Heriot-Watt University Dubai

Praxis Programming B37VB, Year 1

Motor Characterisation Report

Student's name	Aerold Parcon (H00448855)
Student's name	Anas Gomaa (H00481128)
Group number (Buggy number)	B8
Professor's name	Dr Girish Balasubramanian

Table of contents

Introduction

Motors drive the world to move - vehicles, various machines and mechanisms, robotics and various electrical engineering applications. However, you can’t just simply connect motors to anything without extensive characterisation - slight variations in motor characteristics, differences in power consumption (Kuphaldt and Haughery, 2020), or misalignment of motors with wheels, in addition to external conditions such as load variations, friction, or humidity, can greatly affect performance (Mabuchi Motor, n.d.). If motors aren't properly tested and characterized, malfunctions, instability, and potential hazards can occur. These hazards include road safety risks and dangerous accidents. Realistically, components don't perform as their datasheets indicate. Characterising all motors under all actual conditions is important to guarantee dependable performance.

The experiment's goal is to run multiple experiments to figure out all the buggy's motor characteristics - its deviation, its various speeds under different settings and conditions, as well as the correct values to use to ensure the correct operation of the buggy.

Aim

The aim is to characterise DC motors in an Arduino Uno buggy and to identify the right PWM values in order for the buggy to run as straight as possible. The buggy's speed should be measured at low, medium, and high PWM settings.
One must analyze trajectory deviations extensively at all speeds to carefully evaluate every difference between the motors.
To fully achieve the straightest movement possible, calculate corrected PWM values that minimize deviations.

Objective

1- Characterize the differences in speed between the two motors .
2- Correcting the PWM values
3- Calculate the speed of the buggy

Methodology

Three different experiments were used to find the buggy's deviation when using the same PWM values, to calibrate the DC motors to make it run as straight as possible, and to calculate the different speeds depending on the speed levels - low, medium and high.

Buggy set up

Prior to characterising the motors in the buggy, the buggy was set up to run the motors with set PWM values for each DC motor. The buggy was already pre-assembled with an Arduino Uno and an H-Bridge driving the motors, so the only thing left to do was to connect the motors and the Arduino Uno together.

The following connections (as shown on figures 2.1 and 2.2) were made.

Figure 2.1: Wiring diagram. Arduino PWM pins 5, 6, 10 and 11 connected to motor pins.

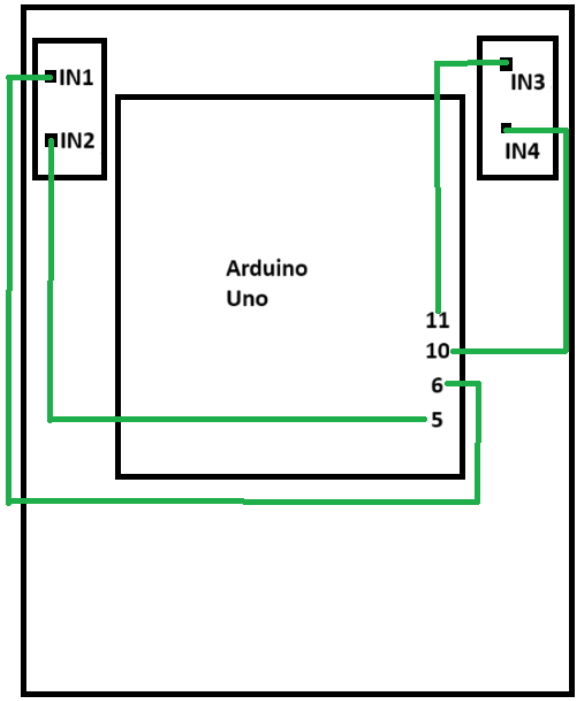
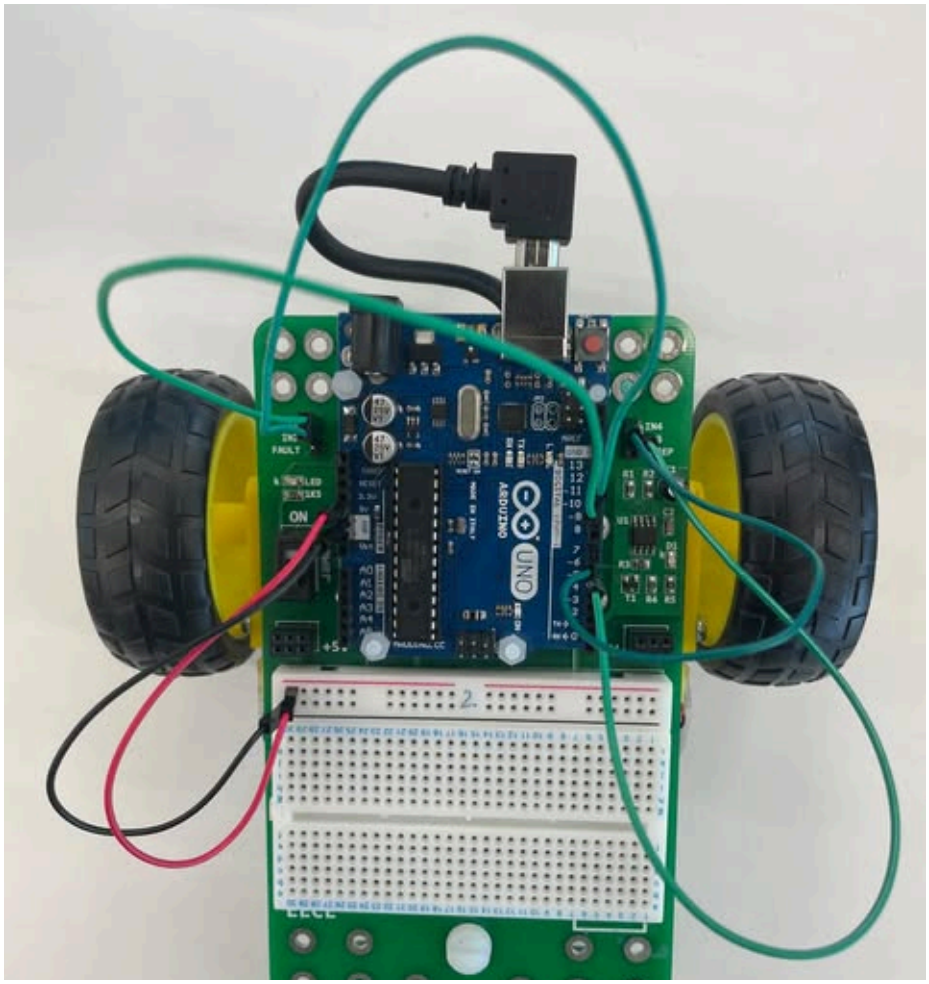


Figure 2.2: The buggy fully wired.



Uploading code

The code for motor control (available on Appendix A) was uploaded on the Arduino. This is taken from *Code for driving motors* available on Canvas (Zeglin, 2016). This code is modified so that the buggy can run at three different speeds - LOW, MEDIUM and HIGH. The variables leftServoSpeed and rightServoSpeed are changed across the experiments for different speeds and setting the corrected PWM values. The code block below shows the section of code where the speeds are changed.

```
/* variables for the speed of motors */
int leftServoSpeed = 100; // These values are changed
int rightServoSpeed = 100; // Changed later on!!!
```

The default non tweaked speed settings used in the following experiments are as followed:
LOW - 100 PWM - Any value lower than 80 isn't enough to power the motors.

MEDIUM - 150 PWM
HIGH - 200 PWM

The corrected calibrated values for the left and right DC motors will be found and characterised in this experiment.

Now that the buggy was set up, the motor characterisation experiments can be done.

Experimental setup

Components needed:

- Fully set-up buggy
- Tape to mark the floor
- Phone camera
- Measuring tape

The following setup as shown on figures 2.3 was made on the floor of the Robotics Lab.

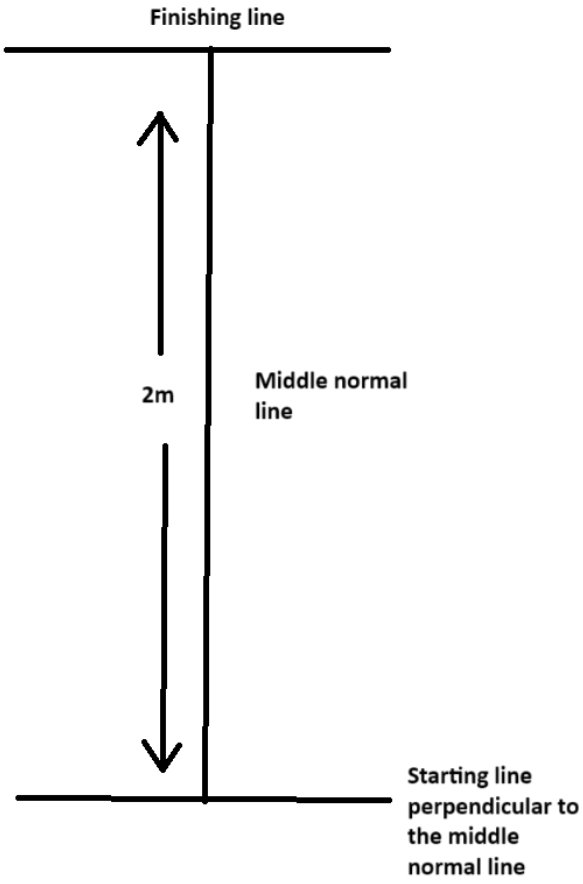


Figure 2.3: Diagram of the experimental setup used for this experiment. The lines were made using electrical tape.

It was made with the following steps:

1) Two points 2 meters apart were marked on the floor using the electrical tape. These points were connected with the tape, creating a line. This will be our reference line to determine the deviation.
2) Two lines perpendicular to the middle line were made. These are the starting and finishing lines.

Now that the lines were marked on the floor, it can be used to characterise the motors.

Procedure

Identifying the deviation when the PWM values are set the same on the motors for three speed settings

- The buggy was precisely aligned on the starting line using the perpendicular start line.
- The Arduino was programmed to set the motors with equal PWM values.
- The buggy was left to run for 5 seconds before stopping it to measure the deviation from the middle line.
- A camera was used to record the exact time the buggy will start and stop to ensure the constant 5 seconds. The stopping point of the buggy was also found using the recorded camera footage.This step was done to reduce human error.
- This was repeated for many runs to calculate the average deviation for each speed setting.

Determining the corrected PWM values for the three speed settings

As previously observed in the previous experiment, the buggy deviates greatly from the reference line, therefore the default speed PWM values need to be corrected in each wheel in order for the buggy to run as straight as possible.

1) The motors were set to the default PWM values in the Arduino Uno.
2) According to the previous results in the first experiment, the PWM values of the motors were tweaked. For example, if the buggy deviated to the left, `rightServoSpeed` would be increased and `leftServoSpeed` decreased in the code, and this would be uploaded again on the Arduino.
3) The buggy was aligned over the starting point, and powered on in the air to accelerate before placing it down. This is so that the slight difference and delay in the motor's

powering wouldn't initially deviate the buggy. It was then set down to drive to the finish line. When it didn't reach the finish line, the values were further tweaked. When the finish line was reached, the buggy was stopped, and the deviation from the middle line was recorded using measuring tape.

4) Depending on the deviation from the previous run, the PWM would be tweaked until it runs as straight as possible.
5) This is repeated for the different PWM speed settings LOW, MEDIUM and HIGH until all of them run as straight as possible.

Determining the speed (in m/s) for each corrected speed settings

Once the corrected values were calibrated onto the DC motors, the motors will be characterised by calculating the speed of the buggy in three different speed settings.

1) Motors were set to the corrected PWMs for a speed setting (low, medium or high) on the Arduino.
2) Phone camera was prepared to record the buggy.
3) Buggy was aligned over the starting point and allowed to accelerate in the air before placing it down.
4) Once placed down, the camera started recording at the same time.
5) Buggy is stopped once it reaches the finishing line, and the ending point is marked. The deviation is measured using measuring tape. The phone camera's recording is stopped.
6) Using the recorded footage, the time taken to reach the end was measured precisely.
7) Speed is calculated using the time taken and the distance travelled

$$distance\ (m)\ =\ \sqrt{length\ of\ track^{\,2} + deviation^{2}}$$
$$speed\ =\ distance/time$$

The distance is calculated with Pythagoras Theorem (figure 2.4)

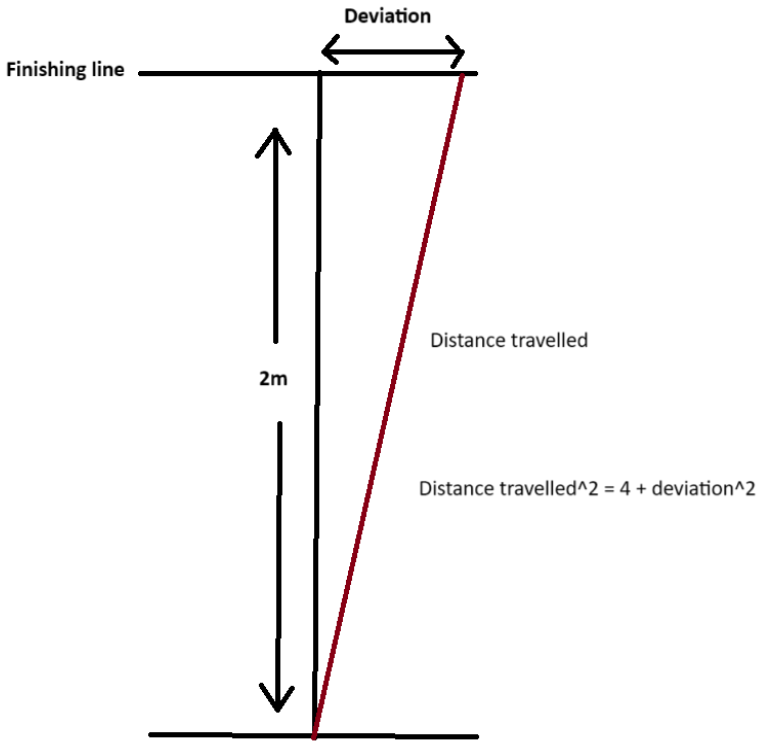


Figure 2.4: Diagram to show how the distance travelled is calculated with Pythagoras Theorem. The distance travelled is the hypotenuse of the triangle formed by the deviation and the distance between the starting and finishing line.

Results

Raw data used to create these tables are in Appendix B. Multiple runs were taken to find the average deviation and speed. The tables below hold the results of the motor characterisation experiments.

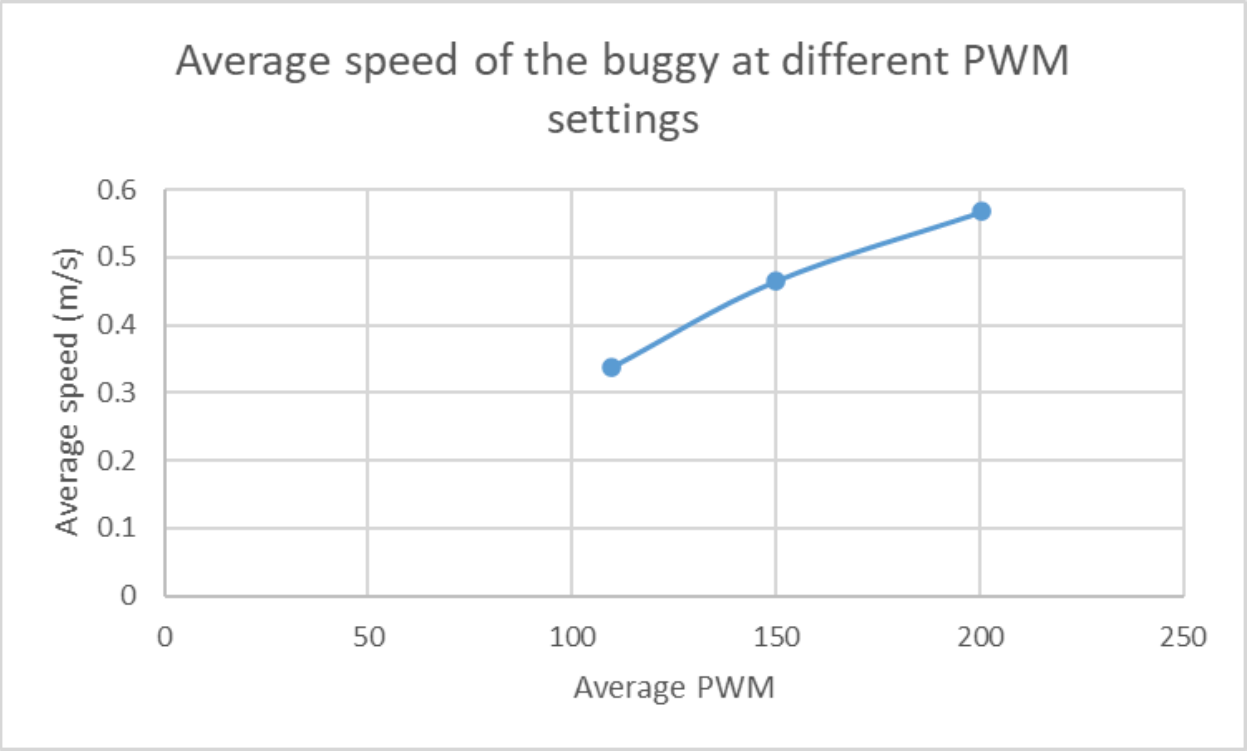
Table 1: Average deviation when using the same PWM values

Speed	**PWM left**	**PWM right**	**Average deviation from the normal (cm)**
Low	100	100	113.50
Medium	150	150	189.75
High	200	200	198.50

Table 2: Corrected PWM values and their speeds

Speed	PWM left	PWM right	Average PWM between the two motors	Average speed (m/s)	Average deviation from the normal (cm)
Low	120	99	109.5	0.337	3.0
Medium	163	137	150	0.465	9.0
High	225	176	200.5	0.567	5.0

Figure 3.1: Average speed plotted against average PWM at each speed setting



Discussion

Deviations and imperfections in the motors

Throughout all the experiments, the results show that there is a deviation - there were no trials where the buggy went perfectly straight, even when finding the corrected values. This is because the motors will have different power distributions due to difference in efficiency and small physical errors in its manufacturing, such as different magnet strengths inside the motor, friction between the gears and mechanical parts, and resistances. In these experiments, the left motor (when viewing it from the buggy facing you) was substantially weaker than the right motor. When the PWM values are the same for each speed setting, the buggy would always deviate to the right, and not even reach the finishing line on the other end of the setup. Upon further inspection in the experiments, the left motor is louder than the right motor, which can mean that there is more friction between the gears of the motor.

First experiment - as speed increases, deviation increases

In the first experiment where the deviations were measured after 5 seconds of running, there can be slight human error in determining the exact stop point of the buggy as we are judging the final position from the video. However it was decided to use the video to get position to keep the time constant as using a stopwatch for time introduces human error by reaction time and time won't be constant.

As the speed is increased in the first experiment when the PWMs are set the same, there is a greater deviation. For example, at low speed setting there is an average of 113.50 cm deviation and at high setting, there is an average of 198.50cm. This could be due to the differences in power of the motors being amplified at higher PWMs.

Second experiment

No matter what values were entered, there were no values where the buggy would run perfectly straight. As mentioned previously this is because of the differences in the motors. However, even when calibrating the motors they wont run straight. For example, in Appendix B table 2, when the PWMs were changed from 163/137 to 164/137, the deviation itself changed from 9 cm to -10cm, which indicates that the correct PWM value is between the 163 and 164 for the left motor. To further improve the corrections of PWM values, float values should be used. However, in an Arduino Uno which is used in this project, PWM values only range from 0-255.

Third experiment - average speed and PWM

PWM and average speed of the buggy have a positive correlation - when PWM is increased, the speed is increased, just as shown on figure 3.1. This is because as there's more power supplied to the motor, it will move faster. Using the graph on figure 3.1, the average PWM can be extrapolated if you want to have a specific average speed. For example, if you wanted the buggy

to run at 0.5m/s, the approximate average PWM value will be 170, and from the 170 value, you further calibrate the values until it goes straight.

Other factors affecting deviation and human errors

There can be more reasons why there were deviations such as the unevenness and friction of the ground which could cause the buggy to tilt towards one direction. To keep this variable constant, we used the same setup on the same floor surface as the final project of this course.

The uneven weight distribution on the buggy due to the different components and wiring can cause deviations too. However the weight distribution must be negligible as the components are small. In future iterations of this buggy with extra modifications and body work, a counterweight can be implemented to ensure the equal weight distribution.

Another source of deviation can be the initial starting position and alignment. When placing the buggy at the starting point there can be a slight error where it’s not heading perfectly straight. Therefore, human error has affected the results gathered from this experiment. To try reducing human error, the starting point was marked with tape, and we ensured that the starting and finishing lines were perfectly perpendicular to the middle. In future experiments, a parallel set of guidelines can be used to ensure the perfect starting alignment of the buggy.

Additionally, the motors power on at a very slightly different time (the left motor being weaker powers later), and accelerate at a different rate. To avoid this issue, before setting the buggy down, it was allowed to accelerate in the air. However this can increase the human error due to the shaking and rattling of the buggy while it's in the air causing the starting position to be off, and the bouncing off the ground. But the initial deviation caused by the uneven starting times of the motors affected the results more, so this method was used.

To try reducing the error and accuracy of the results gathered, multiple runs were taken and averaged.

Conclusion

The aim of this lab report was to investigate and characterise the DC motors in the Arduino buggy. The experiments have successfully identified that the buggy won't run in a completely straight line when the motors are powered with the same PWM values. This is because of manufacturing inconsistencies giving differences in power distribution, efficiency, friction and other various factors. Further experimentation shows that tweaking the PWM values for the different motors can be used to minimize the deviations. The corrected values for three different speed settings were identified and the average speed of the buggy for each setting was determined, characterising the speed of the buggy.

However, the buggy was not able to follow a completely straight line even with the calibrated values due to various variables such as human error, an uneven working surface, weight distribution, and other external factors which cannot be controlled.

To enhance this experiment, the motors can be further characterised using an encoder or other various sensors to identify the rotations per minute depending on the different PWM settings, and acceleration of the buggy at certain points. Testing the buggy under different factors by running it on different surfaces can show how the motors react under various levels of friction. The effects of temperature and heat built up by friction inside the motors can be tested to monitor the motor's performance.

References

Kuphaldt, T.R. and Haughery, J.R. (2020) '5 MOTOR CHARACTERISTICS', in *Applied Industrial Electricity*. Iowa State University, p. 199.

Mabuchi Motor (n.d.) 'Factors Affecting Motor Performance', *Mabuchi Motor* [Online website] Available at: https://www.mabuchi-motor.com/product/knowledge/performance/factors.html. (Accessed on 28th February).

Zeglin, G. (2016) *sketch_motor_drive_new_robot.txt* [Source code]. Heriot-Watt University. Praxis Programming, B37VB, Canvas. Available at: https://canvas.hw.ac.uk/courses/27292/pages/code-for-driving-motors-2?module_item_id=2131948. (Accessed on 28th February).

Appendices

Appendix A - Code used for running the motors in different PWMs

Code is taken from Canvas from the B37VB Praxis Programming Course.

```
/*
Light following control of two Bi-directional Motors
*/
// WheelDrive - move a pair of DC motors at varying rate and direction
//
// Copyright (c) 2016, Garth Zeglin.  All rights reserved. Licensed under
the
// terms of the BSD 3-clause license as included in LICENSE.
//
// This program assumes that:
//
//  1. A DRV8833 dual DC motor driver module is connected to pins 5, 6, 9,
and 10.
//  2. A pair of motors is attached to the driver.
//  3. The serial console on the Arduino IDE is set to 9600 baud
communications speed.

//
================================================================================
======
// Define constant values and global variables.

// Define the pin numbers on which the outputs are generated.
#define MOT_A1_PIN 5
#define MOT_A2_PIN 6
#define MOT_B1_PIN 9
#define MOT_B2_PIN 10

#define LOW_SPEED 100
#define MEDIUM_SPEED 150
#define HIGH_SPEED 200

/* variables for the speed of motors */
```

```
int leftServoSpeed = 100; // These values will be changed in the
experiments
int rightServoSpeed = 100;

//
================================================================================
======
/// Configure the hardware once after booting up.  This runs once after
pressing
//// reset or powering up the board.
void setup(void)
{
  // Initialize the stepper driver control pins to output drive mode.
  pinMode(MOT_A1_PIN, OUTPUT);
  pinMode(MOT_A2_PIN, OUTPUT);
  pinMode(MOT_B1_PIN, OUTPUT);
  pinMode(MOT_B2_PIN, OUTPUT);

  // Start with drivers off, motors coasting.
  digitalWrite(MOT_A1_PIN, LOW);
  digitalWrite(MOT_A2_PIN, LOW);
  digitalWrite(MOT_B1_PIN, LOW);
  digitalWrite(MOT_B2_PIN, LOW);

  // Initialize the serial UART at 9600 bits per second.
  Serial.begin(9600);
}
//
================================================================================
======
/// Set the current on a motor channel using PWM and directional logic.
/// Changing the current will affect the motor speed, but please note this
is
/// not a calibrated speed control.  This function will configure the pin
output
/// state and return.
///
/// \param pwm    PWM duty cycle ranging from -255 full reverse to 255
full forward
/// \param IN1_PIN  pin number xIN1 for the given channel
```

```
/// \param IN2_PIN  pin number xIN2 for the given channel

void set_motor_pwm(int pwm, int IN1_PIN, int IN2_PIN)
{
  if (pwm < 0) {  // reverse speeds
    analogWrite(IN1_PIN, -pwm);
    digitalWrite(IN2_PIN, LOW);

  } else { // stop or forward
    digitalWrite(IN1_PIN, LOW);
    analogWrite(IN2_PIN, pwm);
  }
}
//
================================================================================
======
/// Set the current on both motors.
///
/// \param pwm_A  motor A PWM, -255 to 255
/// \param pwm_B  motor B PWM, -255 to 255

void set_motor_currents(int pwm_A, int pwm_B)
{
  set_motor_pwm(pwm_A, MOT_A1_PIN, MOT_A2_PIN);
  set_motor_pwm(pwm_B, MOT_B1_PIN, MOT_B2_PIN);

  // Print a status message to the console.
  Serial.print("Set motor A PWM = ");
  Serial.print(pwm_A);
  Serial.print(" motor B PWM = ");
  Serial.println(pwm_B);
}

//
================================================================================
======
/// Simple primitive for the motion sequence to set a speed and wait for
an interval.
///
/// \param pwm_A  motor A PWM, -255 to 255
```

```
/// \param pwm_B  motor B PWM, -255 to 255
/// \param duration delay in milliseconds
void spin_and_wait(int pwm_A, int pwm_B, int duration)
{
  set_motor_currents(pwm_A, pwm_B);
  delay(duration);
}

void loop()
{

  spin_and_wait(leftServoSpeed,rightServoSpeed,500); // sets speed of
motors to value entered above for 0.5 sec and keeps repeating
}
```

Appendix B - Raw data

Table 1: Raw data - Deviations for the same PWM values

Trial	PWM	Time (s)	Distance from normal (cm)	Deviation direction		
1	100	5	110	Right		
2	100	5	117	Right		
					Average deviation	113.5
1	150	5	190	Right		
2	150	5	189.5	Right		
					Average deviation	189.75
1	200	5	186	Right		
2	200	5	211	Right		
					Average deviation	198.5

Table 2: Correcting the PWM values

trial no	L	R	deviation notes	Deviation (cm)
	SLOW SPEED			
1	100	100	high deviation, doesn't reach, to the right	
2	120	90	deviates to the left	84
3	120	95	closer to the middle, left	13
4	120	97	closer to the middle, left	8
5	120	99	closest to the middle	4
6	120	99	best one (confirmation)	3
	MEDIUM SPEED			
1	150	150	deviates to the right, doesn't reach the end	
2	160	140	deviates to the right	35
3	163	137	cant get better than this	9
4	164	137		-10
	HIGH SPEED			
1	200	200	doesn't reach, high to the right	
2	220	180	to the right, better	80
3	230	170	to the left	-17.33
4	225	175	to the left, better than before	-7.5
5	225	174	left	-8
6	226	174	left	-5.5

7	225	176	to the right	5

Table 3: Speeds (m/s) for each speed setting

speed	distance between start and finish (m)	deviation (cm)	total distance travelled	time (s)	speed (m/s)		
slow	2	3	2.000224987	5.97	0.3350461		
	2	3	2.000224987	5.91	0.3384475	average	0.336747
medium	2	9	2.002023976	4.4	0.4550054		
	2	9	2.002023976	4.2	0.4766724	average	0.465839
fast	2	5	2.000624902	3.56	0.5619733		
	2	5	2.000624902	3.5	0.5716071	average	0.56679