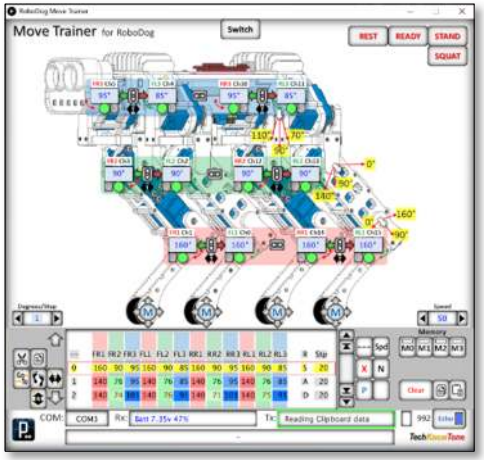
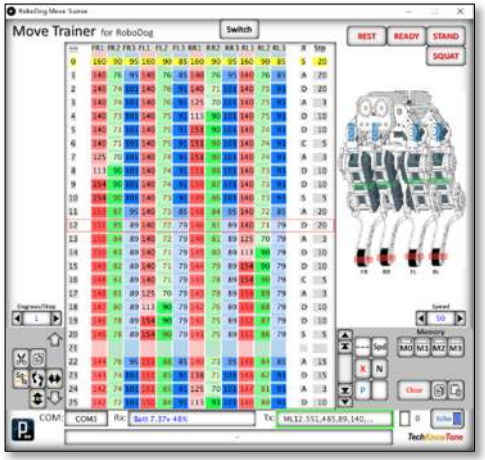
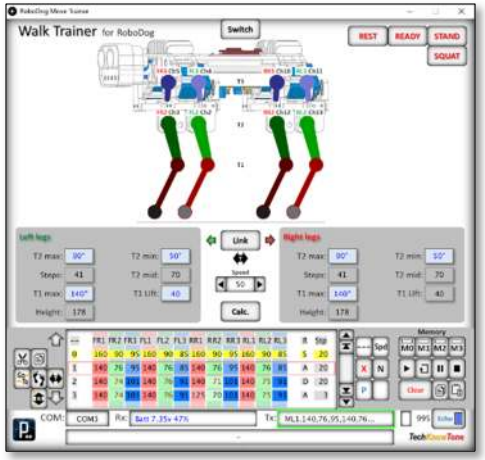
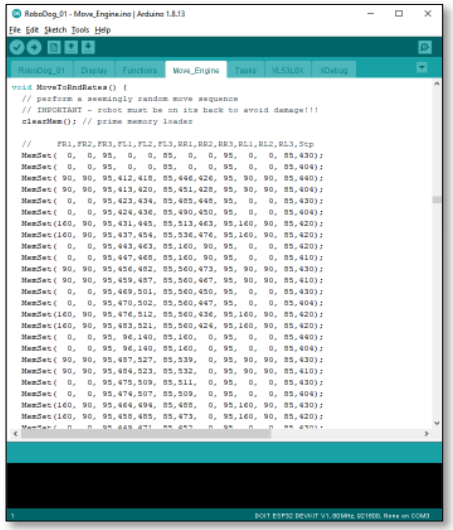
RoboDog

Move Trainer App







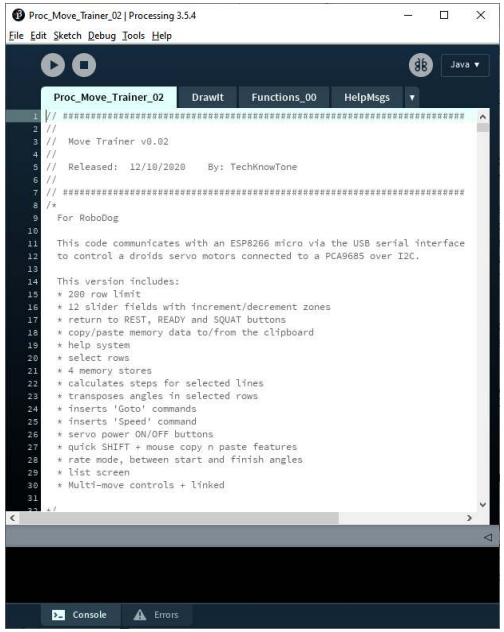


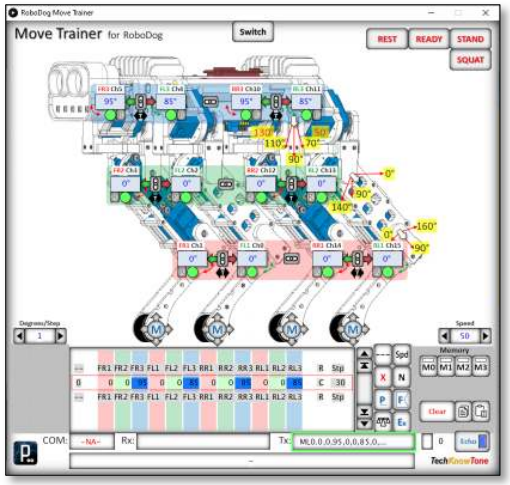
ARDUINO OPEN-SOURCE COMMUNITY

TechKnowTone

Introduction

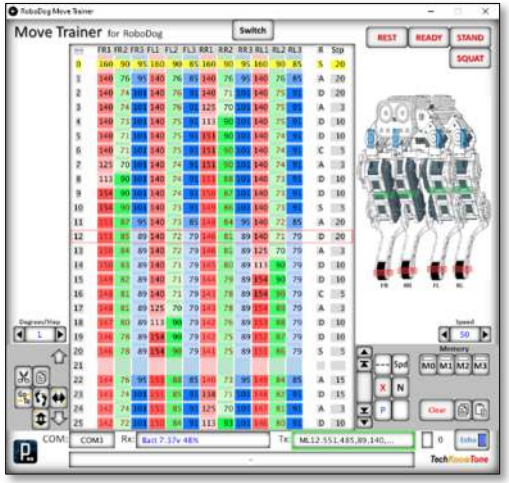
In order to control and develop movements for a complex robot like RoboDog was necessary to create a software application to run on a PC. The ‘Move Trainer’ app was written in C++ using a development environment called Processing. The IDE used in Processing was the forerunner to the Arduino IDE, so it should be a familiar tool to work with, and I recommend it.



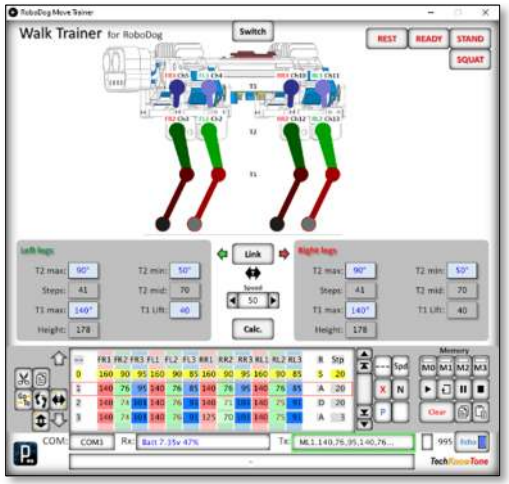


When you invoke the app you are presented with the form shown top right. There is a graphical representation of the robot, and below that a list window, which initially only contains one line of servo values. Start the app now whilst reading this.

Beneath the list window are a number of communication fields, and the bottom field is an integrated help system. As you move the mouse over the form, help messages appear to describe the function of the item beneath the mouse pointer.



As we have 12 servos to control the 4 legs these have been assigned colours and reference numbers to make them more recognisable. The servos are considered as occupying 3 tiers, with tier 1 being the lowest and 3 the highest. For example servo reference **FR3** is the **F**ront **R**ight servo in tier **3**.

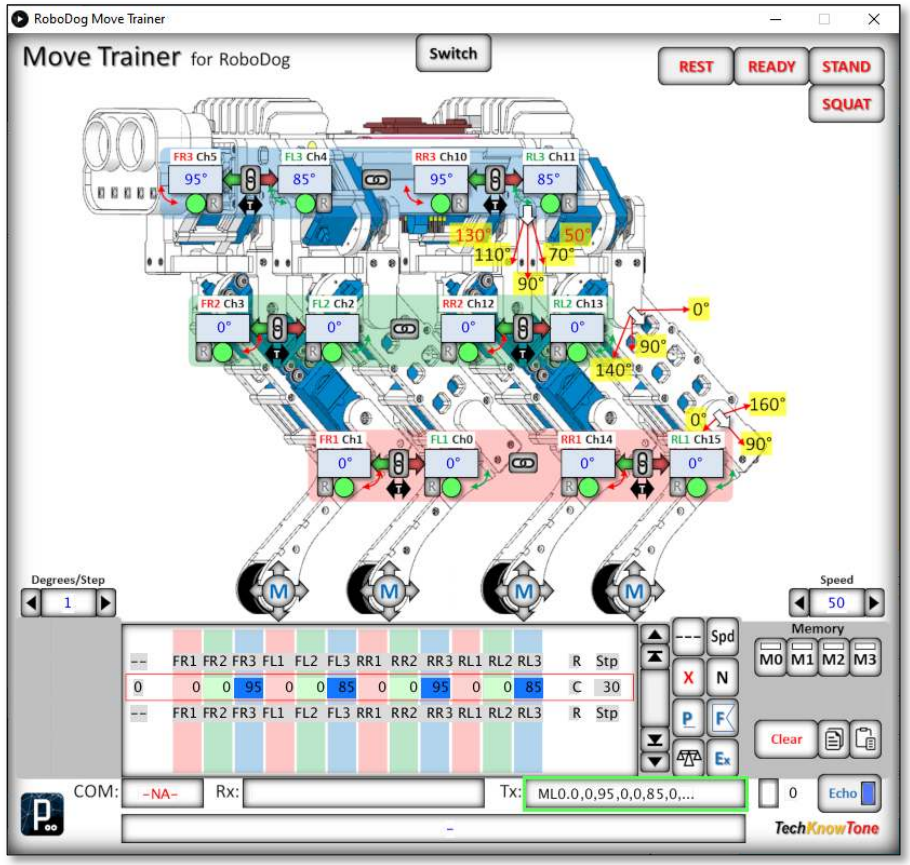


To use this Windows app you need to be connected to the robot via a USB serial link, either directly or over Wi-Fi using the Wii Transceiver. It is recommended that you use the latter, as a USB cable attached directly to the robot will affect its movements and balance.

Note that the app has three forms. You switch between them by clicking on the ‘Switch’ button at the top of each form. All of the background graphics were created in PowerPoint.

Move Trainer

Moving the mouse over the RoboDog image will cause fields to be highlighted in the list, and similarly moving the mouse over the list will cause corresponding fields to by highlighted on the image. This helps you track which servo you are dealing with.



Once you have made the physical USB connection, you can connect the app to the robot by clicking on the COM: field, which displays – NA – initially in red. If the app is able to connect it will display the Windows COM number in black. Note that a left mouse button click makes a connection, and a right button click breaks the connection. This is useful to know as you may want to upload code to you robot from the IDE, and if the app is connected to the serial port the upload will fail.

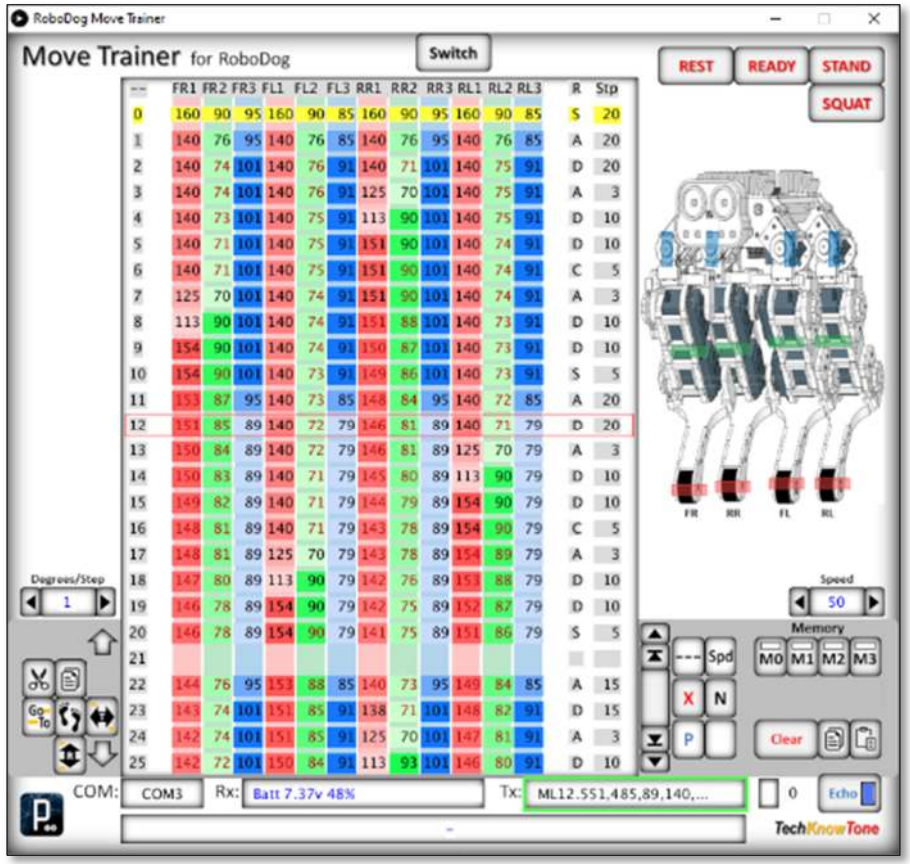
Note that the Move Trainer does not control the robots head servo, as that is controlled by a separate process in the robots code.

Having made the connection to the robot you are now in a position to control it, but by default the robot is set to a '**SAFE**' condition, and this is indicated by the green box drawn around the Tx: field. This allows for changes to be made to the list of servo values without the robot reacting to them, which could cause significant problems. To make the robot '**LIVE**' you simply click the mouse pointer within the Tx: field and the box drawn around it will go red if the robot is connected.

Start with the robot in the kneeling down position, before making the link LIVE. Then click on the servo link button which will effective tether the FR1 and FL1 servos together. A red box outline should appear around the link button. Now click and drag to the right in the FR1 angle field, which should cause the lower front legs to push down and raise the robot up. You've made your first move! Note that the active line in the servo list is in a red box, and that the values change in that line as you click and drag as described above.

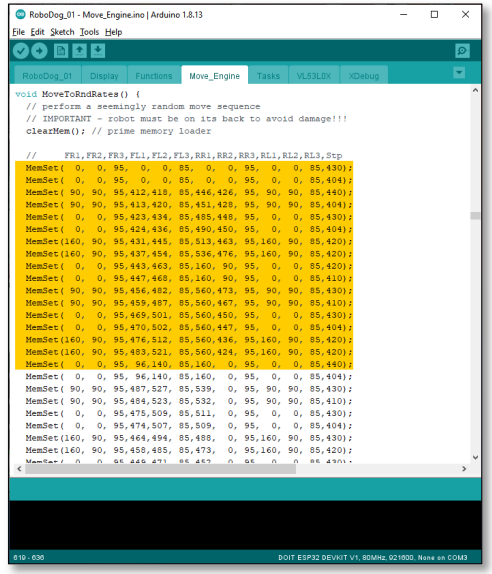
Move Trainer

The second form view is helpful when you have a long list of complex movements, as seen in the image on the right. I've used colour to help indicate where servo values have changed in the list. Note that line 0 is in yellow, indicating that it has been selected. When you select one or more lines in the list, by clicking in the leftmost column, you will see that additional button options become active and appear on the form.



The Move Trainer can convert the list of values into a form used directly in your micros code and place them on the clipboard using the memory copy button. Similarly you can take code from the IDE and paste it into this application. Try opening the .ino file in the IDE and go to the Move Engine tab, then find a function containing MemSet{} servo load statements, like the function MoveToRndRates().

Select and copy several lines of code within the IDE, then return to the Move Trainer and click and hold the memory Paste button.

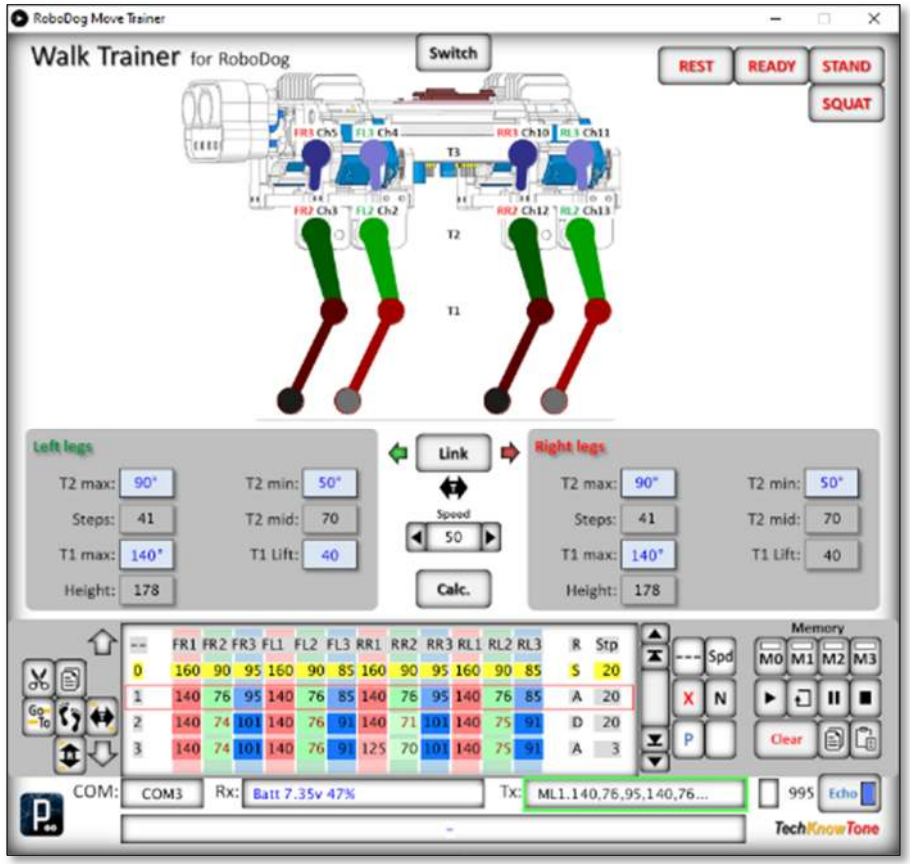


Your Move Trainer list should then fill with the values you selected with the IDE. This shows you how easy it is to bring in existing movements from the code provided. In a similar way you can create your own set of movements and then use the memory copy button to place them on the Windows clipboard in MemSet{} format.

As you move the mouse pointer over the values in the list the highlights drawn over the RoboDog image will move up and down, to correspond with the servo values in the list. This simple form of animation helps you image what the servo movements would achieve if the link was active. You've just created your first list of servo commands.

Walk Trainer

The third form view was developed, as I was struggling to get the robot to walk smoothly. Dynamic effects of movement aren't obvious, as they aren't present when creating individual joint movements. This view presents an animated version of the robot in which the angles of the legs are draw as an overlay, and there are fields in which values can be placed, with the intention of automatically generating a list of servo movements using coded algorithms.



Sadly the solution was not fully developed, but the animated movement of a list can be seen when you click on the play ▶ button, when the USB link is inactive, and the animation actually tracks what the robot is doing when the link is active.

A 'ground' line is also drawn in the image, below the robot, to indicate the lowest point. Which is useful when you are wanting a succession of leg movements all to be at the same extension.

Hopefully this simple overview has been of interest in giving you an insight into what you can do with this app. It has an extensive range of features, which are highlighted by the interactive help system, which would take a lot of effort to document in this form. You can see what can be achieved when coding using Processing, in combination with background images created in PowerPoint. And, as the coding language and IDE are very similar to what you have used with Arduino, it may inspire you to develop similar solutions for your projects where the addition of a PC app can make the difference between success and failure.

Enjoy!