DEVELOPING ALGORITHM FOR OBJECT TRACKING USING PASSIVE SENSORS

By

CHEAM SYEH REN

FINAL PROJECT REPORT

Submitted to the Department of Electrical and Electronic Engineering

in Partial Fulfilment of the Requirements

for the Degree

Bachelor of Engineering (Hons)

(Electrical and Electronic Engineering)

Univeristi Teknologi PETRONAS

Bandar Sri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

DEVELOPING ALGORITHM FOR OBJECT TRACKING USING PASSIVE SENSORS

by

Cheam Syeh Ren
13044

A project dissertation submitted to the
Department of Electrical and Electronic Engineering
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical and Electronic Engineering)

Approved:

_____

Mr Abu Bakar Sayuti b Hj Mohd Saman
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

SEPTEMBER 2013

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

Cheam Syeh Ren

ABSTRACT

Platooning system is a type of cooperative driving system where several vehicles are aligned to form a chain of vehicles just like a train form. In this project, an unguided system of platooning is designed, where a manually controlled leader vehicle is guiding the following vehicle that is moving autonomously. Object tracking is one of the features in platooning. In order to perform object tracking, advanced sensors are usually used in platooning for example laser range finder, camera and GPS. Advanced sensors are usually used because they can handle complex situations for example on the road or highway. However, if the sensors are used for a smaller and simple layout such as in a building, they appear to be high in expenditure. Hence, this paper serves to describe the use of simple and low-cost passive sensors to replace the advanced sensors in developing an object-tracking system which is more appropriate for educational purpose. Integrating simple sensors in object-tracking may not produce excellent results as compared to advanced sensors but with an appropriate algorithm, it is hoped that at least simple leader/follower behaviour could be demonstrated.

ACKNOWLEDGEMENT

Throughout this project, many people has been contributing and assisting the author in succeeding the project. The author would like to acknowledge:

First and foremost, the project supervisor Mr. Abu Bakar Sayuti for the guidance given, the knowledge taught and all his effort in helping the author in her project. He helped a lot in improving the author's work. In fact, the author was able to stay in progress because of the kind assistance from him in discussing the problems and solutions.

The author would like to extend a sincere gratitude and appreciation to the author's family for both the mental and physical support. This helps the author to progress the project smoothly.

Last but not least, the author would like to thank all parties who have helped in this project and making it a success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 Background of Study

Platooning is a formation of vehicles moving in a straight line. One vehicle will lead the platoon while the following vehicles follow it with the same velocity within a safety distance. It is implemented to create a traffic that is organised and increases the road capacity. This can be done since a platoon is a group of vehicles reacting in a coordinated way.[1] Establishing vehicle platoon will results in increased road capacity, better efficiency, reduced congestion and of course less air pollution. California PATH[1, 2] and SARTRE[3] are among the projects that have been immersed in platoon research for a long while.

To produce a platoon, the following vehicles can be in a guided environment or an unguided environment. In guided environment, the routed path is usually equipped with beacons or electromagnet sensors to help the vehicles to stay in the path. Meanwhile, in unguided environment, the routed path is not modified. Therefore, in this case, often the platoon is led by a vehicle that is manually controlled by experienced driver as described in [4].

In platooning system, object tracking would be a vital feature as the follower vehicle needs to track the location of the leader vehicle in order to follow it. To demonstrate object tracking, many tracking sensors are used such as camera, laser range finder, GPS, etc. These advanced sensors definitely produce good results but they are high in expenditure. In this project, the purpose is to develop an object tracking system but at a much affordable cost.

The most famous passive sensors are infrared and ultrasonic sensors which have been very widely used in obstacle avoidance. In comparison between these two types of sensors, IR sensors are lower in cost and have higher response time than ultrasonic sensor. Other than detecting objects, IR sensors can also be used to estimate distance. The disadvantage of using IR sensor is that its reflected amplitude of the surrounding is non-linear and it is very much dependent on the reflectance strength of the object.

Ultrasonic sensors have a higher precision of distance measurement (less than 1 cm and up to 6m) but the fast-response of the IR sensor is more attracting for real-time response [5]. Based on the characteristics of both sensors, IR sensor is selected as the object-tracker while ultrasonic sensor is selected to give distance measurement. More details on the sensors are described in Section 2.0.

To consider object tracking, the exact positioning of the targeted object is important. The position of the targeted object can be identified by collecting two types of information: the distance of the targeted object from the experimental vehicle and the direction angle from the vehicle's heading to the targeted object.

1.2 Problem Statements

The problem statements can be summarised as follow:

- The expenditure of advanced sensors used in object tracking is costly for undergraduate educational level.
- The effectiveness and accuracy of passive sensor for object tracking system is yet to be determined.

Object tracking is a vital feature in platooning system. In the past, prior work on platooning has been using variety approaches such as GPS modules, LIDAR (or laser range finder) and camera for the object tracking. However, these sensors are high in expenditures.

Since the purpose of the project is producing a much affordable platform for object-tracking system, therefore simpler and lower cost sensors like IR sensors are used. There are many applications used in IR sensors for example to detect the brightness of a room, to detect motion and also obstacle avoidance. But in the case of tracking an object, the accuracy and effectiveness of this sensor is yet to be determined.

Object tracking system is much more complex compared to the normal applications of IR sensor as mentioned above because object tracking does not merely consider the presence of the targeted object but it also needs to consider exactly the position of the object and the distance to the object. Therefore, a suitable algorithm needs to be developed and experimented to see if it works. Through the experiments result, the effectiveness of IR sensors is investigated whether they would be appropriate to be enhanced in terms of its application towards object tracking.

1.3 Objectives

The project is to build a working platform where an experimental vehicle is able to track the leader and follow the trail of the leader. In conjunction to this, the following are the objectives of the project:

- To produce an object tracking system appropriate for educational purpose at undergraduate level
- To produce an object tracking system using simple and low-cost sensors
- To investigate the suitability of using passive sensors in object-tracking system
- To develop an algorithm best suited to operate simple sensors for object-tracking

1.4 Scope of Study

This project aims at providing an algorithm to object tracking system using the passive sensors which are lower in cost than advanced sensors. The project is mainly focused on the uses of sensors and any other electronics components. Hence, many past and in-project papers regarding the passive sensors' applications will be the reference to this project. In the end, the algorithm developed shall be simple and understandable. There are also few limitations encountered. These include the inadequate availability of materials required, the effectiveness of the passive sensors may not be comparable to advanced sensors and the lack of experience in the field. However, all these limitations are overcome accordingly to suit the most basic needs and objectives. For example, the unavailable materials are replaced to something similar and thus the program is altered to follow the product specification. Meanwhile, for the case of passive sensors' effectiveness, the results of object tracking system are produced to be as close as possible as long as the proof of concept is observed. Last but not least, many reviews will be done to overcome the lack of experience.

2.0 LITERATURE REVIEW

Platooning system is one of the promising applications in the fields of Intelligent Transportation Systems (ITS). Platooning means creating a cluster of vehicles in the form of a line and the following vehicles following the leader are usually autonomous i.e. the car is moving without the action from the driver. The figure below shows the formation of a platoon. The purpose of platooning in ITS is to reduce traffic congestion as well as improving the safety and comfort in driving. [6]

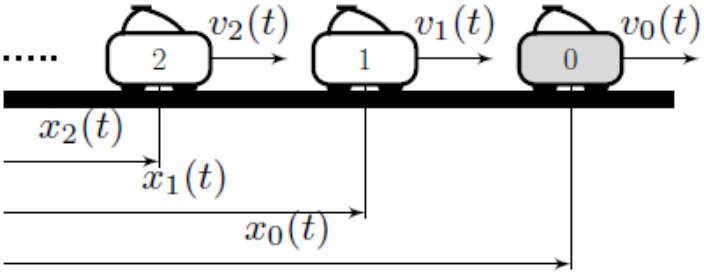


Figure 1: Platooning with a leader vehicle [7]

A platform scenario that is similar to [8] is to be created. The platooning only consists of two vehicles. It is the same for this project except laser range finder is replaced with IR sensor. In [8], the control algorithm used actually considers a more complicated situation i.e. the vehicles can break from platoon and join back into the platoon through the additional function of communication in the system. However, the basis of the algorithm can still be implemented. The control algorithm from the paper is shown in Figure 2. From the algorithm stated, a safety decision factor should be considered into the project. For example, if the distance is too close, then the vehicle needs to back up and if the distance is further than the safety distance then the vehicle proceeds to next operation. It can be considered as well to take into account the best direction driving calculation. Considering these factors, finally the algortihm of this project is designed as shown in Figure 8.

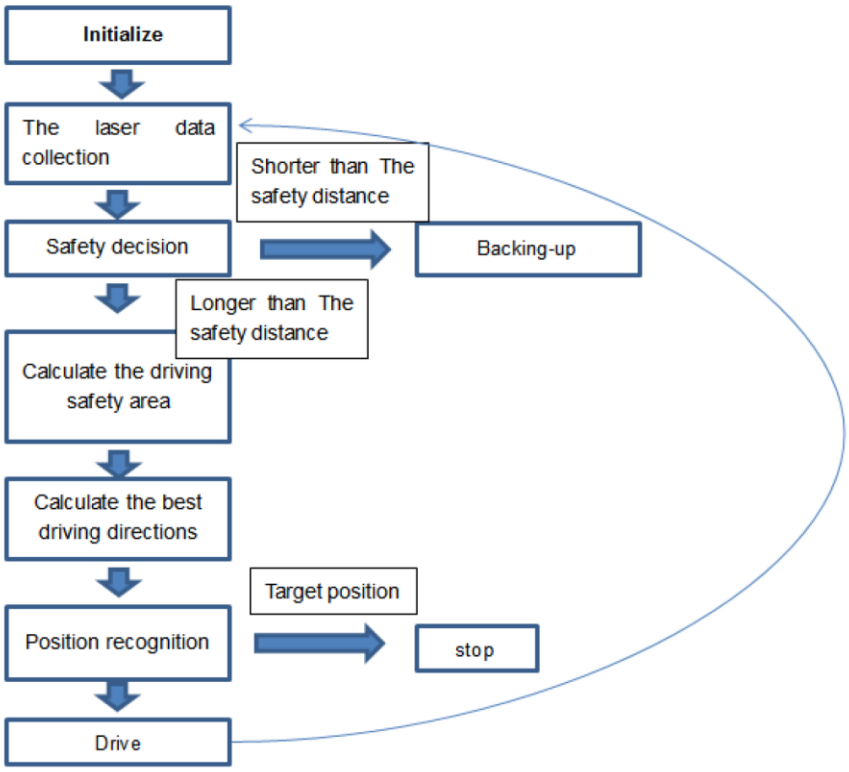


Figure 2: Autonomous driving systems [8]

It is mentioned in [6] that in a vehicle platoon, radar-based techniques are used for Adaptive Cruise Control (ACC) to maintain a safety distance from the leading vehicle. ACC is a system used to allow the following vehicle to maintain a time-gap between itself to the leading vehicle by controlling the throttle speed of the vehicle. If the leading vehicle is far, the throttle will increases in speed and vice versa when the leading vehicle is near. Radar monitoring is a very important feature to ensure that the vehicles do not collide with each other. In the project, instead of using the mechanism like ACC, the project is implementing an ultrasonic sensor to demonstrate the scenario of a following vehicle moving to the targeted object and stops when the targeted object is within a safety distance. This situation can be described like an obstacle-avoidance scenario where the vehicle stops when there is an obstacle and moves when no obstacle.

In reference [9], the paper studies the vehicle platooning and its implementation on robot vehicles. Here, it uses a camera for positioning systems instead but some fundamentals are still applicable for IR sensors. To ensure a good throughput

performance, a desired spacing between the vehicles is required. The desired following distance equation is given by:

$$f_d = r + h_d v \tag{2.1}$$

where r is the distance at standhill, $h_d$is the desired headway time and v is the velocity of the vehicle.

Since the vehicles are moving, there will be a distance difference or gap between the leader and the follower. This will eventually create an error and the gap difference will grow bigger from desired distance if not reduced. For simplicity, the platoon is demonstrated in a general 2-D system. The general equation for the 2-D system is given as:

$$\alpha(s,z)y(s,z) = b(s,z)u(s,z) + c(s,z) \tag{2.2}$$

where $y(s,z)$ and $u(s,z)$ stand for LZ-transform of plant output and input respectively.

A general 2-D controller is driven by error signal of:

$$e(s,z) = y_{ref}(s,z) - y(s,z) \tag{2.3}$$

where $y_{ref}(s,z)$ is the desired reference value and $y(s,z)$ is the output value.

Using the equation in 2.3, each vehicle can be controlled by operating on the error of the real output to the targeted object from the desired reference value. This equation has been proved to be true for i) predecessor following control, ii) leader following control and iii) constant time-headway policy. More information regarding the formation of the equations above can be found in [7].

The error wanted to be eliminated here is the distance error. For the case of the project, the platoon only consists of a leader (targeted object) and a follower which makes the computation even easier since the project does not have to consider the error of n^{th} vehicle. Therefore it is possible to directly reduce the error by simply deducting

the output error from the desired distance. Anyhow, the same equation can be applied for direction errors as well if necessary.

Aside from distance, the velocity of the vehicles is important in a platoon. As mentioned above in [6], the ACC system integrated in platoon controls the throttle speed of the vehicles in order to maintain a safety time-gap between the follower and the leader. In [10] and [11], the transfer function from the vehicle velocity to the distance between vehicles is given by:

$$G_v(s,z) = \frac{\hat{d}(s,z)}{\hat{v}(s,z)} = T_{vel}\frac{(z^{-1}-1)}{s} \tag{2.4}$$

where T_{vel} is the complementary sensitivity function for the velocity loop i.e. the transfer function from the reference velocity to true (measured velocity).

From this equation, the controller is proved to be as first-order system. Then the project can consider using PI or PID controller for the setup. Further experiments need to be conducted to determine the best type of controller tuning.

2.1 Experimental Vehicle

As for the hardware of the project, first of all the experimental vehicle is the Mobile Robot Base Set. It is a startup platform for small size mobile robot. This experimental vehicle is equipped with a base for convenient integration of sensors on it and it also comes with servo motor. The microcontroller board and sensors will be integrated onto the base. Below are the specifications of the robot chassis.

Specifications	
Motor	Servo Motor C36R :360 degrees rotation (2)
Wheel	Servo Wheel (2)
Castor	(2)
Diameter	160mm
Height	80mm
Weight	0.278 kg

Table 1: Specifications of Mobile Robot Base Set [12]

2.2 Servo Motor

The follower vehicle is run by servo motor C36R. It is the most typical low cost actuator used in educational robotic project. Usually RC Servo allows user to control its rotation angle. But in this project, the servo has been modified to operate like a DC motor. Below are the specifications of the C36R motor.

Specifications	
Speed (sec/60deg)	0.16/4.8V, 0.14/6.0V
Torque (kg-cm)	3.5/4.8V, 4.5/6.0V
Weight	36 g
Rotation angle	180 degree but modified to 360 degree
Pulse width range	0.546ms to 2.4ms

Table 2: Specifications of C36R Servo Motor [13]

2.3 Microcontroller

The microcontroller used is the Arduino Uno Rev3-Main Board. On follower vehicle, the Arduino UNO is used to operate with the sensors and computation of the algorithm. The controller will also be giving commands to the vehicle's servo to navigate to the targeted object. On the leader vehicle, UNO is used to run the IR transmitter.

The Arduino Uno is a board based on the ATmega328 microcontroller. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analogue inputs, a USB connection, a power jack, and a reset button.[14] These two Arduino boards are used because they provide ample pins to be used for the IR receivers, servo, ultrasonic sensor and IR LEDs. Furthermore, Arduino is very simple to be used and there are many open-source projects can be found online.

More details on Arduino and some basic projects can be found in [15]. The specifications of the Arduino Uno are described as follow:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Table 3: Specifications of Arduino UNO microcontroller [14]

2.4 Three-pin IR receiver

One of the sensors used in the project is the IR receiver, TSOP34838. The TSOP34838 is a miniaturized receiver for IR remote controls. This sensor operates at a transmission distance of 35m and the directivity is at ±45 degrees [16].

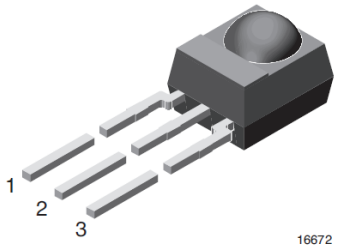


Figure 3: Infrared receiver (TSOP 34838) [16]

This type of sensors is already tuned to detect only infrared lights. The reason IR receiver is used is because the follower vehicle should detect specifically only the targeted object. It is not desirable for the vehicle to be affected by visible light; instead the vehicle should be able to distinguish between its targeted object from its environment surrounding. To minimise the detection error and mistaken other emitting light as the targeted object, the receiver has a demodulator inside that looks for only modulated IR at 38 kHz. This means, if just emitting an IR light will not be enough whereas it should be PWM IR light blinking at 38 kHz. IR receiver is digital out where it either detects the IR light at 38 kHz and returns output LOW or it does not detect any signal and returns output HIGH [17].

2.5 Ultrasonic Distance Sensor

Another sensor used is the Parallax's PING Ultrasonic Distance Sensor. A PING))) sensor works by transmitting an ultrasonic burst (pulse) and then listens for a pulse to return. The width of this pulse corresponds to the distance to the object. It is suitable to be used in this project as the sensor provides precise distance measurement from 2 cm to 3m. A tutorial on PING))) sensor can be found in [18].

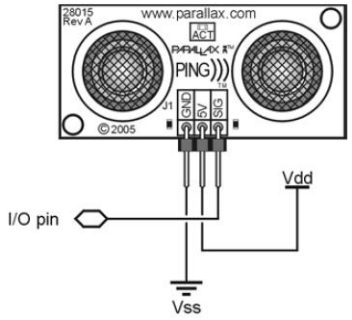


Figure 4: PING))) Ultrasonic Distance Sensor [19]

Specifications	
Operating voltage	+5 VDC
Supply Current	30 mA typ; 35 mA max
Working Distance	2 cm to 3 m
Communication	Positive TTL pulse
Package	3-pin SIP, 0.1" spacing (ground, power, signal)
Weight (kg)	0.009

Table 4: Specifications of PING))) Ultrasonic Distance Sensor [19]

This sensor is used to measure the distance from the targeted object to the vehicle. If the distance is too far, the vehicle will move forward towards the object and if the object is too near the vehicle will move backwards. This sensor is important to ensure the object is within the detection radius of the vehicle.

2.6 Infrared Transmitter

IR transmitter is very similar to LEDs except it emits infrared light that is not visible to human eye. IR transmitter has an angle of directivity of ±24°. This infrared light can only be detected by IR receiver just like how phototransistor works. But the difference is the IR receiver is designed to detect modulated infrared light that is blinking at a certain frequency, in this case 38 kHz.

Figure 5 Infrared Transmitter (LED) [20]

The operation of the IR transmitter and IR receiver is exactly the same as TV remote control. The TV remote flashes IR light at the same frequency but with various patterns according to the buttons pressed. The TV receives the signal sent by the TV remote with an IR receiver like the one used in the follower vehicle as mentioned in Section 2.4.

The IR receiver only responds to infrared if it is blinking at 38 kHz. This prevents external disturbance such as infrared from the sun and incandescent light from being misinterpreted as the signal sent from transmitter.[21] So to send signals that can be detected by the IR receiver, the Arduino is programmed to modulate the IR LED at the frequency rate of 38 kHz. More details on the operation of TV remote can be found in [22].

2.7 Arduino Environment

The programming software used here is the open-source Arduino environment. The Arduino environment makes it easy to write code and upload it to the I/O board. It contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.[23] The environment is written in Java and based on Processing, avr-gcc, and other open source software.

The codes are written and uploaded into Arduino through sketches. It works like text editor and allows copy/paste as well as find/replace functions. Furthermore, Arduino environment can verify the codes while uploading the program into Arduino thus the message area will tell the errors if exist. The best of all, the serial monitor made it easier to read the serial port and observe the incoming data to compare with the expected result data. In short, this programming software is very user-friendly and convenient especially for programmer beginners.

2.8 Microsoft Office

Microsoft Office is used for all the documentations purposes. The respective programs and uses are listed below:

Software	Description
Microsoft Word	For documenting reports
Microsoft PowerPoint	For presentation slides
Microsoft Excel	To compute calculations and organize data in spreadsheet

Table 5: Microsoft Office use in project

3.0 METHODOLOGY

3.1 Research Methodology

In this project, the experimental vehicle is a robot chassis integrated with sensors to track for a leader. The leader vehicle emits IR light at the frequency of 38 kHz and when the experimental vehicle detects the presence of IR light, it will then direct to the position of the leader.

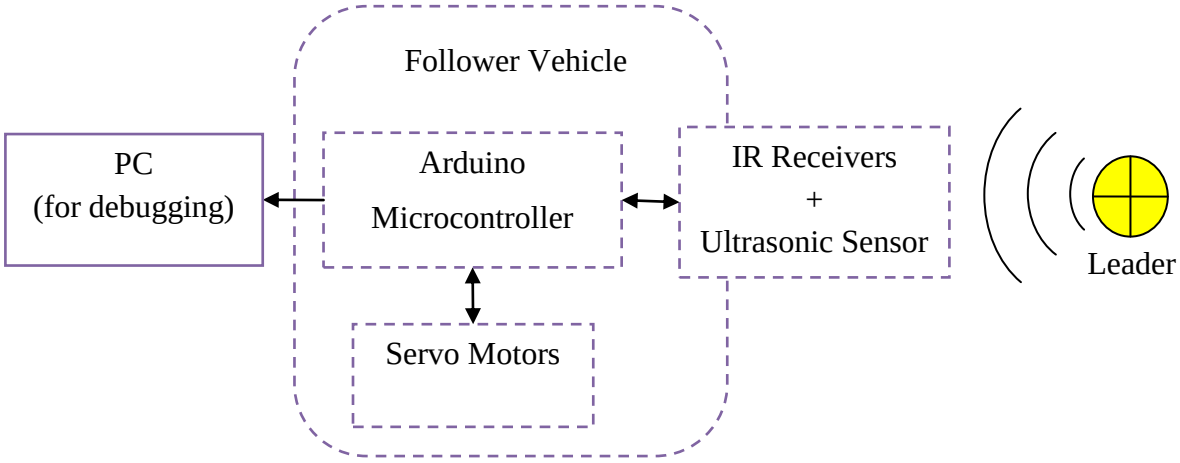
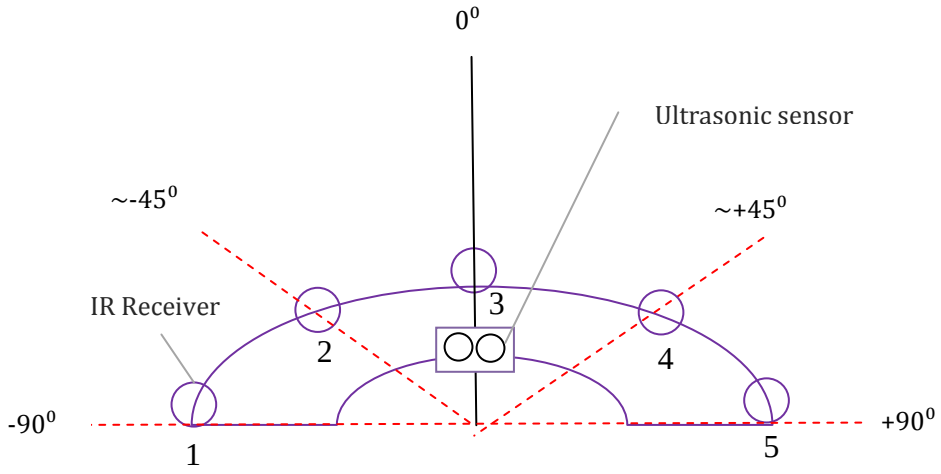


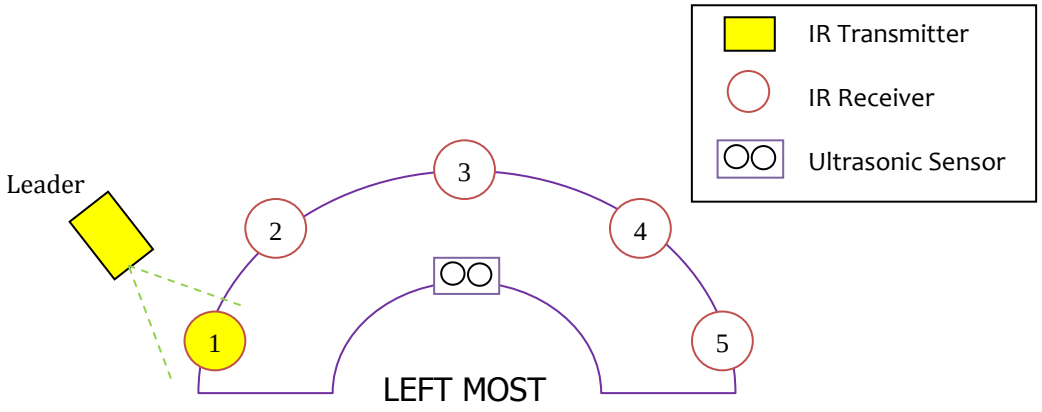
Figure 6: Architecture of object tracking system

The IR receivers need to be integrated onto the vehicle such that the IR detection range is not overlapped. The adjustment for IR detection angle is done physically by wrapping the IR head in a tube with non-reflective sleeve as shown in Appendix 2. For the moment, the IR receivers will be integrated at the front viewing only for proof of concept. Considering only front viewing, the viewing angle of the car is selected at 90 degrees to the left and 90 degrees to the right which made up a total of 180 degrees. Since five sensors will be used, the IR sensor is mounted at approximately 45 degrees apart. Each sensor decides the position of the leader: left, middle, and right. Meanwhile, ultrasonic sensor is assembled in the middle to measure distance. The working range where the object is detectable lies within 10 cm to 40 cm.



Each IR sensor is integrated such that it detects the left, centre and right position. The detection distance range lies between 10cm to 40cm.

The sensors are arranged in this manner (arc form) in order to be able to tell which zone the targeted object is currently at. For example, if the IR 1 lights up that mean the targeted object is at position of far left most.



If IR 1 lights up, leader vehicle is detected at zone left.

IR 1 and IR 2 detects left zone, IR 3 detects middle and IR 4 and IR 5 detects right zone.

To achieve object tracking, there are two factors to consider: i) direction of the leader vehicle and ii) distance of the leader vehicle from the experimental vehicle. The experimental vehicle firstly needs to track the IR light by retrieving data from the IR sensors. Through the data, the microcontroller will compute the algorithm to approximate where the location of the IR light is received and then it will send command to the servo motor to navigate to the source of IR light. The project's architecture is in Figure 7 while the project's algorithm is shown in Figure 8.

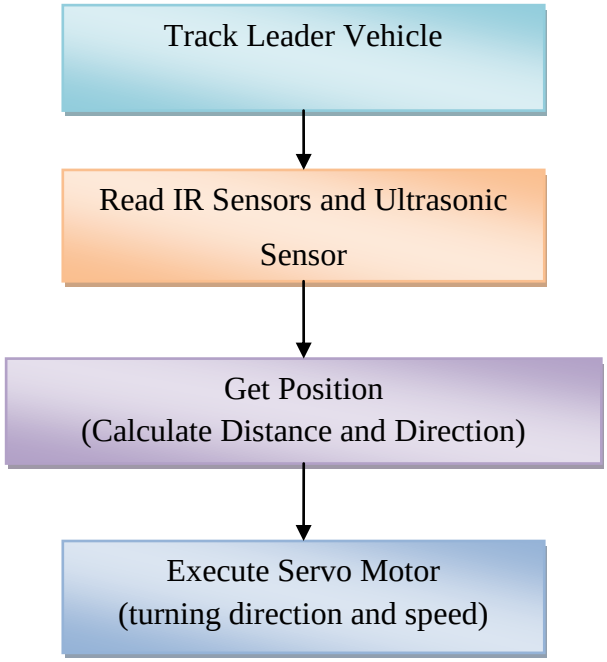


Figure 7: Architecture of software

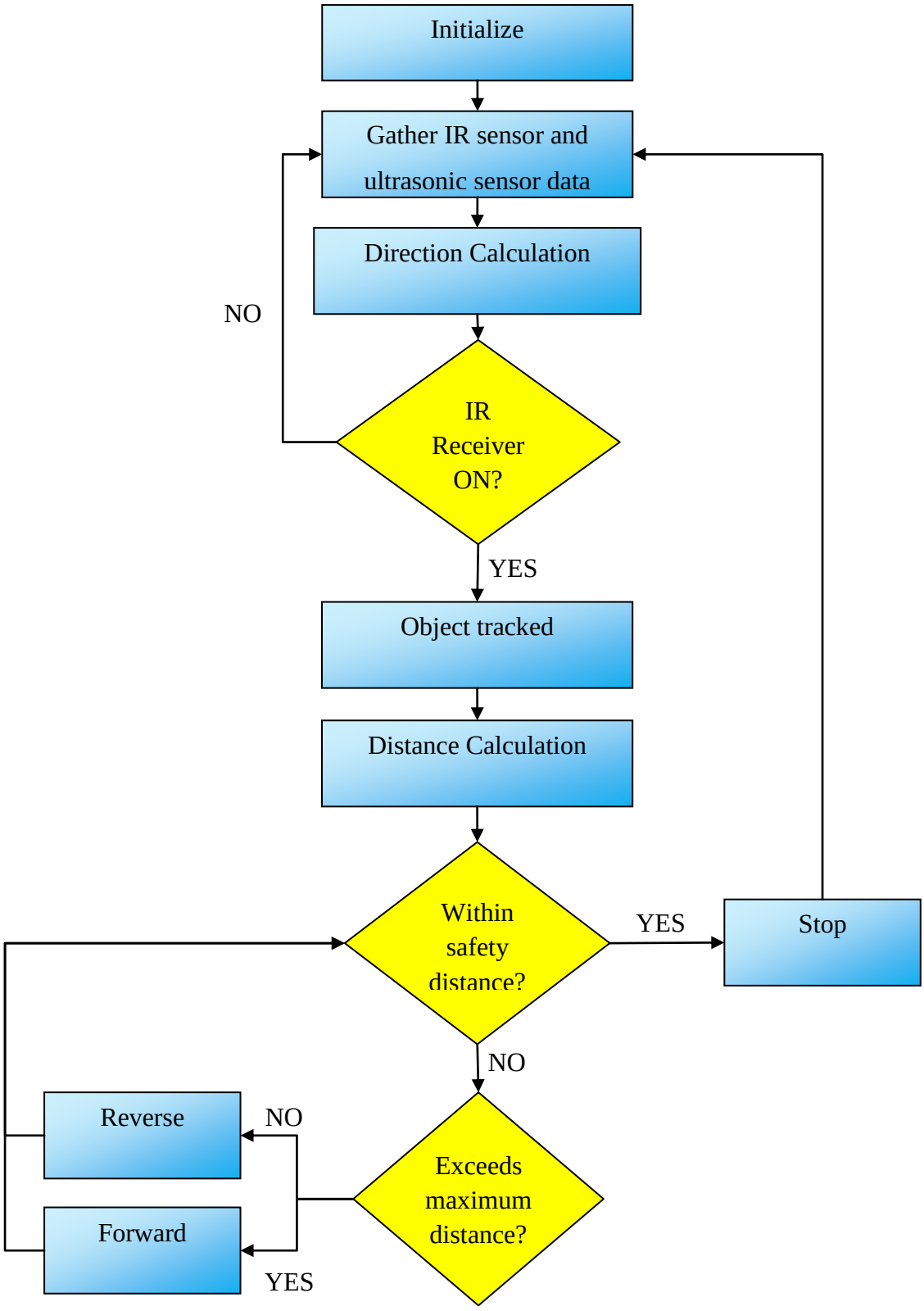


Figure 8: Object-tracking Algorithm

This algorithm can also be represented in Finite State Machine (FSM). A FSM serves like a manager that organises a set of states and it manages the transition between current states to next state. The transition happens when a logic or case is true. FSM can be seen as defining a class of graphs with labels on each arc [24]. Below describes the FSM of this project.

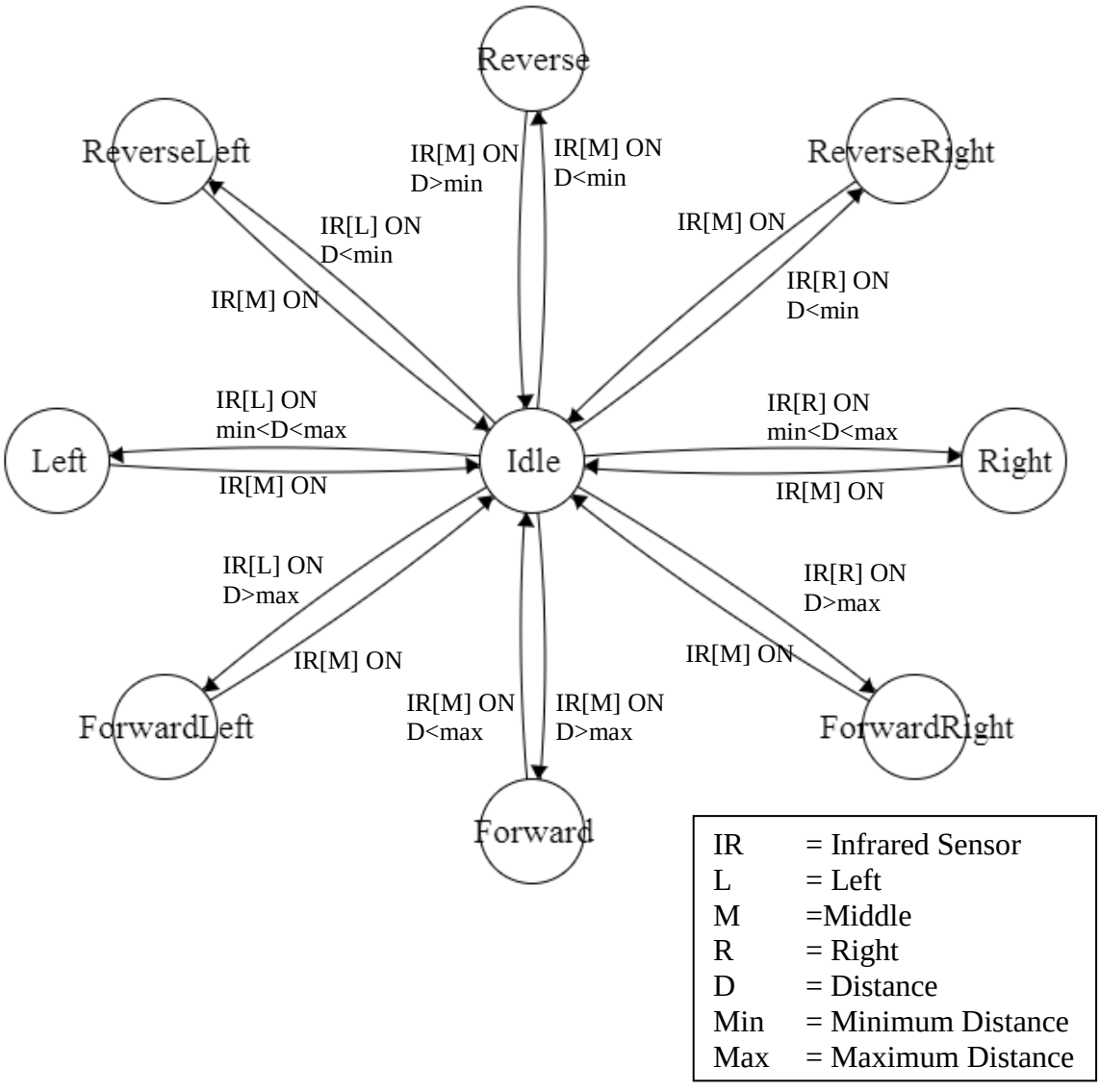


Figure 9: Finite State Machine

IR[L] (IR1,2)	IR[M] (IR3)	IR[R] (IR4,5)	D>Max	D<Min	Min<D<Max	Manoeuvre Result
-	-	-	-	-	-	Idle
-	-	/	-	-	/	Right
-	-	/	-	/	-	Reverse Right
-	-	/	/	-	-	Forward Right
-	/	-	-	-	/	Idle
-	/	-	-	/	-	Reverse
-	/	-	/	-	-	Forward
/	-	-	-	-	/	Left
/	-	-	-	/	-	Reverse Left
/	-	-	/	-	-	Forward Left

Table 6: Conditions and Manoeuvre Result

In object-tracking process, many kinds of disturbance may affect the object tracking thus creating errors to the desired output. Among the disturbances to consider include:

a) Brightness of the environment
b) Sensitivity of the detection range
c) Size of the object

Meanwhile, the ultrasonic sensor can be affected by factors such as the object that:

a) exceeds the distance measurement range (more than 3 m)
b) has reflective surface at shallow angle so the sound will not be reflected back to sensor
c) is too small to reflect the sound back to the sensor

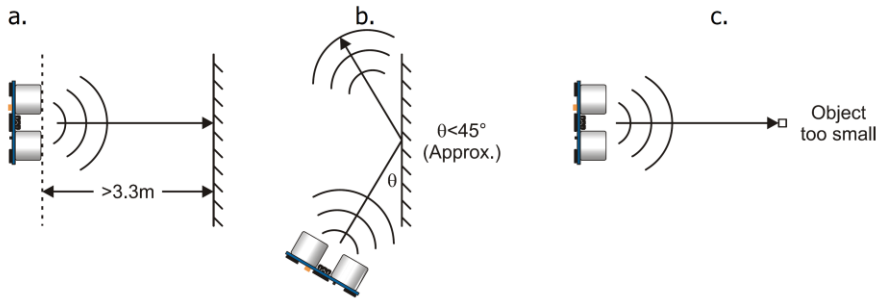


Figure 10: Factors affecting PING))) ultrasonic sensor [19]

These disturbances can be reduced by physical method or software method. Physical method means amendments on the hardware will be made to suit the condition while software method is reducing the errors through coding and mostly involve computations.

3.2 Project Activities

The main objective of this project is to produce an affordable object-tracking vehicle using simple and low-cost sensors. To create all of the different systems of the vehicle the following tasks are assigned:

i. To conduct a literature review to understand how platooning system works as well as reviewing existing an in-project papers about object tracking robots

ii. To study a suitable theory that is well supported to be built. Then design a suitable theoretical framework for the system on a small scale platform in which suitable sensors shall be used for the object tracking.

iii. To develop the algorithms and controller design to be coded into programs. Control strategy will also be developed incorporating the majority of the criteria and disturbances as mentioned in Section 3.0. (detection range, size of object, brightness of environment, etc.)

iv. To test for the algorithm and controller design and perform debugging work if necessary.

v. To come out with a complete prototype and demonstration of object-tracking system.

vi. To produce a report of discussion and experimental results for reference.

3.3 Key Milestones

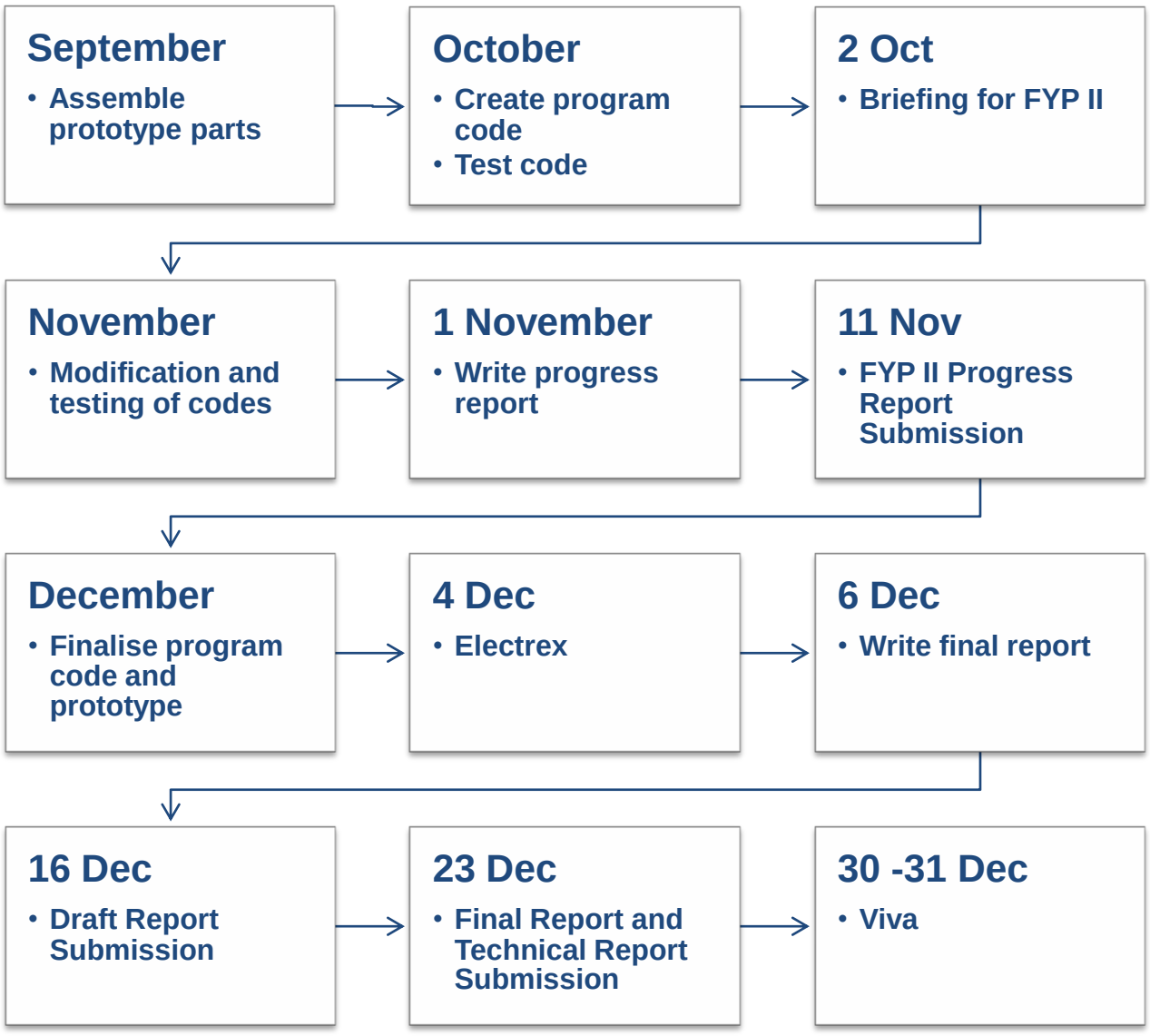


Figure 11: Key Milestones

3.4 Schedule

A Gantt chart is shown in the next page for my schedule. Below describes the general tasks throughout the schedule:

i. Research

This is where various resources are researched to assist in the production of the prototype and program. Based on the researches, a data analysis is conducted to study a suitable theory and algorithm that is well supported to be built into the system.

ii. Designing

During this stage, the technical setup of the system is completed. The prototype parts are assembled. The sensors and microcontroller are integrated onto the robot chassis.

iii. Implementation and Testing

The codes are completed and ready to be tested. A program test is conducted step by step and each sensor is tested individually. Then the full sketch is tested with all sensors integrated together. From then on, debugging work is performed and need to come up with more extensive features if possible. Finally, all the progress and results are documented in a report.

iv. Deadlines

These are the deadlines to be noted for the submissions of reports and presentations.

3.5 Gantt Chart

GANTT CHART														
Stages	Activities	Week Number												
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Research	Search information through various resources (internet, books, etc.)														
	Consult supervisor														
Designing	Assemble vehicle parts														
	Code Controller's Algorithm														
Implementation and Testing	Test IR receiver														
	Test PING))) sensor														
	Test servo														
	Test IR transmitter														
	Test full sketch														
	Analyse result														
	Document findings into a report														
Deadlines	Progress Report								11/11						
	Electrex											4/12			
	Draft Report													16/12	
	Final Report and Technical Report														23/12
	Viva														30/12

3.6 Hardware/ Tools and Software

The table shows the lists of hardware used and their respective purposes.

Hardware	Description
Robot chassis	Mobile robot base
Arduino board (ATmega328 microcontroller)	To compute algorithm and interact with sensors
Infrared receiver	To detect the position of targeted object
Infrared transmitter	To emits infrared light modulated at 38 kHz
PING))) Ultrasonic distance sensor	To measure the distance between the object detected and the sensor
Servo motor	Move the robot to maintain the desired distance to targeted object
L293d	H-bridge to drive motors
Miscellaneous (batteries, wires, etc.)	Other parts of the vehicles

Table 7: List of hardware used

As for the software, the following applications are used:

Hardware	Description
Arduino Environment	To create and upload sketches into Arduino board using USB. Also for debugging purpose.
Microsoft Office	For all the documentation purposes

Table 8: List of software used

The figure below is the setup of connections. On the follower vehicle, the connections are as follow:

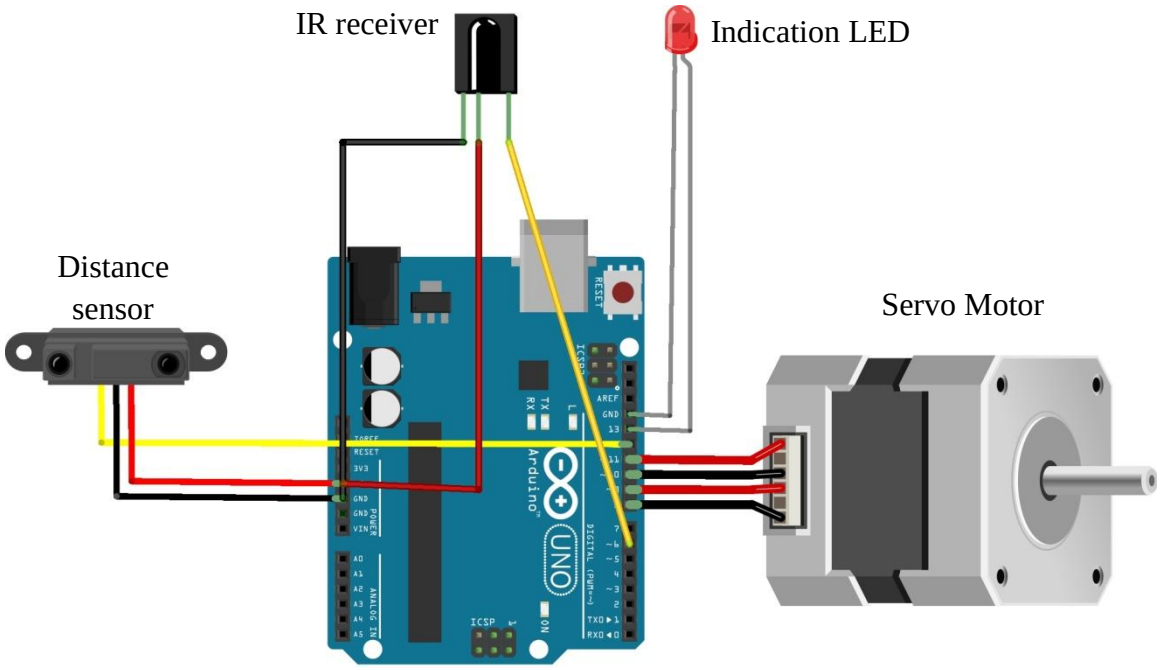


Figure 12: Connection of sensors and other components for follower vehicle

Parts	From Pin on Parts	To pin on Arduino
Infrared Receivers (1-5)	GND	GND
	Vcc	+5 V
	OUT	D3-D7
PING))) Ultrasonic Distance Sensor	GND	GND
	Vcc	+5 V
	OUT	D12 PWM
Servo Motor	OUT 1 (black)	D8, D10
	OUT 2 (red)	D9, D11
LED indicators (1-5)	Anode	A0-A4
	Cathode	GND

Table 9: Connection pins on follower vehicle

For the leader, the connection is very simple as shown in Figure 13. The pin of IR transmitter's anode is connected to a digital output pin, and the cathode to ground. Three transmitters are used because this would increase the detection angle range of the leader vehicle.

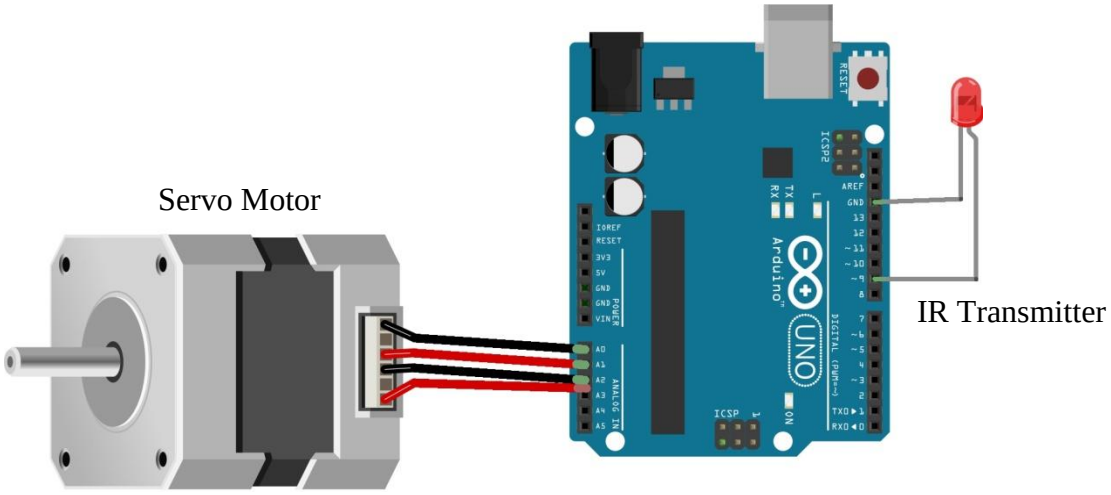


Figure 13: Connection of sensors and other components for leader vehicle

Parts	From Pin on Parts	To pin on Arduino
Infrared Transmitters (1-3)	CATHODE	GND
	ANODE	D3, D6, D9
Servo Motor	OUT 1 (black)	A0, A2
	OUT 2 (red)	A1, A3

Table 10: Connection pins on leader vehicle

3.7 Cost

Parts	Description	Unit	Remark	Cost (RM)
Robot Chassis	As experimental car	1	Cytron	150.00
Infrared Receiver	Detect IR light	6	Shop	24.00
PING))) Ultrasonic Distance Sensor	Measure distance	1	Cytron	50.00
IR transmitter	Transmit IR light	3	Shop	9.00
Arduino UNO	Microcontroller board	1	Cytron	59.00
Prototyping Shield	Uno's prototype shield	1	Cytron	30.00
L293D	H-Bridge for driving motors	2	Shop	12.00
Miscellaneous Parts (Batteries, wires, veroboard, etc.)			Lab and shop	151.30
Total				**485.30**

Table 11: Estimated cost of the project

4.0 RESULTS AND DISCUSSION

4.1 Data Gathering and Analysis

4.1.1 Infrared Transmitter

In this section, the project will discuss about the sketches and work operations of the leader and follower vehicle.

Basically the leader is just infrared transmitters blinking at the rate of 38 kHz. This can be done using the tone function. It generates a square wave of frequency specified i.e. 38 kHz and 50% duty cycle on a pin. This can be done in two ways, using tone function or by using the Arduino timers. The tone function is easier to use than timers but it can be affected if a delay() is called. The tutorial about tone can be found in [25] under *tone()*. While for timers, the information can be found in [26] and [27].

Each Arduino are equipped with three timers: Timer 0, Timer 1 and Timer 2. In this project, Timer 1 and Timer 2 are used to modulate a frequency of 38 kHz. Timer 0 could not be used because the settings changed would interrupt the default delay() values. For example, the default settings delay(100) would delay 100 milliseconds but if Timer 0 is used, the delay(100) would give other values of delay instead. Therefore, a tone function has to be used to replace Timer 0 although it is not preferable since the tone function does not produce a constant frequency of 38 kHz. As mentioned, when a large delay in introduced in the program, the tone function may be out of sync.

The IR transmitters are coded as shown in Figure 14 and named setIrModOutput(). Pin 3, 6 and 9 are used as outputs of transmitters. Pin 10 and 11 are used for timers and therefore shall not be connected to any sensors or devices. The IR transmitters are tested to have average directivity of $\pm 24^{o}$ depending on sensitivity of light.

```
void setIrModOutput(){
  tonel.begin(6); // set up pin 6 to tone()
  pinMode (3, OUTPUT);  //pin 11(OC2A); pin 3(OC2B)
  pinMode (9, OUTPUT);  //pin 9(OC1A); pin 10(OC1B)

  // set up Timer 2 (pin 3 and pin 11 on UNO)
  TCCR2A = _BV (COM2B0);  // toggle OC2B on Compare Match
  TCCR2B = _BV(WGM22) | _BV (CS20);  //Clear Timer on Compare | No prescaler
  OCR2B  = 209;  // compare B register value (16M/(2*210)= 38095 Hz)

  // set up Timer 1 (pin 9 and pin 10 on UNO)
  TCCR1A = _BV (COM1A0) ;  //toggle OC1B on Compare Match
  TCCR1B = _BV(WGM12) | _BV (CS10);  //CTC | No prescaler
  OCR1A  = 209;  //compare A register value (16M/(2*210)= 38095 Hz)

  tonel.play(38000); // blink at 38kHz
  //**Note: Timer 0 settings will affect delay() so cancel command to set up Timer0
  /* set up Timer 0
  pinMode (6, OUTPUT);
  TCCR0A = _BV(COM0A0) | _BV(WGM01);  //toggle OC1B on Compare Match|CTC,
  TCCR0B = _BV (CS02);  //No prescaler
  OCR0A  =  209;*/  //compare A register value (16M/(2*210)= 38095 Hz)
  //Reference: http://www.gammon.com.au/forum/?id=11504 or google TCCR Arduino
`
```

Figure 14: Codes for IR Transmitter

4.1.2 Infrared Receiver

To test for the infrared receiver is simple. The can be done by digital read the pins attached to IR receiver. As mentioned, if the sensor returns HIGH or 1, then there is no detection of infrared light. Otherwise, the sensor will return LOW or 0 when infrared light is present.

```
//assign ir pins
int ir_pin[]={7,6,5,4,3};
int ir[5];

void setup()  // Built-in initialization block
{
  int i;
  for(i=0; i<5; i++){
  pinMode(ir_pin[i], INPUT);
  }
}

void loop()    // MIDDLEain loop auto-repeats
{

  int i;
  for(i=0; i<5; i++){
    ir[i] = digitalRead(ir_pin[i]); //read the status of ir pins
  }
  for(int i = 1; i < 5; i++){
    Serial.print(ir[i]);                    // Display 1/0 no detect/detect
    if (i < 5) Serial.print("  ");
      else Serial.println();
  }
}
```

Figure 15: Codes for IR Receivers

4.1.3 PING))) Ultrasonic Distance Sensor

For PING))) sensor to read a distance, a ping has to be sent by the sensor and measure the time when the ping is reflected back to sensor. The return output is converted into centimetres (cm).

```
void read_ping()
{
  // establish variables for duration of the ping,
  // and the distance result in inches and centimeters:
  long duration, cm;

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH
  // pulse whose duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  // convert the time into a distance
  cm = microsecondsToCentimeters(duration);
}

long microsecondsToCentimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

Figure 16: Codes for PING))) Distance Sensor

4.1.4 Servo Motor

The servo motor is modified to work like a DC motor. The operation is very simple by assigning a HIGH or LOW pulse to respective pin. For example, if the vehicle should move forward then both the wheels have to move clockwise. In this case, to move forward one wheel should be moving in clockwise and another in counter-clockwise. This is because the second wheel is built in a mirrored position to the first wheel.

Condition	Wheel 1	Wheel 2
Forward	CW	CCW
Right	CW	CW
Left	CCW	CCW
Reverse	CCW	CW

Table 12: Servo motor manoeuvre

Follower and leader vehicles have motors attached to them. To drive the servo motors, an H-bridge (L293d) is required because Arduino UNO does not support enough current to the motors. [28]

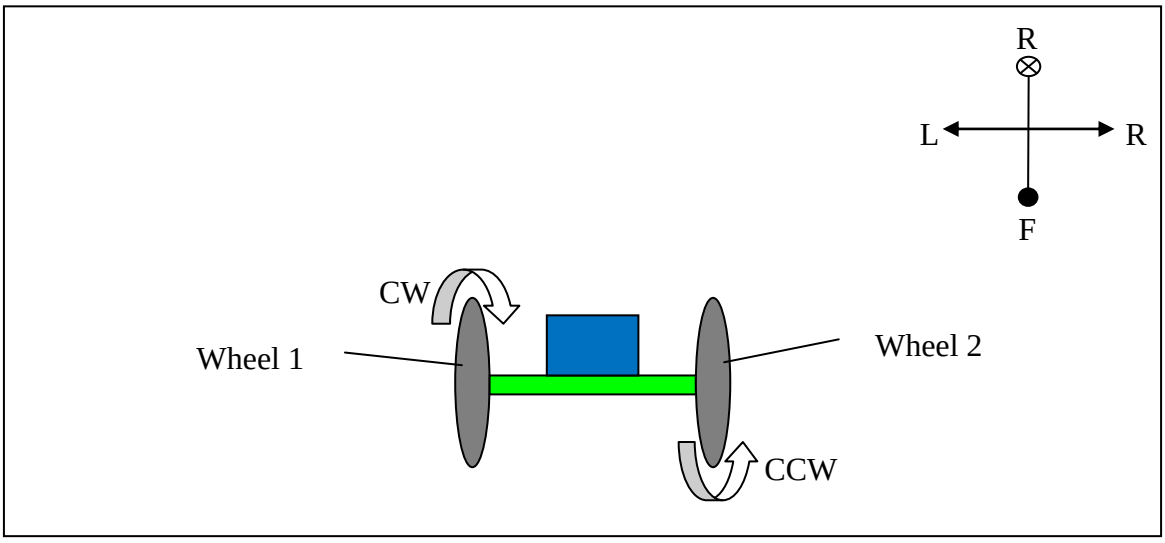


Figure 17: Illustration for Servo Motor Manoeuvre

4.1.5 Follower Vehicle's Algorithm

The manoeuvre of the follower vehicle is categorised into several states as shown in Figure 9 (idle, forward, left, right, etc.). To do this, the *switch-case* statement is used. There are nine possible states altogether. The switch case control is similar to if statement but a break keyword is used at the end of the switch statement to ensure that the program will exit from the loop if a condition is false [25]. The full sketch code is in Appendix 1.

Current State	If	Next State	Remarks
Idle	• IR[M]== ON && cm>max_dist	Forward	ir[L]: Left IR ir[M]: Middle IR ir[R]: Right IR cm: Distance measured min_dist: Minimum distance max_dist: Maximum distance <: More than >: Less than &&: AND operation ||: OR operation
	• IR[M]== ON && cm<min_dist	Reverse	
	• IR[M]== ON && (min_dist<cm<max_dist)	Left	
	• IR[R]== ON && (min_dist<cm<max_dist)	Right	
	• IR[L]== ON && (cm>max_dist)	Forward Left	
	• IR[R]== ON && (cm>max_dist)	Forward Right	
	• IR[L]== ON && cm<min_dist	Reverse Left	
	• IR[R]== ON && cm<min_dist	Reverse Right	
Forward	• IR[M]== ON && cm<max_dist	Idle	
Reverse	• IR[M]== ON && cm>min_dist		
Left	• IR[M]== ON || IR[L]== OFF		
Right	• IR[M]== ON || IR[R]== OFF		
Forward Left	• IR[M]== ON || IR[L]== OFF • IR[L]== ON && cm<max_dist		
Forward Right	• IR[M]== ON || IR[R]== OFF • IR[R]== ON && cm<max_dist		
Reverse Left	• IR[M]== ON || IR[L]== OFF • IR[L]== OFF && cm>min_dist		
Reverse Right	• IR[M]== ON || IR[R]== OFF • IR[R]== OFF && cm>min_dist		

Table 13: States Transition Table

4.2 Experimentation/Modelling

4.2.1 Infrared Field Testing

This is to test if the infrared receivers could detect the presence of infrared light rated at 38 kHz. The emitting infrared light comes from the leader. If the IR light is detected, the receiver will be output LOW as shown below.

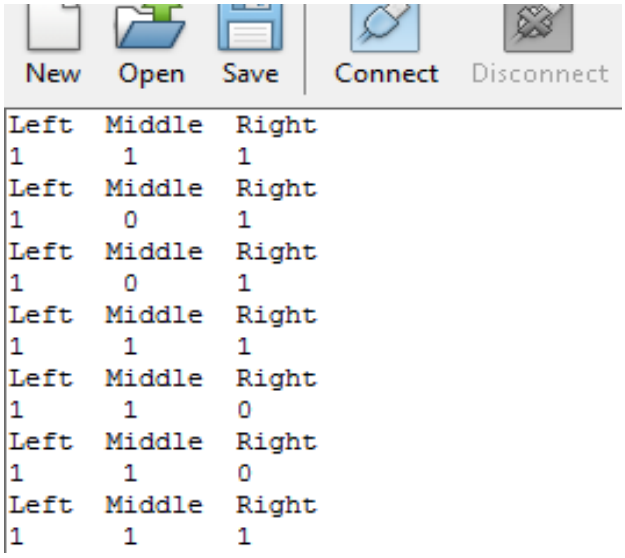


Figure 18: Result of infrared receivers

4.2.2 Distance Testing

In this part, the distance range is tested to observe the maximum and minimum working distance range. From the result obtained, when the distance exceeds around 40 to 41 cm, the IR receiver is no longer able to detect the emission of IR light. Meanwhile, the minimum distance between the leader and follower before collision is about 9 cm. On top of that, when the leader is too close to the follower the IR receiver could not detect the IR light as the detection angle is overlapped. Therefore, the safety distance range is selected to be from 10 cm to 15 cm on average. The safety distance is given a 5 cm gap, because that would provide enough time for the follower to react (stop). The result can be seen in the following chart.

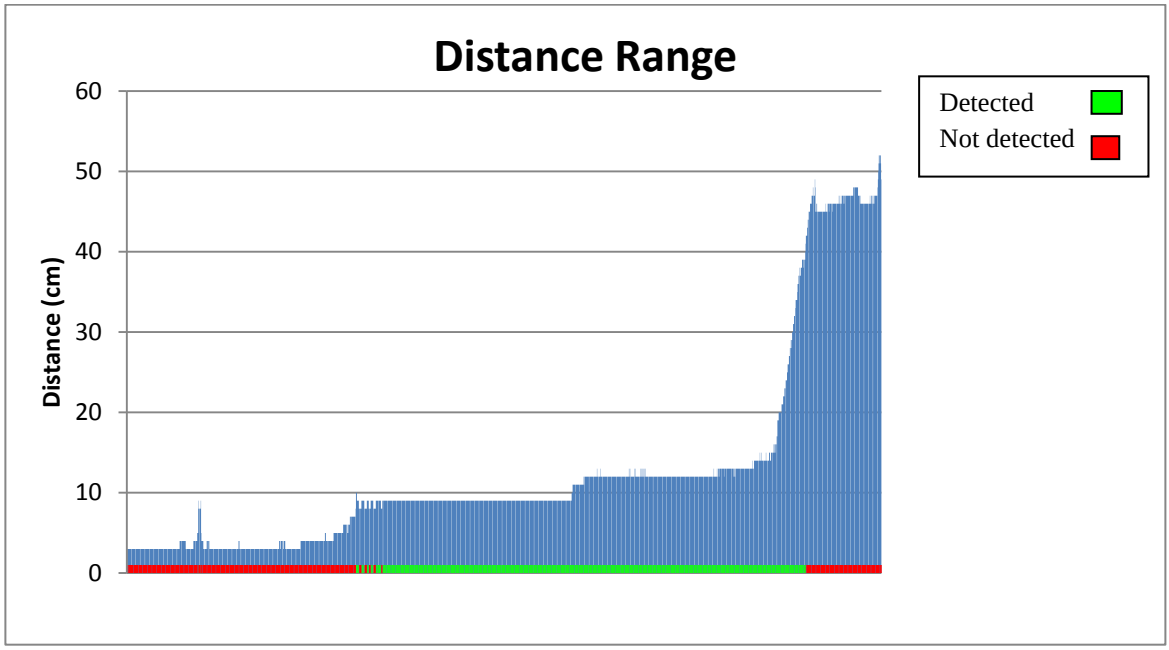


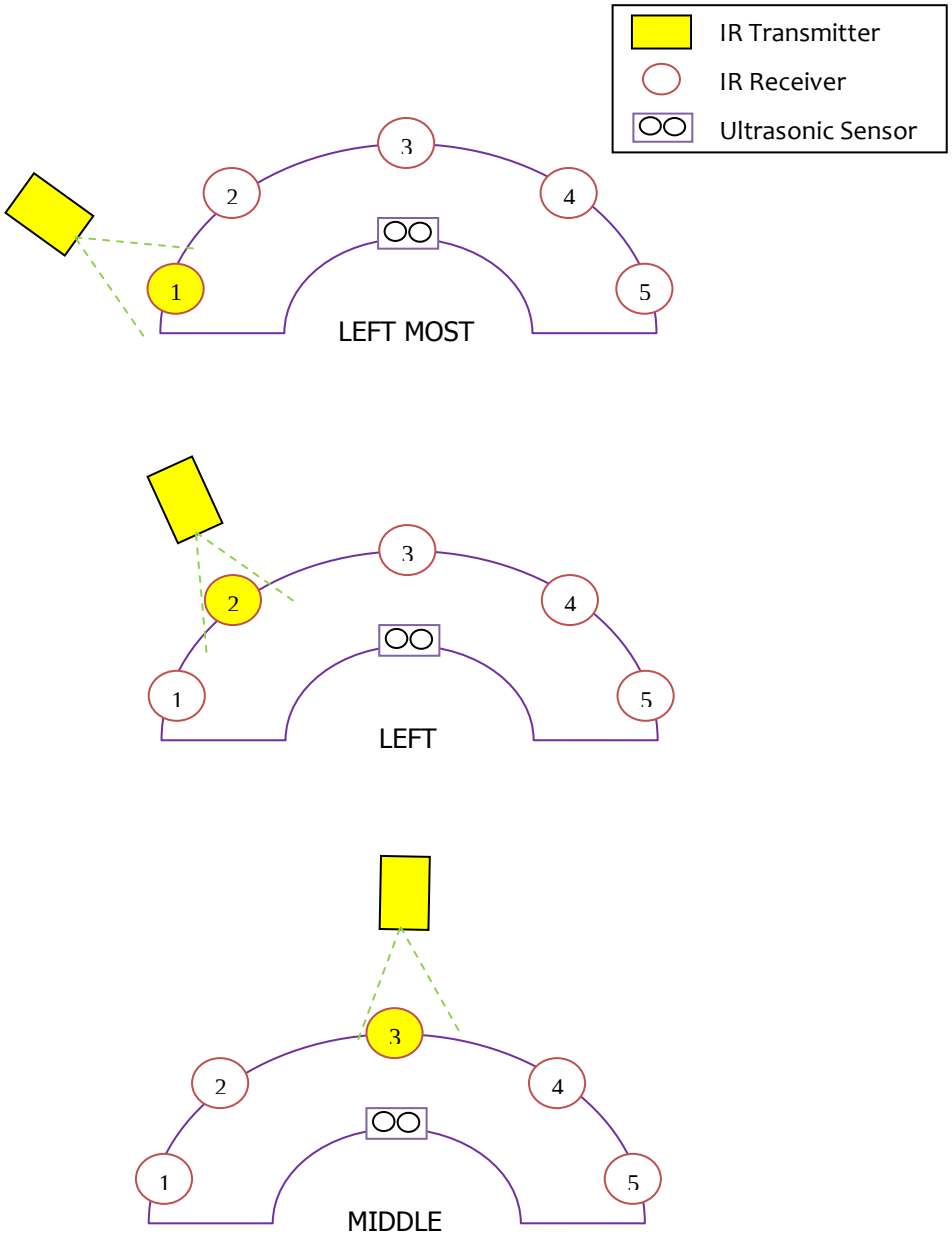
Figure 19: Graph of working distance range

Since working distance is between 9 cm to 40 cm, the safety distance is set at 10 cm to 15 cm.

4.2.3 Overall Testing

Case 1: Working Range Test

The detection of leader works in the following scenarios. The leader is tested within the range from 10 to 40 cm.



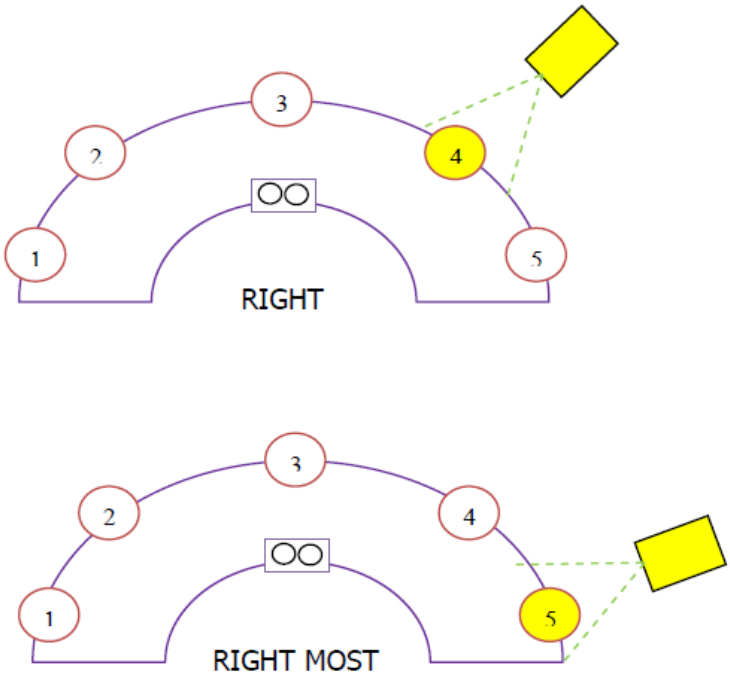


Figure 20: Working Detection Range

Case 2: Overlapping Detection Range Test

Overlapping occurs when the leader is directed at certain angles. Overlapping usually occurs when the leader is facing multiple IRs. The distance of leader is tested within 10 cm to 20 cm.

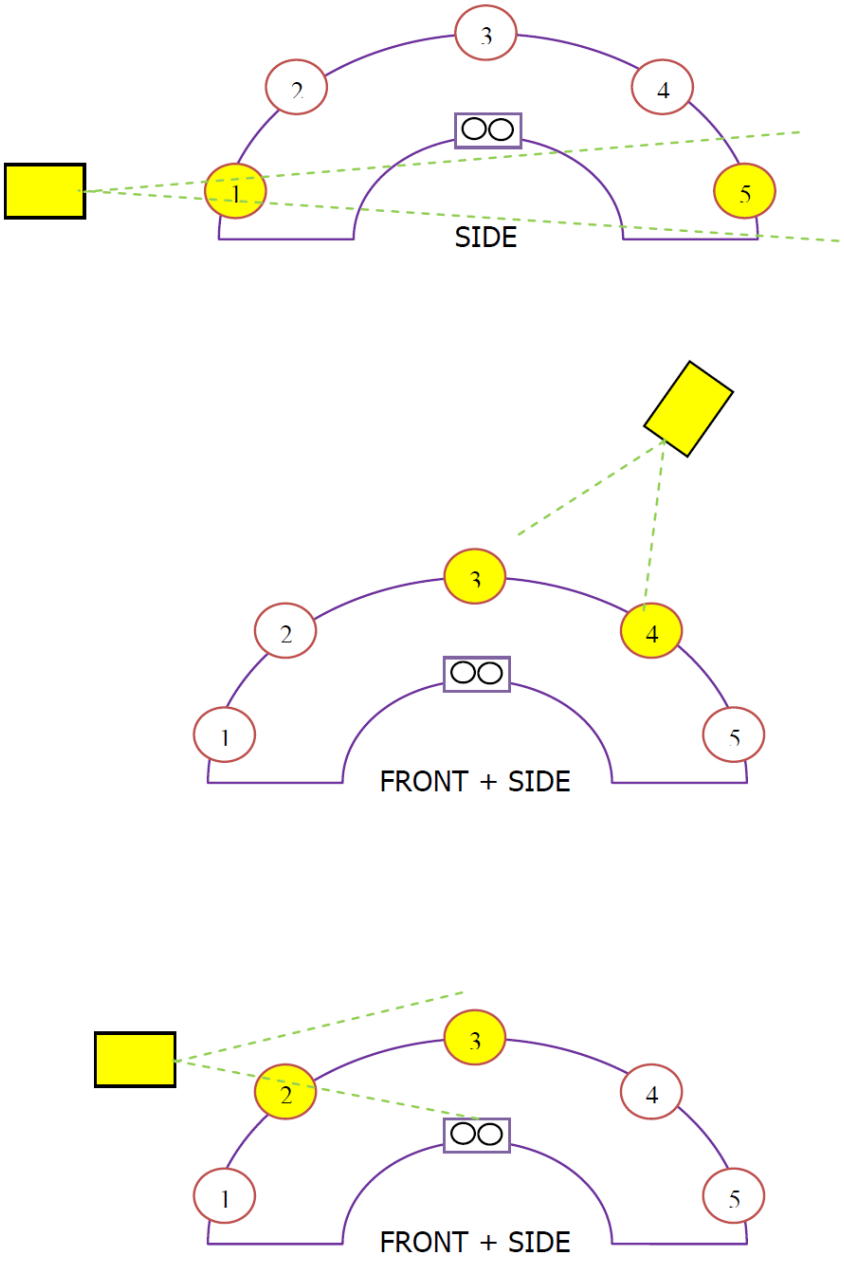
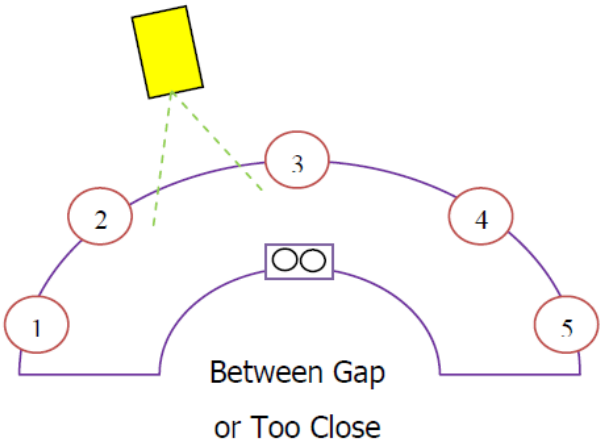


Figure 21: Overlapping Detection Range

Case 3: Off-Detection Range Test

The follower loses detection of the leader when the leader is located in the scenarios below.



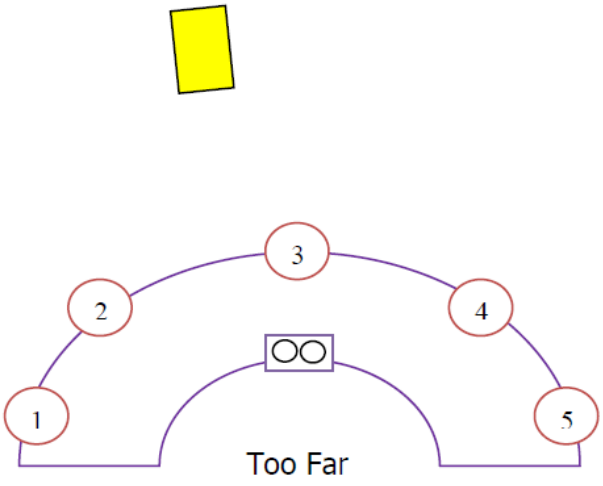


Figure 22: Off-Detection Range

Case 4: Follow-The-Leader Test

The leader is programmed to move in three types of patterns (clockwise, counter-clockwise and figure 8). The results are obtained as shown.

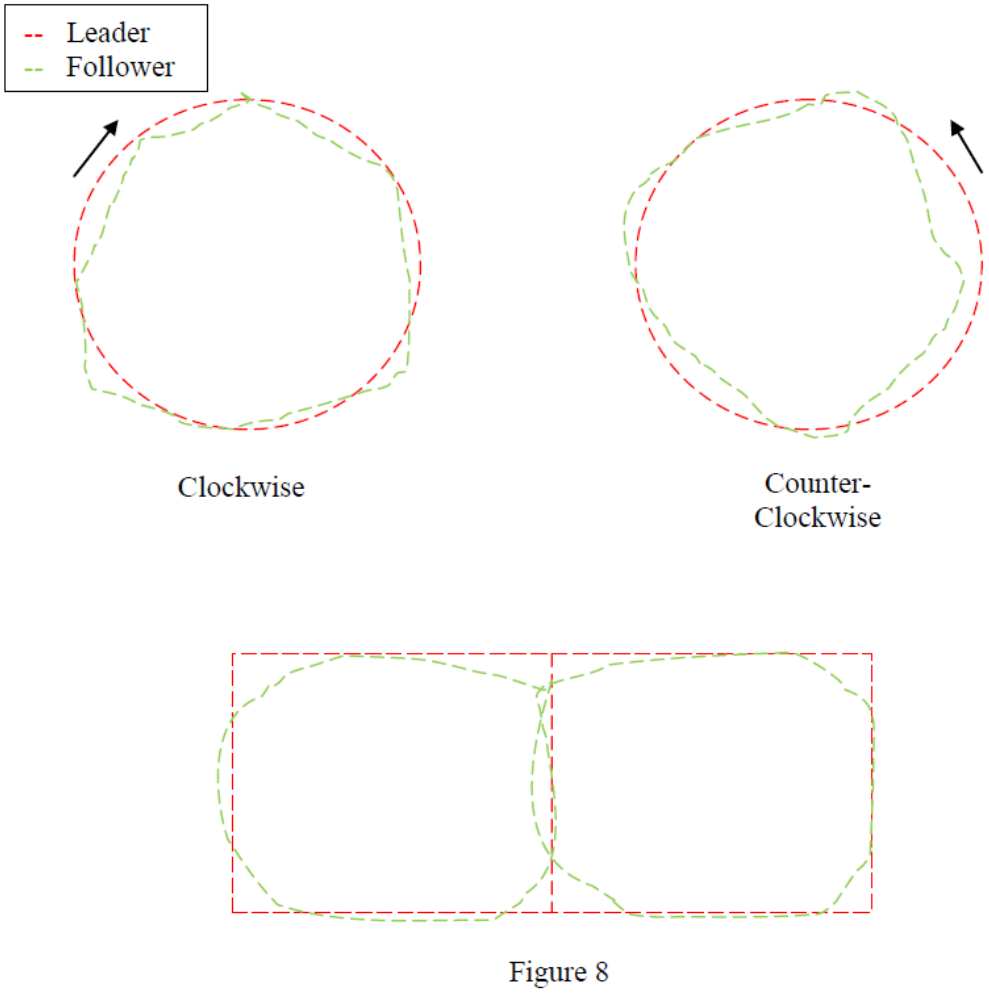


Figure 23: Follow-The-Leader Test

From the result, it can be observed that the following motion is not smooth. For example, when moving in a straight line, then the following pattern is correct. But during turning, the follower tends to get out of path instead of following the path. Hence, when the leader is moving in a circle, the following pattern is not a smooth circle.

However, it can be observed that the follower (green) still does not diverge away from the leader's path (red). Therefore, a leader/follower behaviour is demonstrated.

Next, the pictures show the platooning experiment.

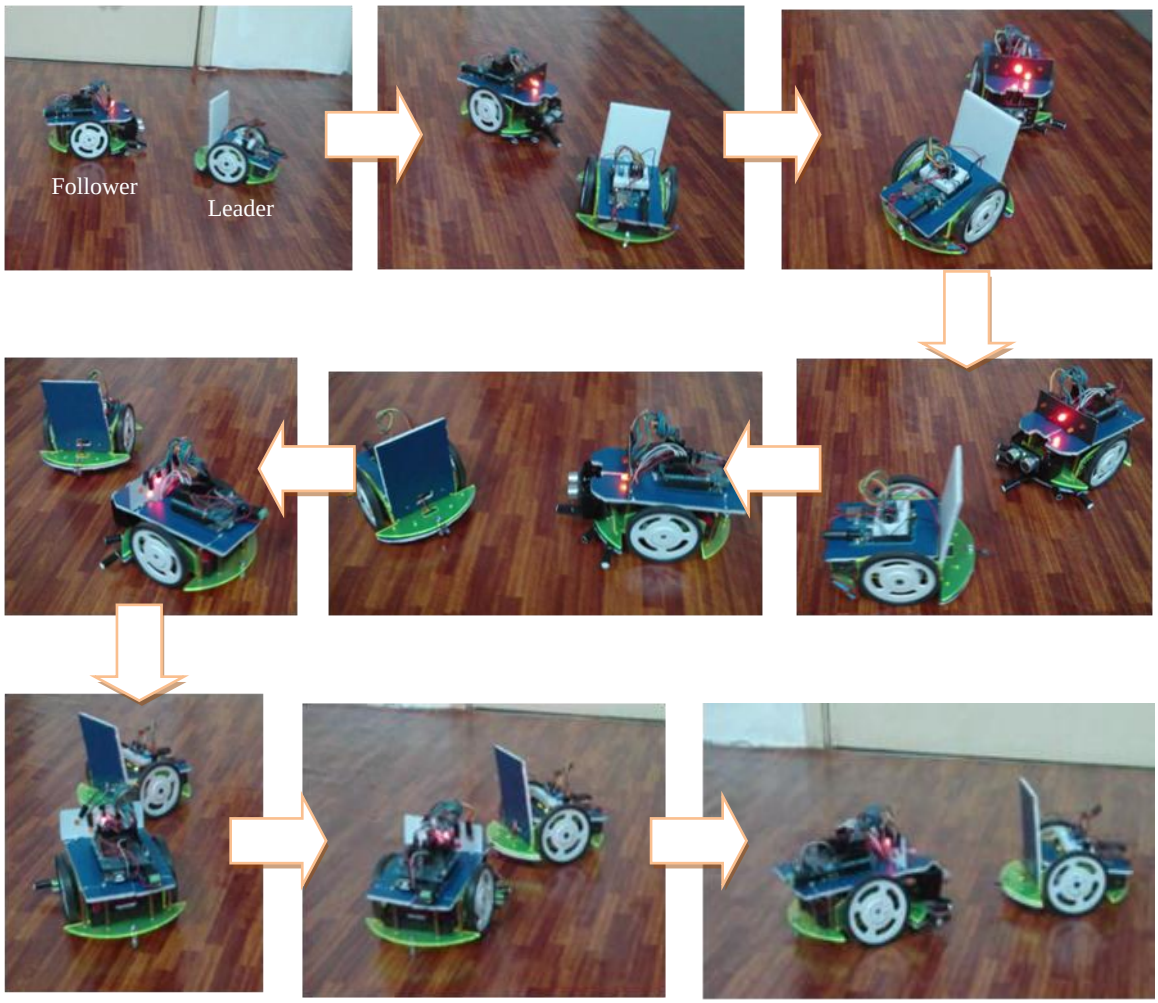


Figure 25: Platooning Experiment

4.3 Prototype

The experimental car is built as shown in the figures below. The experimental car is powered by using two 3.7V 1100mAH Li-Ion Batteries. These batteries are rechargeable thus they are suitable to be used for projects. The car can move by using a pair of C36R Servo Motors which have 360 degree rotation each. After assembling the parts on experimental car, each part is tested with the Arduino Uno microcontroller board. On the other hand, the leader has IR transmitters attached to it.

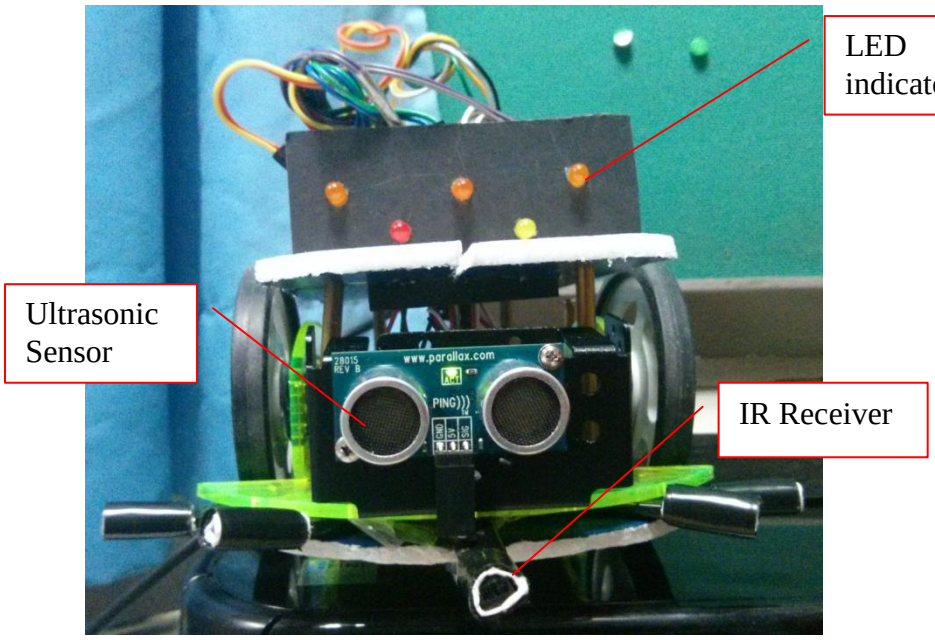


Figure 26: Front View of Follower Vehicle

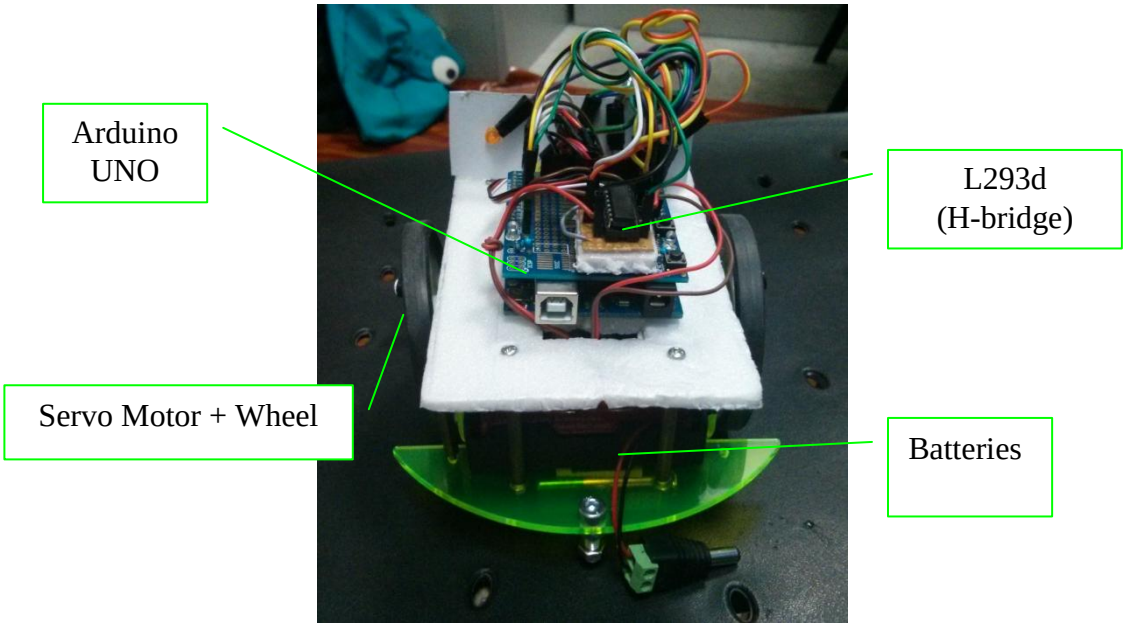


Figure 27: Back view of follower vehicle

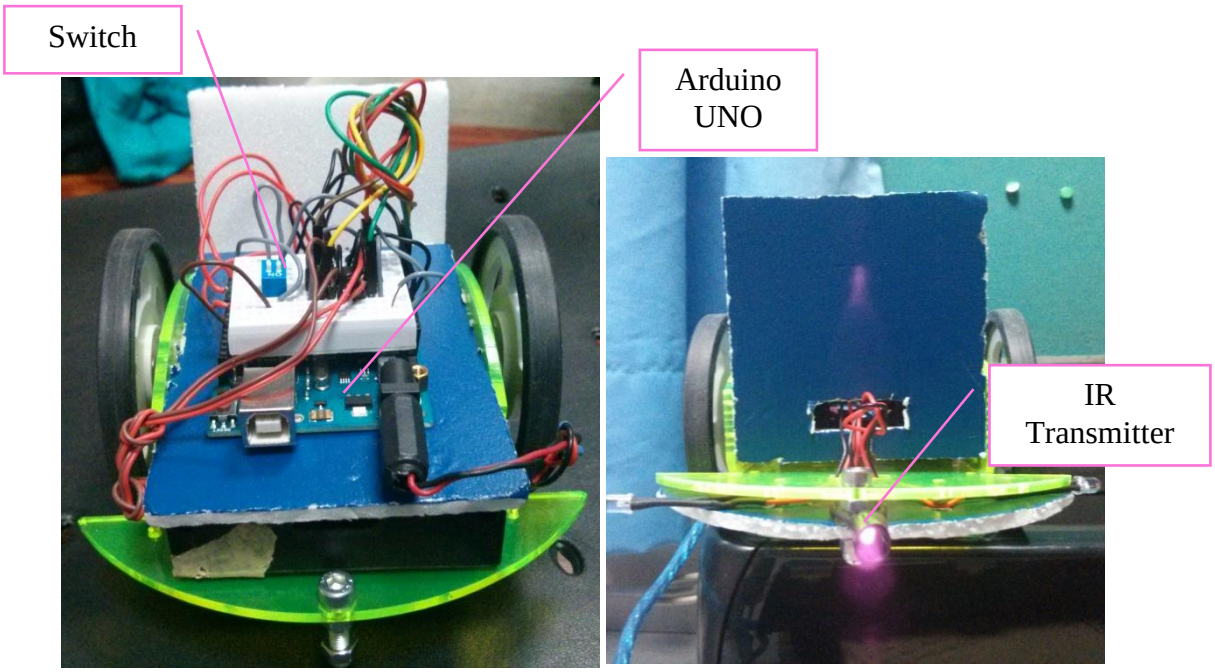


Figure 28: Front and Back View of Leader Vehicle

5.0 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

Considering the literature reviews of past papers, demonstrating the object tracking system using low cost passive sensors is possible. In this project, in order to demonstrate the object tracking system, the infrared sensors are used. Whereby they are aligned in an array to detect the position of the object i.e. the zone where targeted object lies (left, middle or right).

Indeed, from the tests conducted there are several lessons learnt. First of all, position estimation is essential in determining the position of targeted object that the car should be heading to. Distance factor is also important to ensure that the experimental car and the targeted object always maintained themselves in the working range. Last but not least, is having a fast respond rate for the vehicles to react immediately.

5.2 Recommendations

For future works, several recommendations could be done to encourage better results. For example, use a mobile robot equipped with servo motors instead of DC because servo motor gives better precision down to every degree. The motors are preferably the one that allows separate control of servo angle and servo speed.

Other alternatives or algorithms are possible, such as:

- Adds communication e.g. XBee module so the leader can send its position information to follower.
- Still using passive sensors, to keep the cost low but improve on the range and sensitivity of the IR sensors. It is possible to do this because TV remote has a much longer range but the technique to do so is yet to be figured out.
- Alternatively, the sensors could be replaced with low cost camera with pre-processing done on board. CMU cam is suitable because it has on board

microcontroller and CMU cam costs around 80 USD. However the approach will be different.

- The final approach, which has not been considered at the moment is using laser range finder, which is way more expensive but more flexible. It can be used for obstacle and terrain detection.

REFERENCES

[1] P. Hong, B. Marcu, F. Browand, and A. Tucker, "Drag forces experienced by two, full-scale vehicles at close spacing," 1998.

[2] V. Platooning and A. Highways, "VPlatooning," ed, 2010.

[3] A. Dávila and M. Nombela, "SARTRE: Safe Road Trains for the Environment," in *Conference on Personal Rapid Transit PRT@ LHR*, 2010.

[4] C. Bergenhem, Q. Huang, A. Benmimoun, and T. Robinson, "Challenges of platooning on public motorways."

[5] G. Benet, F. Blanes, J. Simó, and P. Pérez, "Using infrared sensors for distance measurement in mobile robots," *Robotics and autonomous systems*, vol. 40, pp. 255-266, 2002.

[6] M. Segata, F. Dressler, R. Lo Cigno, and M. Gerla, "A simulation tool for automated platooning in mixed highway scenarios," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, 2012, pp. 389-392.

[7] M. Šebek and Z. Hurák, "2-D polynomial approach to control of leader following vehicular platoons," in *18th World Congress of the International Federation of Automatic Control (IFAC),(Milano, Italy), IFAC*, 2011.

[8] Y. Zhao and H. Ogai, "Development of a platooning control algorithm based on RoboCar," in *SICE Annual Conference (SICE), 2011 Proceedings of*, 2011, pp. 352-355.

[9] E. G. Woldu and F. A. Jokhio, "Experiments with Vehicle Platooning," Halmstad University, 2010.

[10] D. Martinec and Z. Hurák, "Vehicular platooning experiments with LEGO MINDSTORMS NXT," in *Control Applications (CCA), 2011 IEEE International Conference on*, 2011, pp. 927-932.

[11] D. Martinec, M. Sebek, and Z. Hurak, "Vehicular platooning experiments with racing slot cars," in *Control Applications (CCA), 2012 IEEE International Conference on*, 2012, pp. 166-171.

[12] (23 June). *Mobile Robot Base Set (Orange). Cytron Technologies Sdn Bhd*. Available: http://cytron.com.my/viewProduct.php?pcode=HD-BSC-O&name=Mobile%20Robot%20Base%20Set%20(Orange)

[13] *RC Servo Motor. Cytron Technologies Sdn Bhd*. Available: http://cytron.com.my/viewProduct.php?pcode=C36R&name=RC%20Servo%20Motor

[14] (23 June). *Arduino Uno Rev3-Main Board. Cytron Technologies Sdn Bhd*. Available: http://cytron.com.my/viewProduct.php?pcode=ARDUINO-UNO&name=Arduino%20Uno%20Rev3-Main%20Board

[15] M. McRoberts, "Arduino starter kit manual," *A complete beginner's guide to the Arduino. First edition. Miami, USA: Earthshine electronics*, 2009.

[16] "IR Receiver Modules for Remote Control Systems (TSOP348..)," ed: Vishay Semiconductors, 2003, p. 7.

[17] Ladyada. (2012, June 23). *IR Sensor*. Available: http://learn.adafruit.com/ir-sensor

[18] D. A. Mellis. (3 Nov 2008). *Ping))) Sensor*. Available: http://www.arduino.cc/en/Tutorial/Ping

[19] "PING))) Ultrasonic Distance Sensor," ed: Parallax Inc, 2008, p. 12.

[20] "Infrared Light Emitting Diode," ed: Fairchild Semiconductor, 2001, p. 4.

[21] A. Lindsay. (2012). *Navigating with Infrared Headlights- Infrared Light Signals*. Available: http://learn.parallax.com/node/300

[22] K. Shirriff. (2009, November 9). *A Multi-Protocol Infrared Remote Library for the Arduino*. Available: http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

[23] (23 June). *ARDUINO*. Available: http://arduino.cc/en/

[24] E. Roche and Y. Schabes, *Finite-state language processing*: The MIT Press, 1997.

[25] *Language Reference. Arduino*. Available: http://arduino.cc/en/Reference/HomePage

[26] (2012). *How to create a 38 Khz pulse with arduino using timer or PWM?* Available:

http://forum.arduino.cc/index.php?PHPSESSID=4comitqdbb9iru48ikp07e4gm6&topic=102430.30

[27] *Adjusting PWM Frequencies*. Available: http://playground.arduino.cc/Main/TimerPWMCheatsheet

[28] guibot. *Control your motors with L293D and Arduino*. Available: http://www.instructables.com/id/Control-your-motors-with-L293D-and-Arduino/

APPENDICES

```
/* Author: Cheam Syeh Ren
 *         13044
 *         Electrical and Electronics Eng.
 *         Universiti Teknologi PETRONAS
 *
 * Follower Vehicle
 * Last Updated: 9 Dec 2013
 */

//assign ir pins
int ir_pin[]={7,6,5,4,3};
int ir[5];

//define states
int state = 0;  //declare current state as 0=idle
#define idle        0
#define go_left     1
#define go_forward  2
#define go_right    3
#define go_reverse  4
#define go_forward_left  5
#define go_forward_right 6
#define go_reverse_left  7
#define go_reverse_right 8

//init for distance
const int pingPin = 12;
#define max_dist 15 //maximum working distance
#define min_dist 10 //minimum working distance
long duration, cm;  //for read_ping()

//assign motor pins
int motor_left[]={8,9};
int motor_right[]={11,10};
long ms = 200;  //time in millisecs for motor to move

//assign led pins
int led_close = 14;
int led_far = 15;
int led_right = 16;
int led_middle = 17;
int led_left = 18;
```

```
//define variables
#define ON  0
#define OFF 1
#define LEFT_MOST  0  //most left
#define LEFT       1  //left
#define MIDDLE     2  //middle
#define RIGHT      3  //right
#define RIGHT_MOST 4  //most right

void read_ping()
{
  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);
  cm = duration / 29 / 2;  // convert time to distance cm
}
void setup()  // Built-in initialization block
{
  int i;
  for(i=0; i<5; i++){
  pinMode(ir_pin[i], INPUT);
  }

  for(i=0; i<2; i++){
  pinMode(motor_left[i], OUTPUT);
  pinMode(motor_right[i], OUTPUT);
  }

  pinMode(led_left, OUTPUT);
  pinMode(led_middle, OUTPUT);
  pinMode(led_right, OUTPUT);
  pinMode(led_close, OUTPUT);
  pinMode(led_far, OUTPUT);
  Serial.begin(9600);      //Set Serial rate as 9600 bps.
                           //Serial is used for debugging purposes only.
}
```

```
void loop()    // Main loop auto-repeats
{
  for(int i=0; i<5; i++){
    ir[i] = digitalRead(ir_pin[i]); //read the status of ir pins
  }

  read_ping();  //read distance from ping sensor
  delay(5);  //puts delay so sensor has time to read


  /* Codes for LED indicators */
  if (ir[LEFT_MOST]==ON || ir[LEFT]==ON) digitalWrite(led_left, HIGH);
    else digitalWrite(led_left, LOW);

  if (ir[MIDDLE]==ON) digitalWrite(led_middle, HIGH);
    else digitalWrite(led_middle, LOW);

  if (ir[RIGHT]==ON || ir[RIGHT_MOST]==ON) digitalWrite(led_right, HIGH);
    else digitalWrite(led_right, LOW);

  if (cm<min_dist) digitalWrite(led_close, HIGH);
    else digitalWrite(led_close, LOW);

  if (cm>max_dist) digitalWrite(led_far, HIGH);
    else digitalWrite(led_far, LOW);
  /* End of LED Indicators' codes*/

  /* Finite State Machine of all possible states */
  switch(state){
    case idle:
      stop();
      //go forward if too far
      if (ir[MIDDLE] == ON && cm>max_dist)
        state = go_forward;
      //go reverse if too close
      if (ir[MIDDLE] == ON && cm<min_dist)
        state = go_reverse;
      //go left if left IR on and within safety distance
      if ((ir[LEFT] || ir[LEFT_MOST]==ON) && min_dist>cm>max_dist))
        state = go_left;
      //go right if right IR on and within safety distance
      if ((ir[RIGHT] || ir[RIGHT_MOST]==ON) && min_dist>cm>max_dist))
        state = go_right;
      //go forward left if left IR ON and too far
      if (ir[LEFT_MOST] == ON && (cm>max_dist))
        state = go_forward_left;
      if (ir[LEFT] == ON && (cm>max_dist))
        state = go_forward_left;
      //go forward right if right IR ON and too far
      if (ir[RIGHT] == ON && (cm>max_dist))
        state = go_forward_right;
      if (ir[RIGHT_MOST] == ON && (cm>max_dist))
        state = go_forward_right;
```

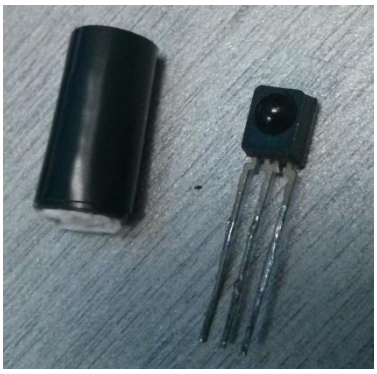
```
  //go reverse left if left IR ON and too close
  if (ir[LEFT_MOST] == ON && cm<min_dist)
    state = go_reverse_left;
  if (ir[LEFT] == ON && cm<min_dist)
    state = go_reverse_left;
  //go reverse right if right IR ON and too close
  if (ir[RIGHT] == ON && cm<min_dist)
    state = go_reverse_right;
  if (ir[RIGHT_MOST] == ON && cm<min_dist)
    state = go_reverse_right;
  break;

case go_forward:
  forward(ms); //drive forward for ms time
  if (ir[MIDDLE] == OFF)// no detection
    state=idle;
  if (ir[MIDDLE] == ON && cm<max_dist)//within safety distance
    state=idle;
  break;

case go_reverse:
  reverse(ms); //reverse for ms time
  if (ir[MIDDLE] == OFF)//no detection
    state=idle;
  if (ir[MIDDLE] == ON && cm>min_dist) //within safety distance
    state=idle;
  break;

case go_left:
  left(ms); //turn left for ms time
  if (ir[MIDDLE] == ON)//object tracked
    state=idle;
  if (ir[LEFT_MOST] == OFF && ir[LEFT] == OFF)//no detection
    state=idle;
  break;

case go_right:
  right(ms); //turn right for ms time
  if (ir[MIDDLE] == ON)//object tracked
    state=idle;
  if (ir[RIGHT] == OFF && ir[RIGHT_MOST] == OFF)//no detection
   state=idle;
  break;

case go_forward_left:
  forward_left(ms);// go forward left for ms time
  if (ir[LEFT_MOST] == OFF && ir[LEFT] == OFF)//no detection
    state=idle;
  if (ir[MIDDLE] == ON)//object tracked
    state=idle;
  break;
```

```
    case go_forward_right:
      forward_right(ms);// go forward right for ms time
      if (ir[RIGHT] == OFF && ir[RIGHT_MOST] == OFF)//no detction
        state=idle;
      if (ir[MIDDLE] == ON)//object tracked
        state=idle;
      break;

    case go_reverse_left:
      reverse_left(ms);//reverse left for ms time
      if ((ir[LEFT_MOST]== OFF) && (ir[LEFT] == OFF))//no detction
        state=idle;
      if (ir[MIDDLE] == ON)//object tracked
        state=idle;
      break;

    case go_reverse_right: //Reverse right
      reverse_right(ms);
      if ((ir[RIGHT] == OFF) && (ir[RIGHT_MOST] == OFF))//no detection
        state=idle;
      if (ir[MIDDLE] == ON)//object tracked
        state=idle;
      break;

    default:
      stop();
      break;
  }
}

/* Vehicle Manoeuvre */
void forward(int time)
{
  digitalWrite(motor_right[0], HIGH);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[0],HIGH);
  digitalWrite(motor_left[1], LOW);
  delay(time);
}

void left(int time)
{
  digitalWrite(motor_right[0], HIGH);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[0],LOW);
  digitalWrite(motor_left[1], HIGH);
  delay(time);
}
```

```
void right(int time)
{
  digitalWrite(motor_right[0], LOW);
  digitalWrite(motor_right[1], HIGH);
  digitalWrite(motor_left[0],HIGH);
  digitalWrite(motor_left[1], LOW);
  delay(time);
}

void reverse(int time)
{
  digitalWrite(motor_right[0], LOW);
  digitalWrite(motor_right[1], HIGH);
  digitalWrite(motor_left[0],LOW);
  digitalWrite(motor_left[1], HIGH);
  delay(time);
}

void reverse_right(int time)
{
  digitalWrite(motor_right[0], LOW);
  digitalWrite(motor_right[1], HIGH);
  digitalWrite(motor_left[0],LOW);
  digitalWrite(motor_left[1], LOW);
  delay(time);
}

void reverse_left(int time)
{
  digitalWrite(motor_right[0], LOW);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[0],LOW);
  digitalWrite(motor_left[1], HIGH);
  delay(time);
}

void forward_right(int time)
{
  digitalWrite(motor_right[0], LOW);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[0],HIGH);
  digitalWrite(motor_left[1], LOW);
  delay(time);
}
```

```
void forward_right(int time)
{
  digitalWrite(motor_right[0], LOW);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[0],HIGH);
  digitalWrite(motor_left[1], LOW);
  delay(time);
}

void forward_left(int time)
{
  digitalWrite(motor_right[0], HIGH);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[0],LOW);
  digitalWrite(motor_left[1], LOW);
  delay(time);
}

void stop()
{
  digitalWrite(motor_right[0],LOW);
  digitalWrite(motor_right[1], LOW);
  digitalWrite(motor_left[1],LOW);
  digitalWrite(motor_left[0], LOW);
}
```

Appendix 1: Follower Vehicle's Sketch

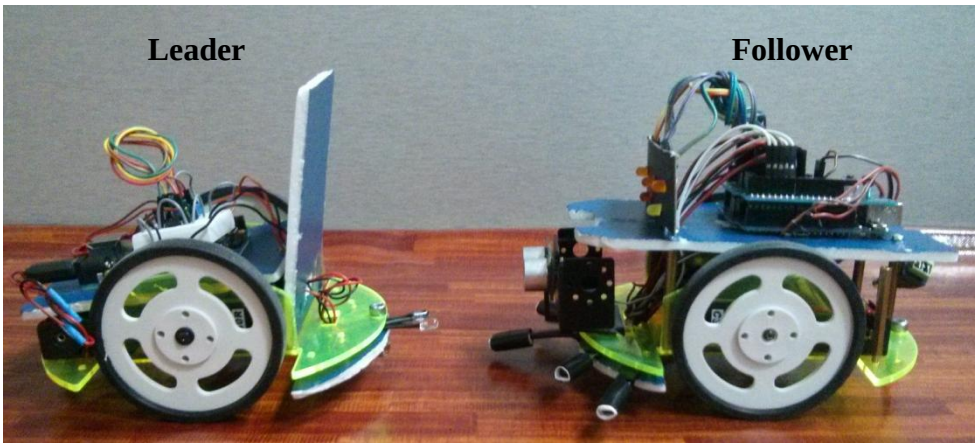


Appendix 2: Infrared Receiver with non-reflective sleeve

Appendix 3: PING))) Ultrasonic Distance Sensor [19]

Appendix 4: Mobile Robot Base Set (Robot Chassis) [12]

Appendix 5: Arduino UNO board [14]



Appendix 6: Leader and Follower Vehicle