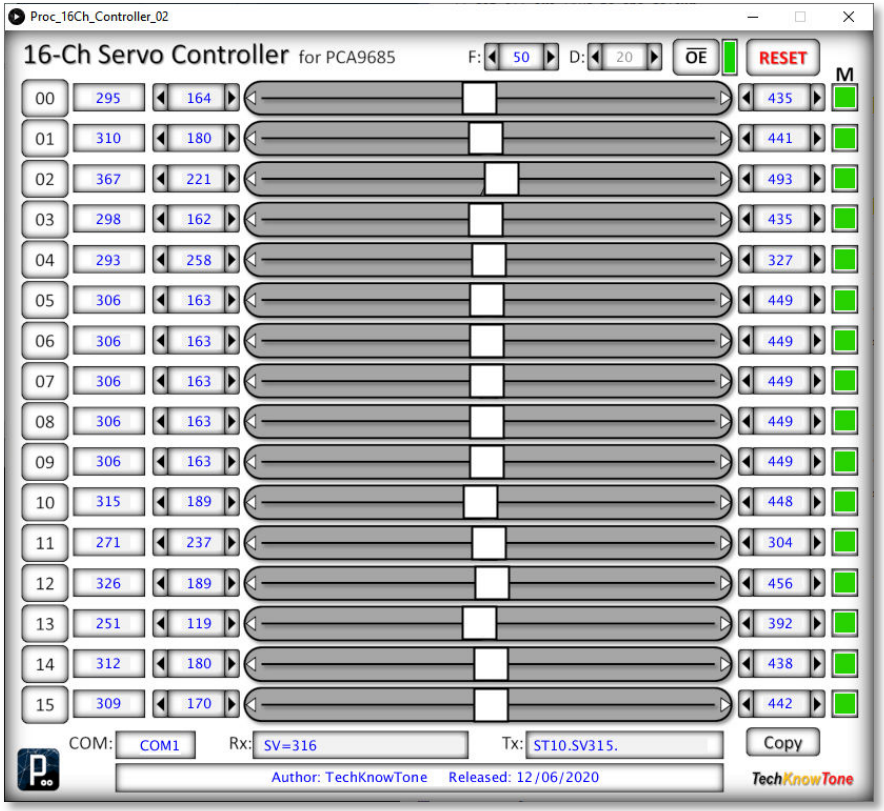
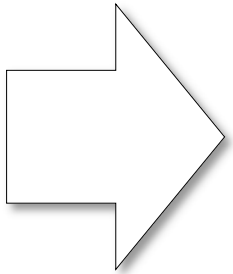
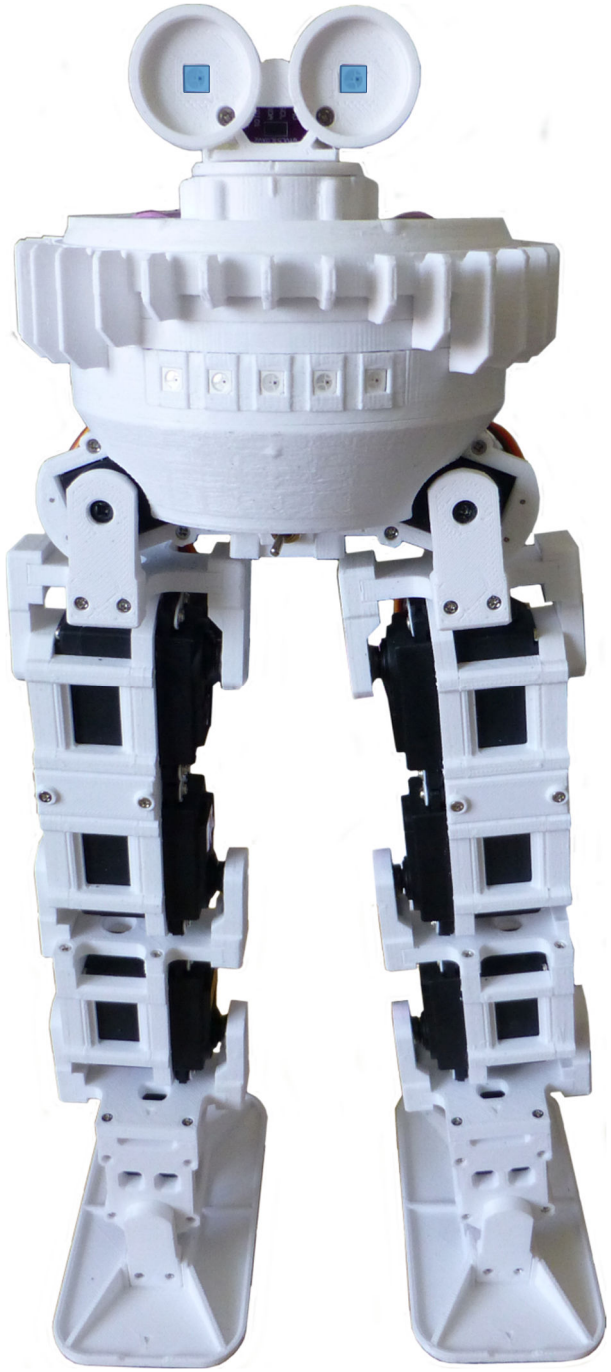
Biped Droid

Servo Calibration



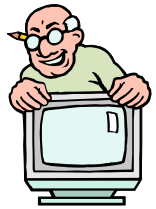




ARDUINO OPEN-SOURCE COMMUNITY

Issue: 1.1 Released: 08/12/2025 TechKnowTone

CAUTION



Lithium batteries can be extremely dangerous, if not handled and cared for properly. This design does not include any form of current limiting circuit, like a fuse. So, care must be taken to ensure that the wiring guidelines are followed accurately, that checks are made for short-circuits, and that battery polarities are marked, and they are inserted the correct way round. Failure to do so, could result in an explosive fire.

Charging Practices: Always remove batteries from your project to charge them. Use a charger, designed for the battery used, and from a trusted supplier. Choose a flat, non-flammable surface to charge on, away from flammable materials. Never leave unattended when charging. Don't charge overnight. Monitor charging to ensure charge characteristics are as expected. Only pair batteries with similar characteristics. Do not overcharge, or leave charging for prolonged periods. This increases the risk of damage and fire.

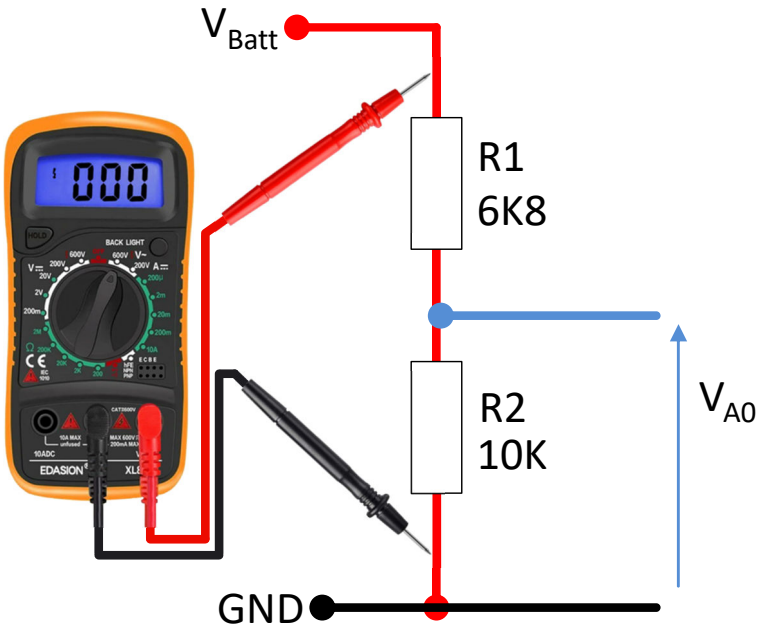
Battery care & maintenance: Stop using a battery if it is swollen, damaged, dented or leaking. Never charge a damaged battery. Never allow a Lithium battery to discharge below 3.2 volts, as cell damage will occur. Avoid extreme temperatures. Do not charge or store batteries in very hot or cold environments. Don't cover batteries whilst charging, as this can trap heat, causing overheating.

In case of fire: Get out and stay out. If a fire starts, leave immediately, and call the fire brigade. For low voltage Lithium batteries, water is a safe extinguisher.

Built-in Monitoring: Most of my project designs include code, and circuitry, to monitor battery voltage, whilst in use. This code then seeks to alert the operator, when the battery has reached a critical low voltage, before shutting down power consuming circuitry; including the micro. Time should therefore be spent on calibrating this feature, as a precaution, for good battery management and maintenance.

Carefully dispose of batteries that damaged, or discharged below their critical voltage.

Battery Monitor (Protection)



$$V_{A0} = \frac{V_{Batt} \times R2}{R1 + R2} \qquad V_{A0D} = \frac{V_{A0} \times 1023}{5}$$ voltage read by 10-bit ADC

$$V_{A0} = \frac{V_{Batt} \times 10K}{16K8} \qquad V_{A0D} = \frac{V_{Batt} \times 0.5952 \times 1023}{5}$$

$$V_{Batt} = \frac{V_{A0D} \times 5.0}{608.9}$$

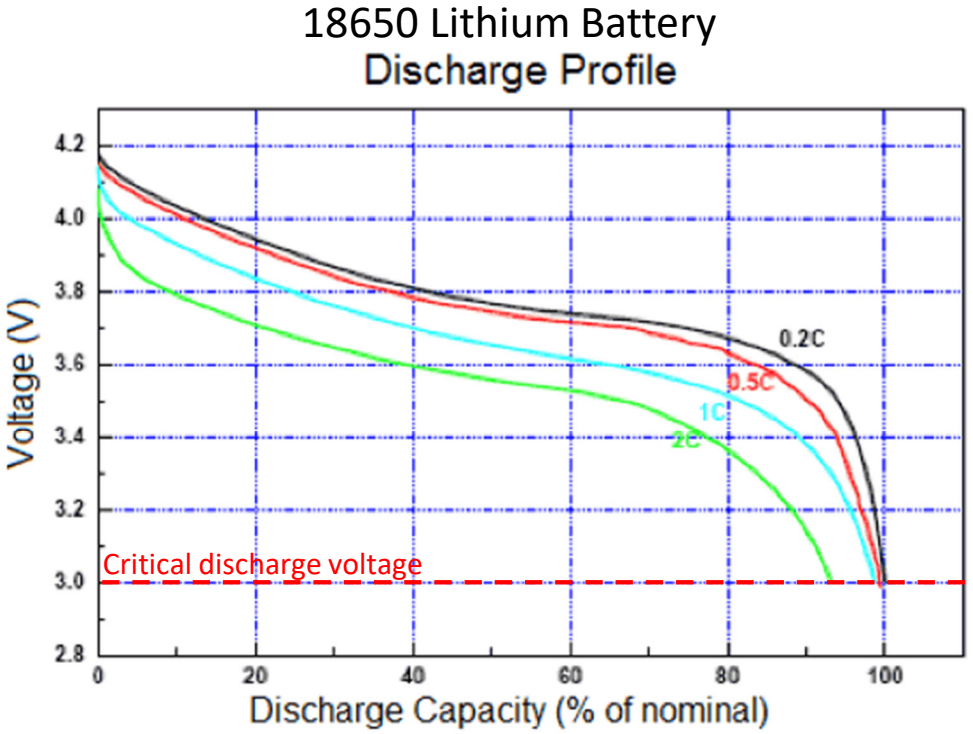
$V_{FSD} = 8.4v$ @ $V_{A0} = 5v$

Two cells in series gives a nominal 7.4v constant discharge voltage. To prevent damage, stop using once the following conditions are reached:

3.60 + 3.00 = 6.60v (one battery fades early)
3.30 + 3.30 = 6.60v (both batteries fade together)

Hence $V_{A0D} = 540$ @ $V_{Batt} = 6.60v$

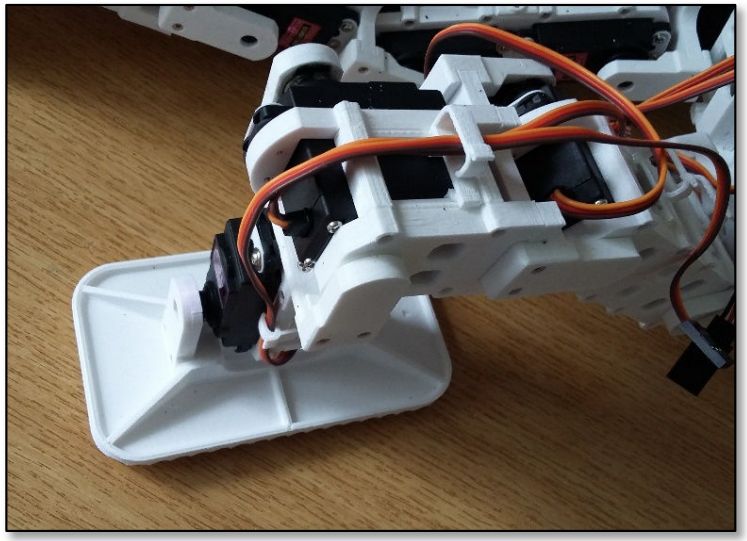
The code will shut down when the value drops to 540.

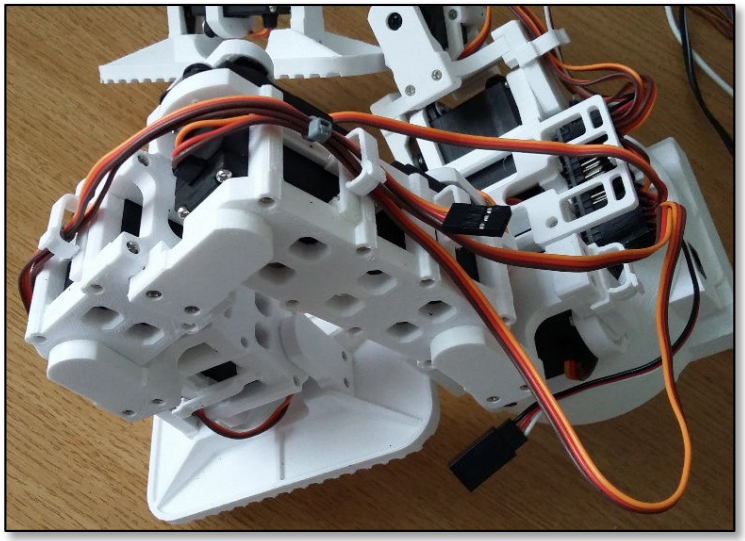


Discharge: 3.0V cutoff at room temperature.

Issue: 1.1 Released: 08/12/2025

Course Calibration

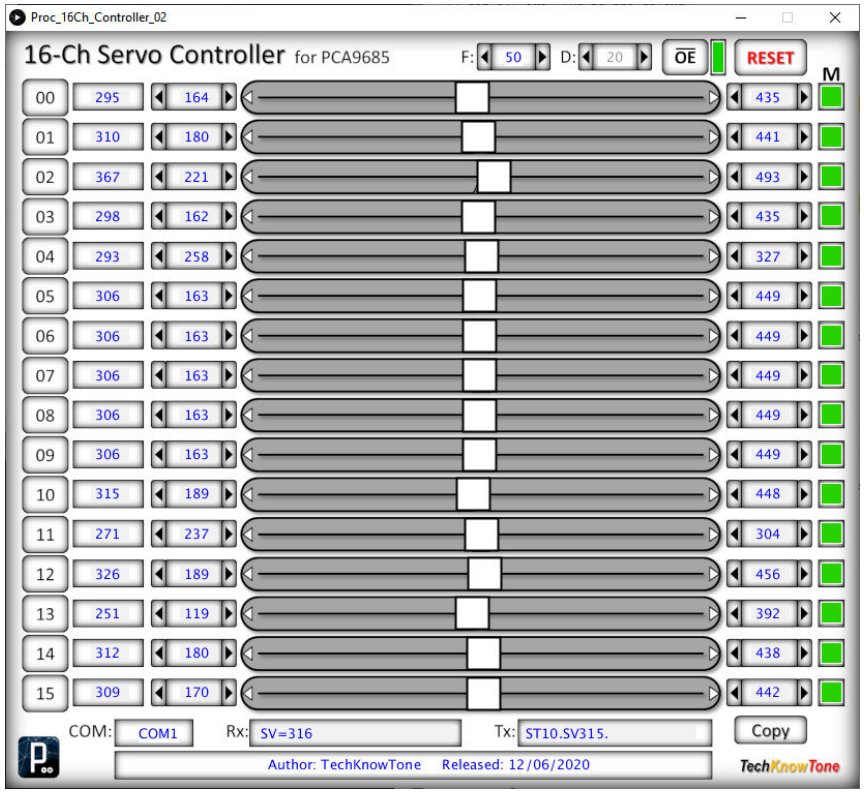




Use a servo tester when attaching the servos to your droid initially, to set the angles of the attached levers in their approximate positions. The next slide shows the PWM values to be set for each servo.

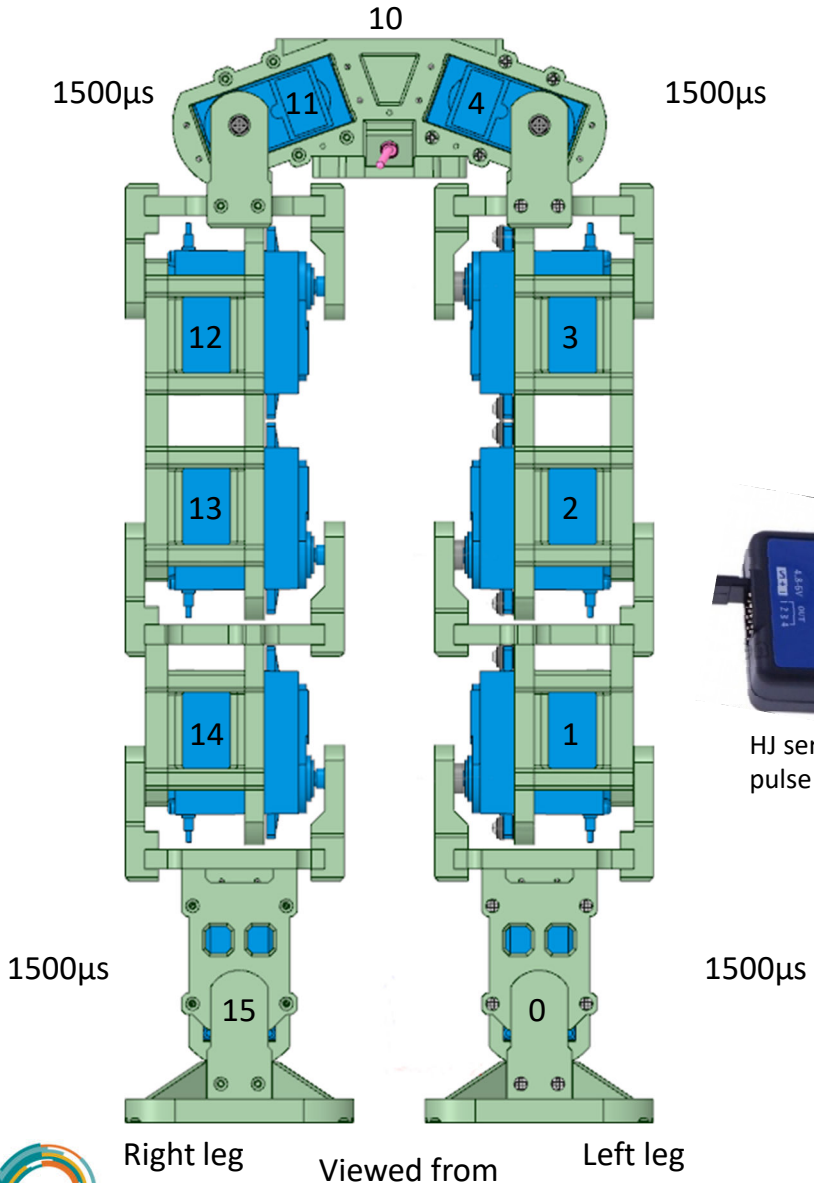
Once you have assembled and wired up your droid to the PCA9685 driver board you can use the application I have provided. This works with special code you need to install temporarily on the NANO EVERY. With this app you can select each servo in turn and move the slider to determine values needed to achieve the calibration angles shown in the subsequent pages of this guide. With all of the values determined you simply press the COPY button and paste them directly into your Arduino code.

Note that as no two servos are alike, the values I have given in this guide, and included in the sample code, will be similar but different from the ones you derive from your robot. That's the nature of servos!

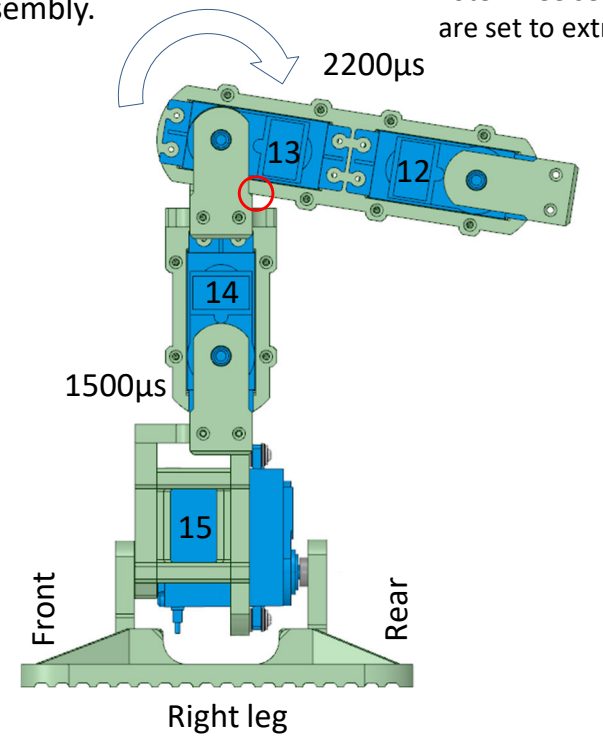


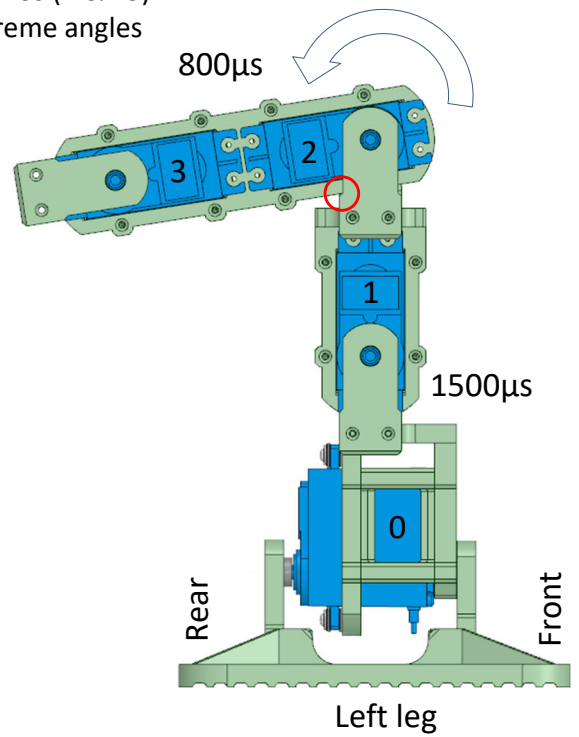
Biped Droid Servo setup – use these nominal values during assembly.

During assembly set each servos shaft to these nominal values using a servo tester before attaching the leaver arms. Choose the best spline fit to give you the least angular error. In all cases there is likely to be a compromise. More accurate values will be determined when the assembled droid is calibrated later.

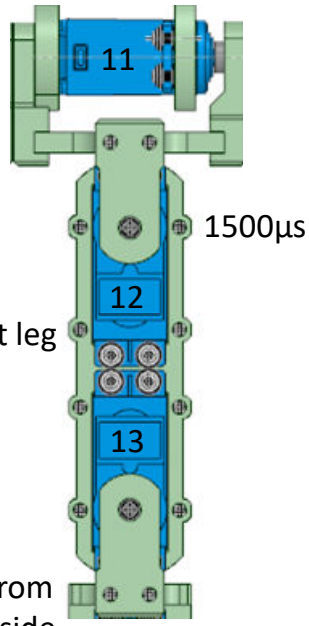


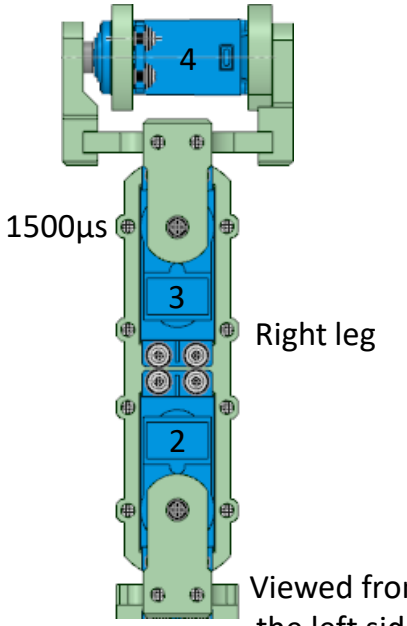
Note: knee servos (2 & 13) are set to extreme angles





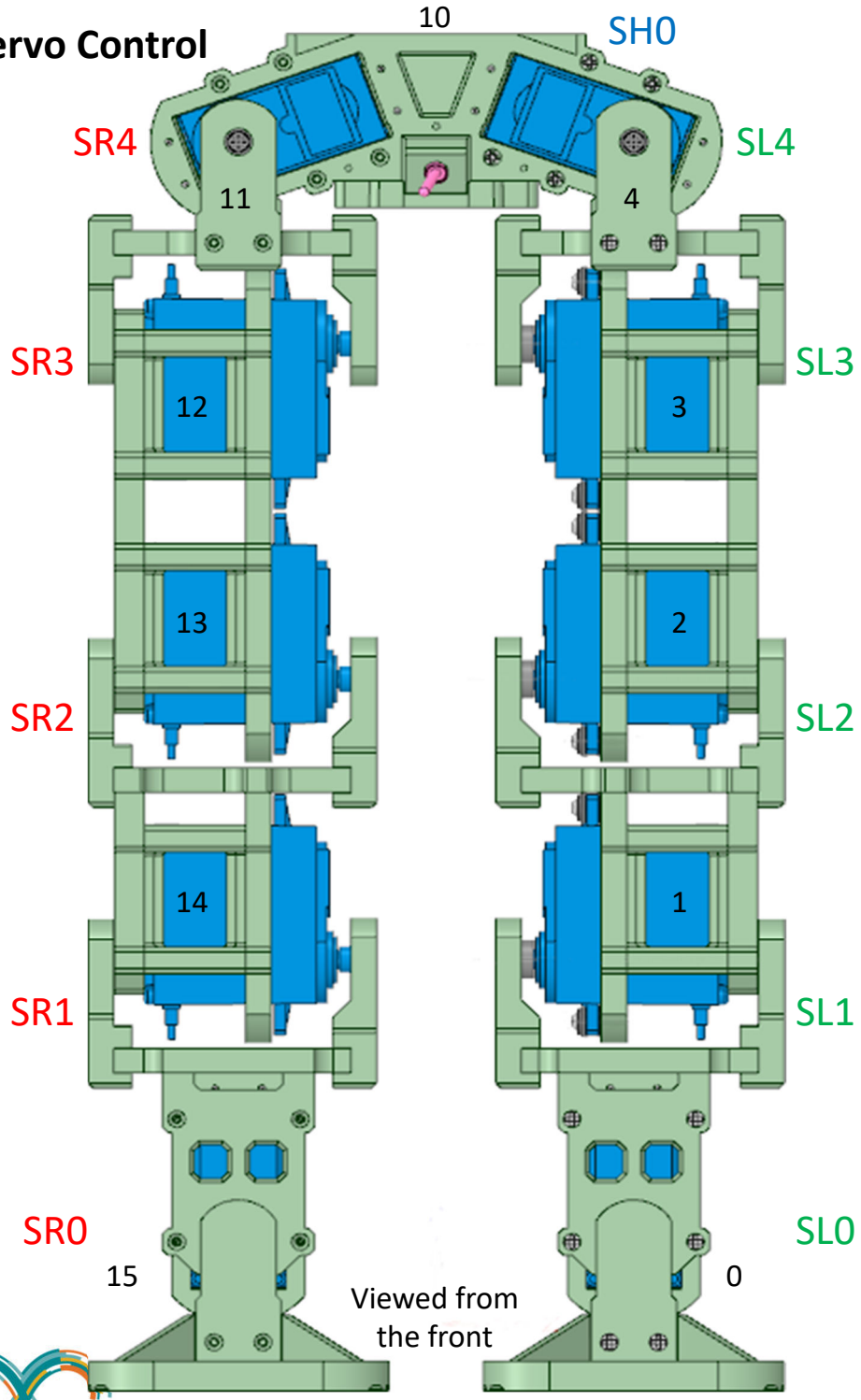
HJ servo tester displays pulse widths directly in µs.





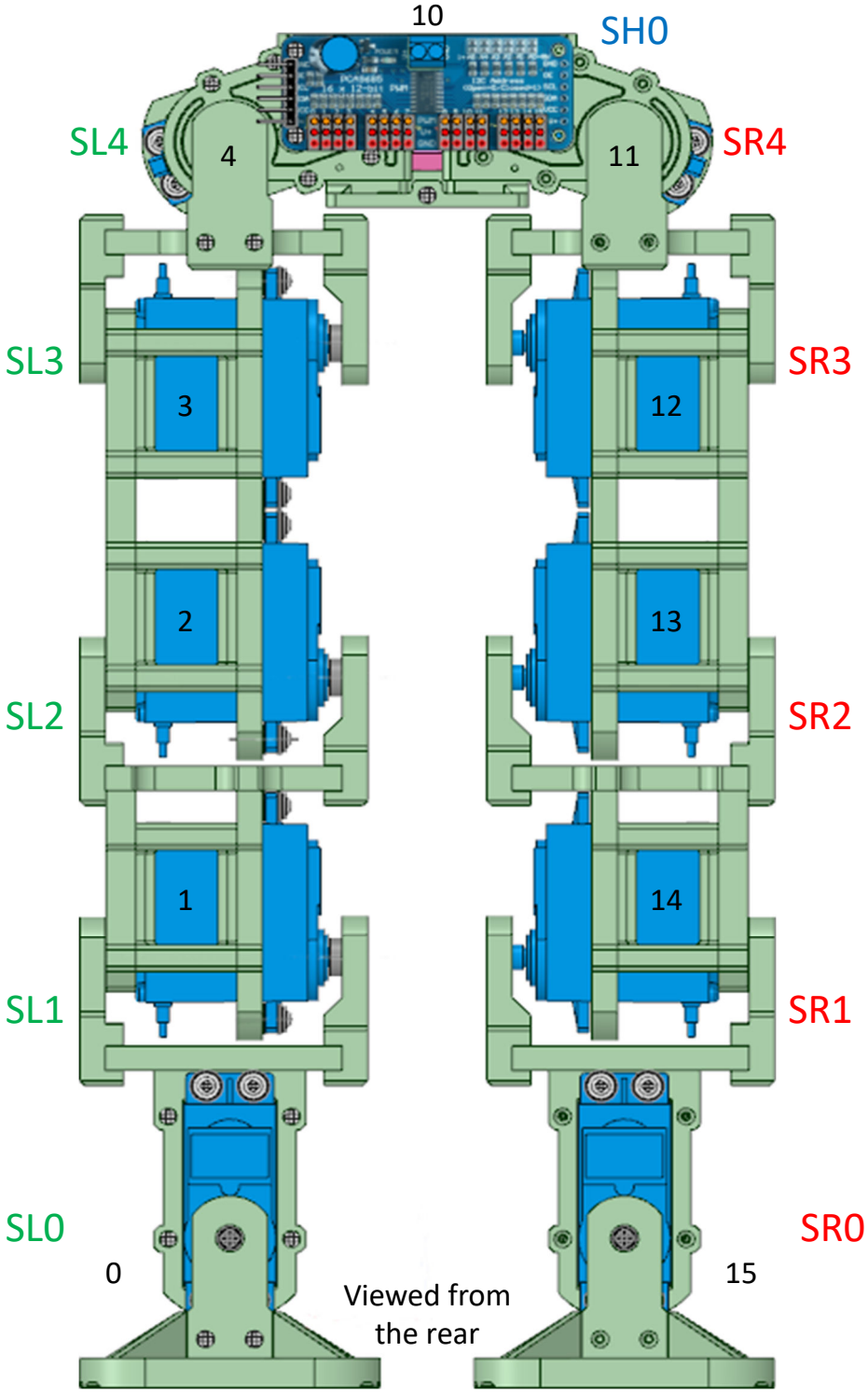
Issue: 1.1 Released: 08/12/2025

Servo Control



Servo Mapping

SL0	0
SL1	1
SL2	2
SL3	3
SL4	4
NC	6
NC	7
SR0	15
SR1	14
SR2	13
SR3	12
SR4	11
SH0	10
NC	9
NC	8

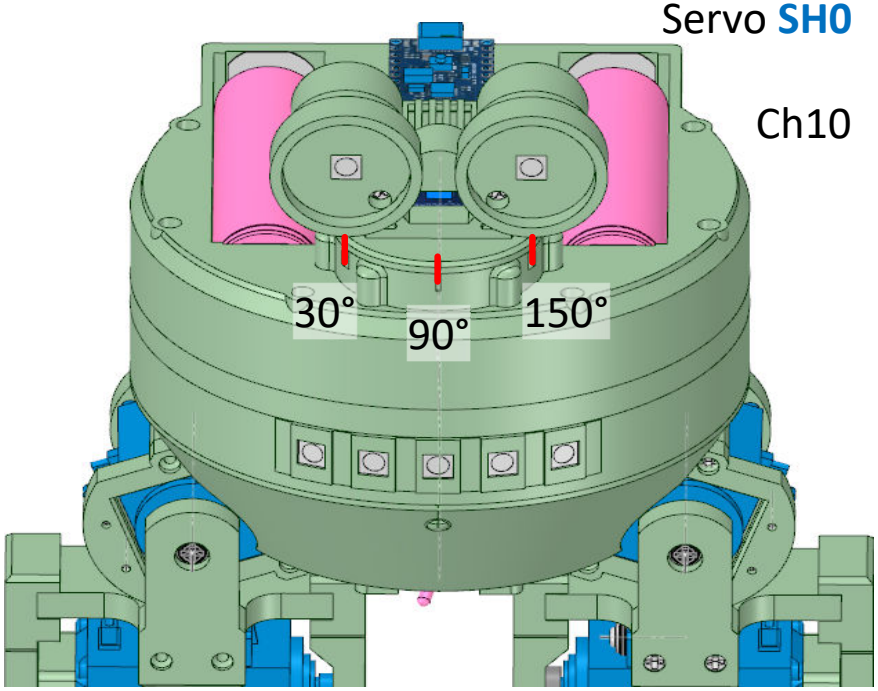


Issue: 1.1 Released: 08/12/2025

Head Calibration

Servo **SH0**

Ch10



Start by calibrating the head servo SH0, which is connected to channel 10 of the PCA9685 driver board. Whilst the MG90S servo may move over a range of 180° it is not desirable to drive it to its mechanical end stops, as excessive current will be drawn, wasting battery power and reducing the life of the servo. So in this design I have chosen to move it through 120°, +/- 60° from the centre position. Note that the 3-D models include reference marks at these angles to make this calibration process simple to perform.

Enable channel 10 only on the 16-Ch controller app and move the corresponding slider left to determine the lower setting. Note that the memory 'M' lamp for channel 10 will change from green to yellow. It should be possible to move the servo beyond this target point if the servo arm was fitted correctly during the build process. If this is not the case the arm will need to be removed and reattached to the servo in a more counter-clockwise position. With the marker on the rotation head opposite the 30° mark, seen in the image, record that value and click in the lower limit text field of the 16-Ch controller to set this channel 10 lower limit to the current slider PWM position.

Now move the channel 10 slider to the right, and in the same way determine the PWM value needed to reach the leftmost 150° marker position. Record this value and click in the upper limit text field of the 16-Ch controller to set the channel 10 upper limit to the current slider position.

Finally move the slider towards the centre 90° position to determine the PWM value needed for that marker. If the servo is linear the value should be half-way between the lower 30° and upper 150° limits found, but this may not quite be true, as seen in the values here.

Now click on the memory 'M' lamp for channel 10, holding the mouse key down for more than 1 second for it to turn from yellow to blue. The values you have set will then be memorised by the app and will be included in the data copied to the clipboard for pasting into your Arduino code later.

You should now be able to move the channel 10 slider freely from left to right between the upper and lower limits. Finally turn OFF channel 10.

Angle	PWM	
30°	189	LL
90°	315	Centre
150°	458	UL

Centre PWM = LL + ((UL – LL)/2)
= 189 + ((458 – 189)/2)
= 323

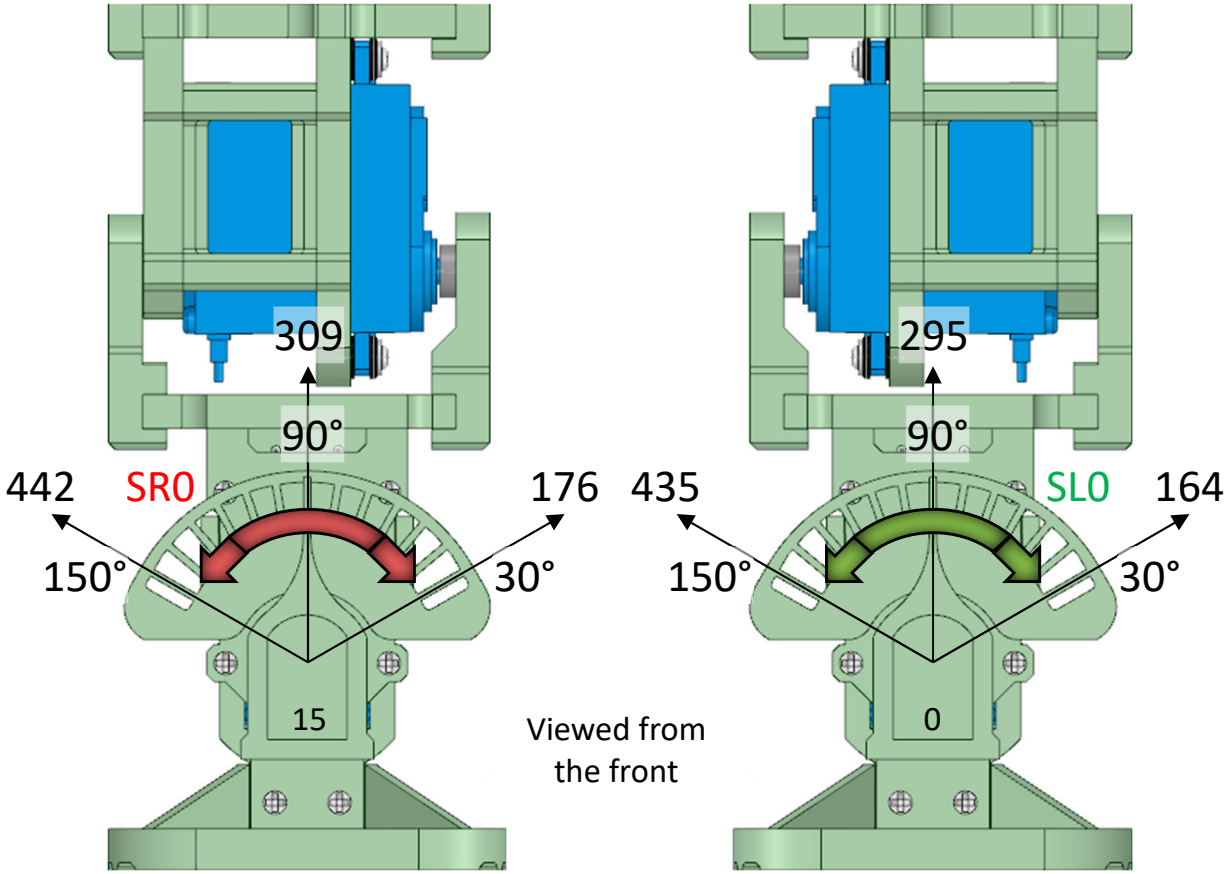
The calculated and measured centre values may not match if the servo is not linear.

Feet Calibration +/- 60° (30° – 90° – 150°)

Next we will calibrate the feet servos SR0 and SL0, which are connected to channels 15 and 0 respectively on the PCA9685 driver board. Lay the droid on its back and apply the angle gauge and pointer to the left foot as seen in the diagram. The angle gauge is graduated in 10° steps, with 90° in the centre and +/- 60°.

Enable channel 15 only on the 16-Ch controller app and move the corresponding slider left to determine the lower setting. Note that the memory 'M' lamp for channel 15 will change from green to yellow. You may need to reduce the lower limits value to achieve the 30° point. It should be possible to move the servo beyond this target point if the servo arm was fitted correctly during the build process. If this is not the case the arm will need to be removed and reattached to the servo in a more clockwise position. With the pointer opposite the 30° mark, seen in the image, record that value and click in the lower limit text field of the 16-Ch controller to set this channel 15 lower limit to the current slider PWM position.

Now move the channel 15 slider to the right, and in the same way determine the PWM value needed to reach the leftmost 150° marker position. Record this value and click in the upper limit text field of the 16-Ch controller to set the channel 15 upper limit to the current slider position.



Note:
* Foot angle conventions match hips.
* Code limits used are +/-80° (10° – 90° – 170°)
* Try LL and UL extremes to ensure servo can exceed +/-80°

Finally move the slider towards the centre 90° position to determine the PWM value needed. If the servo is linear the value should be half-way between the lower 30° and upper 150° limits found, but this may not quite be true, as mentioned earlier. Now click on the memory 'M' lamp for channel 15, holding the mouse key down for more than 1 second for it to turn from yellow to blue. The values you have set will then be memorised by the app and will be included in the data copied to the clipboard for pasting into your Arduino code later. Now turn OFF channel 15.

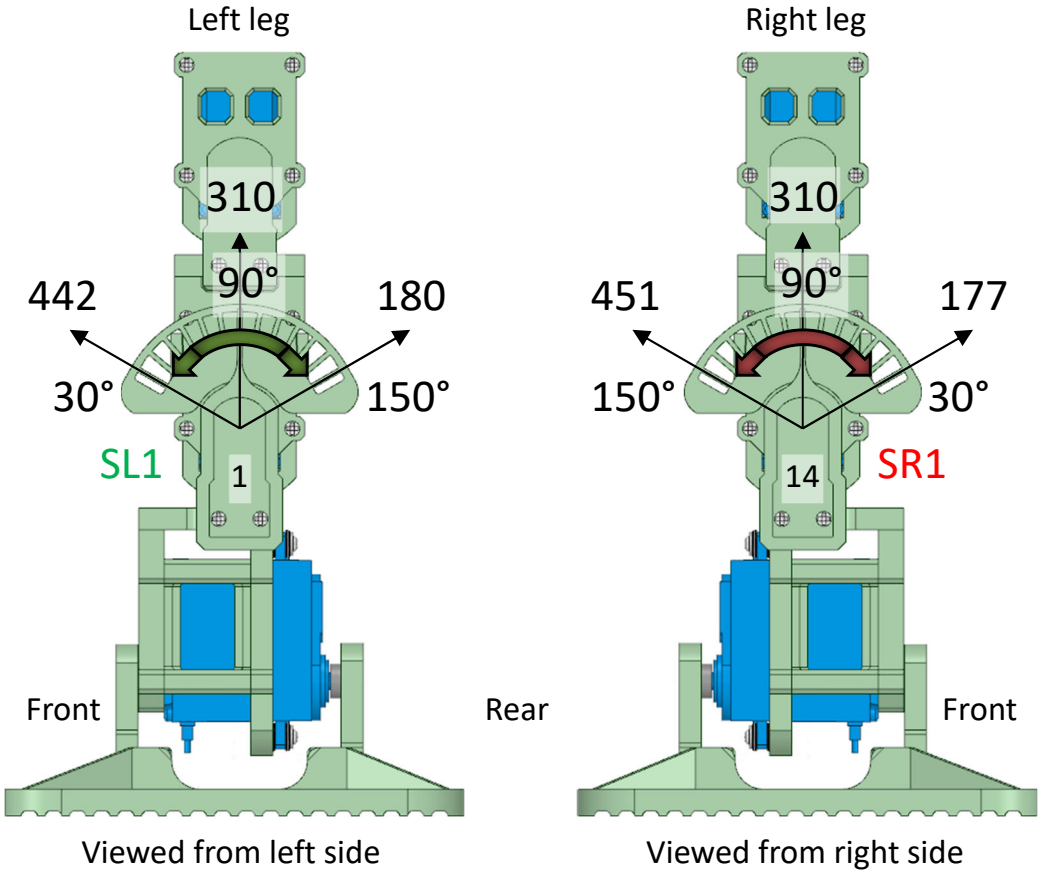
Now fit the angle gauge and pointer to the left foot and repeat the process for channel 0, setting the lower, upper and centre point values. Then memorise the values using the 'M' button in the same way. Note that at any point you can copy the memorised values to the clipboard and paste them into the Arduino code as a permanent record. When the 16-channel controller app connects to the Arduino over USB it receives these default values at the outset. Disconnecting and reconnecting the COM link will reset the Arduino and cause this to happen.

Ankle Calibration +/- 60° (30° – 90° – 150°)

Next we will calibrate the ankle servos SR1 and SL1, which are connected to channels 14 and 1 respectively on the PCA9685 driver board. Lay the droid on its left side and apply the angle gauge and pointer to the right ankle as seen in the diagram. The angle gauge is graduated in 10° steps, with 90° in the centre and +/- 60°.

Enable channel 14 only on the 16-Ch controller app and move the corresponding slider left to determine the lower setting. Note that the memory 'M' lamp for channel 14 will change from green to yellow. You may need to reduce the lower limits value to achieve the 30° point. It should be possible to move the servo beyond this target point if the servo arm was fitted correctly during the build process. If this is not the case the arm will need to be removed and reattached to the servo in a more clockwise position. With the pointer opposite the 30° mark, seen in the image, record that value and click in the lower limit text field of the 16-Ch controller to set this channel 14 lower limit to the current slider PWM position.

Now move the channel 14 slider to the right, and in the same way determine the PWM value needed to reach the leftmost 150° marker position. Record this value and click in the upper limit text field of the 16-Ch controller to set the channel 14 upper limit to the current slider position.



Viewed from left side

Viewed from right side

Note: * Ankle angle conventions match hips swing.
* Code limits used are +/-80° (10° – 90° – 170°)
* Try LL and UL extremes to ensure servo can exceed +/-80°

Finally move the slider towards the centre 90° position to determine the PWM value needed. If the servo is linear the value should be half-way between the lower 30° and upper 150° limits found, but this may not quite be true, as mentioned earlier. Now click on the memory 'M' lamp for channel 14, holding the mouse key down for more than 1 second for it to turn from yellow to blue. The values you have set will then be memorised by the app and will be included in the data copied to the clipboard for pasting into your Arduino code later. Now turn OFF channel 14.

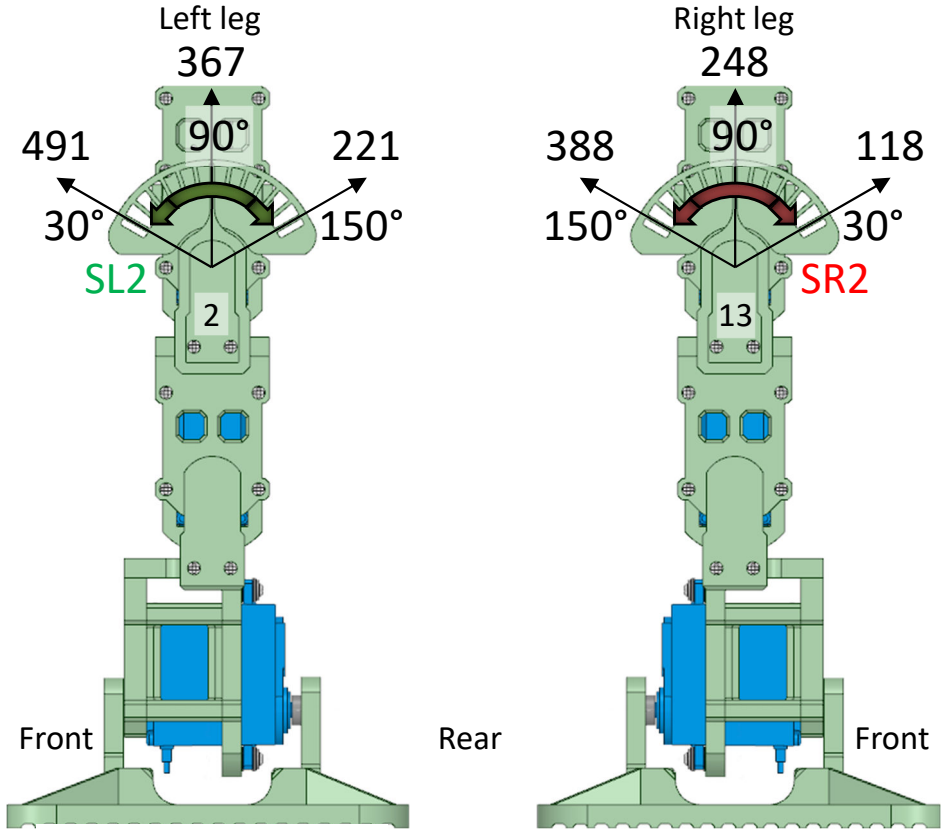
Now turn the droid onto its right side fit the angle gauge and pointer to the left ankle and repeat the process for channel 1, setting the lower, upper and centre point values. Then memorise the values using the 'M' button in the same way. Now turn OFF channel 1. Note that at any point you can copy the memorised values to the clipboard and paste them into the Arduino code as a permanent record. When the 16-channel controller app connects to the Arduino over USB it receives these default values at the outset. Disconnecting and reconnecting the COM link will reset the Arduino and cause this to happen.

Knees Calibration +/- 60° (30° – 90° – 150°)

Next we will calibrate the knee servos SR2 and SL2, which are connected to channels 13 and 2 respectively on the PCA9685 driver board. Lay the droid on its left side and apply the angle gauge and pointer to the right knee, as seen in the diagram. The angle gauge is graduated in 10° steps, with 90° in the centre and +/- 60°.

Enable channel 13 only on the 16-Ch controller app and move the corresponding slider left to determine the lower setting. Note that the memory 'M' lamp for channel 13 will change from green to yellow. You may need to reduce the lower limits value to achieve the 30° point. It should be possible to move the servo beyond this target point if the servo arm was fitted correctly during the build process. If this is not the case the arm will need to be removed and reattached to the servo in a more clockwise position. With the pointer opposite the 30° mark, seen in the image, record that value and click in the lower limit text field of the 16-Ch controller to set this channel 13 lower limit to the current slider PWM position.

Now move the channel 13 slider to the right, and in the same way determine the PWM value needed to reach the leftmost 150° marker position. Record this value and click in the upper limit text field of the 16-Ch controller to set the channel 13 upper limit to the current slider position.



Viewed from left side Viewed from right side

Note: * Ankle angle conventions match hips swing.
* Code limits used are +90°/-60° (30° – 90° – 180°)

Finally move the slider towards the centre 90° position to determine the PWM value needed. If the servo is linear the value should be half-way between the lower 30° and upper 150° limits found, but this may not quite be true, as mentioned earlier. Now click on the memory 'M' lamp for channel 13, holding the mouse key down for more than 1 second for it to turn from yellow to blue. The values you have set will then be memorised by the app and will be included in the data copied to the clipboard for pasting into your Arduino code later. Now turn OFF channel 13.

Now turn the droid onto its right side and fit the angle gauge and pointer to the left knee and repeat the process for channel 2, setting the lower, upper and centre point values. Then memorise the values using the 'M' button in the same way. Now turn OFF channel 2. Note that at any point you can copy the memorised values to the clipboard and paste them into the Arduino code as a permanent record. When the 16-channel controller app connects to the Arduino over USB it receives these default values at the outset. Disconnecting and reconnecting the COM link will reset the Arduino and cause this to happen.

Hips Swing Calibration +/- 60° (30° – 90° – 150°)

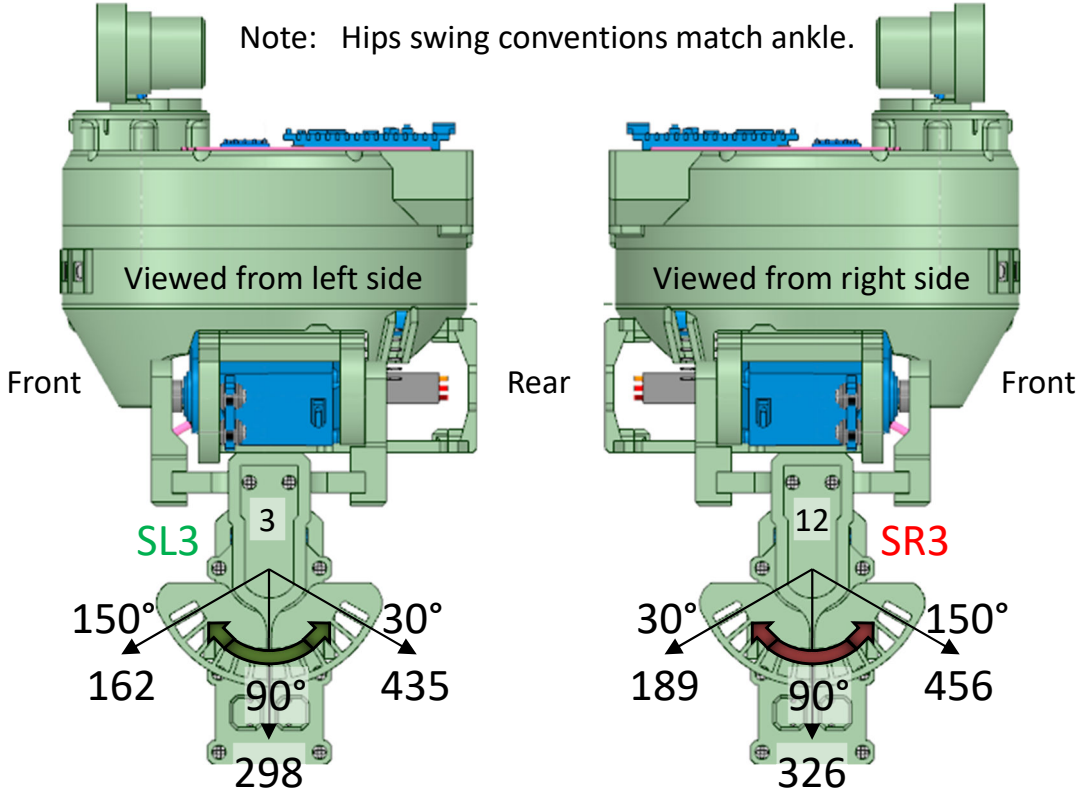
Next we will calibrate the hip swing servos SR3 and SL3, which are connected to channels 12 and 3 respectively on the PCA9685 driver board. Lay the droid on its left side and apply the angle gauge and pointer to the right knee, as seen in the diagram. The angle gauge is graduated in 10° steps, with 90° in the centre and +/- 60°.

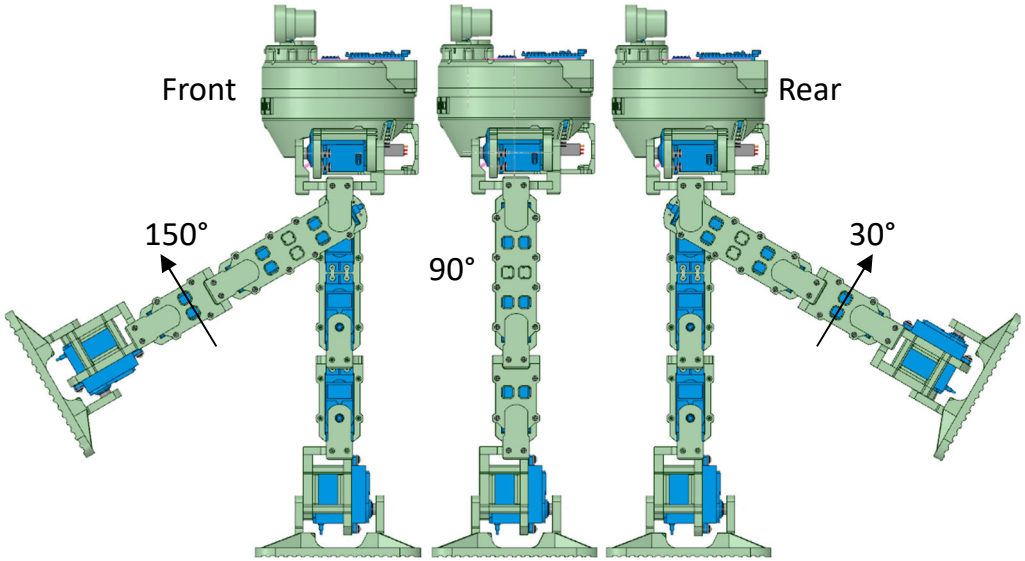
Enable channel 12 only on the 16-Ch controller app and move the corresponding slider left to determine the lower setting. Note that the memory 'M' lamp for channel 12 will change from green to yellow. You may need to reduce the lower limits value to achieve the 30° point. It should be possible to move the servo beyond this target point if the servo arm was fitted correctly during the build process. If this is not the case the arm will need to be removed and reattached to the servo in a more clockwise position. With the pointer opposite the 30° mark, seen in the image, record that value and click in the lower limit text field of the 16-Ch controller to set this channel 12 lower limit to the current slider PWM position.

Now move the channel 12 slider to the right, and in the same way determine the PWM value needed to reach the leftmost 150° marker position. Record this value and click in the upper limit text field of the 16-Ch controller to set the channel 12 upper limit to the current slider position.

Finally move the slider towards the centre 90° position to determine the PWM value needed. If the servo is linear the value should be half-way between the lower 30° and upper 150° limits found, but this may not quite be true, as mentioned earlier. Now click on the memory 'M' lamp for channel 12, holding the mouse key down for more than 1 second for it to turn from yellow to blue. The values you have set will then be memorised by the app and will be included in the data copied to the clipboard for pasting into your Arduino code later. Now turn OFF channel 12.

Now turn the droid onto its right side and fit the angle gauge and pointer to the left hip swing servo and repeat the process for channel 3, setting the lower, upper and centre point values. Then memorise the values using the 'M' button in the same way. Now turn OFF channel 3.





Hips Sway Calibration +/-20° (70 – 90 - 150°)

Finally we will calibrate the hip sway servos SR4 and SL4, which are connected to channels 11 and 4 respectively on the PCA9685 driver board. Lay the droid on its back, facing upwards. There is no angle gauge associated with these setting.

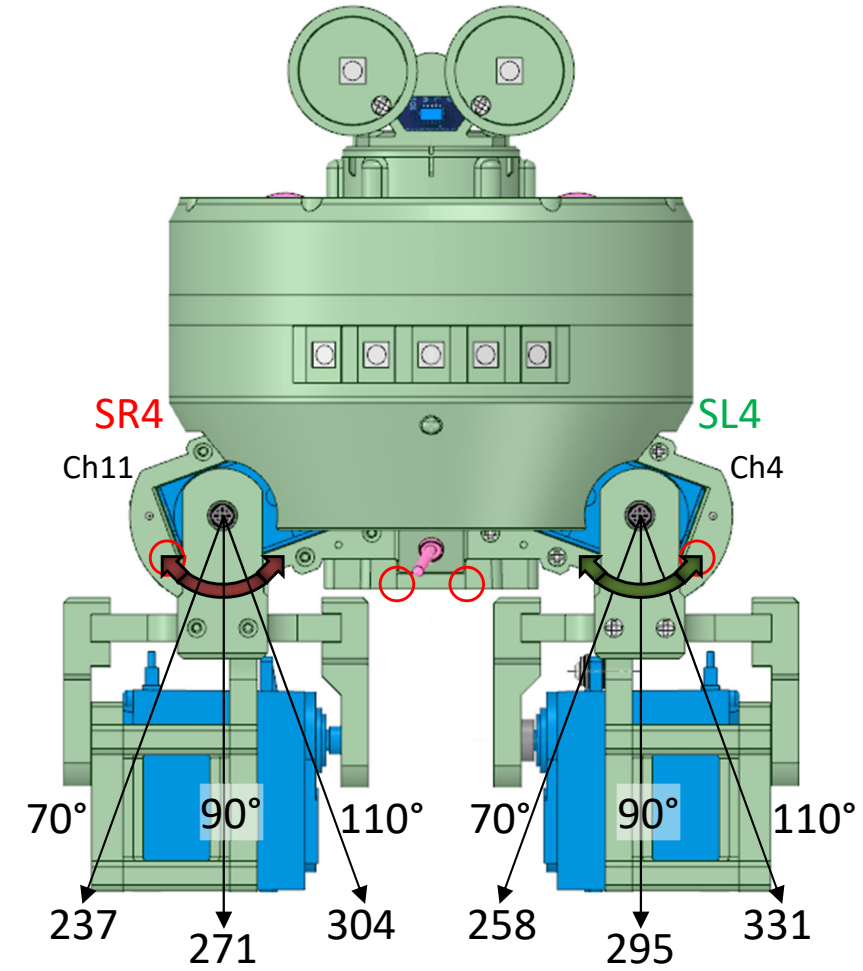
Enable channel 11 only on the 16-Ch controller app and move the corresponding slider left to determine the lower setting. Note that the memory 'M' lamp for channel 11 will change from green to yellow. You should not need to reduce the lower limits value to achieve the 70° point. It should be possible to move the servo beyond this target point if the servo arm was fitted correctly during the build process. If this is not the case the arm will need to be removed and reattached to the servo in a more clockwise position. Move the slider until the leg pivot arm just obscures the lower servo mounting screw (see figure B showing the left-hand leg) With the leg in this position record that value and click in the lower limit text field of the 16-Ch controller to set this channel 11 lower limit to the current slider PWM position.

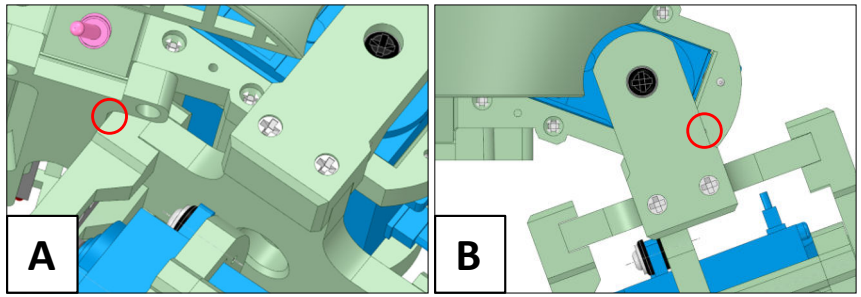
Now move the channel 11 slider to the right cautiously, until the inner side of the leg just touches the lower body (see figure A showing the left-hand leg). Ensure that you do not try to go beyond this collision point as you may damage the servo. Record this 110° value and click in the upper limit text field of the 16-Ch controller to set the channel 11 upper limit to the current slider position. Now calculate the 90° point from these to readings as follows:

$$PWM^{90} = PWM^{LL} + (PWM^{UL} – PWMLL)/2$$
$$PWM^{90} = 237 + (304 – 237)/2 = 271$$

Set the slider to this value before holding down the 'M' memory lamp to record the three values. Now turn OFF channel 11, and repeat this exercise for servo SL4 using channel 4 on the PCA9685 driver board.

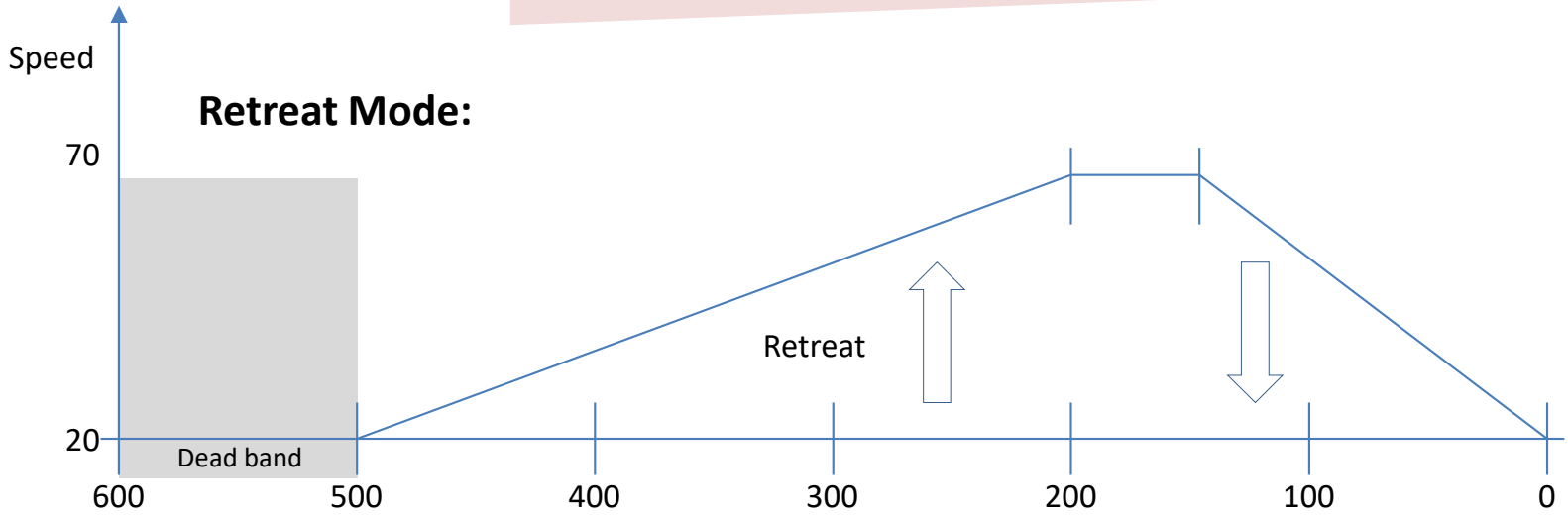
In practise you will be able to swing the legs out beyond the figure determined, but not inward due to the collision point.

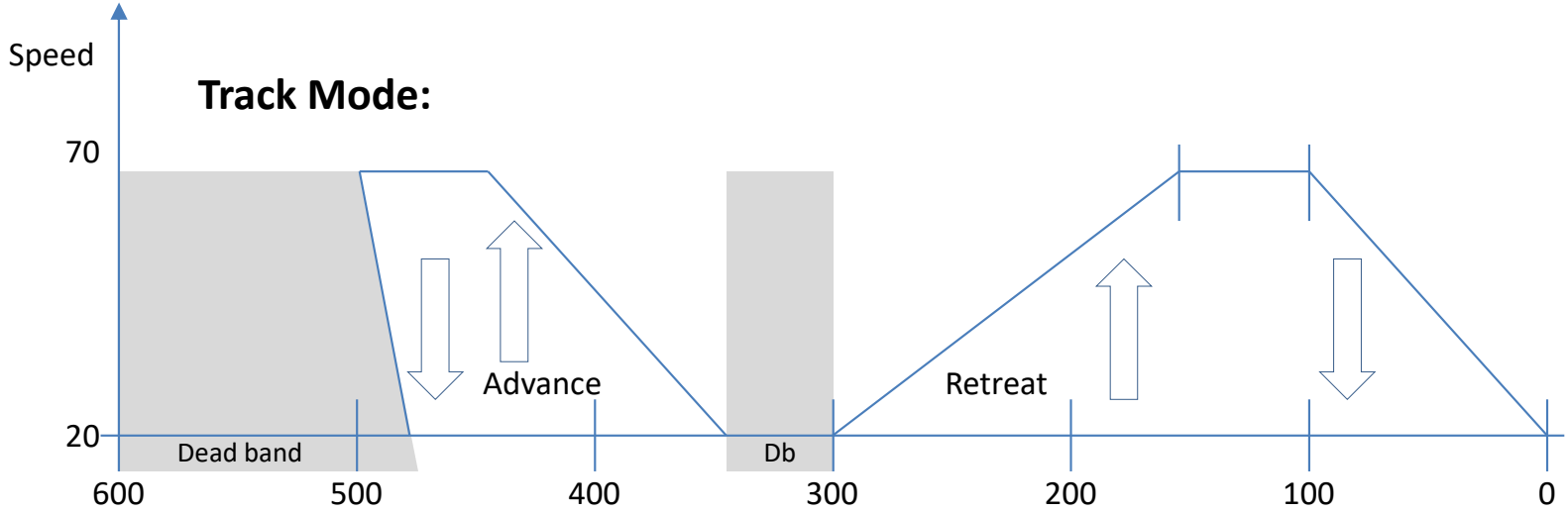


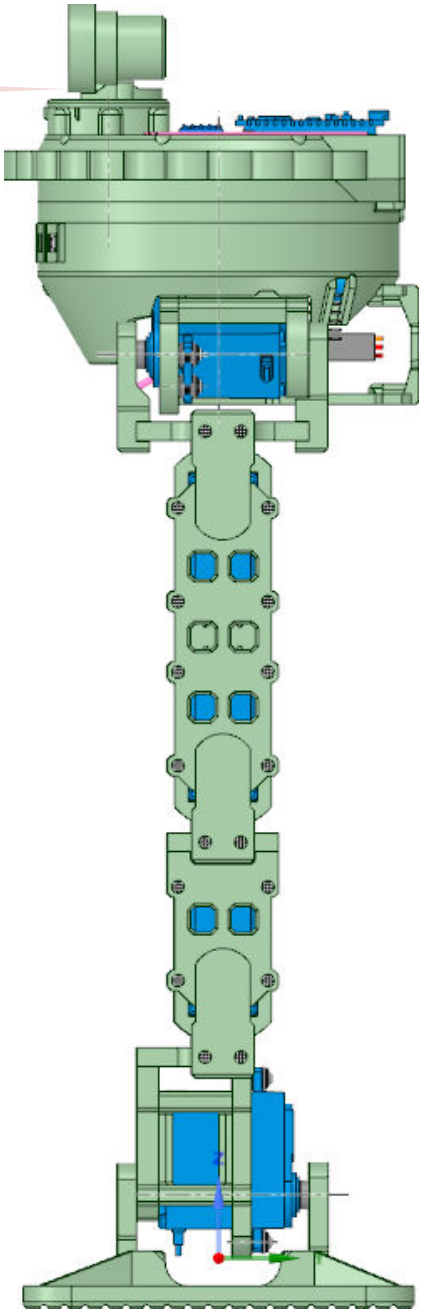


Note: *Hip angle conventions match feet.
* Code limits: 50°/110 and 70°/130° (right / left)

Range finder







Issue: 1.1 Released: 08/12/2025