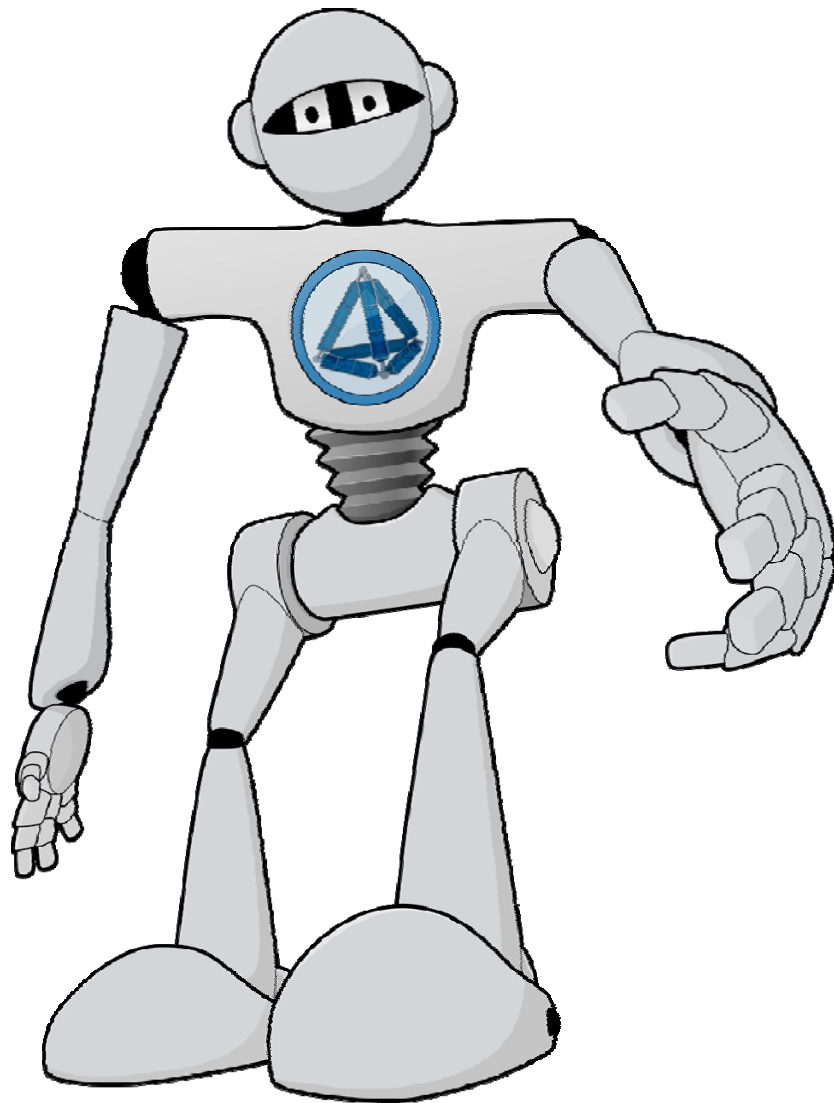


Tribotix

Mathematics required for Legged Robotic Motion





1. Introduction

This article is meant as a brief introduction to the mathematics required for Legged Robotic Motion. In this article I will discuss the three topics needed to understand how you can make a robotic limb track a certain trajectory. The three topics that will be discussed are:

1. Forward Kinematics,
2. Inverse Kinematics, and finally
3. Trajectory Planning.

In this article I will use a 2-joint leg from a Robot Dog as the basis for my discussions. I have chosen this as an example as the 2-joint leg can only move in two dimensions, this means the 2-joint leg can be defined by an xy Cartesian co-ordinate system and as such the mathematics is relatively simple.

If you were to go beyond a 2-joint robotic limb the mathematics becomes significantly harder (involving matrix algebra) and possibly a three dimensional xyz Cartesian co-ordinate system.

I have decided to keep this article as simple and practical as possible, with detailed graphs and complete mathematical solutions where applicable. I hope you find the contents of this article interesting and informative.



2. Forward Kinematics

The Forward Kinematics problem can be simply stated as follows:

If we are given:

1. the angles (A_1 & A_2), and
 2. the length of our robotic limbs (L_1 & L_2),
- what will be the (x , y) position of the robotic limbs 'tip'?

The solution to this problem is based on simple trigonometry. Consider the 2-joint robotic limb shown below in figure 2.1.

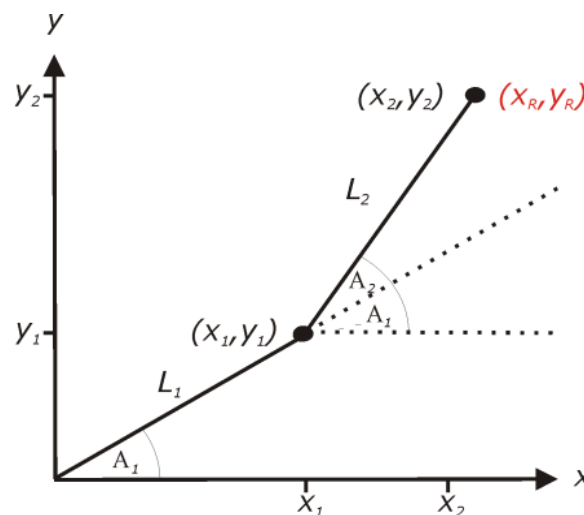


Figure 2.1 – Graphical Representation of a 2-joint Robotic Limb

To calculate the resultant position (x_R , y_R) you simply need to determine the x and y component of each robotic limb and add these together.

For those of us who have forgotten our High School Maths:

$$\sin A = \frac{\textit{opposite}}{\textit{hypotenuse}} \quad \& \quad \cos A = \frac{\textit{adjacent}}{\textit{hypotenuse}}$$

We can then apply this to our problem, the result is shown below in figure 2.2. There is really only one trick here, that is the angle associated with the robotic limb L_2 must be referenced to the x -axis.

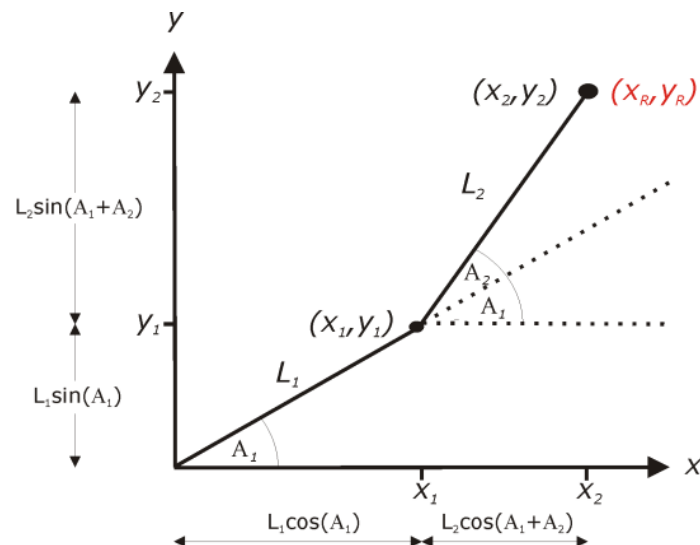


Figure 2.2 – Graphical Representation of a 2-joint Robotic Limb

So, the resultant position (x_R, y_R) for a 2-joint robotic limb can be given by the equations:

$$x_R = L_1 \cos(A_1) + L_2 \cos(A_1 + A_2) \quad (2.1)$$

$$y_R = L_1 \sin(A_1) + L_2 \sin(A_1 + A_2) \quad (2.2)$$



3. Inverse Kinematics

Inverse Kinematics is much more complicated than that of Forward Kinematics – the mathematics and geometry associated with Inverse Kinematics is quite complicated. The Inverse Kinematics problem can be simplified to the following statement:

If we are given:

1. the length of our robotic limbs (L_1 & L_2), and
 2. the (x, y) position of the robotic limbs 'tip',
- what will be the angles (A_1 & A_2) required to achieve this position ?

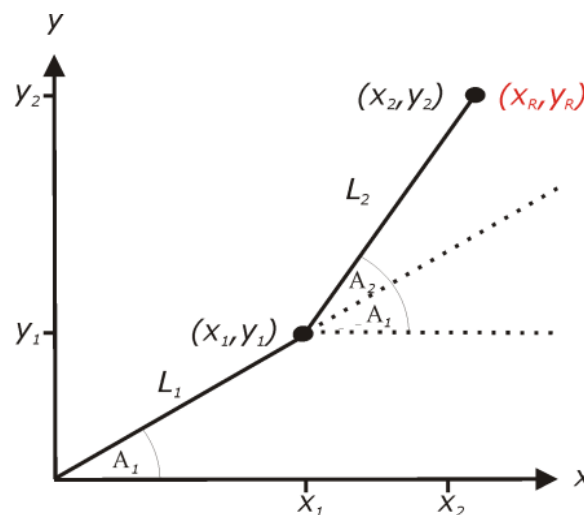


Figure 3.1 – Graphical Representation of a 2-joint Robotic Limb

There are two ways that Inverse Kinematics problems can be solved. The first is algebraically, but this is relatively complicated. The second method is the use of geometry. Both these methods, along with full working solutions, will be presented in the following subsections.

3.1 Algebraic Solution

Previously we have shown that the Forward Kinematic Equations are:

$$x = L_1 \cos(A_1) + L_2 \cos(A_1 + A_2) \quad (2.1)$$

$$y = L_1 \sin(A_1) + L_2 \sin(A_1 + A_2) \quad (2.2)$$

The first step in the algebraic solution is to square both sides of equation (2.1) and (2.2) to create equations (3.1) and (3.2) respectively:



$$\begin{aligned} [x]^2 &= [L_1 \cos(A_1) + L_2 \cos(A_1 + A_2)]^2 \\ x^2 &= L_1^2 \cos^2(A_1) + 2L_1L_2 \cos(A_1)\cos(A_1 + A_2) + L_2^2 \cos^2(A_1 + A_2) \end{aligned} \quad (3.1)$$

$$\begin{aligned} [y]^2 &= [L_1 \sin(A_1) + L_2 \sin(A_1 + A_2)]^2 \\ y^2 &= L_1^2 \sin^2(A_1) + 2L_1L_2 \sin(A_1)\sin(A_1 + A_2) + L_2^2 \sin^2(A_1 + A_2) \end{aligned} \quad (3.2)$$

Now, we must add equations (3.1) & (3.2) together:

$$\begin{aligned} x^2 + y^2 &= L_1^2 [\sin^2(A_1) + \cos^2(A_1)] + L_2^2 [\sin^2(A_1 + A_2) + \cos^2(A_1 + A_2)] \\ &\quad + 2L_1L_2 [\sin(A_1)\sin(A_1 + A_2) + \cos(A_1)\cos(A_1 + A_2)] \end{aligned} \quad (3.3)$$

Now we need to use the following trigonometric identity (the Pythagorean formula for sines and cosines):

$$\sin^2(\alpha) + \cos^2(\alpha) = 1$$

Substituting this into equation (3.3) gives us:

$$x^2 + y^2 = L_1^2 + L_2^2 + 2L_1L_2 [\sin(A_1)\sin(A_1 + A_2) + \cos(A_1)\cos(A_1 + A_2)]$$

Moving L_1 and L_2 to the LHS of the equation gives us:

$$x^2 + y^2 - L_1^2 - L_2^2 = 2L_1L_2 [\sin(A_1)\sin(A_1 + A_2) + \cos(A_1)\cos(A_1 + A_2)]$$

Moving $2L_1L_2$ to the LHS of the equation gives us:

$$\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} = \sin(A_1)\sin(A_1 + A_2) + \cos(A_1)\cos(A_1 + A_2) \quad (3.4)$$

Again we need to use 2 trigonometric identities (Sum formulas for sine and cosine):

$$\sin(\alpha + \beta) = \cos(\alpha)\sin(\beta) + \sin(\alpha)\cos(\beta)$$

$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$$

Substituting these equations into equation (3.4) for $\sin(A_1 + A_2)$ and $\cos(A_1 + A_2)$ gives us:



$$\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} = \sin(A_1)[\cos(A_1)\sin(A_2) + \sin(A_1)\cos(A_2)] \\ + \cos(A_1)[\cos(A_1)\cos(A_2) - \sin(A_1)\sin(A_2)]$$

Expanding the RHS of the equation gives us:

$$\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} = \sin(A_1)\cos(A_1)\sin(A_2) + \sin^2(A_1)\cos(A_2) \\ + \cos^2(A_1)\cos(A_2) - \sin(A_1)\cos(A_1)\sin(A_2)$$

The term $\sin(A_1)\cos(A_1)\sin(A_2)$ can now be eliminated to give us:

$$\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} = \sin^2(A_1)\cos(A_2) + \cos^2(A_1)\cos(A_2)$$

Factoring $\cos(A_2)$ on the RHS of the equation gives us:

$$\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} = \cos(A_2)[\sin^2(A_1) + \cos^2(A_1)]$$

Using the trigonometric identity to eliminate $\sin^2(A_1) + \cos^2(A_1)$ we can now make $\cos(A_2)$ the subject of the equation:

$$\cos(A_2) = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \quad (3.5)$$

We now have a solution for one of the angles that we needed, the only problem now is that there is maybe a multiple solution as:

$$\cos(A_2) = \cos(-A_2)$$

To ensure that we only have one solution for A_2 we must also find the \sin of A_2 .

Again, we use a trigonometric identity (the Pythagorean formula for sines and cosines), but first we need to manipulate the identity so that $\sin(A_2)$ is the subject of the equation:



$$\begin{aligned}\sin^2(A_2) + \cos^2(A_2) &= 1 \\ \sin^2(A_2) &= 1 - \cos^2(A_2) \\ \sin(A_2) &= \pm\sqrt{1 - \cos^2(A_2)}\end{aligned}$$

We can have a unique result for A_2 :

$$A_2 = \sin^{-1} \left[\pm\sqrt{1 - \cos^2(A_2)} \right]$$

Substituting equation (3.5) into this result gives:

$$A_2 = \sin^{-1} \left[\pm\sqrt{1 - \left[\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \right]^2} \right] \quad (3.6)$$

Now, we must find a solution for A_1 . Again we revert to the Forward Kinematic Equations:

$$x = L_1 \cos(A_1) + L_2 \cos(A_1 + A_2) \quad (2.1)$$

$$y = L_1 \sin(A_1) + L_2 \sin(A_1 + A_2) \quad (2.2)$$

If we initially concentrate on equation (2.1), replacing $\cos(A_1 + A_2)$ with it's trigonometric identity we get:

$$\begin{aligned}x &= L_1 \cos(A_1) + L_2 \cos(A_1 + A_2) \\ x &= L_1 \cos(A_1) + L_2 \cos(A_1) \cos(A_2) - L_2 \sin(A_1) \sin(A_2) \\ x &= [L_1 + L_2 \cos(A_2)] \cos(A_1) + [-L_2 \sin(A_2)] \sin(A_1)\end{aligned}$$

If we then do similarly with equation (2.2), replacing $\sin(A_1 + A_2)$ with it's trigonometric identity, we get:

$$\begin{aligned}y &= L_1 \sin(A_1) + L_2 \sin(A_1 + A_2) \\ y &= L_1 \sin(A_1) + L_2 \cos(A_1) \sin(A_2) + L_2 \sin(A_1) \cos(A_2) \\ y &= [L_2 \sin(A_2)] \cos(A_1) + [L_1 + L_2 \cos(A_2)] \sin(A_1)\end{aligned}$$

Now we have 2 simultaneous equations, 2 equations with 2 unknowns ($\sin(A_1)$ & $\cos(A_1)$):

$$\begin{aligned}[L_1 + L_2 \cos(A_2)] \cos(A_1) + [-L_2 \sin(A_2)] \sin(A_1) &= x \\ [L_2 \sin(A_2)] \cos(A_1) + [L_1 + L_2 \cos(A_2)] \sin(A_1) &= y\end{aligned} \quad (3.7)$$



An easy way to solve these simultaneous equations is using Cramer's Rule, i.e. given:

$$\begin{aligned} a \cos \alpha + c \sin \alpha &= e \\ b \cos \alpha + d \sin \alpha &= f \end{aligned}$$

then

$$\cos \alpha = \frac{\begin{vmatrix} e & c \\ f & d \end{vmatrix}}{\begin{vmatrix} a & c \\ b & d \end{vmatrix}} = \frac{ed - fc}{ad - bc} \quad \& \quad \sin \alpha = \frac{\begin{vmatrix} a & e \\ b & f \end{vmatrix}}{\begin{vmatrix} a & c \\ b & d \end{vmatrix}} = \frac{af - be}{ad - bc}$$

Therefore, applying Carson's rule to the equation set (3.7) gives:

$$\cos(A_1) = \frac{[L_1 + L_2 \cos(A_2)]x - [-L_2 \sin(A_2)]y}{[L_1 + L_2 \cos(A_2)]^2 + [-L_2 \sin(A_2)]^2}$$

$$\sin(A_1) = \frac{[-L_2 \sin(A_2)]x + [L_1 + L_2 \cos(A_2)]y}{[L_1 + L_2 \cos(A_2)]^2 + [-L_2 \sin(A_2)]^2}$$

So, we finally have a solution for both A_1 and A_2 :

$$A_2 = \cos^{-1} \left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \right) \quad (3.8)$$

and

$$A_1 = \cos^{-1} \left(\frac{[L_1 + L_2 \cos(A_2)]x - [-L_2 \sin(A_2)]y}{[L_1 + L_2 \cos(A_2)]^2 + [-L_2 \sin(A_2)]^2} \right) \quad (3.9)$$

or

$$A_1 = \sin^{-1} \left(\frac{[-L_2 \sin(A_2)]x + [L_1 + L_2 \cos(A_2)]y}{[L_1 + L_2 \cos(A_2)]^2 + [-L_2 \sin(A_2)]^2} \right) \quad (3.10)$$



3.2 Geometric Solution

The geometric is much simpler than the algebraic solution presented in the previous section. We start by considering the diagram shown in figure 3.2.

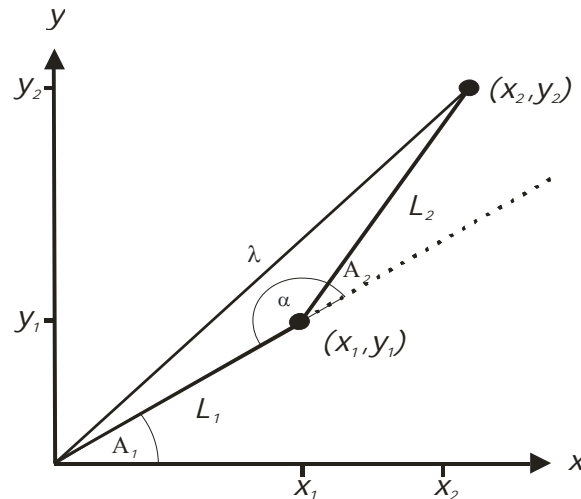


Figure 3.2 – Determining angle A_2 via geometric approach

First, we need to determine the length of the line from the origin $(0, 0)$ to the target position (x, y) , this is easily done by using the *Distance Between 2 Points* formula:

$$\lambda = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (3.11)$$

In this case though, (x_1, y_1) , is the origin $(0, 0)$ so equation (3.11) simplifies to:

$$\lambda = \sqrt{x_2^2 + y_2^2}$$

We will also need to determine the size of the angle α in figure 3.2. If we extend the line from the origin $(0, 0)$ to (x_1, y_1) , we will create the angle A_2 . As this is a straight line,

$$\alpha = (180 - A_2)$$

We can now use the *cosine rule* to determine A_2 .

$$\text{cosine rule: } c^2 = a^2 + b^2 - 2ab \cos(C)$$

In this situation:



$$\begin{aligned}a &= L_1 \\b &= L_2 \\c &= \sqrt{x^2 + y^2} \\C &= (180 - A_2)\end{aligned}$$

Substituting these values into the *cosine rule* gives us:

$$\begin{aligned}\left(\sqrt{x^2 + y^2}\right)^2 &= L_1^2 + L_2^2 - 2L_1L_2 \cos(180 - A_2) \\x^2 + y^2 &= L_1^2 + L_2^2 - 2L_1L_2 \cos(180 - A_2) \\x^2 + y^2 - L_1^2 - L_2^2 &= -2L_1L_2 \cos(180 - A_2) \\\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} &= -\cos(180 - A_2) \\-\cos(180 - A_2) &= \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\end{aligned}$$

Now, $\cos(A_2) = -\cos(180 - A_2)$ - this allows us to write:

$$\cos(A_2) = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \quad (3.12)$$

Therefore, A_2 can be written as:

$$A_2 = \cos^{-1}\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \quad (3.13)$$

Now that A_2 has been calculated, we need to determine A_1 . To do this though, we need to determine the angle β shown in figure 3.3.

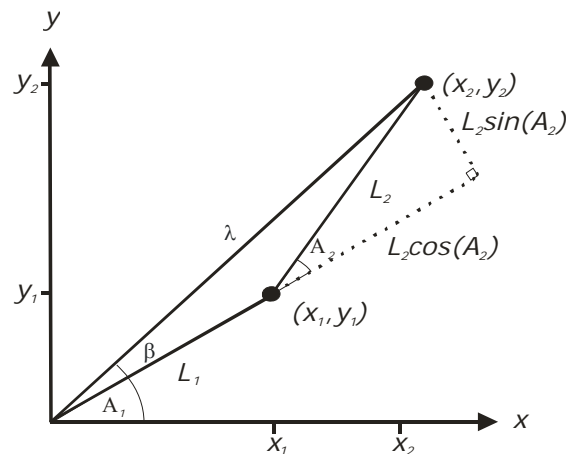


Figure 3.3 – Determining angle A_1 via geometric approach – Step #1

First, let's consider the small right hand triangle with (x_1, y_1) and (x_2, y_2) as 2 of the 3 co-ordinates - shown in figure 3.3 as dotted lines. Here the lengths of the opposite and adjacent sides of this right handed triangle are given by:

$$\begin{aligned} \text{adjacent} &= L_2 \cos(A_2) \\ \text{opposite} &= L_2 \sin(A_2) \end{aligned}$$

Now, let's consider the large right hand triangle with the origin $(0,0)$ and (x_2, y_2) as 2 of the 3 co-ordinates – again shown in figure 3.3 with some of the lines dotted. Here the lengths of the opposite and adjacent sides of this right handed triangle are given by:

$$\begin{aligned} \cos(\beta) &= \frac{L_1 + L_2 \cos(A_2)}{\lambda} \\ &= \frac{L_1 + L_2 \cos(A_2)}{\sqrt{x_2^2 + y_2^2}} \\ \beta &= \cos^{-1} \left(\frac{L_1 + L_2 \cos(A_2)}{\sqrt{x_2^2 + y_2^2}} \right) \end{aligned} \quad (3.14)$$

$$\begin{aligned} \sin(\beta) &= \frac{L_2 \sin(A_2)}{\lambda} \\ &= \frac{L_2 \sin(A_2)}{\sqrt{x_2^2 + y_2^2}} \\ \beta &= \sin^{-1} \left(\frac{L_2 \sin(A_2)}{\sqrt{x_2^2 + y_2^2}} \right) \end{aligned} \quad (3.15)$$

These 2 expressions give us the ability to determine a unique value for β .



Now that we know β we can use the right angled triangle with the angle $(A_1 + \beta)$, as shown in figure 3.3, to determine A_1 .

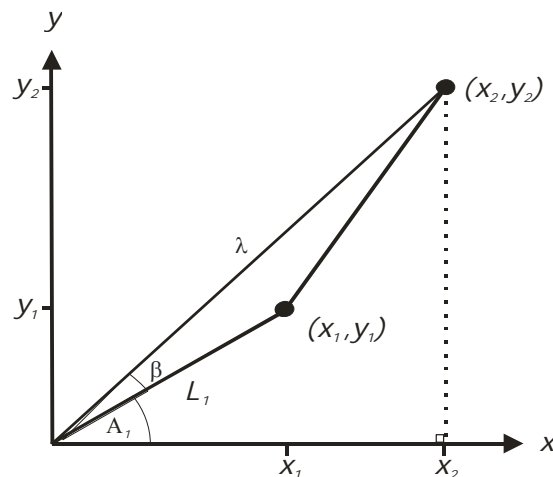


Figure 3.3 – Determining angle A_1 via geometric approach – Step #2

We can determine A_1 two ways, i.e. by using *cos* and *sin* as shown below:

$$\begin{aligned}\cos(A_1 + \beta) &= \frac{x_2}{\sqrt{x_2^2 + y_2^2}} \\ A_1 + \beta &= \cos^{-1}\left(\frac{x_2}{\sqrt{x_2^2 + y_2^2}}\right) \\ A_1 &= \cos^{-1}\left(\frac{x_2}{\sqrt{x_2^2 + y_2^2}}\right) - \beta\end{aligned}\quad (3.16)$$

$$\begin{aligned}\sin(A_1 + \beta) &= \frac{y_2}{\sqrt{x_2^2 + y_2^2}} \\ A_1 + \beta &= \sin^{-1}\left(\frac{y_2}{\sqrt{x_2^2 + y_2^2}}\right) \\ A_1 &= \sin^{-1}\left(\frac{y_2}{\sqrt{x_2^2 + y_2^2}}\right) - \beta\end{aligned}\quad (3.17)$$

These 2 expressions give us the ability to determine a unique value for A_1 .

So, we have now determined A_1 & A_2 . It hasn't been easy but it's a lot easier than the algebraic solution presented in Section 3.1.



4. Trajectory Planning

Now, let's take the formula's we derived in Section 3 and see how these apply to making a 2DOF robot leg track a trajectory. In this discussion I will use an ellipse as the trajectory – this has been shown to be an efficient method of moving a robotic dog's legs to allow it to walk.

4.1 Ellipse

The Parametric Equations for an Ellipse are:

$$\begin{aligned}x &= h + a \cos(t) \\y &= k + b \sin(t) \\0 &\leq t \leq 2\pi\end{aligned}\quad (4.1)$$

There are four parameters here that define the position and size of the ellipse, i.e. a , b , h & k . Figure 4.1 shows the physical significance of these variables.

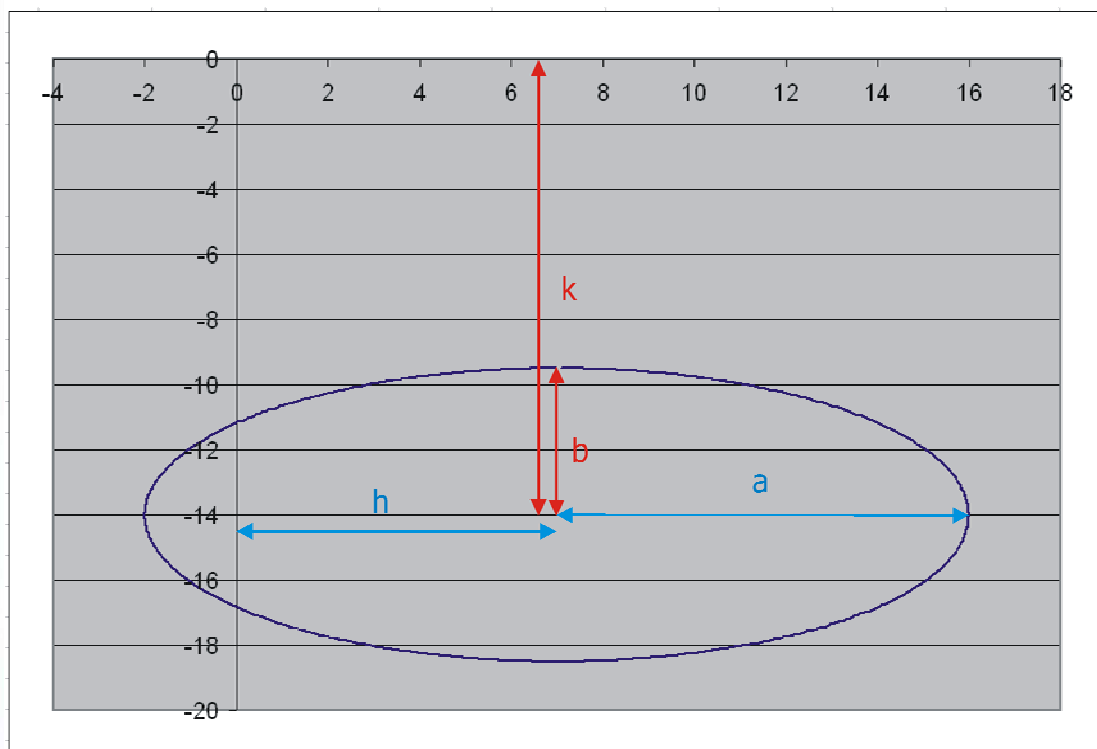


Figure 4.1 Ellipse with Parametric Equation Parameters



In summary:

Position: (h, k) is the centre of the ellipse.

Size: a is the size of the radius for the x -axis whilst b is the size of the radius for the y -axis.

Obviously the selection of these parameters depends on the physical anatomy of the robotic limb. The next section will look at the implementation of an elliptical trajectory for a 2-joint robotic dogs leg.

4.2 Anatomy of example Robot Dog

Figure 4.2 shows the dimensions of the robotic dog's leg that we'll use as an example. Here the 1st limb (L_1) is 9.0cm whilst the 2nd limb (L_2) is 8.5cm.

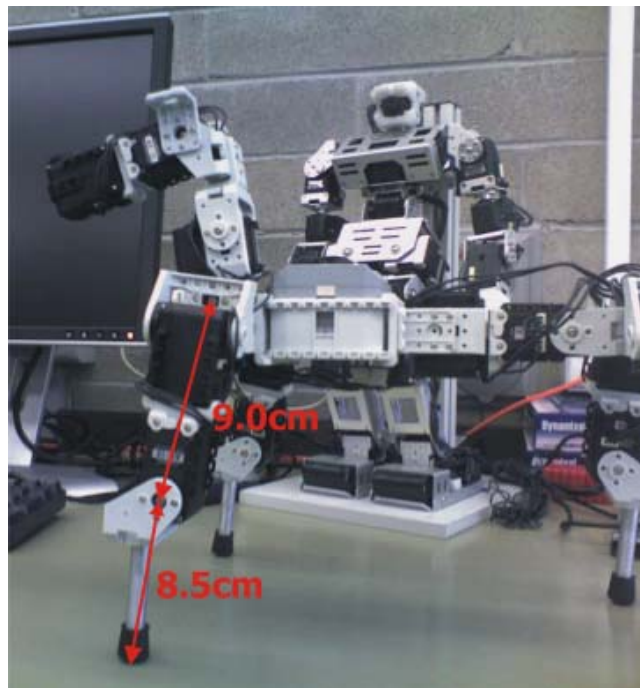


Figure 4.2 Robotic Dog leg dimensions

To determine the parametric equation parameters (a, b, h & k) for a 2-joint robotic limb (L_1 & L_2), I use the following rules of thumb:

1. the centre ellipse is $(0, -(L_1 + (L_2/2)))$
i.e. $x=0$ & $y=-(L_1 + (L_2/2))$
This sets the centre of the ellipse on the $-ve$ y -axis.
2. the parameter b is set to $(L_2/2)$
This means that the maximum length the limbs can be extended to on the y -axis is $-(L_1 + (L_2/2)) - (L_2/2) = -(L_1 + L_2)$ – this is obviously the physical maximum as well.



- the parameter a is set to a maximum of $0.95 * (L_1/2)$
This again restricts the calculations from exceeding that which is physically achievable by the 2-joint configuration.

NB. these *rules of thumb* outlined above set the **maximum sized ellipse** possible with the centre at its maximum distance from the axis of the 1st joint – not necessarily the most efficient.

Figure 4.3 shows the result of implementing these rules of thumb with the following the variables:

$$L_1 = 9.0$$

$$L_2 = 8.5$$

$$a = 8.55$$

$$b = 4.25$$

$$h = 0$$

$$k = -13.25$$

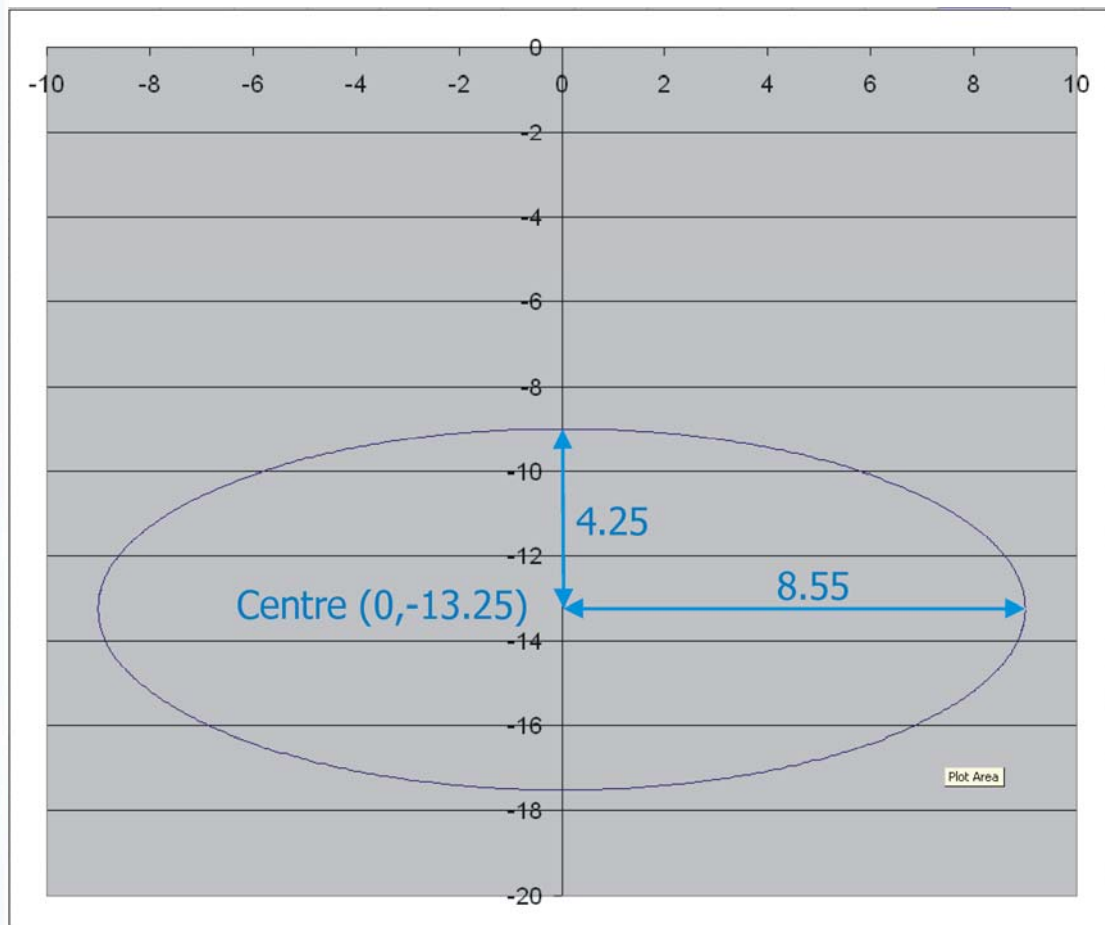


Figure 4.3 Planned trajectory using maximum parameters for a , b , h & k



Now that we have our equations, our robots limb dimensions and a set of parametric equation parameters we can now look at the implementation.

4.3 Implementation

Figure 4.4 shows the planned elliptical trajectory as defined by the parameters we selected in the previous section. As we were able to show in equation 3.5, there can be multiple solutions to an inverse kinematic problem – this is commonly known as ‘*elbow-out*’ and ‘*elbow-in*’.

To graphically highlight this I have:

- Selected a random point on the planned trajectory,
- Drawn a circle with the radius L_2 at this point,
- Drawn a circle with the radius L_1 at the origin, and
- Where these 2 circles intersect are the 2 possible solutions for our inverse kinematic problem.

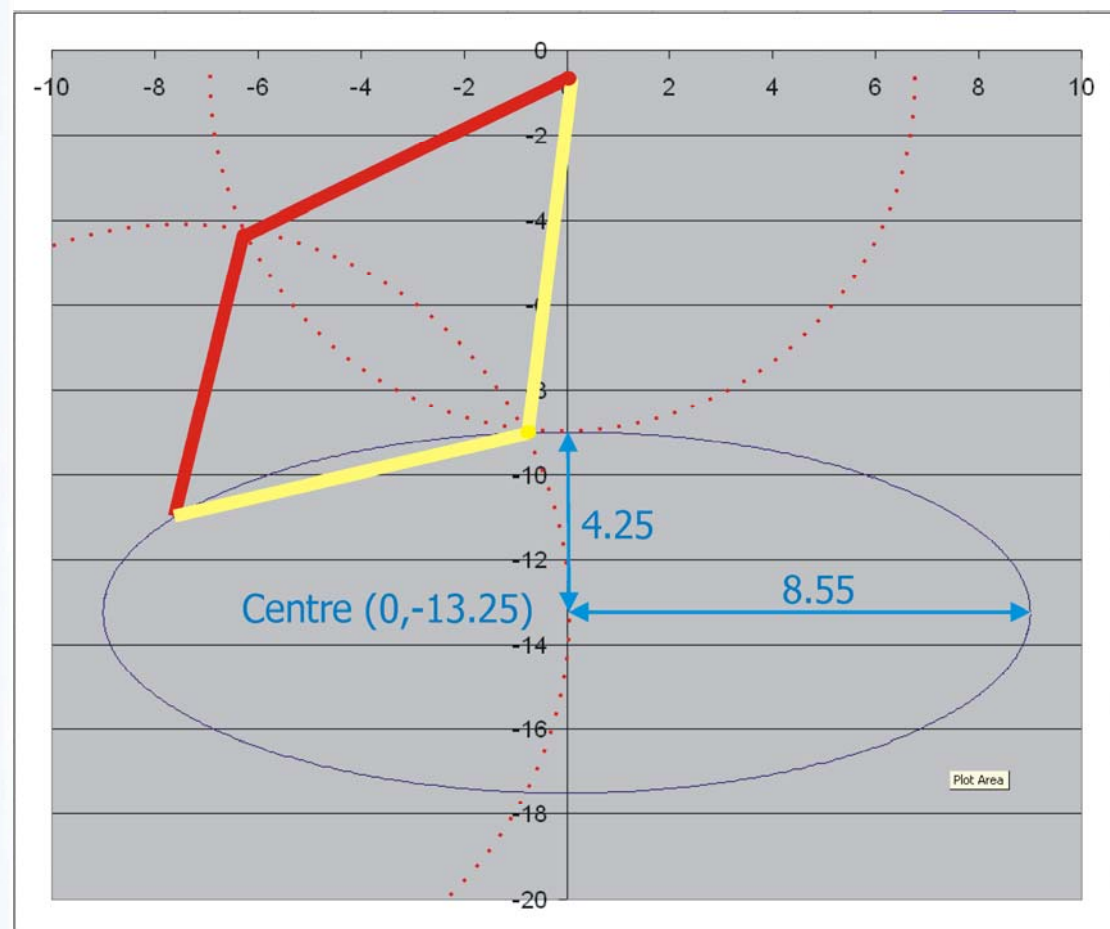


Figure 4.4 *Elbow-In* & *Elbow-Out* solutions to our Inverse Kinematic Problem



You'll notice in this diagram that one solution (the RED) would be better suited to carry a load based at the origin in comparison to the other solution (the YELLOW). So, the final choice of A_1 must take this physical requirement into consideration.

As shown in figure 4.2, I have decided to demonstrate my solution on a Robotic Dog constructed from the *Bioid Robotic kit*. I have also decided to use 'static' control of the robot dogs legs, i.e. the angles will be pre-calculated and made available in a data-table. To generate my data-tables I have created a spreadsheet with solutions for both the Algebraic and Geometric solutions.

The *Bioid Robotic kit* allows such data-tables to be implemented via a *Motion Editor* software package. The *Motion Editor* defines these data-tables in *Motion Pages*. These *Motion Pages* allow for 128 motions to be stored, each motion having a maximum of 7 poses. I decided to use 28 equally spaced time intervals to generate the data for the data-tables, this will give us 4 motion pages which I can 'string' together. The resultant movement will be 'rather chunky', as 28 positions won't give great resolution over such a big elliptical movement, but it will illustrate that this does work.

Table 4.1 shows the resultant data for our data-table. Of main interest in this table are the resultant angles A_1 & A_2 . Figure 4.5 graphically illustrates how the angles vary as a function of time – note the smoothness of the wave like curves.



| | t (ms) | x (cm) | y (cm) | $Dist$ (cm) | A_1 ($^\circ$) | A_2 ($^\circ$) |
|----|----------|----------|----------|-------------|--------------------|--------------------|
| 0 | 0.00 | 8.55 | -13.25 | 15.77 | 277.91 | 51.42 |
| 1 | 0.22 | 8.34 | -12.30 | 14.86 | 273.25 | 63.77 |
| 2 | 0.45 | 7.70 | -11.41 | 13.76 | 267.16 | 76.32 |
| 3 | 0.67 | 6.68 | -10.60 | 12.53 | 259.54 | 88.58 |
| 4 | 0.90 | 5.33 | -9.93 | 11.27 | 250.24 | 99.89 |
| 5 | 1.12 | 3.71 | -9.42 | 10.12 | 239.12 | 109.37 |
| 6 | 1.35 | 1.90 | -9.11 | 9.30 | 226.49 | 115.85 |
| 7 | 1.57 | 0.00 | -9.00 | 9.00 | 213.64 | 118.18 |
| 8 | 1.80 | -1.90 | -9.11 | 9.30 | 202.89 | 115.85 |
| 9 | 2.02 | -3.71 | -9.42 | 10.12 | 196.13 | 109.37 |
| 10 | 2.24 | -5.33 | -9.93 | 11.27 | 193.77 | 99.89 |
| 11 | 2.47 | -6.68 | -10.60 | 12.53 | 195.07 | 88.58 |
| 12 | 2.69 | -7.70 | -11.41 | 13.76 | 199.09 | 76.32 |
| 13 | 2.92 | -8.34 | -12.30 | 14.86 | 205.02 | 63.77 |
| 14 | 3.14 | -8.55 | -13.25 | 15.77 | 212.25 | 51.42 |
| 15 | 3.37 | -8.34 | -14.20 | 16.46 | 220.33 | 39.68 |
| 16 | 3.59 | -7.70 | -15.09 | 16.95 | 228.92 | 28.92 |
| 17 | 3.81 | -6.68 | -15.90 | 17.25 | 237.74 | 19.48 |
| 18 | 4.04 | -5.33 | -16.57 | 17.41 | 246.49 | 11.69 |
| 19 | 4.26 | -3.71 | -17.08 | 17.48 | 254.91 | 5.83 |
| 20 | 4.49 | -1.90 | -17.39 | 17.50 | 262.76 | 2.06 |
| 21 | 4.71 | 0.00 | -17.50 | 17.50 | 270.00 | 0.00 |
| 22 | 4.94 | 1.90 | -17.39 | 17.50 | 275.24 | 2.06 |
| 23 | 5.16 | 3.71 | -17.08 | 17.48 | 279.42 | 5.83 |
| 24 | 5.39 | 5.33 | -16.57 | 17.41 | 282.15 | 11.69 |
| 25 | 5.61 | 6.68 | -15.90 | 17.25 | 283.34 | 19.48 |
| 26 | 5.83 | 7.70 | -15.09 | 16.95 | 283.00 | 28.92 |
| 27 | 6.06 | 8.34 | -14.20 | 16.46 | 281.17 | 39.68 |

Table 4.1 Data-tables for Elliptical movement of Robotic Dogs Legs

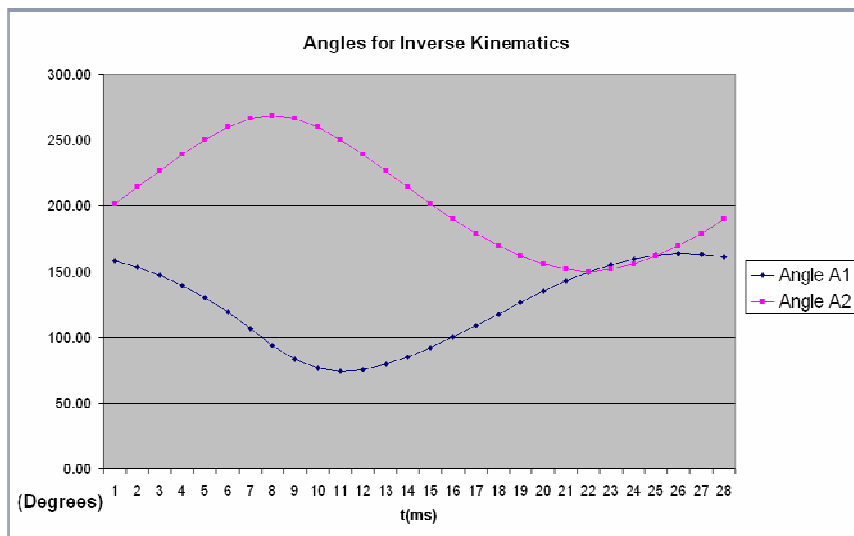


Figure 4.5 Graph showing Joint Angles vs Time for Elliptical Trajectory



Now, we will need to make sure our mathematical model matches the physical model, i.e. we'll need to check and see that the angles A_1 & A_2 use the same mathematical and physical references – if they don't we'll need to adjust these angles to match the physical mounting of the motors.

As can be see in figure 4.6, the centre position of the AX-12 motor is 150° . Given this physical constraint, it would be best to mount the AX-12's vertically as shown in this diagram. Physically, this will give us the maximum movement of the AX-12's in either direction, i.e. this will give us a 150° centre and $\pm 150^\circ$ movement.

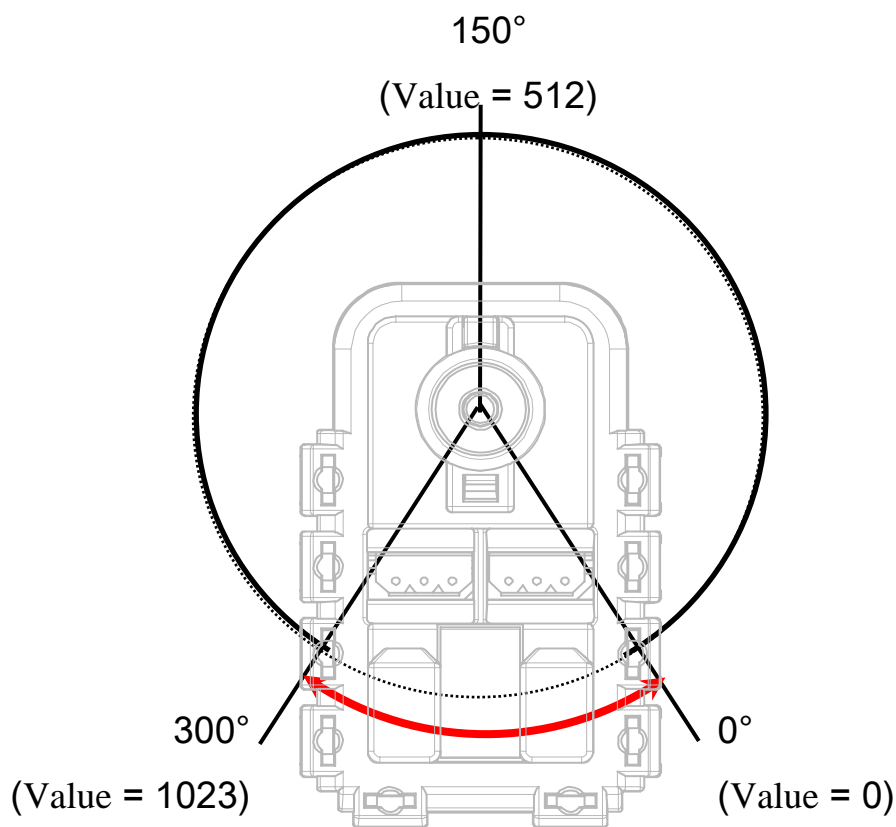


Figure 4.6 this is the rear view of a Dynamixel module (AX-12)

Having decided where our reference will be, we will need to look at the data from the spreadsheet and see how this can be mapped to our physical configuration.

The data for A_2 is fine, as this angle was referenced orthogonally to tip of the first joint. But, A_2 was referenced to *+ve x-axis*. If we choose to mount the AX-12 vertically then our physical reference will no longer be the *+ve x-axis* but *-ve y-axis* – this will require our angles for A_2 to be rotated by -90° . This is achieved simply by subtracting 270° from A_2 .



Now that we are happy with the referencing of our angles we will need to take into account the 150° offset caused by the physical characteristics of the AX-12. This is achieved simply by adding 150° to our correctly referenced angles A_2 and A_3 .

The last thing we need to do is convert these *angles* to *numbers* which the AX-12 uses for positioning. As can be seen in figure 4.6, 0° has a value of 0 whilst the maximum angle 300° has a value of 1023. This is because the microcontroller within the AX-12 uses an AtoD with 10bit resolution, $2^{10}=1024$.

The block diagram shown in figure 4.7 illustrates how the position of the AX-12's motor's shaft is determined, and then how it is converted from an angle to a digital value used by the microcontroller.

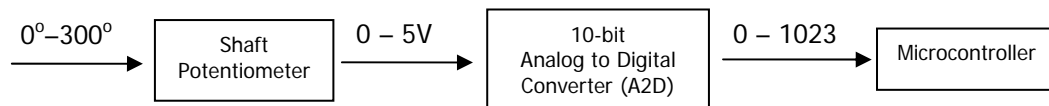


Figure 4.7 Conversion of the AX-12's position to a digital value

As the angle to number relationship is linear, we can determine the required number by using the following equation:

$$n = \frac{1024}{300}d = 3.41d \quad (4.2)$$

where: n is the digital value, and
 d is the position of the AX-12 in degrees.

The mapping of the resultant data (provided by the mathematical solution) to the numerical values (used by the Motion Editor to position the AX-12's) are shown in the *Bioid Implementation* worksheet within the main spreadsheet.

These values were entered manually into the Motion Editor and the robot dogs *paw* tracked an elliptical trajectory as required. As expected, this movement was a little 'jumpy' due to only having 28 positions around the ellipses circumference BUT it did track an ellipse. Obviously the next step would be to take this movement from one leg and transfer this to all the robot dogs legs, I won't attempt this in this article though – I think this article has been long enough ...