2. Wiring the circuit

The connections required to complete the circuit are detailed in the table below:

ULTRASONIC SENSOR	
Trigger pin	Digital pin 8
Echo pin	Digital pin 9
Gnd	Motor shield GND pin
Vcc	Motor shield VCC pin

DC MOTOR 1	
Pin 1	Motor shield M1 slot
Pin 2	Motor shield M1 slot

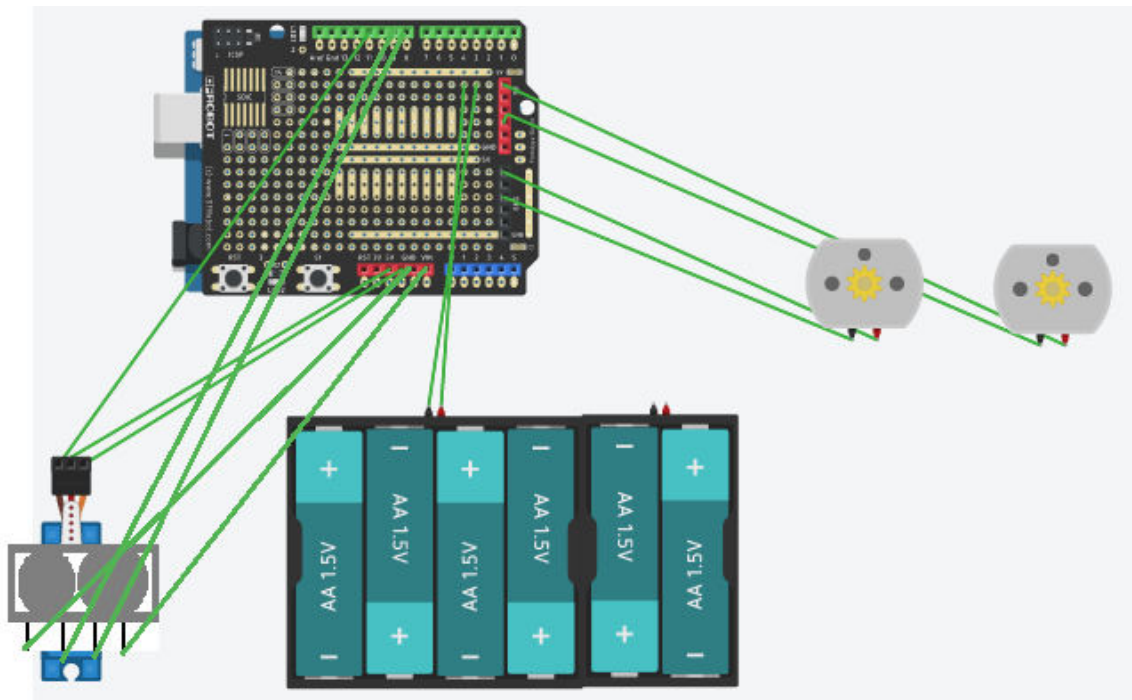
POWER SUPPLY	
VCC	Motor shield VCC
GND	Motor shield GND

SERVO MOTOR	
Output pin	Digital pin 11
Gnd	Arduino GND pin
Vcc	3.3V motor shield pin

DC MOTOR 2	
Pin 1	Motor shield M2 slot
Pin 2	Motor shield M2 slot

Notes: The servo motor should be connected to the 3.3V power supply pin. The 5V power supply pin should be ignored for this particular servo motor as servo motors take significantly lesser power than DC motors. Using the 5V pin will hence cause the sensor to start stuttering.

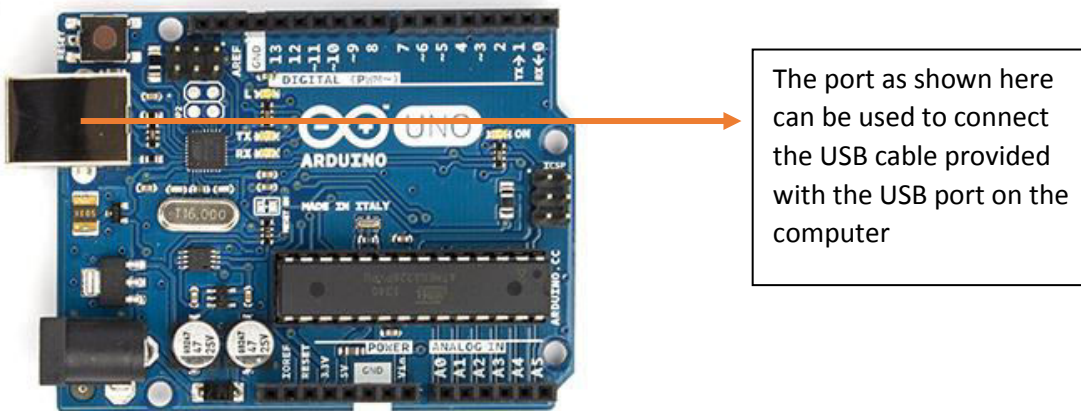
A circuit diagram is also shown below. Note that the diagram is drawn solely for illustrating the connections of the circuit and thus not show the actual equipment being used:



7.2.3 Uploading program to the Arduino board

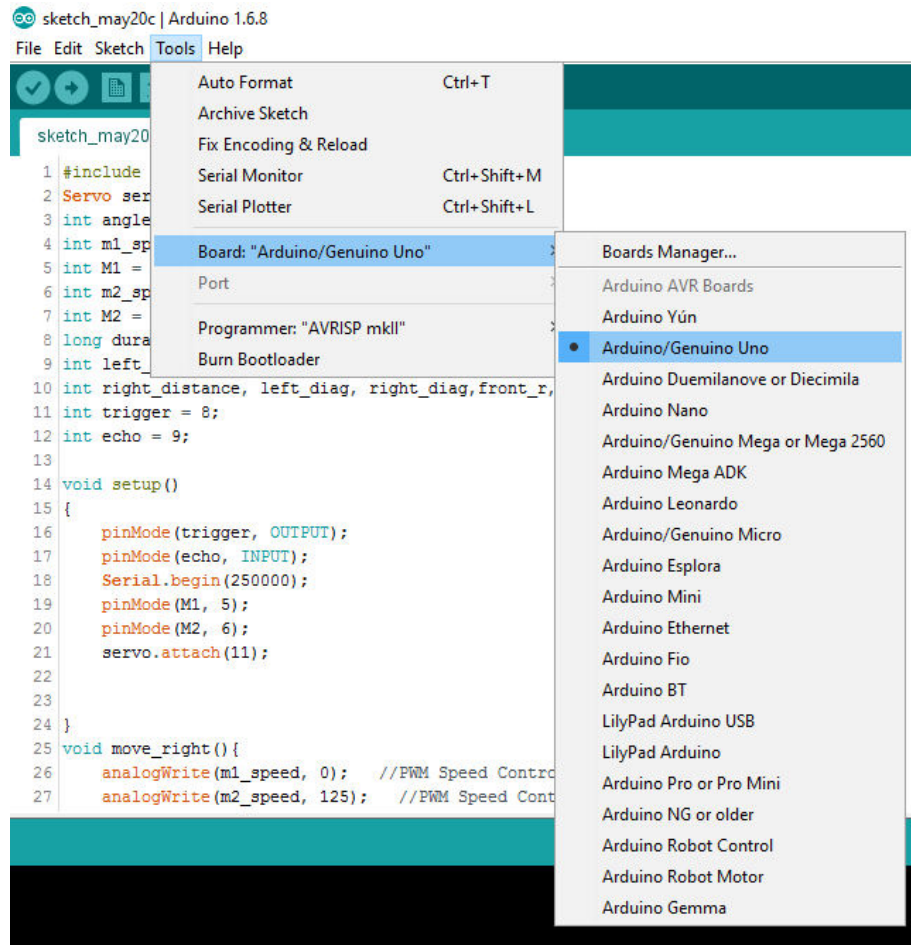
To upload the program onto the Arduino board, we firstly need to install the Arduino IDE on a computer. The IDE is available for free and can be downloaded from their website (<https://www.arduino.cc/en/Main/Software>). The installation of the IDE is very simple and does not need to be discussed. Once the IDE is installed, open the program in the IDE. Before we can upload the program, we need to take a simple step to ensure that Arduino UNO board is recognized by the IDE. In a series of steps below, a detailed description is provided on how to upload the program to the Arduino board:

Step 1: Connect the Arduino board to the computer.



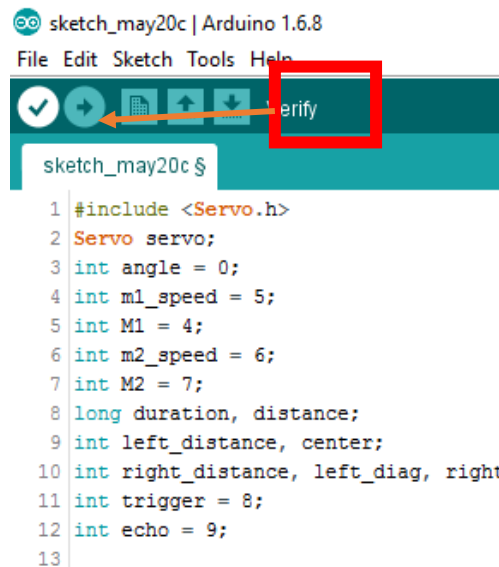
FINAL REPORT

Step 2: Make sure that once connected, the IDE recognizes the Arduino UNO board.



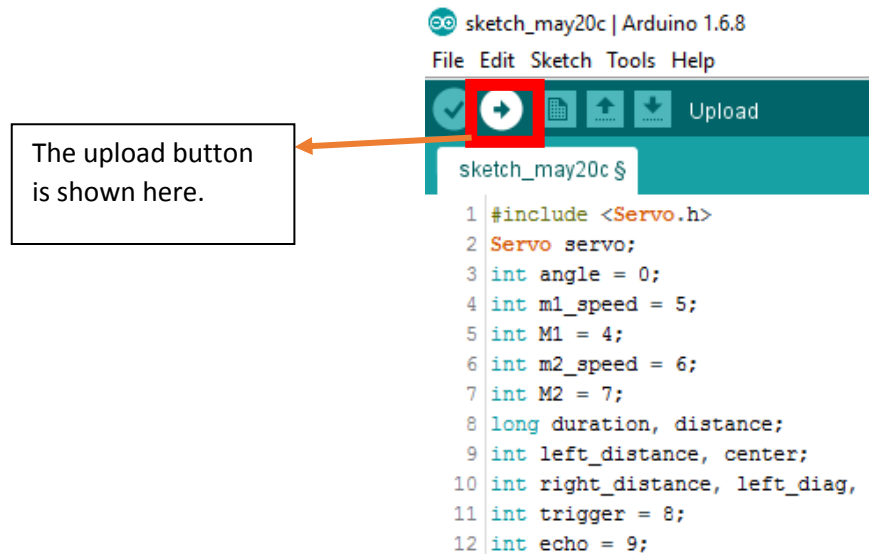
Step 3: Compile the program by clicking on the 'verify button' to ensure there are no errors in the program.

The verify button is shown here.



FINAL REPORT

Step 4: Click on the 'upload button' to upload the program onto the Arduino board.



7.2.4. Provide power supply to vehicle

Finally, we insert 6 AA batteries into the battery holder. As soon as the batteries are inserted, the LED light on the Arduino should turn on. The vehicle powers itself on and the program starts running on the Arduino board.

7.3. Appendix B: Test Report

The system was divided into multiple components for testing: the operation of the ultrasonic sensor, servo motor, DC motors, power supply, and the OAS combined:

Test 1:

Start Date: 18/05/2016

End Date: 19/05/2016

The testing will be conducted on individual components first to ensure they are functioning accurately. The following features are included in the test:

- Testing to check if the ultrasonic sensor is still functioning accurately.
- Testing to check if the servo motor is functions accurately.

Expected outcomes:

- The ultrasonic sensor should be able to detect obstacles up to an accuracy of 5cms.

FINAL REPORT

- The servo motor should turn accurately at the programmed angles and should not overshoot or undershoot the target angle.

Results:

The goal of the test is to mitigate any technical flaws that may arise during the operation of the actual system:

- Use the test stub for the ultrasonic sensor to measure distances of objects. The sensor should provide an accuracy of up to 5cms which is chosen to be acceptable for our system.
- Use the test stub for the servo motor to test the component individually. By turning the servo at 0 degrees and 180 degrees, we can check to what accuracy the motor turns. If the servo overshoots, we would have to tune the servo through our program by making up for the difference.

Test 2:

Start date: 19/05/2016

End data: 20/05/2016

Next, we will test another individual component called the DC motors to test that they are working as programmed:

- Test the DC motor to see that they are producing power and the wheels turn forward.
- Test the DC motors to check if they can reverse current and turn the wheel backwards.
- Test the DC motors to check if the speed can be controlled through the program

Expected outcomes:

- The DC motors should run and produce a 'forward' current to turn the wheels forward.
- The DC motor should run and produce a reverse current to turn the wheel backwards.
- Test the motor at different speeds to see if the speed of the motors can be controlled.

Results:

- The DC motors are able to turn the wheels forward.
- The DC motors are able to reverse current to turn the wheels backwards.

FINAL REPORT

- Setting the speeds at (0 (minimum speed), 100, 180, 225(maximum speed)), it was discovered that the DC motors reacted sufficiently although it was already previously discovered that one of the DC motors was slightly faulty and produced less power and hence had to be tuned in comparison to the second DC motor.

Test 3:

Start date: 20/05/2016

End date: 20/05/2016

In this test, we have measured the power supply for power output and stability:

- Test that the 6 AA batteries are able to provide the power needed to operate the vehicle.
- Also, ensure that the supply of power is stable to keep the device running smoothly.

Expected outcomes:

- The DC motors or the servo motors should not stall. Motor stall is a sign of an unstable power source. If the motor stops running anytime during operation, it means that the power supply is not stable. Stuttering of servo motors is also another sign of unstable power source.
- A volt meter is used to test that at least 9V of power is provided to the Arduino board for smooth operation.

Results:

- The motors never stalled during operation and ran smoothly throughout.
- A volt meter confirmed that a new pack of 6 AA batteries was able to generate at least 9V of power enabling the device to run smoothly.

Test 4:

Start date: 20/05/2016

End data: 20/05/2016

In this test, we will check to see if the OAS brakes the vehicle when it detects an obstacle. Consequently, the vehicle should also go forward if there are no obstacles:

- The vehicle should brake if there is an obstacle ahead and it cannot go forward.
- The vehicle should go forward if there are no obstacles ahead.

FINAL REPORT

- The vehicle should reverse if it is too close to an obstacle and is unable to turn.

Expected outcomes:

- Power the vehicle on and keep an obstacle approximately 40cms in front of the vehicle. The vehicle should stop when it detects the obstacle.
- Power the vehicle on and keep no obstacles ahead of the vehicle. The car should then travel forward.
- Power the vehicle on and keep an obstacle at approximately 5 cms in front of the vehicle. The vehicle should not go forward and should reverse.

Results:

- The vehicle brakes when it detects an obstacle ahead of it.
- The vehicle goes forward when there are no obstacles ahead of it.
- The vehicle reverses when it is too close to an obstacle.

Test 5:

Start date: 20/05/2016

End date: 20/05/2016

In the following section, we will test to check if the vehicle turns when there is an obstacle in front of the vehicle:

- The vehicle should turn left when there is an obstacle ahead and on the right and there are no obstacles at the left of the vehicle.
- The vehicle should turn right when there is an obstacle ahead and on the left of the vehicle and there are no obstacles at the right of the vehicle.

Expected outcomes:

- The vehicle should avoid the obstacle and turn left when there is an obstacle ahead and on the right.
- The vehicle should avoid the obstacle and turn right when there is an obstacle ahead and on the left.

Results:

- An obstacle was kept at approximately 10cms away at the front and on the left of the vehicle. The vehicle scans the area and then appropriately turns right.
- An obstacle was kept at approximately 10cms away at the front and on the right of the vehicle. The vehicle scans the area and then appropriately turns left.

FINAL REPORT

Test 6:

Start date: 21/05/2016

End date: 21/05/2016

Tests were done to ensure that obstacles were detected on the diagonals sections of the vehicle in front of it. Scanning the entire field of view ahead ensures there are no blind spots.

- Check if obstacles are detected at approximately 45 degrees ahead of the vehicles in relation to the sensor.
- Also, test if obstacle detection is done at the right diagonal (135 degrees).
- When the vehicle detects an obstacle, the system should turn the servo motors at left and right (0 degrees and 180 degrees) to scan the area on the left and on the right. Ensure that the operation is done as expected.

Expected outcomes:

- The vehicle should stop when it detects obstacles at its diagonal.
- The servo motor should turn at 0 and 180 degrees to scan area when there is an obstacle ahead and it cannot go forward.

Results:

- We tested the first condition by placing obstacles at diagonals in front of the vehicle at approximately 10cms away. The vehicle was able to detect the obstacle and does not go forward expected.
- When there is an obstacle ahead, the system stops the vehicle and the servo motor moves at 0 and 180 degrees to scan the area on the left and on the right to find best route.

REFERENCES

CAA *Distracted Driving*. (2016). *Distracteddriving.caa.ca*. Retrieved 29 May 2016, from <http://distracteddriving.caa.ca/education/>

Satzinger, J., Jackson, R., & Burd, S. (2000). *Systems analysis and design in a changing world*. Cambridge, MA: Course Technology.

NHTSA,. (2008). *National Pedestrian Crash Reprt*. Virginia: NHTSA.

Cui, G., Wei, W., & Guo, J. (2011). Obstacle Avoidance System Based on Single Rotary Ultrasonic Sensor. *AMM*, 55-57, 521-526. <http://dx.doi.org/10.4028/www.scientific.net/amm.55-57.521>