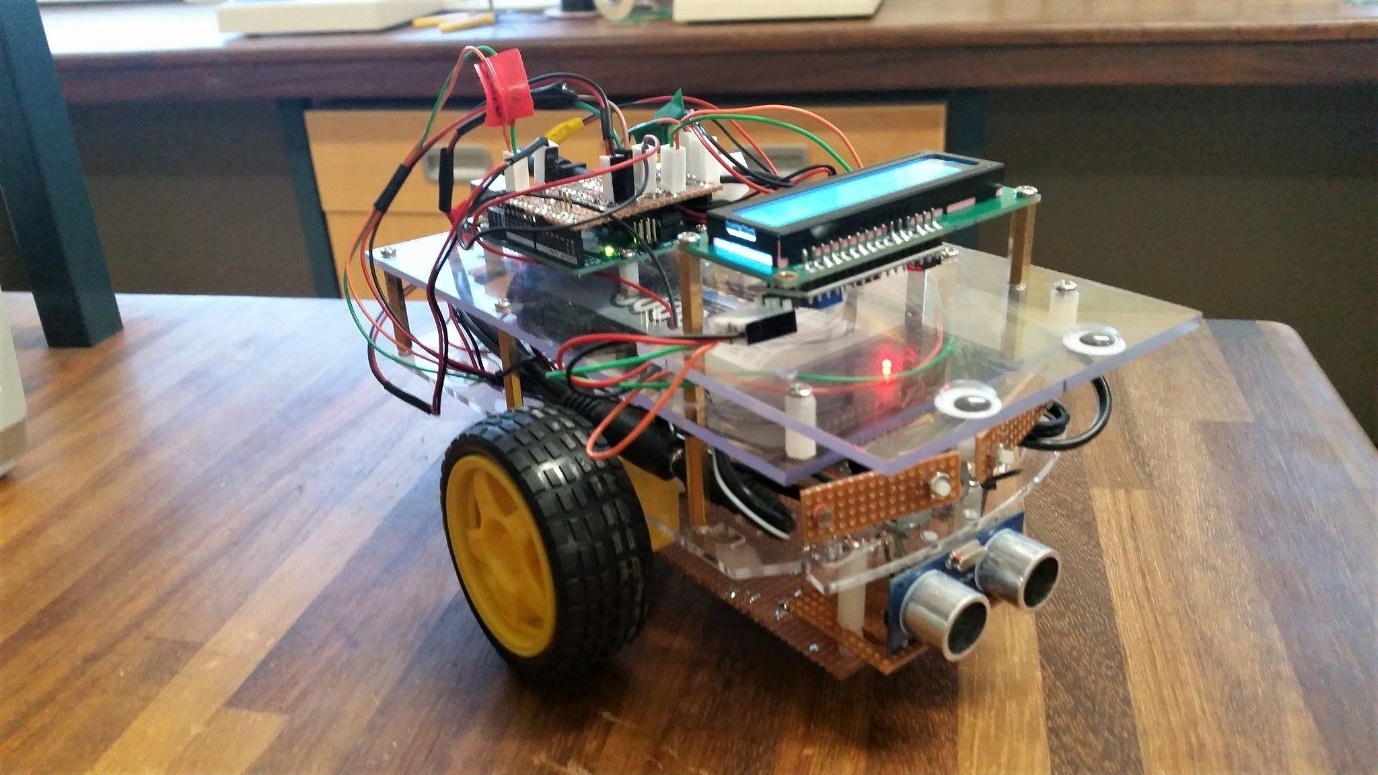
Group robotics project

Individual Report

Jack Rome

H00219766



# Contents Page

|  |  |
| --- | --- |
| Title | Page |
| Introduction | 2 |
| Background Theory | 3 |
| Program | 4,5 |
| Testing Procedures | 6 |
| Performance and Discussion | 7 |
| Conclusion | 8 |

# Introduction

The following report will discuss the workings of the collision avoidance sub-system. The report will cover background theory on the workings of the sensor, the physics behind the ultrasonic as well as explaining the program the Arduino board uses. The design choices used for this sub-system will be explained.

This report will also detail the testing of the subs-system and will include a discussion of its final performance.

The aim of the sub-system is to provide the robot with the ability to autonomously roam without crashing into any obstacles. This will be achieved by using an ultrasonic sensor distance sensor as the only input to the Arduino board.

An example application of this technology would be parking assists in modern vehicles. ‘In the case of the rear sonar, two to four ultrasonic sensors are mounted on the rear bumper to detect an obstacle up to 2 to 2.5m away’ (newelectronics.co.uk). This parallels the method that this subsystem uses to avoid collision.

# 2. Background Theory

The collision avoidance sub-system uses an ultrasonic sensor as its input. This sensor finds the distance to the closest object ahead of it. The sensor’s manual quotes an effective range of 30O.

Setup:



The sensor has 4 pins. These are labelled: Vcc, Trig, Echo and Gnd. The voltage pin (red) is connected to the Arduino to the +5V rail and provides the power to the sensory component. The Trigger input of the sensor (orange) was connected to pin 13 of the Arduino board and the Echo input of the sensor (green) was connected to pin 12. The ground (black) was connected to a ground pin on the Arduino board.

Figure 1: Ultrasonic sensor showing the colours of wires used

The sensor was installed onto the front of the chassis on a piece of Veroboard, with the wires soldered onto the Veroboard behind it.

Theory:

The sensor works by sending and receiving pulses and using the difference in receival time to calculate the distance to the closest object. The sequence initiates with a 5V signal to the trig pin for at least 10µs. The sensor will then send a burst of 8 40kHz ultrasonic pulses and wait for the reflection of the burst. When the reflection is received, the echo pin will delay proportionately to the distance. Formulae can now be used to obtain the distance in either centimetres or inches. Centimetres will be used as the unit of distance with the program.

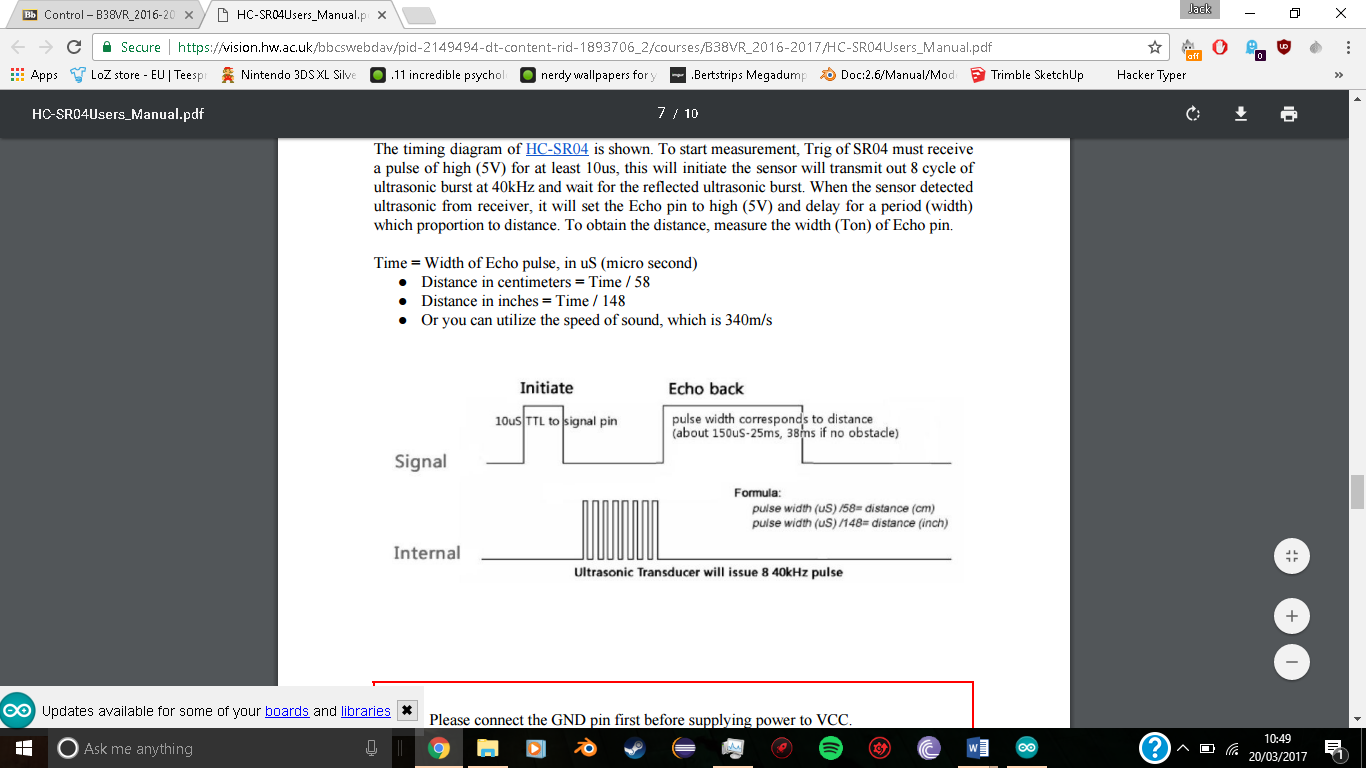


Figure 2: Timing Diagram of the sensor’s sequence

# Program

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C address

int trigPin = 13; //Trig - orange Jumper

int echoPin = 12; //Echo - green Jumper

int in1Pin = 6;

int in2Pin = 10; //motor control

int in3Pin = 9;

int in4Pin = 8;

int enablePin1 = 11;

int enablePin2 = 5;

long duration, cm;

int leftspeed = 252;

int rightspeed = 255;

long dist = 20;

void setup() {

//Serial Port begin

**Serial**.begin (9600);

//Define inputs and outputs

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(2, INPUT);

pinMode(in1Pin, OUTPUT);

pinMode(in2Pin, OUTPUT);

pinMode(in3Pin, OUTPUT);

pinMode(in4Pin, OUTPUT);

pinMode(enablePin1, OUTPUT);

pinMode(enablePin2, OUTPUT);

initiateLCD();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Hello Jack");

delay(2000);

}

void loop()

{

boolean reverse = 0;

setMotor(leftspeed, rightspeed, reverse, reverse);

// The sensor is triggered by a HIGH pulse of 10 or more microseconds.

// Give a short LOW pulse beforehand to ensure a clean HIGH pulse:

scan();

**Serial**.print("forward ");

**Serial**.print(cm);

**Serial**.print("cm");

**Serial**.println();

while (cm <= dist)

{

setMotor(0, 0, 0, 0);

delay(10);

setMotor(leftspeed, rightspeed, 0, 1);

**Serial**.println("Rotate");

delay(250);

setMotor(0, 0, 0, 0);

delay(10);

scan();

}

}

void scan()

{

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(5);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

cm = (duration / 2) / 29.1;

lcd.setCursor(0, 1);

lcd.print("Distance: ");

lcd.print(cm);

lcd.print("cm");

delay(50);

}

void setMotor(int leftSpeed, int rightSpeed, boolean reverseL, boolean reverseR)

{

analogWrite(enablePin1, rightSpeed); //pwm the enable pin with the speed value for the right motor

digitalWrite(in1Pin, ! reverseL);

digitalWrite(in2Pin, reverseL);

analogWrite(enablePin2, leftSpeed); //pwm the enable pin with the speed value for the left motor

digitalWrite(in3Pin, ! reverseR);

digitalWrite(in4Pin, reverseR);

}

void initiateLCD() {

lcd.begin(16, 2); // initialize the lcd for 16 chars 2 lines, turn on backlight

// ------- Quick 3 blinks of backlight -------------

for (int i = 0; i < 3; i++)

{

lcd.backlight();

delay(250);

lcd.noBacklight();

delay(250);

}

lcd.backlight(); // finish with backlight on

}

# Testing Procedures

Two sets of tests were set used to assess the functionality of the sub-system. The first test was a component test and the next tested the program with the robot.

**Component Testing:**

This test was accomplished using the Arduino software’s ‘Serial Monitor’ tool to check the input the board would receive from the sensor. The program would print the distance in centimetres that had been calculated using the sensor. The aim of this test was to check that the sub-system was wired correctly and that it could differentiate between near and distant objects.

The test will involve placing the sensor a measured 30cm from a wall and slowly moving the sensor towards the wall with the distance readings being printed to the serial monitor once every half second. If the sensor performs correctly, it is expected that the output readings will begin at 30 and decrement accordingly towards 0.

**Program Testing:**

This test assessed the program used to control the robot. This involved letting the robot attempt to autonomously navigate using the sensor. Testing the program allowed for improvements to be made to the program and the overall system. The aim of this test was to check if the entire sub-system operated as stated in the introduction. That is, to navigate autonomously without colliding with obstacles.

# 5. Performance and Discussion

|  |
| --- |
| Serial monitor output |
| 30cm, 30cm, 30cm, 29cm, 26cm, 24cm, 22cm, 19cm, 17cm, 16cm, 14cm, 12cm, 11cm, 10cm, 8cm, 6cm, 6cm, 4cm, 3cm, 3cm, 2cm, 2cm, 1cm, 1cm |

The result for the component testing are in the table below:

The results decremented, as expected, while the robot was moved continuously closer to the wall. This shows that the sensor behaves reliably and that the wiring of the components is correct.

It should be noted that the serial monitor began to fluctuate when the robot was left idle. The serial monitor produced outputs that increased and decreased by a centimetre randomly despite the sensor not being moved.

Whilst testing the program, the distance to the closest object was displayed on an LCD screen. This acted as a portable version of the serial monitor and allowed us to view the input the sensor received without having the Arduino board connected to a laptop.

In practice the robot performed best when the motors were set at maximum speed (255 pulse width modulation value), but it should be noted that the right motor did not operate as fast as the left motor so the left motor speed had to be reduced to compensate for that. The tests also found that the maximum distance should be set to 20cm before a turning method should be followed.

The optimal motor speeds were achieved by a trial and error approach. A simple program was created to drive both motors at maximum speed and the left motor speed was adjusted after every test in attempt to fix this motor error and drive the robot in as straight a path as possible. The result was a reduction of 3 out of 255 to the left motor.

Other amendments that had to be made to the program code were adjusting timings of the delays in the turning, waiting, sensor operation, etc. I also made the code more efficient by creating a subprogram for the sensor use rather than repeat the same code multiple times and finally I made amendments to the motor control subprogram so that the Boolean ‘reverse’ was changed to two spate parameters: ‘reverseL’ and ‘reverseR’. This was so that the motors could be reversed separately.

# 6. Conclusion

Overall, the collision avoidance sub-system performed very well and achieved the aims set in the introduction. The robot could roam around the lab completely autonomously and could even be left unmonitored for a while and still not collide with anything.

As construction of the robot’s chassis continued, a plate glass was installed above the sensor to hold the LCD screen. This acted as a safety guard whilst testing the program as, in the early stages, the robot would often require protection against crashing, at full speed, into a wall or stool leg.

A trouble with this setup was the fact that there was only one sensor. This meant that the robot would often have trouble with objects that were out with its 30O range and, therefore, would sometimes collide with narrow objects such as stool legs. This could be solved by adding more sensors and different positions on the robot and using these to obtain a better judgement on the best direction to travel.

On the other hand, a different improvement that could be made with this sub-system would be the inclusion of a servo motor. This would provide the ability to scan an area and select the best path rather than the robot’s current method of rotating until the first free path is found. In this scenario, the sensor would need to be moved to the top of the robot, with the Veroboard it is soldered on being mounted onto the servo motor. This idea has its advantages and disadvantages when compared to the previous idea.

# 7. References

1. The information on ultrasonic parking assist was taken from this website:

<http://www.newelectronics.co.uk/electronics-technology/an-introduction-to-ultrasonic-sensors-for-vehicle-parking/24966/>

1. Figure 1 was taken from this website:

<http://www.mtechlog.com/2015/09/using-ultrasonic-sensor-with-arduino.html>

1. Figure 2 was taken from ultrasonic distance sensor manual provided on vision:

<https://vision.hw.ac.uk/bbcswebdav/pid-2149494-dt-content-rid-1893706_2/courses/B38VR_2016-2017/HC-SR04Users_Manual.pdf>

1. The code on page 4 and 5 were adapted from the program on the Arduino website:

<http://playground.arduino.cc/Main/UltrasonicSensor>