Toby Thomas

Building a line following robot using an Arduino Uno

Word count: 2985 not including diagrams and references

2985

Introduction

The ability of a robot to perform tasks autonomously has applications in every industry including healthcare. The aim of this report is to design and evaluate the ability of a robot to follow a line around a track made of black paint and detect obstacles in its path.

The aging population and the lack of sufficient staffing numbers in NHS combined with the poor effectiveness of recent flu vaccines has increased the pressure on hospitals across the UK. This increased pressure has caused longer waiting times in A&E departments and been a cause of cancelation of many elective operations. Some staff must distribute resources such as equipment or medicine among wards and departments which can become a time-consuming process. By automating this process, resources can be distributed more efficiently, freeing up staff who are then free to perform other jobs, helping reduce the workload on the front-line nurses and doctors. One method of implementing this would be to use a line-following robot to transport supplies within the hospital. This report details the design and testing process of building such a robot that uses Infra-red (IR) LED and receiver modules to track a black line on a white surface. It also needs to be able to detect obstacles on its track and is fitted with an ultrasonic (US) distance transducer to detect distance and speed of objects in its way. The robot will be controlled using the popular micro-controller: Arduino Uno.

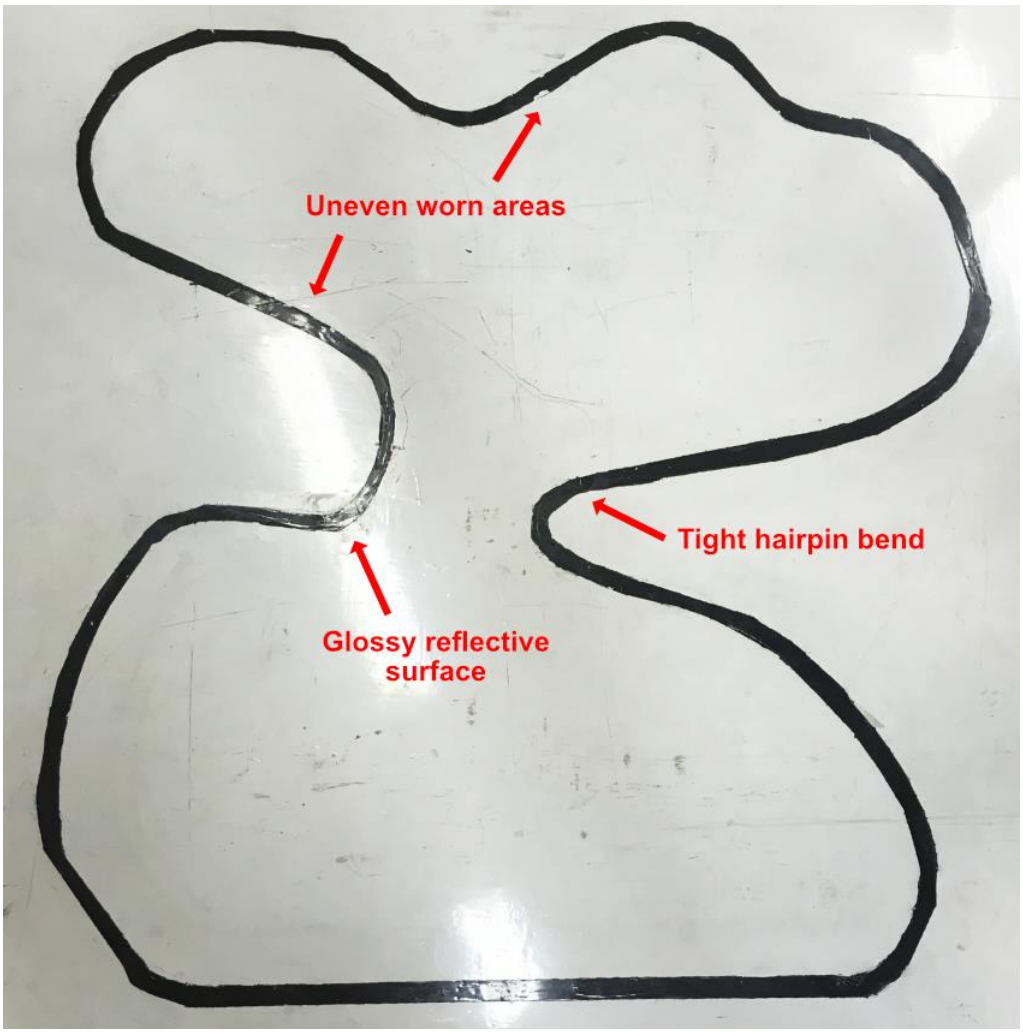
Contents:

1. Description of problem

The objectives of this project are to:

- Construct a mechanical system that forms a vehicle platform with two independently driven wheels based on the Dagu Magican chassis;
- Operate the driven wheels with PM DC brush monitors through the Adafruit motor shield V2 and Arduino Uno;
- Attach 1 or more IR line sensors to control orientation of the buggy and operate these with the Arduino Uno;
- Attach a servo-driven pan and tilt mechanism to the buggy to control the orientation of an US distance transducer;
- Program the 'buggy' to follow a black line on a white background whilst avoiding collisions with the surroundings.

The evaluation of the buggy will be related on its performance relating to each of these objectives. The test track was constructed from a plywood sheet pained white with a black loop painted on. The track consisted of one long straight section, several gentle corners and one hairpin bend.



2. Description of components

2.1 Degu Magican chassis

The chassis of the robot is based on the Dagu Magical Chassis. It comprises of 2 main plates with attachments for 2 DC brush Motors which allow for wheels of width 65mm. There are a variety of mounting holds for the attachment of sensors, micro-controller and battery pack. It also comes with a rear caster mounted on spacers.

[Image 1]

[Image 2]

2.2 Arduino Uno

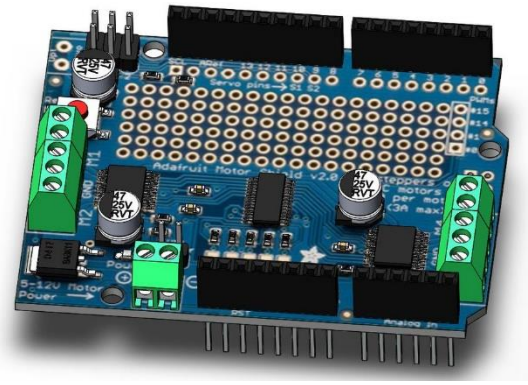
Th Arduino Uno is a micro-controller based on the ATMega328. It has 14 I/O connections and 6 analogue inputs with 32kb of programmable memory, 2kb of SRAM and 1kb of EEPROM. It can be powers with via the USB serial connection or by external power supplies from 7V to 12V. The voltage output of the I/O pins was set to 5V. The code was written using the Arduino IDE version 8.1.5 in c/c++.

[Image 3]

2.3 Adafurit Motor Shield V2

The Motor Shield allows for the connection of up to 4 bi-directional DC brush motors and 2 servos without using up any of the I/O ports by being installed directly on top of the Arduino. This is useful for projects such as robots that have multiple moving parts to keep

the ports free for sensors. It has a dedicated Pulse Width Modulation driver (PWM) which allows more smooth motor speed control compared to using a L293D motor driver. The direction of the motors is controlled using 4 separate H-bridges with transistors and there is also a dedicated PWM control chip. It also has built in diodes to prevent back emfs from the motor affecting the rest of the circuit. The motors can all be controlled using the functions defined in the Adafruit Motor Shield V2 Library. Power can be supplied from either external batteries, or by using a jumper to bypass these pins, directly from the usb or onboard power supply.



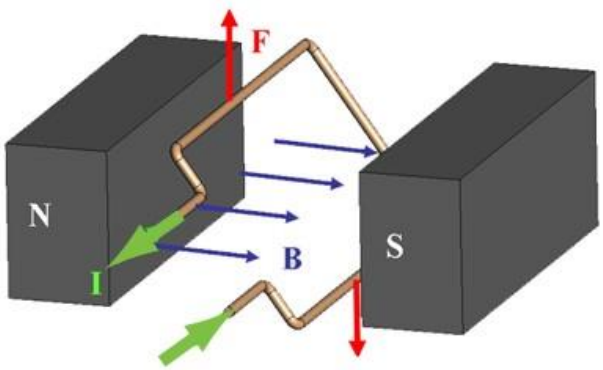
[image 4]

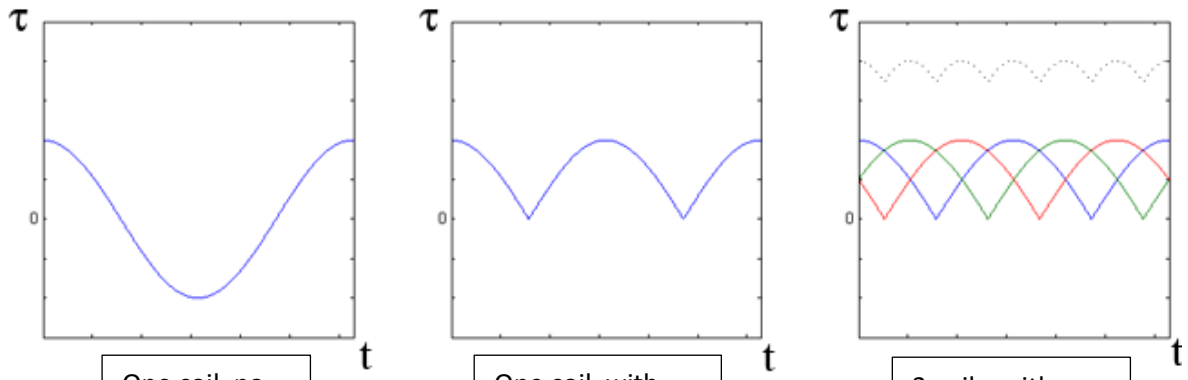
2.4 DC Brush Motors

These motors consist of a wire placed inside a magnetic field that is free to rotate on an armature. When current passes through the wire, forces are created in directions perpendicular to the current and magnetic field according to the right-hand rule of electromagnetism, causing the coil to rotate. A commutator switches the direction of the current through the coil rotates, ensuring the forces continue to cause rotation rather than slowing the coil down after half a turn. Peak force occurs when the wire is moving perpendicular to the coil. Positions either side of this produce a lower torque. To prevent this drop off, motors have multiple coils and so the torque output is more stable.

The Speed of a motor can be controlled by using PWM to simulate an analogue signal to the motor. The coil has mass and therefore inertia, which is why the acceleration is not istant when a signal is applied

[Image 5]





One coil, no commutator

One coil, with commutator

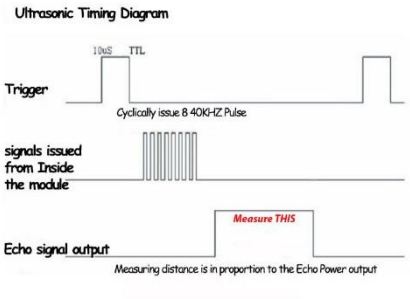
3 coils, with commutator

[Image 15]

2.5 Ultrasonic distance transducer

This device has four pins: 5v voltage in, trigger, echo and ground. By setting the trigger pin to HIGH for 10µs, a 40 000 Hz eight pulse burst is emitted which bounces back off any objects in front of the sensor. The distance of an obstacle can be calculated by considering the travel time of the reflected pulse and the speed of sound in the medium. The relative speed of an approaching object can be calculated by considering the change in distance between two measurements.

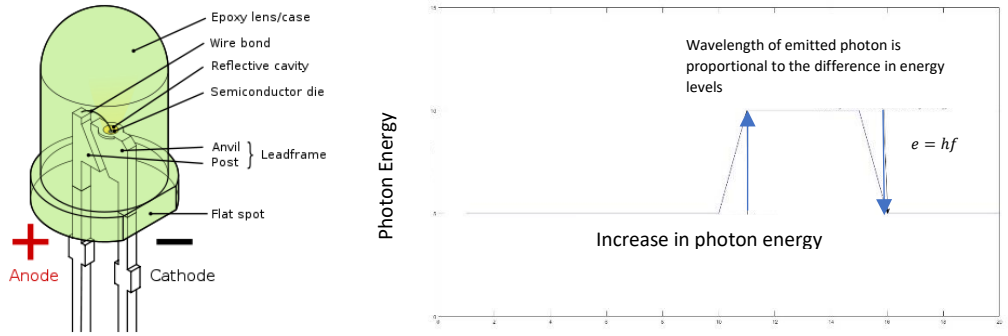
[Image 6]



[Image 7]

2.6 Light Emitting Diodes(LED) and resistor

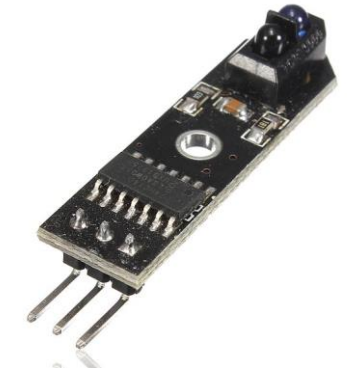
A LED uses the photoelectric effect to create emission of photons of a specific wavelength. Electrons in the semiconductor are excited to higher energy levels by an applied voltage. As they drop back down to lower energy levels the difference in energy is given in the form of photons of specific wavelength ($e = hf$). LEDs are diodes and only allow current to flow through one direction through them. This is a useful property when using a motor without a driver to prevent back emfs. LEDs have very low resistance so to prevent short circuiting they should be used in series with a resistor. LEDs were used in this project as indicators of proximity and objects approaching at high speed.



[Image 8]

2.7 Infra-Red Line tracker

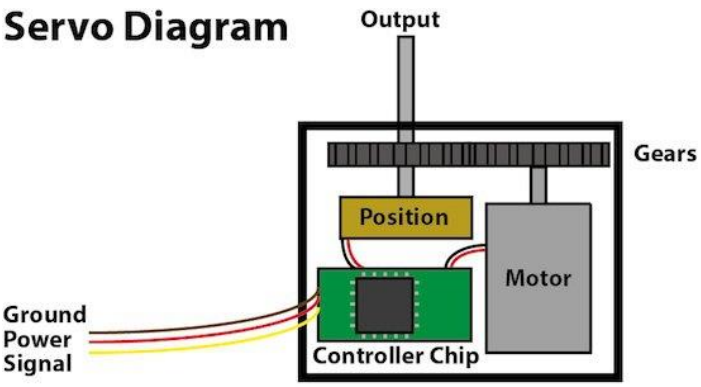
The Line tracker module consists of an LED that emits IR light and an IR light detector. If the light from the LED is reflected and detected by the module then, the output pin is given as HIGH. A white surface reflects all light wavelengths including IR, however if the sensor moves onto a black surface, all the light is absorbed and the output is LOW.



[Image 9]

2.8 Servo Motor

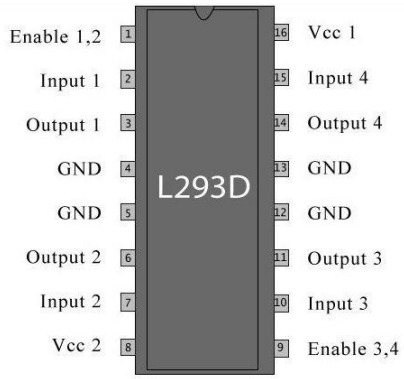
A servo motor can be used to accurately control the position of a part using a motor and controller chip. They consist of a motor, gearing and control circuitry. Servos are controlled using Pulse Width Modulated (PWM) signals which determines the position of the arm. To use a servo the <servo.h> library must be included in the code to create a servo object. This buggy used 2 servo motors and a hinge to create a bracket for the US sensor with 2 degrees of freedom.



[Image 10]

2.9 L293D Motor driver

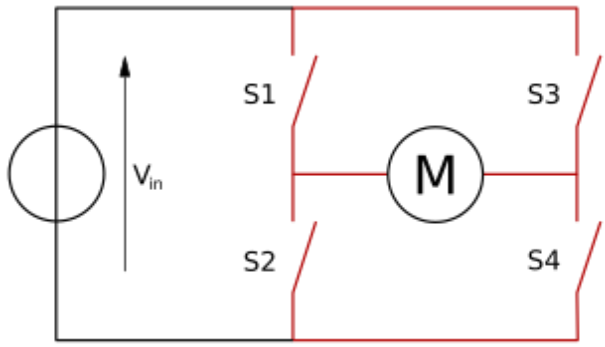
The L293D Motor Driver is a chip containing logic circuits that can be used to control up to 2 DC motors bi-directionally. Each motor is controlled from the pins of each side of the chip, by setting Input 1/3 and 2/4 to opposite states. Vcc 1 (5v) provides the voltage for the logic circuit whereas vcc 2 (>32v) provides the voltage to run the motors. These chips are used on the Adafruit Motor Shield V1 but can be used independently.



[Image 11]

2.10 H-bridge Circuit

The H-bridge Circuit is a very useful method of direction control for a motor. It consists of a motor with 4 switches (or MOSFETs with diodes) in the arrangement described by [Image 12]. The logic can be summarised in the table below



[Image 12]

S1	S2	S3	S4	Result
1	0	0	1	Motor right
0	1	1	0	Motor left
0	0	0	0	Free run
0	1	0	1	Breaking
1	0	1	0	Breaking
1	1	0	0	Shoot Through
0	0	1	1	Shoot Through
1	1	1	1	Shoot Through

3 Design and Testing

3.1 Line Sensor placement

The most important design feature of the buggy was the placement of the line sensors. Many different designs were trailed, each if which would have to be adjusted. When experimenting with different designs there were 3 main considerations that were considered:

- Performance on a straight line
- Performance on gentle corners
- Performance on the hairpin bend

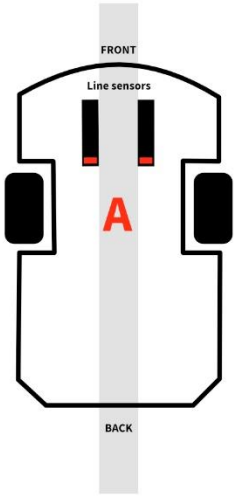
The configurations were categorised into 5 designs according to the angle between the line sensor and wheel axis, distance from the wheel axis and distance between sensors.

Testing of designs A, C and E showed that larger angles provided better sensitivity to corners, however short distances from the axis reduced that maximum turning radius of the buggy to the point at which it could not complete the hairpin bend. It was also noticed in design A that as the buggy attempted tight corners, the line between the sensors did not cross the line perpendicular. Because it crosses at an angle, both sensors detect black during the turn, stopping the buggy (see track below).

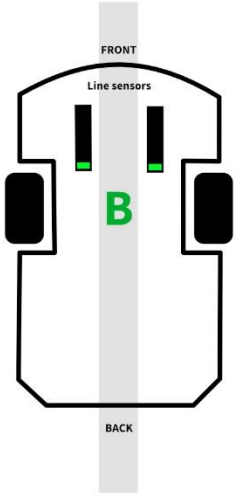
The large distance between sensors in Design E meant that the line was not detected until the wheel was on the line, at which point it was too late to correct for.

Design D performed well on the sharp corners, but was too sensitive on the straight section causing the buggy to make lots of unnecessary micro corrections, wasting a lot of time.

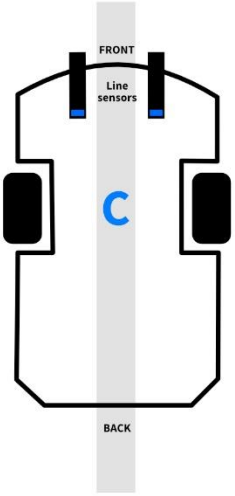
Design A proved to be the best design: it navigated the corner with reasonable ease while completing the track in an average on 32 seconds.



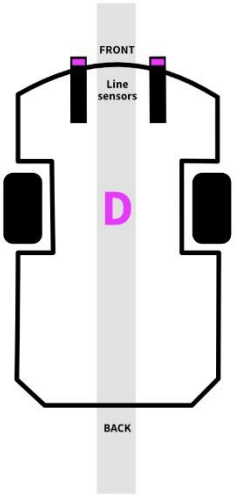
Design A (60-70°, 8cm, 4 cm)



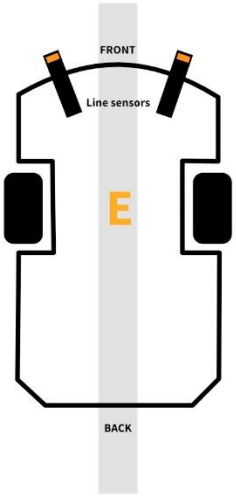
Design B (50-60°, 5cm, 4cm)



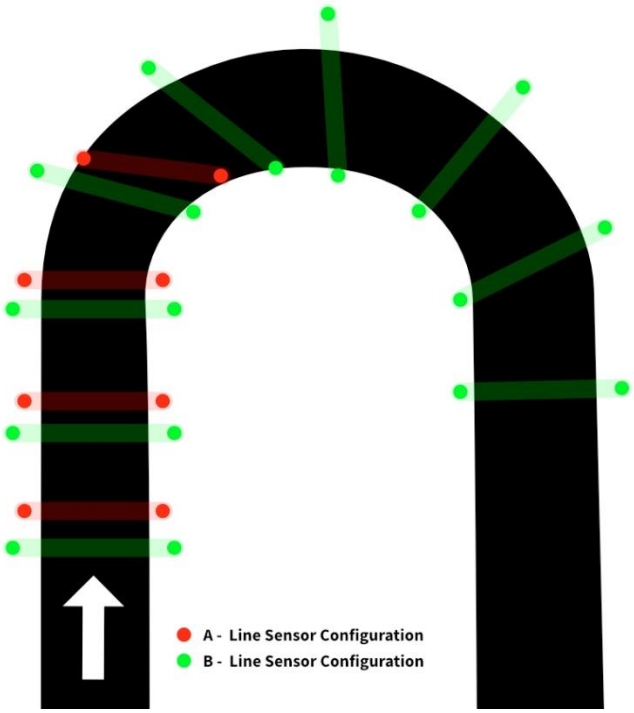
Design C (90°, 2cm, 8cm)



Design D (20-45°, 1cm, 3cm)

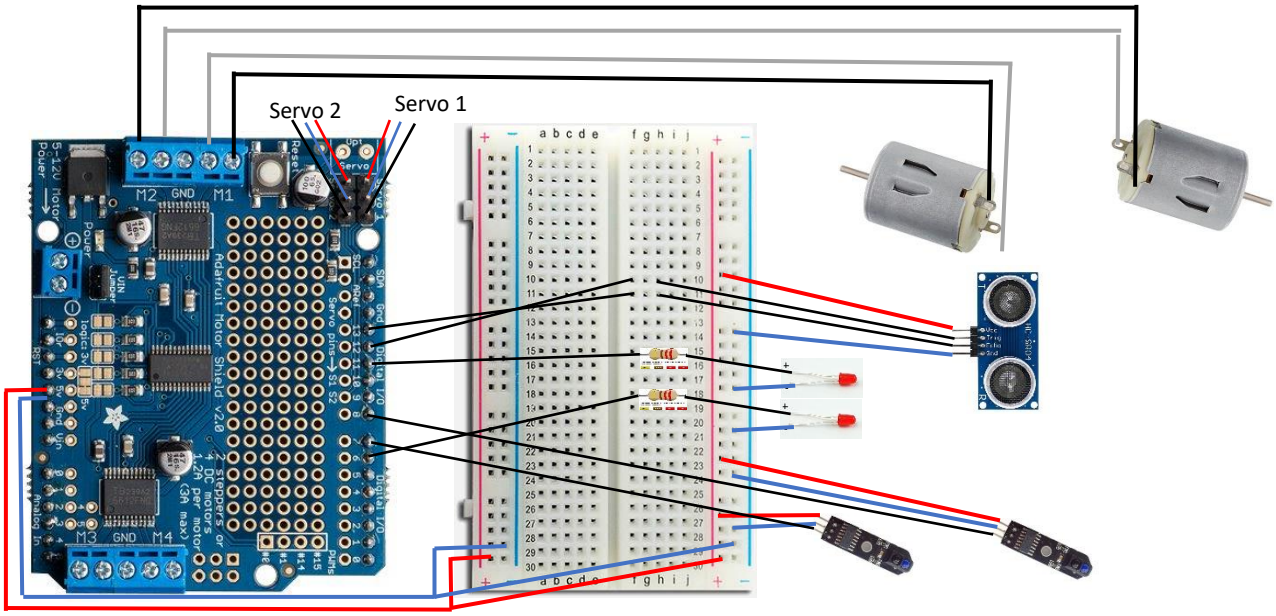


Design E (90-100°, 8cm, 10cm)

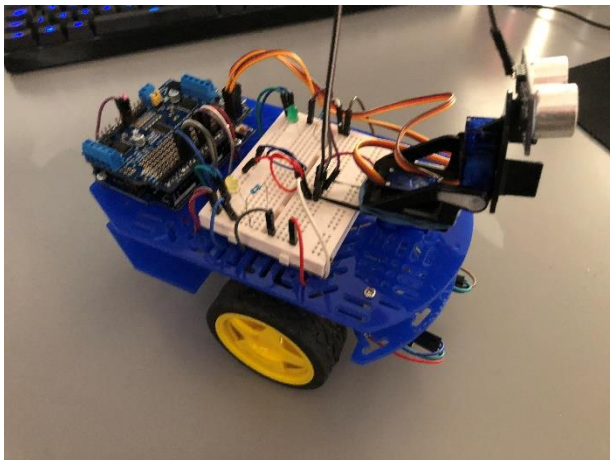

A - Line Sensor Configuration
B - Line Sensor Configuration

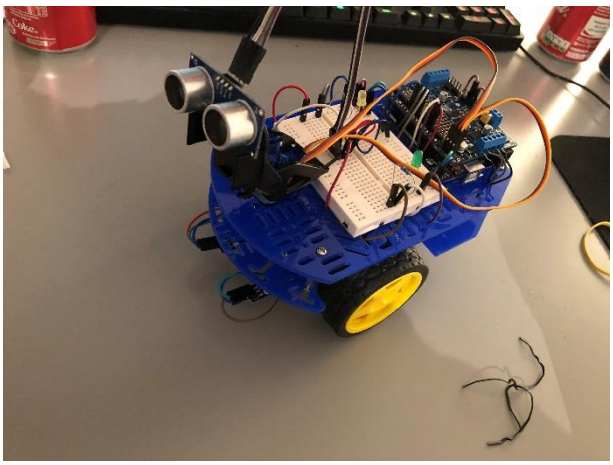
4 Assembly of components

The sensors were bolted on to the base of the baseplate of the buggy chassis using hexagonal spacers. The baseplate had attachment sites for the two DC motors which sat between the base and top plates. The Arduino, breadboard and servo were fixed to the top plate wired up according to the following diagram:



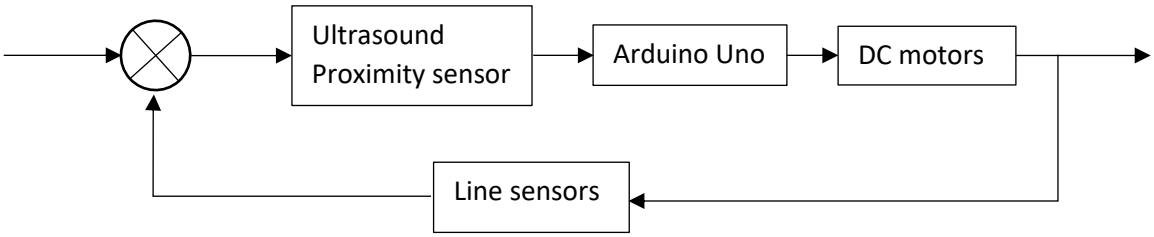
[Image 13]





5 Description of software

The buggy has a closed-loop design, where the output of both wheel speeds is dependent on the inputs from the line sensors.



There are 3 distinct functions that the code: the operation of the Ultrasound sensor; the output signals for the motor depending on the inputs from the line sensors; and the positioning of the pan and tilt motors.

5.1 Ultrasound

The first action in the ultrasound code is to ensure the trigger pin is set to LOW. After a short delay the trigger is set to HIGH which emits the 8 x 40kHz pulses. The echo is then received and the time is read from the module using pulseIn().

Now that the time has been measured, the distance can be calculated by considering the pulse has travelled to the object and returned i.e. twice the distance.

$$Distance = \frac{1}{2} * speed * time$$

This distance can then be stored for use in the next iteration to calculate the speed of an approaching object as the differential of change in distance.

$$Speed = \frac{x_i - x_{i-1}}{dt}$$

The distance and object speed are then compared to threshold values which determine whether to slow down or stop the motors.

```
6     //Ultrasound sensor
7     int motor_speed = 65;
8     int motor_speed_t = 30;
9     if (count_l > count_thresh || count_r > count_thresh) {
10      motor_speed = 100;
11    }
12    digitalWrite(TRIGPIN, LOW);                    //Force initial Trigger state to LOW
13    delayMicroseconds(2);
14    digitalWrite(TRIGPIN, HIGH);                   //generate US wave
15    delayMicroseconds(10);
16    duration = pulseIn(ECHOPIN, HIGH);              //read response time
17    distance = duration * 0.034 / 2;               //2 * distance = speed of sound * time
18    D_dist = (distance - prev_dist);               //find change in distance
19    obsped = D_dist / 0.1;                         //speed of obstacle = change in distance / time
20
```

```
21    // if object is within 40cm and approaching quickly - WARNING
22    if (obsped < obsped_thresh && distance < 40) {
23      Serial.println("WARNING OBJECT APROACHING QUICKLY");
24      digitalWrite(SLEDPIN, HIGH);
25      FLAG_slow_down = 1;
26    } else {
27      digitalWrite(SLEDPIN, LOW);                    // Turn Speed LED on
28      FLAG_slow_down = 0;                            //FLAG_slow_down to slow motors
29    }
30
31    // if object in within tol cm - STOP
32    if (distance < dist_thresh) {
33      Serial.println("WARNING obstruction");
34      digitalWrite(PLEDPIN, HIGH);                   // Turn Proximity LED on
35      FLAG_stop = 1;                                 // FLAG_stop to stop motors
36    } else {
37      digitalWrite(PLEDPIN, LOW);
38      FLAG_stop = 0;
39    }
40    prev_dist = distance;
41
```

5.2 Motor control and Line sensor input

The line sensors returned a value of HIGH if they were registering white or LOW if the surface is black. The robot is programmed to track the white floor wither side of the line and correct its direction when it detects that one side has crossed one line.

The direction of the buggy was changed by either stopping/slowing one wheel or reversing one of the wheels direction. Stopping/slowing one wheel made the robot smoother on most of the track but the radius the turning curve was larger than that of the hairpin corner, preventing it from turning tight enough to get around. This issue was overcome by stopping the wheel on a corner and beginning a counter which, after it exceeds a threshold value, causes the inside wheel to turn backwards according to some ratio, significantly decreasing the radius of the turn and allowing to successfully navigate the hairpin turn.

The motors can be controlled using a driver with H-bridges. The initial approach was to use a single L293D driver to control both motors, but at low speeds PWM was causing stuttering. The Motor Shield's dedicated PWM chip provided a better signal and produced a more consistent torque at low speeds.

The motors and

F = forward; B = backward; S_l = sensor 1; S_r = sensor_2; ` = not

Left Sensor	Right Sensor	Left Motor	Right Motor
1	1	F	F
1	0	F	B
0	1	B	F

0	0	B	B

	S_l . S_r	S_l . S_r`	S_l`, S_r`	S_l` . S_r
Motor_l - forward	1	1	0	0
Motor_l - backward	0	0	1	1

	S_l . S_r	S_l . S_r`	S_l`, S_r`	S_l` . S_r
Motor_r - forward	1	0	0	1
Motor_r - backward	0	1	1	0

$$Motor_l(forward) = S_l.(S_r + S_r`)$$

$$Motor_l(backward) = S_l`.(S_r + S_r`)$$

$$Motor_r(forward) = S_r.(S_l + S_l`)$$

$$Motor_r(backward) = S_r`.(S_l + S_l`)$$

Note: S_l and S_r are intended as arbitrary labels i.e S_l is not necessarily the left sensor, vice versa.

```
5   // Motor Control using line sensors
6   if (FLAG_slow_down) {                          //reduce speed if FLAG_slow_down== 1
7     motor_speed = motor_speed / 2;
8     Serial.println("go slow");
9   }
10  if (FLAG_stop == 1) {                                 //stop if FLAG_stop ==1
11    motor_1->setSpeed(RELEASE);
12    motor_2->setSpeed(RELEASE);
13    motor_speed = 0;
14    Serial.println("    stop - obs");
15  }
16  S_r = digitalRead(LINEPIN_1);                         //read signal from line sensor 1
17  S_l = digitalRead(LINEPIN_2);                         //read signal from line sensor 2
18  Serial.println(S_l, S_r);                      //print to serial monitor
19
20  if (S_l == HIGH && S_r == HIGH) {              // If both white
21    motor_1->setSpeed(motor_speed);                     //set both speeds to normalspeed
22    motor_2->setSpeed(motor_speed);
23    Serial.println("    FORWARD");
24    motor_1->run(FORWARD);                              //run motors in forward direction
25    motor_2->run(FORWARD);
26    count_l = 0;                                 //reset left and right turn counters
27    count_r = 0;
28  }
29  if (S_l == HIGH && S_r == LOW) {               //if sensor 1 is black
30    motor_1->setSpeed(motor_speed_t + count_l * 2);     //set inside wheel to increase by
    2*count
31    motor_2->setSpeed(motor_speed);                     //set outside wheel to normal speed
32    Serial.println("    left");
```

```
33    motor_1->run(BACKWARD);                          //run inside wheel backwards
34    motor_2->run(FORWARD);                           //run outside wheel forward
35    count_l = count_l + 1;            //Increment counter to keep track of number of interactions
36   }
37   if (S_l == LOW && S_r == HIGH) {                  //If sensor 2 is black
38    motor_1->setSpeed(motor_speed);                  //set outside wheel to normal speed
39    motor_2->setSpeed(motor_speed_t + count_r * 2);  //set inside wheel to increase by
   2*count
40    Serial.println("    right");
41    motor_1->run(FORWARD);                           //run outside wheel forward
42    motor_2->run(BACKWARD);                          //run inside wheel Backward
43    count_r = count_r + 1;            //Increment counter to keep track of number of interactions
44
45   if (S_l == LOW && S_r == LOW) {                   //if both black
46    motor_1->setSpeed(30);                           //set both motor speeds to as low as
   possible
47    motor_2->setSpeed(30);
48    Serial.println("    STOP - BOTH BLACK");
49    motor_1->run(BACKWARD);                          /*set both motors to creep back to  try
   and get back on track*/
50    motor_2->run(BACKWARD);
51    Serial.println("   ");
52   }
53   Serial.print("       COUNT L R: ");               //print both counts to help identify a good…
54   Serial.print(count_l);                           //…threshold value
55   Serial.print("      ");
56   Serial.println(count_r);
57
58  }
```

4.3 Servo control

The pan and tilt servos are controlled by including <servo.h> library and creating an object for each servo. A counter was incremented in every iteration which was used as the argument of sine and cosine then scaled to -90 to +90. These sinusoidal values were then written to the servo objects to define their position. The two servos were mounted with the US sensor on a hinge and orientated such that one controlled the panning direction and the other controlling the pitch.

```
42   //servo positioning
43   pos = pos + 1;                                   //increment pos in each iteration
44   servo_1.write(90 * sin(pos));
45   servo_2.write(90 * cos(pos));
```

5 Further Work

This design included an ultrasound sensor would prevent the buggy from colliding with obstacles. Further work will include using the pan and tilt servos to navigate around an obstacle and continue on the path.

Application of such a robot would require destination programming. One method of doing this is to have sensors that use different wavelengths of light, allowing them to track different colour paths to different destinations.

Practical implementation of a system like this would require remote control override of the buggy. This can be done simply using an IR remote are receiver module. Hospitals have wireless networks throughout departments which can be taken advantage of by installing a WLAN module on the buggy.

This project would have been trivial, except that the track used was designed to be difficult to navigate due to shiny surfaces, chipped paint and the hairpin bend. For more reliable results, further work should include investigation into the features of a track which is easier to navigate.

Finally, one significant cause of error in this project was the transient response of the motors to a step response when a corner was detected which caused the robot to overshoot. This behaviour can be investigated and used to optimise values to produce more reliable results

FINAL Code:

```
#include <Servo.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"

// Define PINS
const int ECHOPIN = 13;                          //echo pin
const int TRIGPIN = 12;                          //trigger pin
const int SLEDPIN = 11;                          //speed LED
const int PLEDPIN = 6;                                   //proximity LED
const int LINEPIN_1 = 7;                         //line sensor
const int LINEPIN_2 = 8;                         //line sensor
const int SERPIN_1 = 9;                          //server pin
const int SERPIN_2 = 10;                         //server pin

//Define global variables
long duration;                                   //duration of pulse
double distance = 0;                                     //distance to object
double prev_dist;                                //distance in previous iteration
double D_dist = 0;                                       //change in distance from last
measurement
double obsped;                                   //speed of object
const double obsped_thresh = -20;                //speed threshold of object
const int dist_thresh = 15;                      //distance threshold
int S_r;                                         //line sensor 1
int S_l;                                         //line sensor 2
bool FLAG_slow_down;
bool FLAG_stop;
int count_l;                                             //counter - left turn
int count_r;                                             //counter - right turn
int count_thresh = 10;                           // threshold for number of turn counts
int pos = 0;                                             //servo position

//create servo objects
Servo servo_1;
Servo servo_2;

//create motor sheild object
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *motor_1 = AFMS.getMotor(2);
Adafruit_DCMotor *motor_2 = AFMS.getMotor(1);

void setup() {
 // Define PIN inputs
 pinMode(LINEPIN_1, INPUT);
 pinMode(LINEPIN_2, INPUT);
 pinMode(ECHOPIN, INPUT);

 //Define PIN outputs
 pinMode(TRIGPIN, OUTPUT);
 pinMode(SLEDPIN, OUTPUT);
 pinMode(PLEDPIN, OUTPUT);
```

```
109
110  //attach servos
111  servo_1.attach(SERPIN_1);
112  servo_2.attach(SERPIN_2);
113
114  //Begin Serial monitor
115  Serial.begin(9600);
116  AFMS.begin();
117 }
118
119 void loop() {
120  //servo positioning
121  pos = pos + 1;                                    //increment pos in each iteration
122  servo_1.write(90 * sin(pos));
123  servo_2.write(90 * cos(pos));
124
125  //Ultrasound sensor
126  int motor_speed = 65;
127  int motor_speed_t = 30;
128  if (count_l > count_thresh || count_r > count_thresh) {
129    motor_speed = 100;
130  }
131  digitalWrite(TRIGPIN, LOW);                              //Force initial Trigger state to LOW
132  delayMicroseconds(2);
133  digitalWrite(TRIGPIN, HIGH);                             //generate US wave
134  delayMicroseconds(10);
135  duration = pulseIn(ECHOPIN, HIGH);                        //read response time
136  distance = duration * 0.034 / 2;                  //2 * distance = speed of sound * time
137  D_dist = (distance - prev_dist);                  //find change in distance
138  obsped = D_dist / 0.1;                            //speed of obstacle = change in distance / time
139
140  // if object is within 40cm and approaching quickly - WARNING
141  if (obsped < obsped_thresh && distance < 40) {
142    Serial.println("WARNING OBJECT APROACHING QUICKLY");
143    digitalWrite(SLEDPIN, HIGH);
144    FLAG_slow_down = 1;
145  } else {
146    digitalWrite(SLEDPIN, LOW);                             // Turn Speed LED on
147    FLAG_slow_down = 0;                             //FLAG_slow_down to slow motors
148  }
149
150  // if object in within tol cm - STOP
151  if (distance < dist_thresh) {
152    Serial.println("WARNING obstruction");
153    digitalWrite(PLEDPIN, HIGH);                    // Turn Proximity LED on
154    FLAG_stop = 1;                                  // FLAG_stop to stop motors
155  } else {
156    digitalWrite(PLEDPIN, LOW);
157    FLAG_stop = 0;
158  }
159  prev_dist = distance;
160
161  // Motor Control using line sensors
```

```
162  if (FLAG_slow_down) {                          //reduce speed if FLAG_slow_down== 1
163    motor_speed = motor_speed / 2;
164    Serial.println("go slow");
165  }
166  if (FLAG_stop == 1) {                                  //stop if FLAG_stop ==1
167    motor_1->setSpeed(RELEASE);
168    motor_2->setSpeed(RELEASE);
169    motor_speed = 0;
170    Serial.println("    stop - obs");
171  }
172  S_r = digitalRead(LINEPIN_1);                          //read signal from line sensor 1
173  S_l = digitalRead(LINEPIN_2);                          //read signal from line sensor 2
174  Serial.println(S_l, S_r);                      //print to serial monitor
175
176  if (S_l == HIGH && S_r == HIGH) {              // If both white
177    motor_1->setSpeed(motor_speed);                      //set both speeds to normalspeed
178    motor_2->setSpeed(motor_speed);
179    Serial.println("    FORWARD");
180    motor_1->run(FORWARD);                               //run motors in forward direction
181    motor_2->run(FORWARD);
182    count_l = 0;                                 //reset left and right turn counters
183    count_r = 0;
184  }
185  if (S_l == HIGH && S_r == LOW) {               //if sensor 1 is black
186    motor_1->setSpeed(motor_speed_t + count_l * 2);      //set inside wheel to increase by
     2*count
187    motor_2->setSpeed(motor_speed);                      //set outside wheel to normal speed
188    Serial.println("    left");
189    motor_1->run(BACKWARD);                              //run inside wheel backwards
190    motor_2->run(FORWARD);                               //run outside wheel forward
191    count_l = count_l + 1;          //Increment counter to keep track of number of interactions
192  }
193  if (S_l == LOW && S_r == HIGH) {               //If sensor 2 is black
194    motor_1->setSpeed(motor_speed);                      //set outside wheel to normal speed
195    motor_2->setSpeed(motor_speed_t + count_r * 2);      //set inside wheel to increase by
     2*count
196    Serial.println("    right");
197    motor_1->run(FORWARD);                               //run outside wheel forward
198    motor_2->run(BACKWARD);                              //run inside wheel Backward
199    count_r = count_r + 1;          //Increment counter to keep track of number of interactions
200
201  if (S_l == LOW && S_r == LOW) {                //if both black
202    motor_1->setSpeed(30);                               //set both motor speeds to as low as
     possible
203    motor_2->setSpeed(30);
204    Serial.println("    STOP - BOTH BLACK");
205    motor_1->run(BACKWARD);                              /*set both motors to creep back to  try
     and get back on track*/
206    motor_2->run(BACKWARD);
207    Serial.println("   ");
208  }
209  Serial.print("       COUNT L R: ");                  //print both counts to help identify a good…
210  Serial.print(count_l);                               //…threshold value
```

```
211   Serial.print("       ");
212   Serial.println(count_r);
213
214 }
```

References

[1] Statistical Press Notice, Quarter 2, 2017-28, NHS England, 2017

[2] https://robosavvy.com/store/dagu-magician-chassis.html, retrieved Jan 2018

[Image 1] https://www.jameco.com/Jameco/Products/MakeImag/2210001.jpg

[Image 2] http://www.dagurobot.com/images/201502/goods_img/53_P_1425026826960.png

[image 3] https://images-na.ssl-images-amazon.com/images/I/81A621O1eoL._SX466_.jpg

[image 4] https://d2t1xqejof9utc.cloudfront.net/screenshots/pics/f31ea15a267eda5dc618412653ebc78b/large.jpg

[Image 5] http://hades.mech.northwestern.edu/images/2/24/Motor_coils_corrected.jpg

[Image 6] http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/

[Image 7] http://www.upgradeindustries.com/media/ebay/Photos/SR04/explanation_of_pulses.jpg

[Image 8] http://www.extremetacticaldynamics.com/knowledge-base/wp-content/uploads/2016/03/LED-diagram2.png

[Image 9] https://alexnld.com/wp-content/uploads/2015/03/SKU119276-2.jpg

[Image 11] http://www.instructables.com/file/FHVX4AOIQS4O1FD/

[image 13] https://cdn.sparkfun.com/assets/9/1/e/4/8/515b4656ce395f8a38000000.png

[Image 14] https://www.wiltronics.com.au/wp-content/uploads/pages/servo-motorstepper-shield-by-adafruit.jpg

[Image 15] http://hades.mech.northwestern.edu/images/c/cd/Torque_graphs.png