Chapter 2
Robot Kinematics

A brief Introduction

Kinematics is the study of movement (well almost) so in this chapter we are looking at the principles of robot movement, for our 2-wheeled differential drive robot. The explanation is mathematical; this is necessary for us to write code to get the robot moving. The most important expressions are highlighted. So how does a wheeled robot move? Simply by driving its wheels. If you drive both wheels at the same angular speed, then the robot moves forward. If you drive the left wheel faster, then the robot will arc to the right. Driving both wheels at the same speed, but in opposite directions, will make the robot spin about its axis.

Linear and Angular Velocities

Let's see how wheels work. Fig.1 shows a wheel completing a full revolution. Imagine the type is coated with paint, so as the wheel rotates it paints a nice line on the surface. How long is this line? Simply the circumference of the wheel.

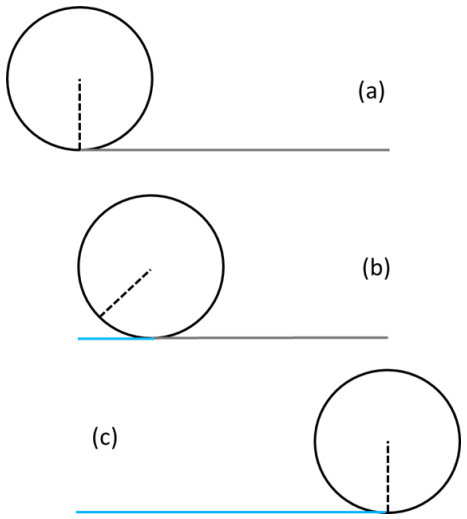


Figure 1. Rotating wheel leaving a track of blue paint on the surface

Now let's do a quick calculation. I guess you will remember the expression for the circumference of a circle radius r. This is of course $2\pi r$. So if we have a wheel of radius 33mm then one rotation will shift the robot a distance (2)(3.1415)(33) = 207mm.

And now for some maths. If one rotation gives a distance of $2\pi r$ then half a rotation will give half of this distance, but what about if the angle of rotation is $\Delta\theta$? (Here the symbol Δ means a change and θ is the angle. Well, the arc of the circle corresponding to the angle $\Delta\theta$ is just

$$\Delta s = r\Delta\theta \qquad (1)$$

where the angle is in radians (more on that later). So, this is the length of the line painted by the robot, and is the distance moved forward, look at Fig.2.

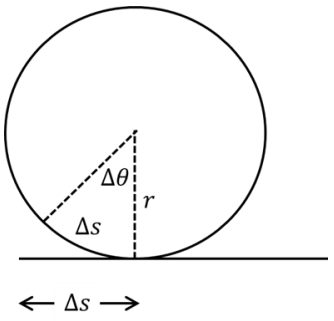


Figure 2. Wheel rotating with a change in angle

Now let's think about the robot's speed in mm/sec. Speed is the distance moved Δs in a time interval Δt. Here's the expression for speed.

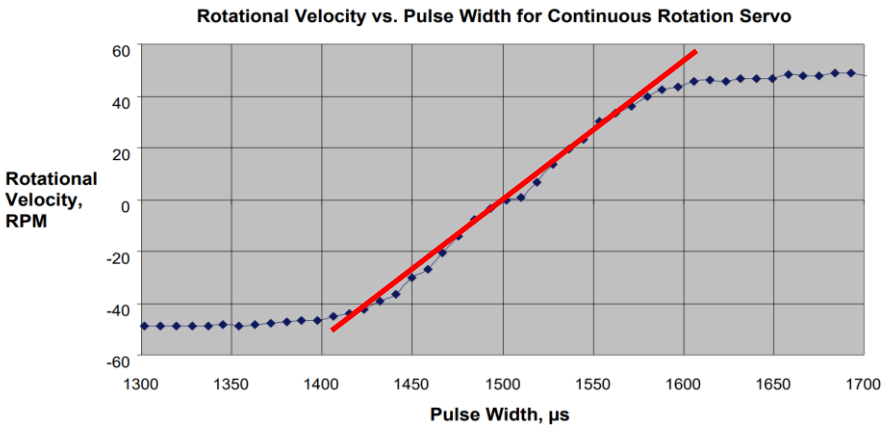
$$v = \frac{\Delta s}{\Delta t} \qquad (2)$$

Continuing with our example, if the robot wheels make one revolution in 2 seconds, then the robot speed is

$$v = \frac{2\pi r}{2} = \frac{207}{2} = 103.5\ mm/sec$$

Try to imagine what that means. The symbol for speed is v since what we are really talking about is *velocity*, speed in a particular direction (forward or backward).

But robots move by turning their wheel with their servomotors and our computer programs must provide *drive* to the motors to make them rotate. To make a servomotor rotate, we must give it a series of 'pulses' where the pulse width determines the rotational speed. For the robot we shall be using, the 'Parallax Activity Bot /BoE Bot' the relationship between pulse width and speed is shown in the diagram below. The points show real measurements, and the red line has been placed to capture the most useful part of the experimental curve, where speed is proportional to pulse width. You can see that with a pulse width of 1500 μs (microseconds) the motor does not rotate. Therefore, in our code (and in our thinking) we shall define a quantity **drive** *relative to this 1500*. So, a drive of 50 will create a rotational velocity of around 30 rpm, and a drive of -50 will rotate the motor at this rpm, but in the opposite direction.



It's easy to get a relationship between drive and rpm from the red line above; here we find this is

$$drive = \frac{50.0}{30.0} rpm \qquad (3)$$

In our code, you will see the variables **driveL** and **driveR** which are, of course, the drives sent to the left and right servomotors.

Now we need to understand how the values of driveL and driveR will determine the robot's speed; let's assume these are the same, so the robot will move forwards (we'll look at other possibilities later). If we stick expression (1) for the distance gone when the wheel rotates through an angle, into expression (2) which is the definition for linear robot velocity moving forward, we get

$$v = \frac{\Delta s}{\Delta t} = \frac{r\Delta\theta}{\Delta t} = r\,\frac{\Delta\theta}{\Delta t}$$

Here $\Delta\theta/\Delta t$ tells us how fast the wheel angle changes with time, so this is the *angular velocity* of the wheel; we give this the symbol ω so we have a fundamental expression

$$v = \omega r \qquad (4)$$

This makes sense; if the wheel radius *r* is increased, then the linear speed *v* is increased, and if the angular velocity ω is increased (the wheel rotates faster) then the linear speed is increased. All seems good, and it is. There may be a conceptual stumbling block, however, it's due to the *units* of angular velocity ω. The angle change in (1) is measured in *radians* (in one revolution, there are 360 degrees which is 2π radians, a little over 6). So, we need to connect angular velocity in radians/sec to angular velocity in rpm, so we can use the above drive graph.

Help! What are radians?

Think of a wheel rotating once, through 360 degrees. We know the distance gone is the circumference of the wheel $2\pi r$. Now expression (1) tells us that this distance is $r\Delta\theta$. So, we have

$$2\pi r = r\Delta\theta$$

and cancelling the *r* gives us

$$\Delta\theta = 2\pi$$

therefore, in a circle, there are 2π radians.

Let's say the wheels are rotating at *n* rpm, i.e., n_{rpm}. Therefore, the revs per second is

$$n_{rps} = \frac{n_{rpm}}{60}$$

Each rotation the wheel rotates 2π radians, therefore the radians per second (ω) is just

$$\omega = 2\pi \frac{n_{rpm}}{60} = \frac{2\pi}{60} n_{rpm}$$

i.e.,

$$\omega = \frac{2\pi}{60} n_{rpm} \qquad (5)$$

Let's work through an example how we would use this maths to make a robot move forwards a desired distance, in a desired time.

Let's take the case of driving the robot a desired distance of 80mm in a desired time of 2 seconds, a speed of 40 mm/s.

$v = \frac{\Delta s}{\Delta t}$ (2)	*v = 80/2 = 40 mm/sec*
$\omega = \frac{v}{r}$ from (4)	ω=40/33.0 = 1.21 rad/sec
$n_{rpm} = \frac{60}{2\pi}\omega$ from (5)	n_{rpm}= (60*1.21)/6.28 = 12 rpm
	drive = (50.0/30.0)*12 = 20

We shall revisit this below when we discuss how to code our robot.

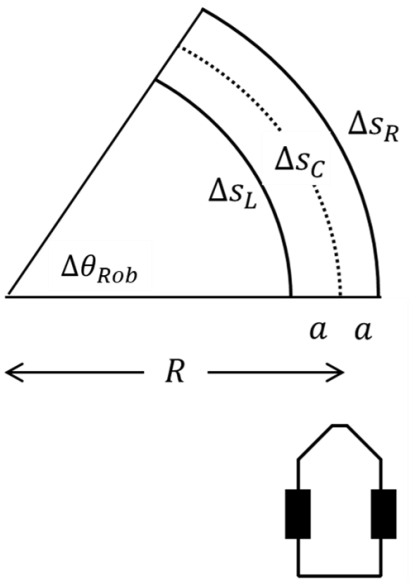


Figure 3. Robot moving on an arc. Distance a is between each wheel and the robot centre.

Movement on an arc

Have a look at Fig.3, at the bottom you will see the robot. The length of its axle, connecting its wheels is $2a$ and for the Parallax robot we are using, this is 104.0 mm, so we have a = 52 mm.

The diagram shows that the centre of the robot (between the wheels) moves along an arc of radius *R*. So the left wheel moves

along an arc of radius $(R - a)$ and the right wheel moves along a larger arc of radius $(R + a)$. Clearly the right wheel is moving faster than the left. The robot's *pose* changes, it started off moving North, and it ends up moving Northwest, having changed its bearing by an angle θ_{Rob}. The subscript *Rob* reminds us we are thinking about the entire robot, rather than its wheels.

To understand the maths which follows, we apply the relationship $\Delta s = r\Delta\theta$ used above. So, for the left wheel we have

$$\Delta s_L = (R - a)\Delta\theta_{Rob} \qquad (6a)$$

and for the right wheel

$$\Delta s_R = (R + a)\Delta\theta_{Rob} \qquad (6b)$$

Now let's imagine that the robot takes a certain time interval Δt to complete its trajectory along the arc. To find the robot wheel speeds, we must divide distance gone by each wheel by this time interval, see expression (2). So, we get for the left wheel

$$v_L = \frac{\Delta s_L}{\Delta t} = (R - a)\frac{\Delta\theta_{Rob}}{\Delta t} \qquad (7a)$$

and for the right wheel

$$v_R = \frac{\Delta s_R}{\Delta t} = (R + a)\frac{\Delta\theta_{Rob}}{\Delta t} \qquad (7b)$$

This is fine, but these expressions are not really telling us much. So, we must go forwards a bit. Remember the expression (1) connecting wheel distance and angle. For the left and right wheels these become, since both wheels have the same radius.

$$\Delta s_L = r\Delta\theta_L, \quad and \quad \Delta s_R = r\Delta\theta_R$$

and putting these into expressions (7a) and (7b) we find

$$r\left(\frac{\Delta\theta_L}{\Delta t}\right) = (R - a)\left(\frac{\Delta\theta_{Rob}}{\Delta t}\right) \qquad (8a)$$

and

$$r\left(\frac{\Delta\theta_R}{\Delta t}\right) = (R + a)\left(\frac{\Delta\theta_{Rob}}{\Delta t}\right) \qquad (8b)$$

I've stuck in some brackets here, where changes in angles are divided by a corresponding change in time. These are angular velocities, speeds of rotation.

The symbol for angular velocity is **omega** ω (lower case) or Ω (upper case). Lower case ω is the angular velocity of the *wheels* and upper case Ω is the angular velocity of the robot body, when viewed from above; this is just the rotation speed of the robot.

So, we can rewrite equations (8) like this

$$r\omega_L = (R - a)\Omega_{Rob}$$

$$r\omega_R = (R + a)\Omega_{Rob}$$

therefore

$$\omega_L = \frac{(R - a)\Omega_{Rob}}{r} \qquad (9a)$$

$$\omega_R = \frac{(R + a)\Omega_{Rob}}{r} \qquad (9b)$$

These are very useful expressions. As with all expressions, think of the stuff on the right of the = sign as an *input* to a computation, what we want to calculate, and the stuff on the left is what we have to code to make this happen. So, if we want the robot to go around an arc of radius R with an angular velocity Ω_{Rob} then we have to make the motors rotate with angular velocities ω_L and ω_R.

Special Cases

There are two special cases of the maths in expressions (9). First, when the robot is moving straight, then we can write $R \to \infty$ so we can neglect a in the expressions, and R divides out. You can see this by calculating the ratio of the wheel omega's

$$\frac{\omega_L}{\omega_R} = \frac{(R - a)}{(R + a)} \qquad (10)$$

where the ratio becomes $R/R = 1$ which tells us that the omegas are the same, i.e., both wheels rotate at the same speed.

The other special case is when $R = 0$, this is where the robot rotates about its centre. Putting this into expressions (10) we find

$$\frac{\omega_L}{\omega_R} = -1 \qquad (10)$$

so, the omegas are equal and opposite!

A Worked Example

Let's say we want our robot to travel along an arc of radius 300mm and change its pose by 45 degrees, and it does this in 2 seconds, see Fig.4.

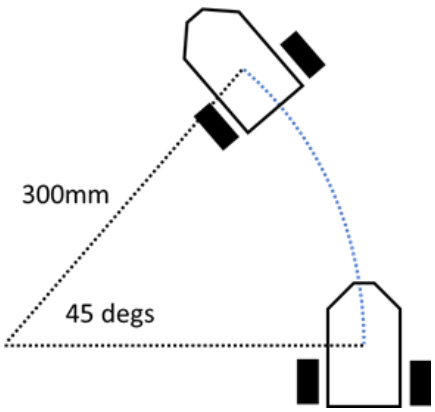


Figure 4 Worked example. The robot rotates 45 degs on an arc 300mm and takes 2 seconds.

The angle expressed in radians is $45\pi/180$ = 0.785rad. The angular velocity of the robot Ω_{Rob} is 0.785/2.0 = 0.393 rad/sec.

Since a = 52mm for the Parallax robot and our arc has a radius of 300 mm, plugging these into expressions (8) gives us

$$\omega_L = \frac{(300-52)0.393}{33}, \quad \omega_R = \frac{(300+52)0.393}{33}$$

and so we calculate

$$\omega_L = 2.85\ rad/s, \qquad \omega_R = 4.19\ rad/s$$

Now we need to convert these *omegas* to *rpms*. Since an omega of 2π rad/s corresponds to 1 rev/sec, then 1 rad/s corresponds to $1/2\pi$ rev/sec. Then ω rad/s corresponds to $\omega/2\pi$ rev/sec, and therefore to $60\omega/2\pi$ rpm. To get revs per second we divide the omegas by 2π which gives us

left 0.45 revs/sec right 0.67 revs/sec

and to get the revs/min we multiply by 60

left 27 revs/min right 40.2 revs/min

and using the expression (3) for drive, we finally have

left drive 45 right drive 67

which are the drive signals we send to our motors.

All of these calculations are done in our Arduino code. The purpose of this worked example is simply to explain what the code actually does.

How to Code a **Real** Robot

We need to get the robot to move forward a certain distance we specify or rotate an angle we specify or rotate on an arc. In our code, we have to specify **drives** which will make the robot move as we want, then send these drives to the servos. Here's a code snippet which gets the robot moving forwards

```
driveL = 30;
driveR = 30;
driveServos(driveL,driveR);
```

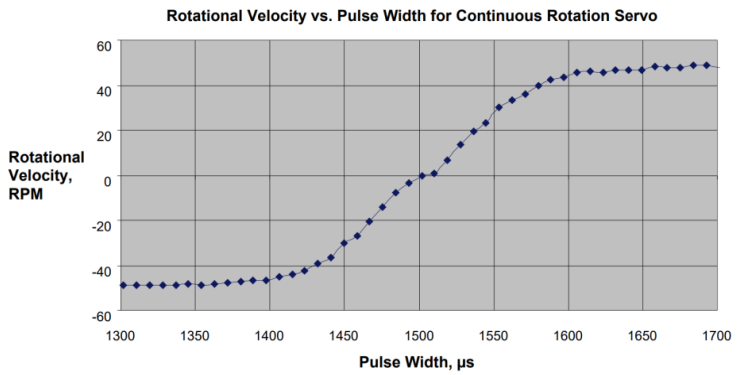
This code will get the robot moving (for ever) but we would rather like to tell the robot *how far to move* and *with what speed*. So, here's some code to get the robot moving forward for a specified distance (in mm) over a specified time. We relate the code to the corresponding maths. All the variables have been declared for you in the code templates.

`desDist = 80;` `desTime = 2.0;`	
`desSpeed = desDist/desTime;`	expression (2)
`omega = desSpeed/wheelRad;`	from expression (4)
`rpm = (60/(2*PI))*omega;`	from expression (5)
`driveL = (50.0/30.0)*rpm;`	expression (3)
`driveR = driveL;`	
`driveServos(driveL,driveR);`	
`delayTime = (int)(desTime*1000)`	
`delay(delayTime);`	
`driveServos(0.0,0.0);`	

The last three lines deserve some comment. The lines before them set the motor drives and therefore their speeds. These last three lines specify how long the servos must spin. So we calculate the **delayTime** in milliseconds from our **desTime** (in seconds), and pass it to the **delay(…)** function which suspends the MPU for this time. Then, we drive the servos with driveL = 0.0 and driveR = 0.0

which will make them stop. So, the final result is that the robot will move 80mm forwards and it will take 2secs to do this, both values we specify.

Now the maths is perfect, and the code is perfect (they are both perfect *abstract* systems of thinking), but when you get the robot to execute this code, it will not move 80mm forwards, it will be more, or it will be less. Why? Because the robot is not *abstract*, it is not a *simulation*, it is really *embodied in the real world*. Think of its motors, their datasheet may specify their 'accuracy' as 10%. This means that if you ask them to rotate with an angular velocity of 10 rpm, they will rotate with anywhere between 9 and 11 rpm. In the worst-case scenario, the left motor could rotate at 9 rpm and the right at 11 rpm; the robot would not go straight forward but arc to the left! Also, we need to know the wheel radius, and so we measure it, but our measurements are subject to errors. And, also the relationship between **drive** and rpm may not be the one we presented earlier, each motor is different. Looking at the drive-rpm curve again (reproduced below) we see a real issue.



If we look at pulse Width 1500 (drive = 0) we see that increasing the drive to about 20 does not make the motor turn! There is a 'dead band'. This means that we cannot use small forward drives or motor speeds.

So, back to our problem. There are two ways to cope with this problem: The first relies on direct observation of the robot, and measurement of how it moves. Let's say we ask the robot to move 80mm and we measure how far it moves, 90mm; it's gone too far. So we could reduce the wheel velocity **omega**, but we know this is

dangerous, since if omega is made too small, then the motor will not turn.

There is another way to make the robot travel less far; we can reduce the amount of time we drive the motors. So, we change the **desTime**

```
correctionFactor = desDist/actualDist;
desTime = desTime*correctionFactor;
```

It is important that we make this change at the appropriate place in the code, it must not be made before we calculate the omega's since they depend on the desTime. Here's where to put the correction, so that it only affects the **delayTime** which tells the motors how long to rotate.

```
correctionFactor = desDist/actualDist;

desTime = desTime*correctionFactor;
delayTime = (int)(desTime*1000)
delay(delayTime);
driveServos(0.0,0.0);
```