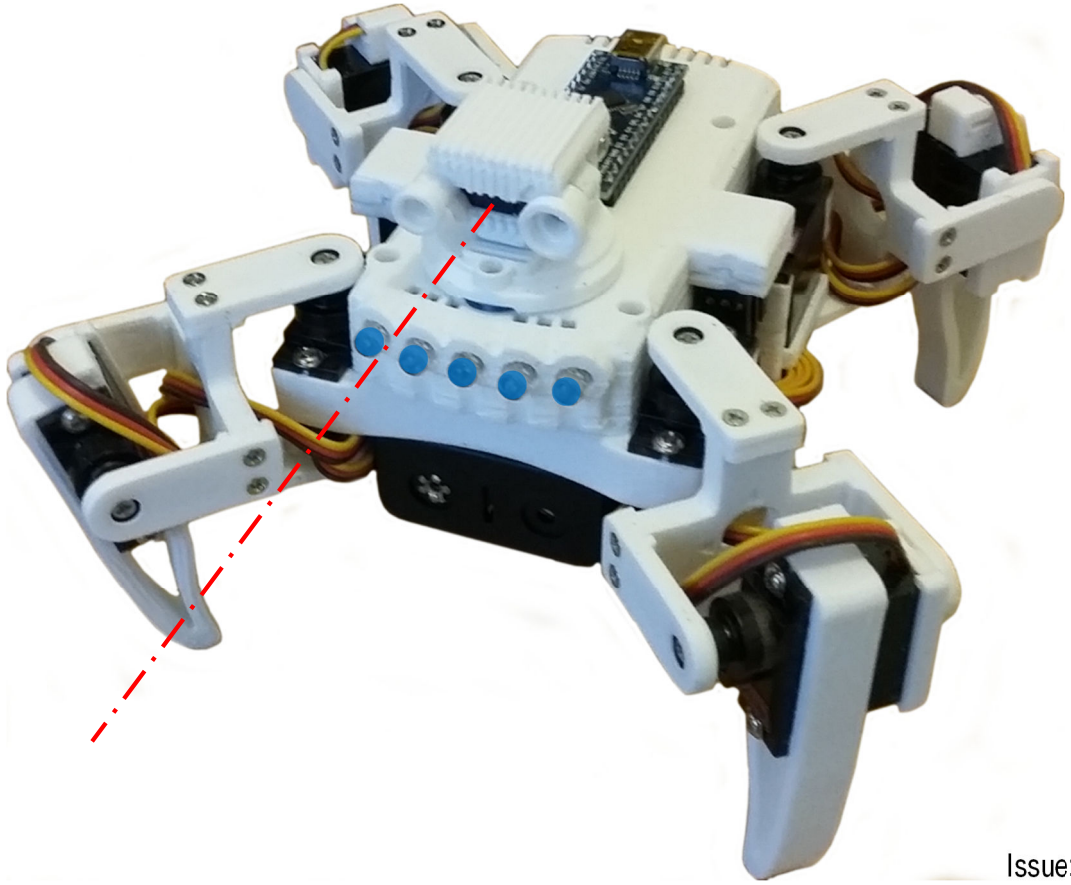
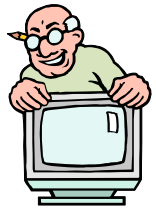
Autonomous Quadruped Robot

Calibration



Issue: 1.1 Released: 09/12/2025 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.

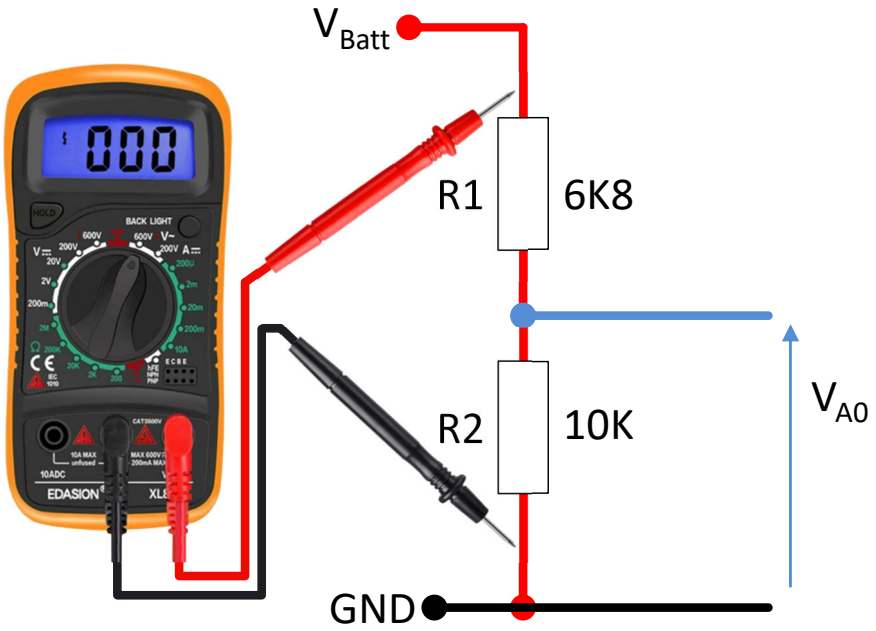
Battery care & maintenance: Stop using a battery if it is swollen, damaged, dented or leaking. Never charge a damaged battery. Never allow a Lithium battery to discharge below 3.2 volts, as cell damage will occur. Avoid extreme temperatures. Do not charge or store batteries in very hot or cold environments. 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.

Quadruped Battery Monitor (Protection)



$$V_{A0} = \frac{V_{Batt} \times R2}{R1 + R2}$$

$$V_{A0} = \frac{V_{Batt} \times 10K}{16K8}$$

$$V_{FSD} = 8.4 \text{ volts}$$

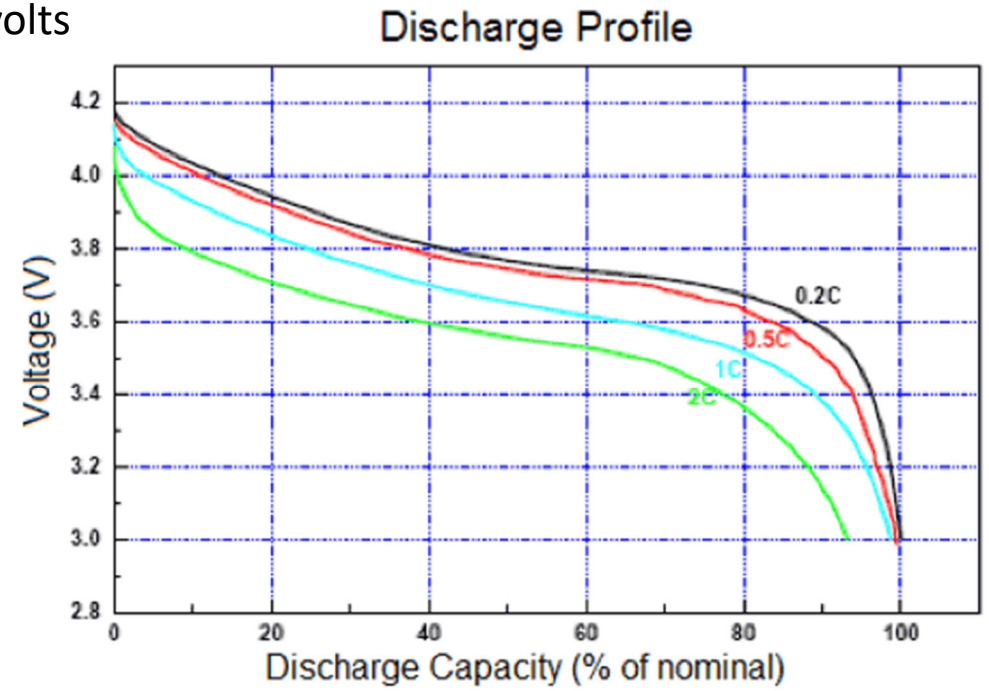
$$V_{A0D} = \frac{V_{A0} \times 1023}{5}$$

$$V_{A0D} = \frac{V_{Batt} \times 0.5952 \times 1023}{5}$$

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

3.70 + 3.00 = 6.70v (one battery fades early)
3.30 + 3.30 = 6.70v (both batteries fade together)

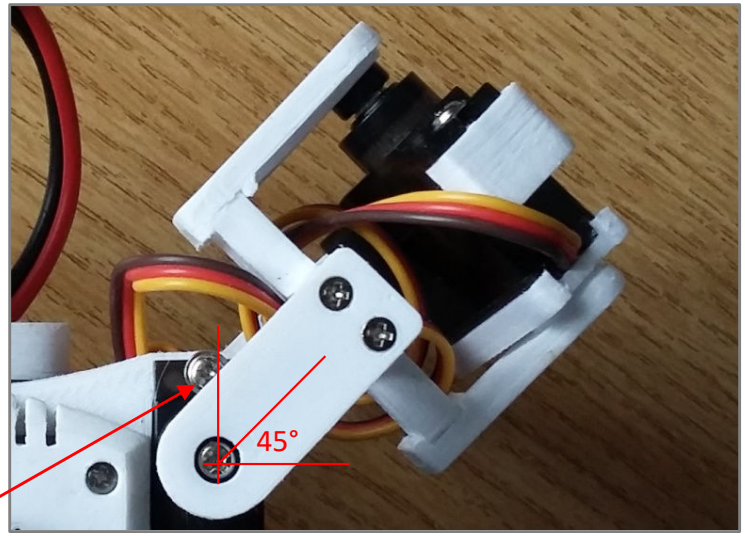
Hence V_{A0D} = 804 @ V_{Batt} = 6.60v



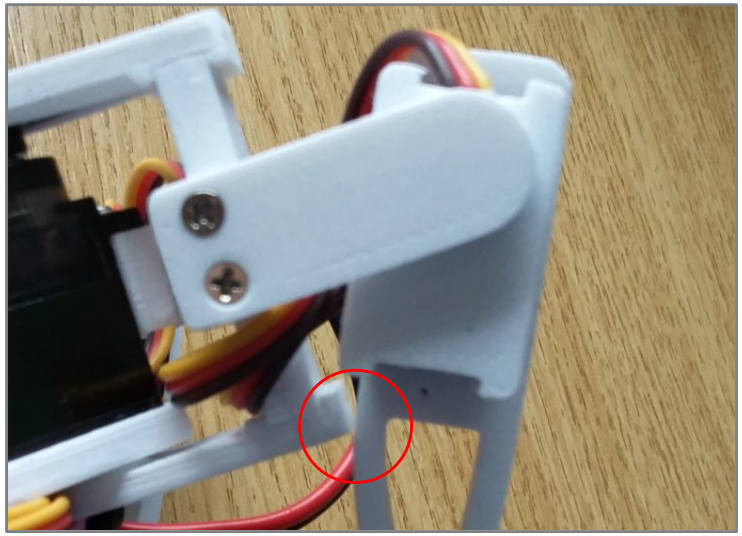
Discharge: 3.0V cutoff at room temperature.

Course Calibration

Use a servo tester when attaching the servos to your robot initially, to set the angles of the attached levers in their approximate positions. The head of the clamping screw is a useful guide.



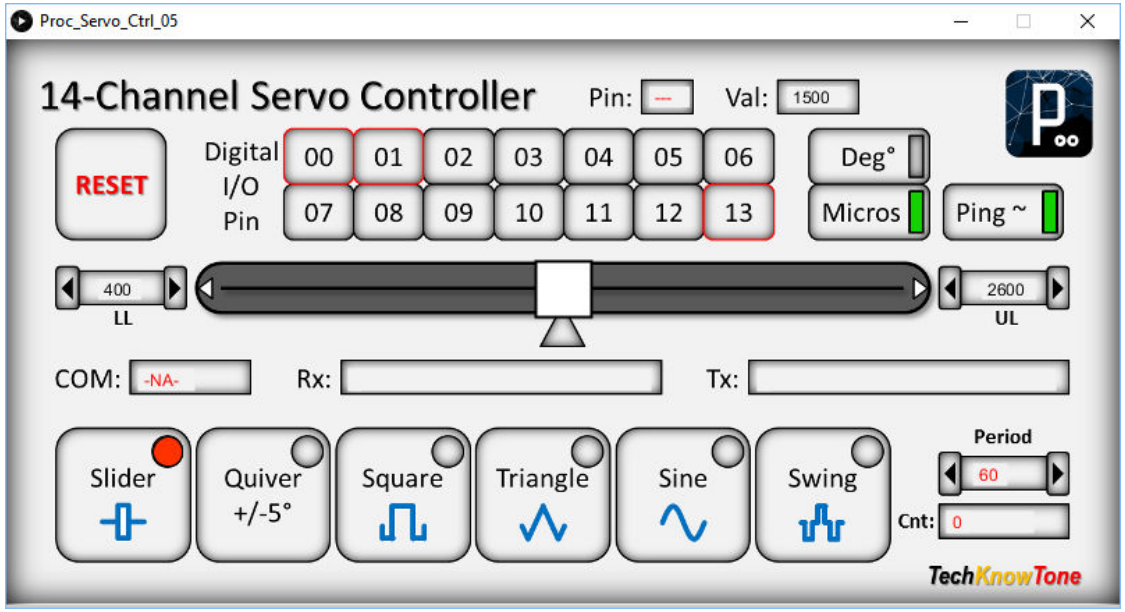
Fit the hip joints at 1500µs = 45°



Turn leg servos with their leavers to their mechanical stops, then back off by 5-10 ° . Fit legs just touching their collision points.

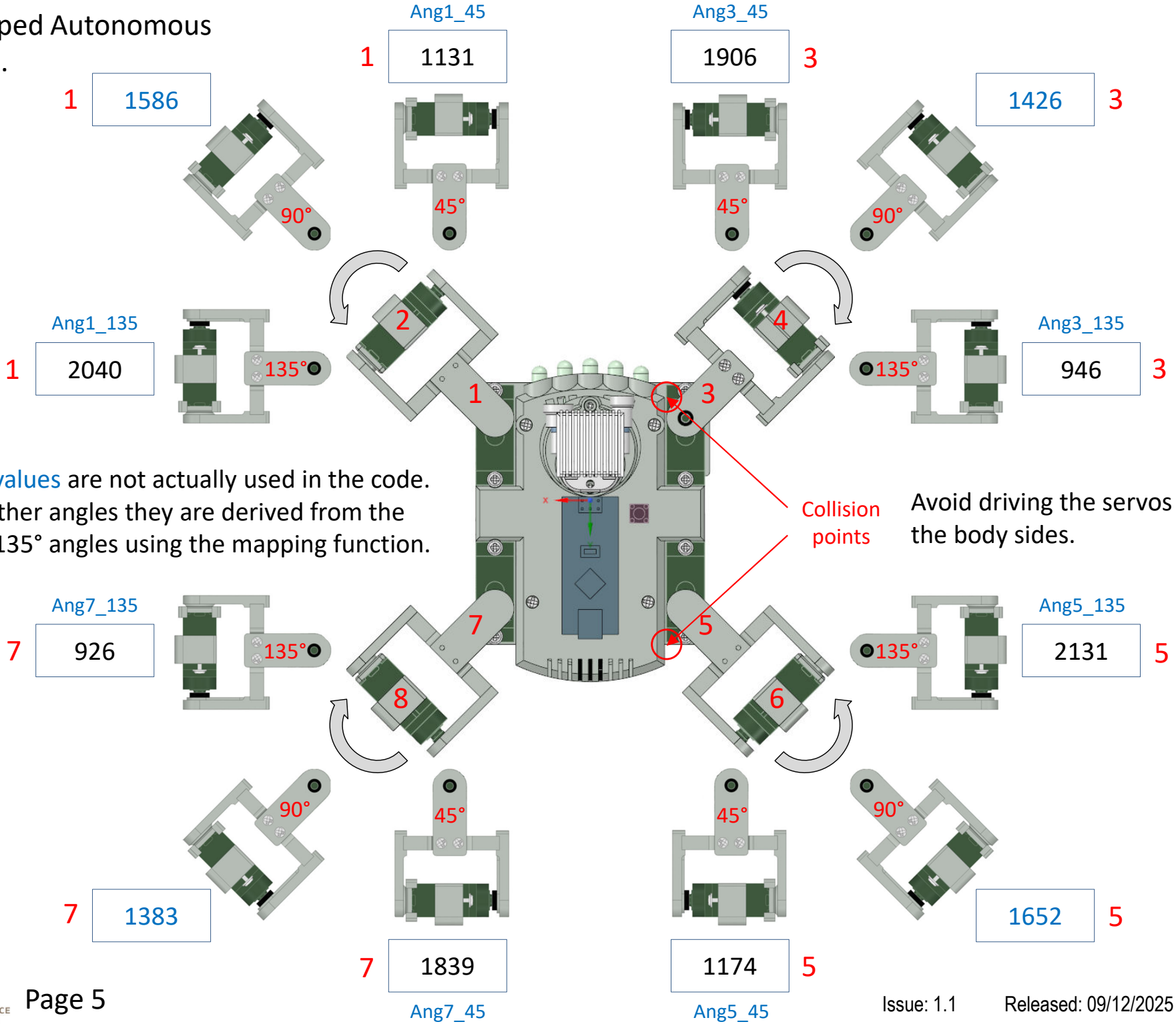
Once you have assembled and wired up your robot you can use the application I have provided. This works with special code you need to install temporarily on the NANO. You can select each servo in turn and move the slider to determine values needed to achieve the calibration angles shown in the subsequent pages of this guide.

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!



Issue: 1.1 Released: 09/12/2025

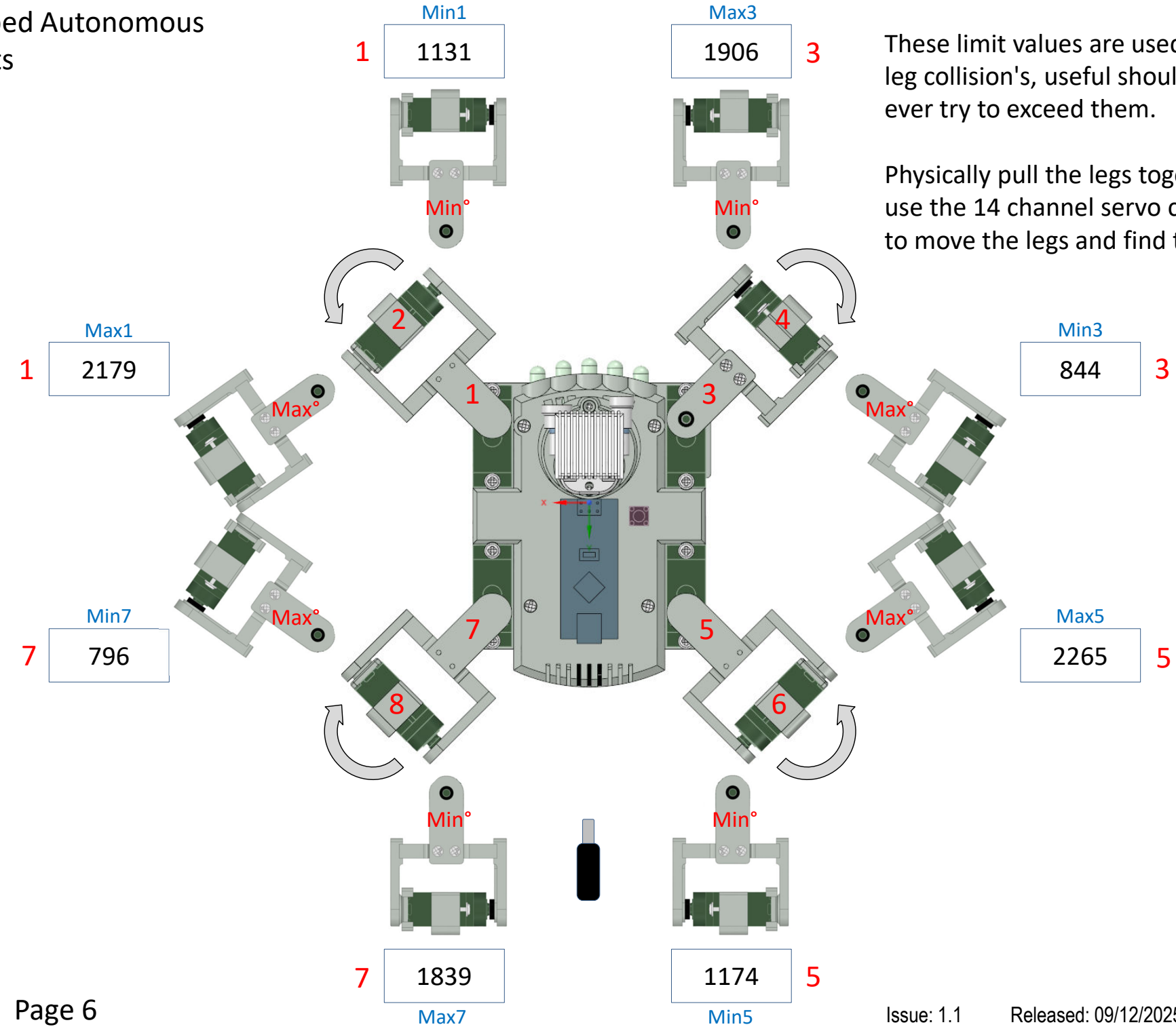
Quadruped Autonomous Hips Cal.



The 90° values are not actually used in the code. Like all other angles they are derived from the 45° and 135° angles using the mapping function.

Issue: 1.1 Released: 09/12/2025 TechKnowTone

Quadruped Autonomous Hip Limits

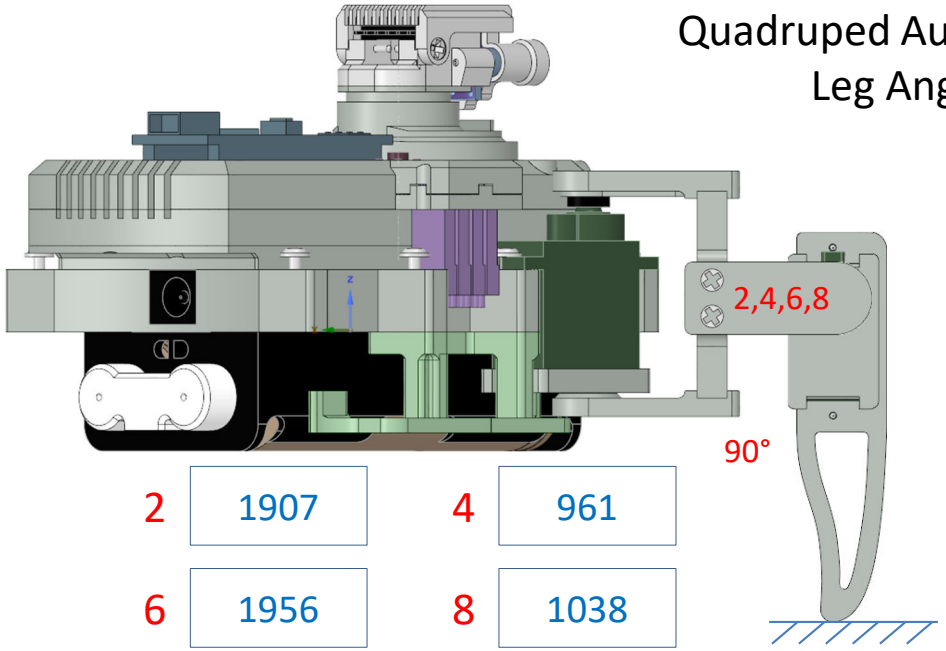


These limit values are used to prevent leg collision's, useful should your coded ever try to exceed them.

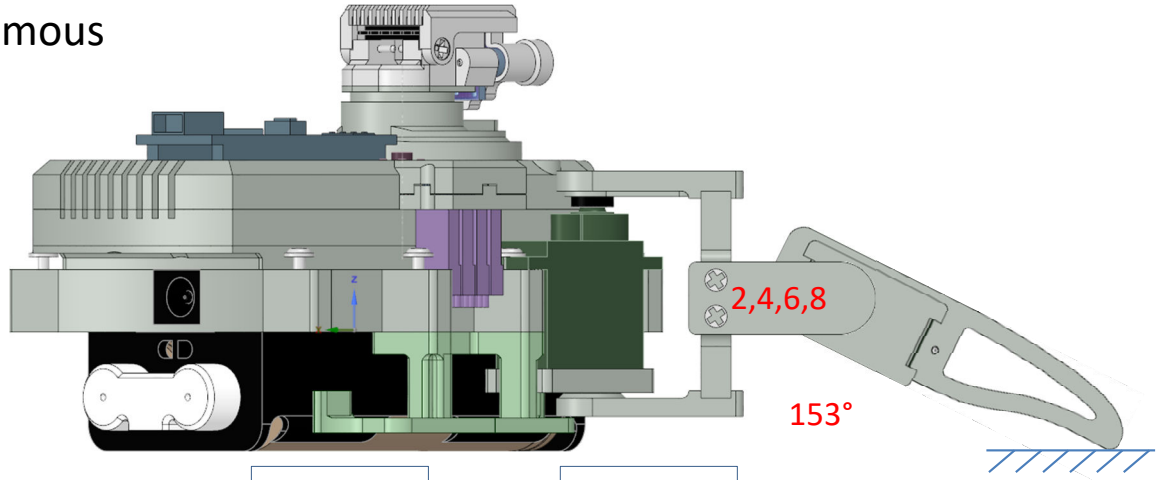
Physically pull the legs together, then use the 14 channel servo controller app to move the legs and find these values.

Issue: 1.1 Released: 09/12/2025 TechKnowTone

Quadruped Autonomous
Leg Angles

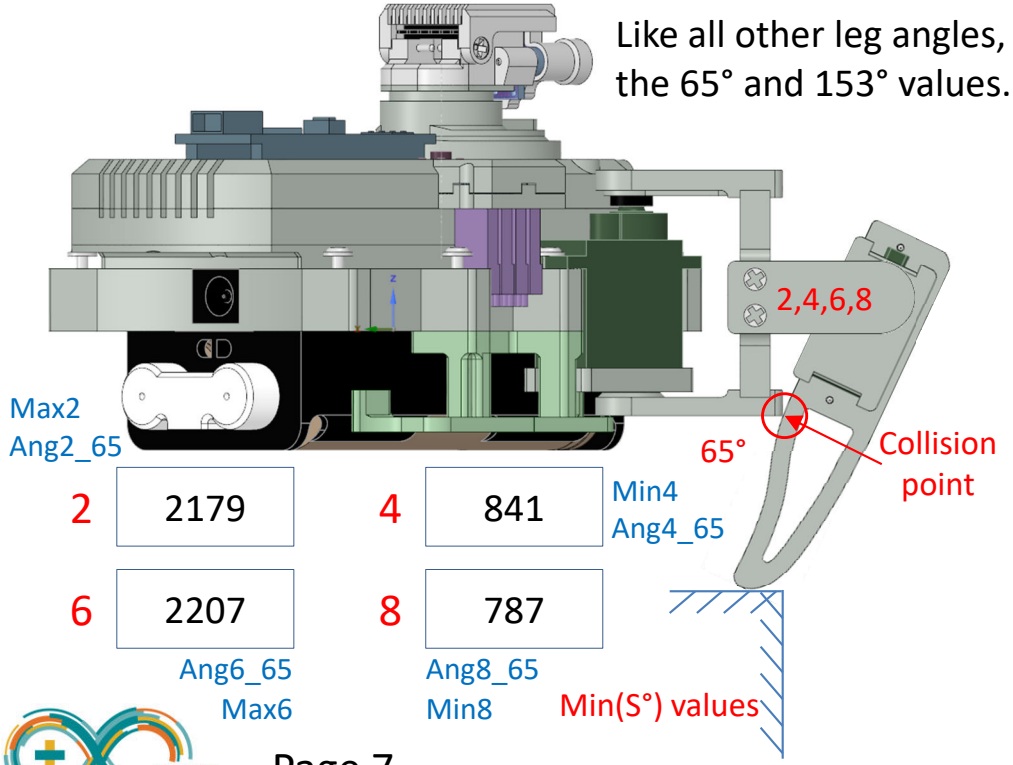


2	1907	4	961	
6	1956	8	1038	

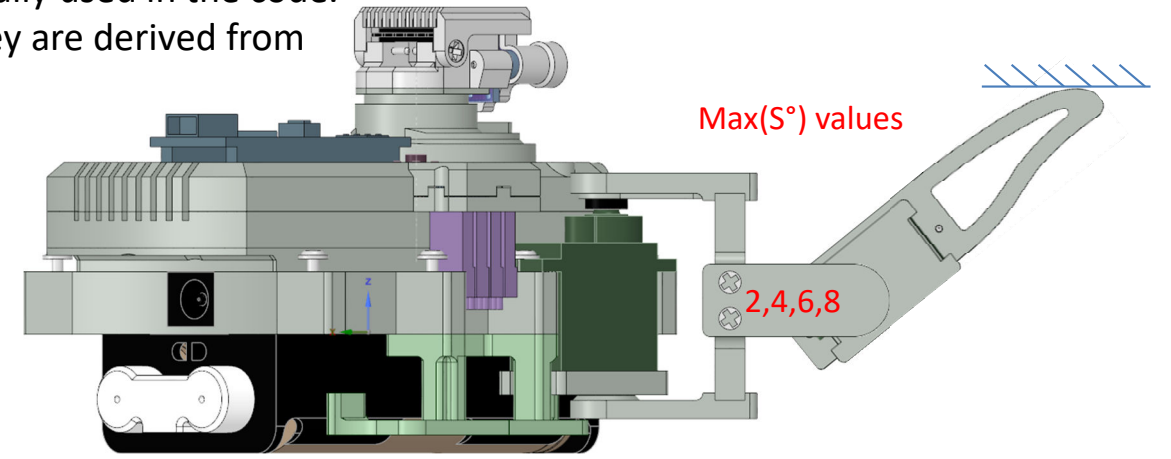


Ang2_153	2	1222	4	1736	Ang4_153
Ang6_153	6	1323	8	1672	Ang8_153

The 90° values are not actually used in the code.
Like all other leg angles, they are derived from the 65° and 153° values.



2	2179	4	841		
6	2207	8	787		



Min2	2	701	4	2284	Max4
Min6	6	696	8	2303	Max8

Quadruped Auton
Head Limits

9

Head_0
1155

0°

These limit values are used to prevent leg collision's, useful should your coded ever try to exceed them.

Physically pull the legs together, then use the 14 channel servo controller app to move the legs and find these values.

Head_65N
690

-65°

Head_65P
1651

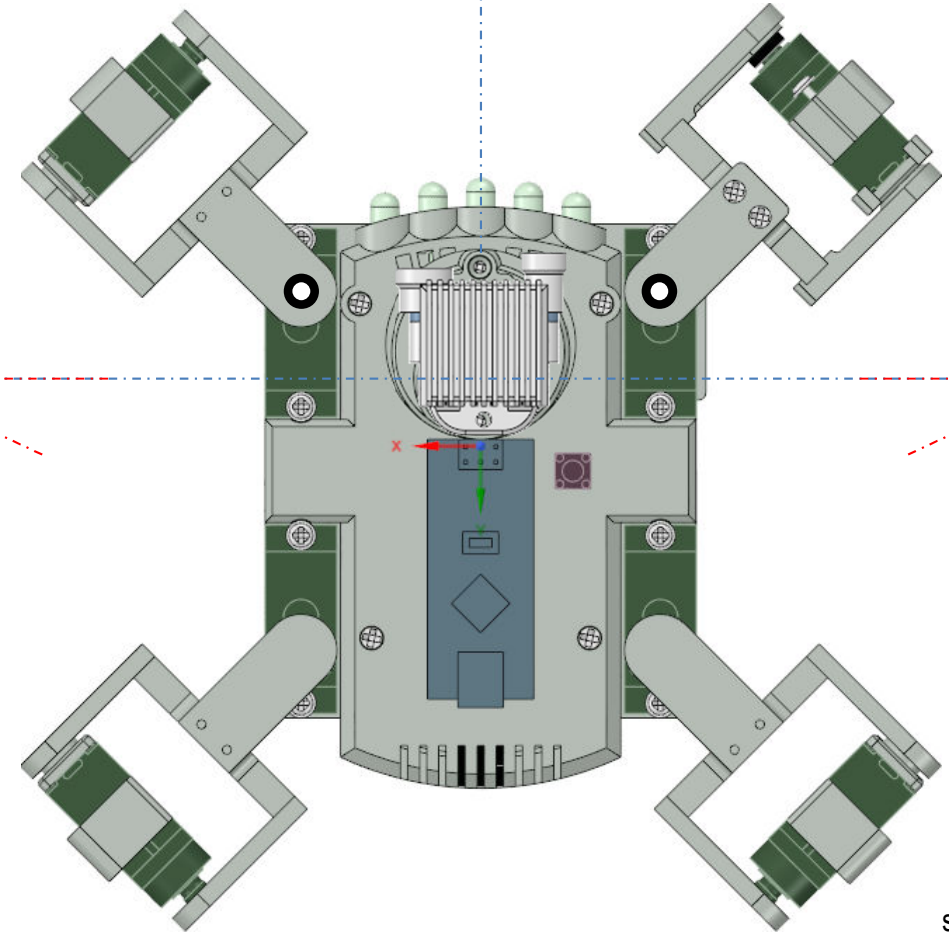
+65°

-90°

+90°

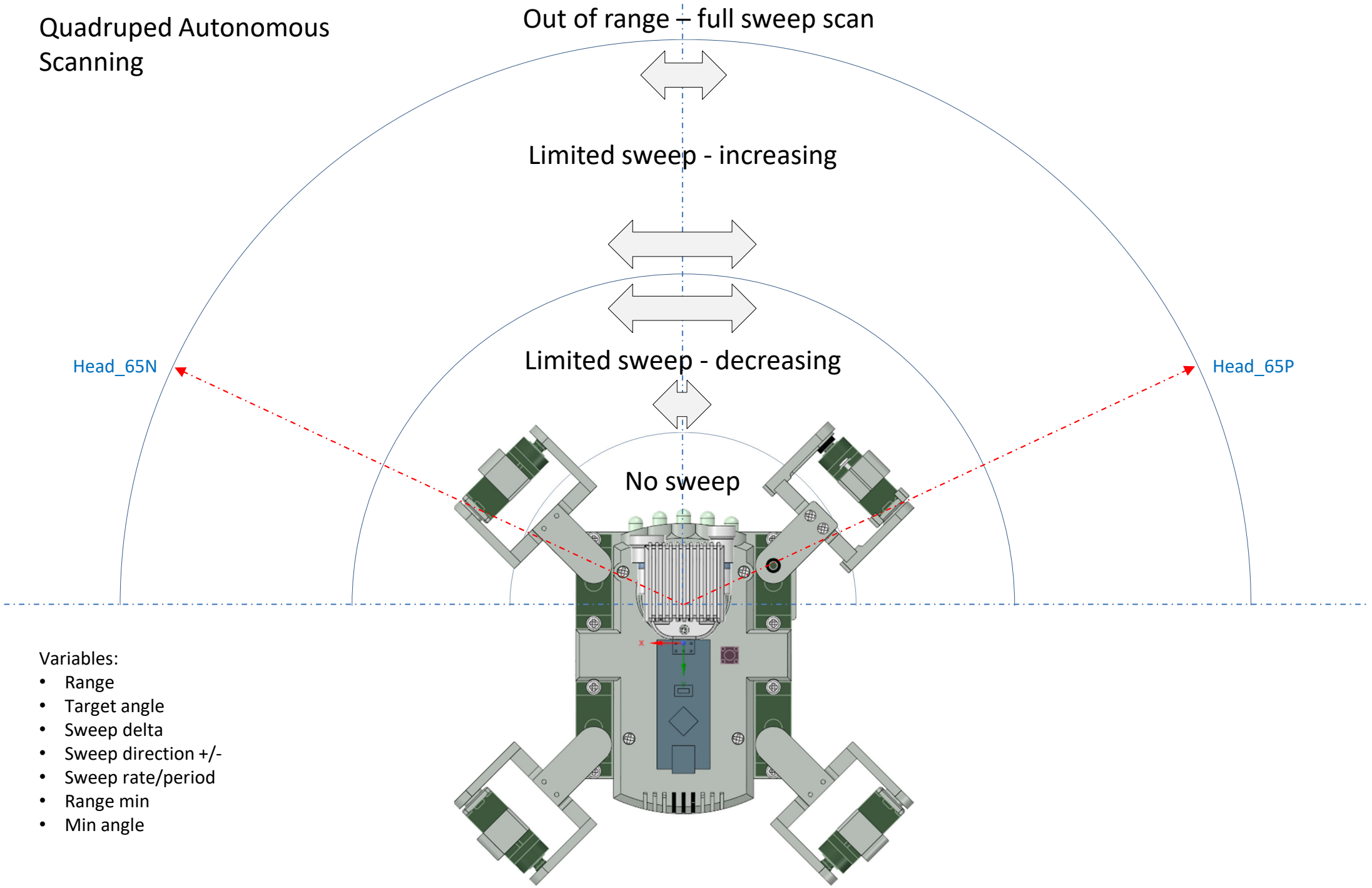
Head_90N
520

Head_90P
1849



sue: 1.1 Released: 09/12/2025

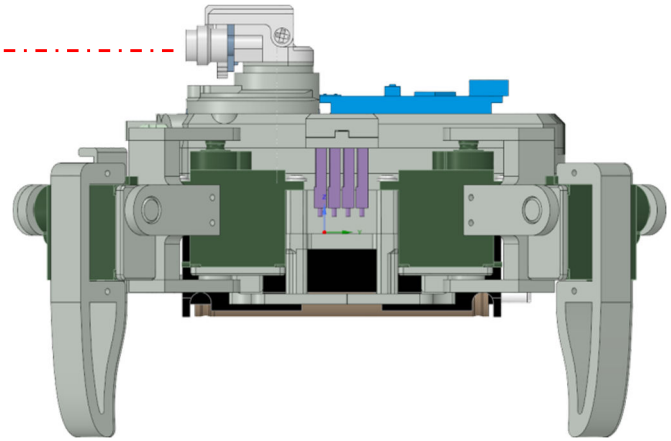
Quadruped Autonomous Scanning

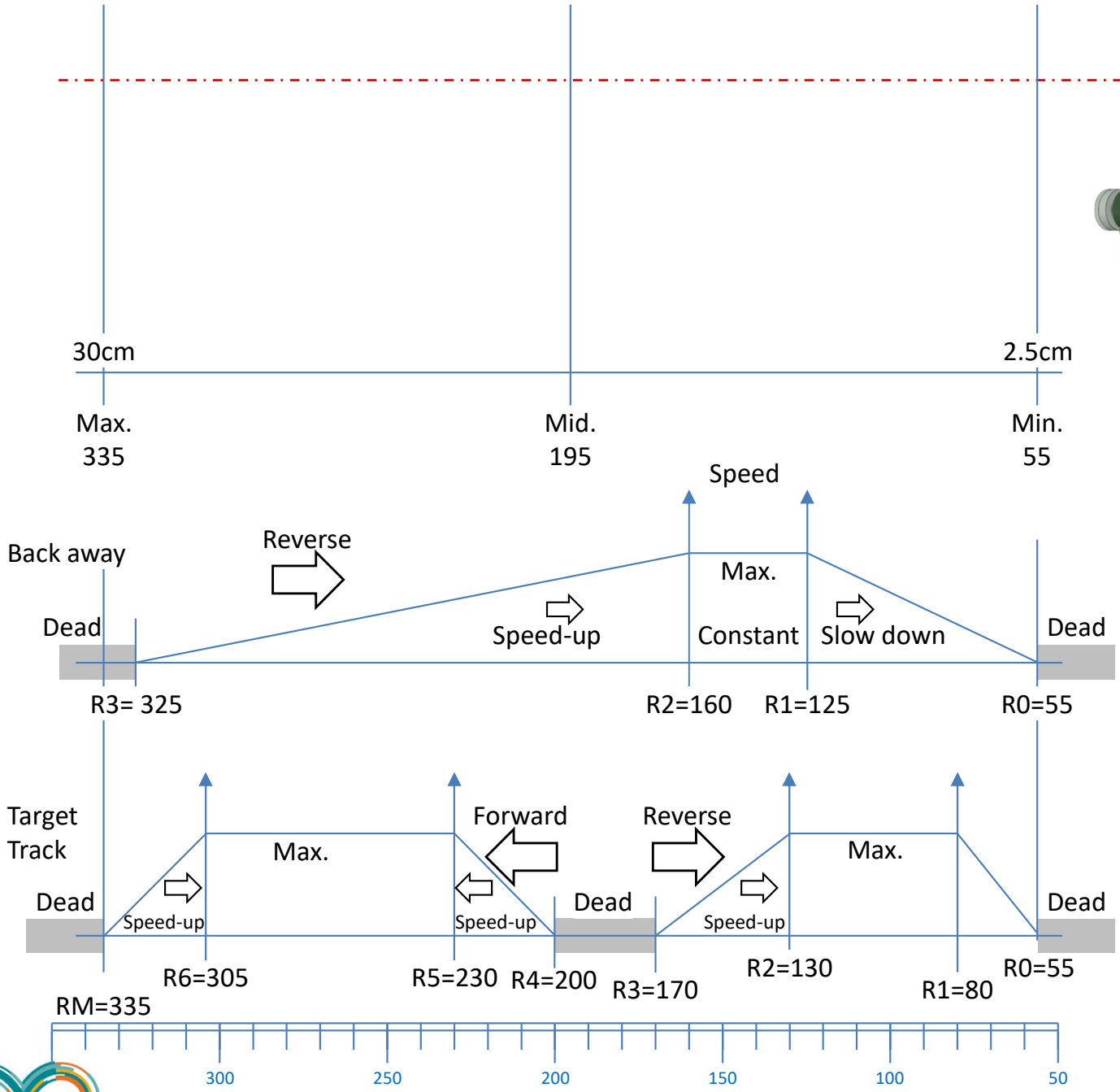


Variables:

- Range
- Target angle
- Sweep delta
- Sweep direction +/-
- Sweep rate/period
- Range min
- Min angle

LTOF Rotating Sensor

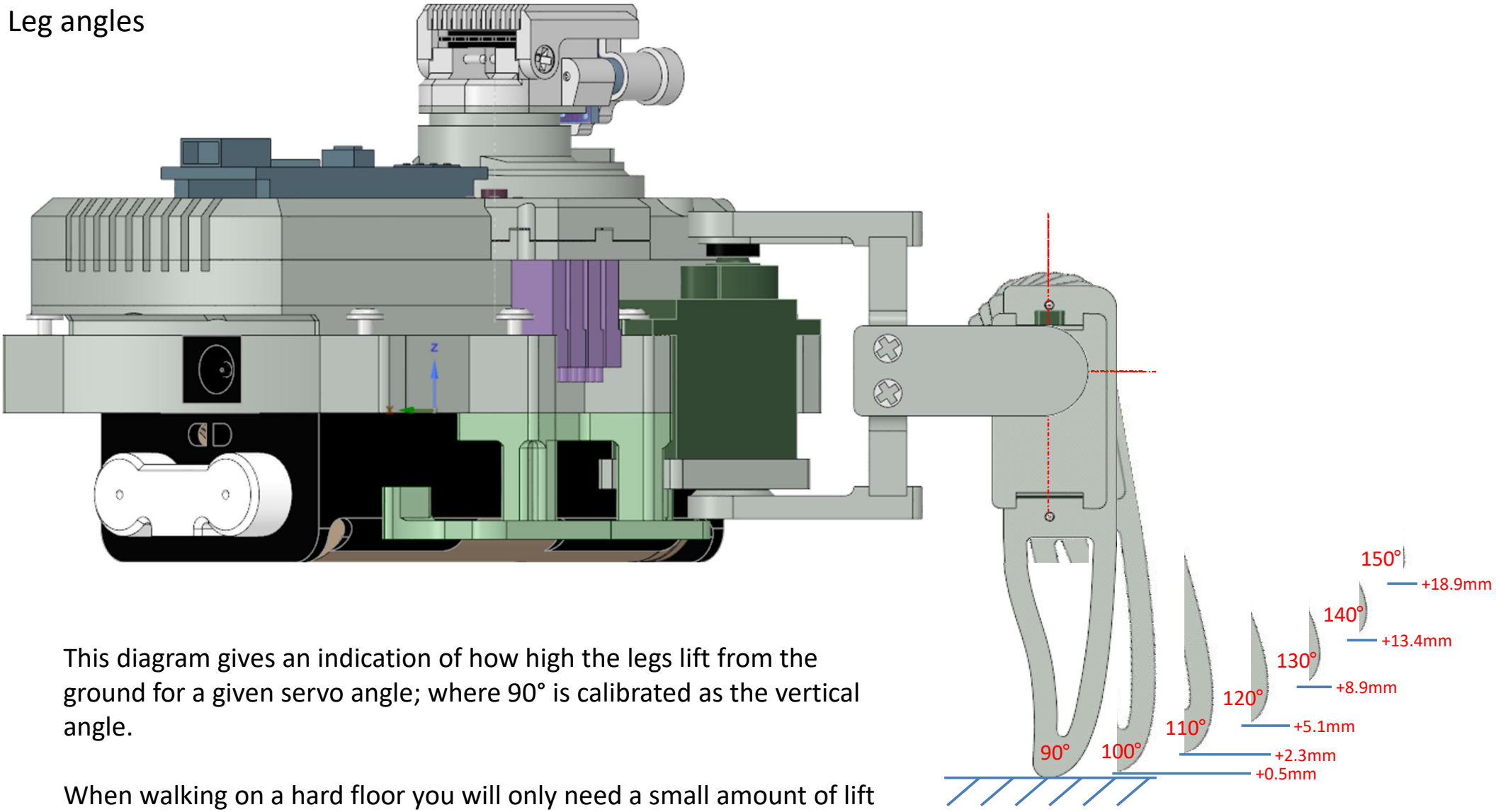




These are the values I determined from my LTOF sensor, and the speed maps I developed for the back away and target tracking functions.

You should be able to find these values in the code and if necessary substitute the values you have determined from your sensor. Use the serial monitor in the IDE along with Serial.print() functions to display your values.

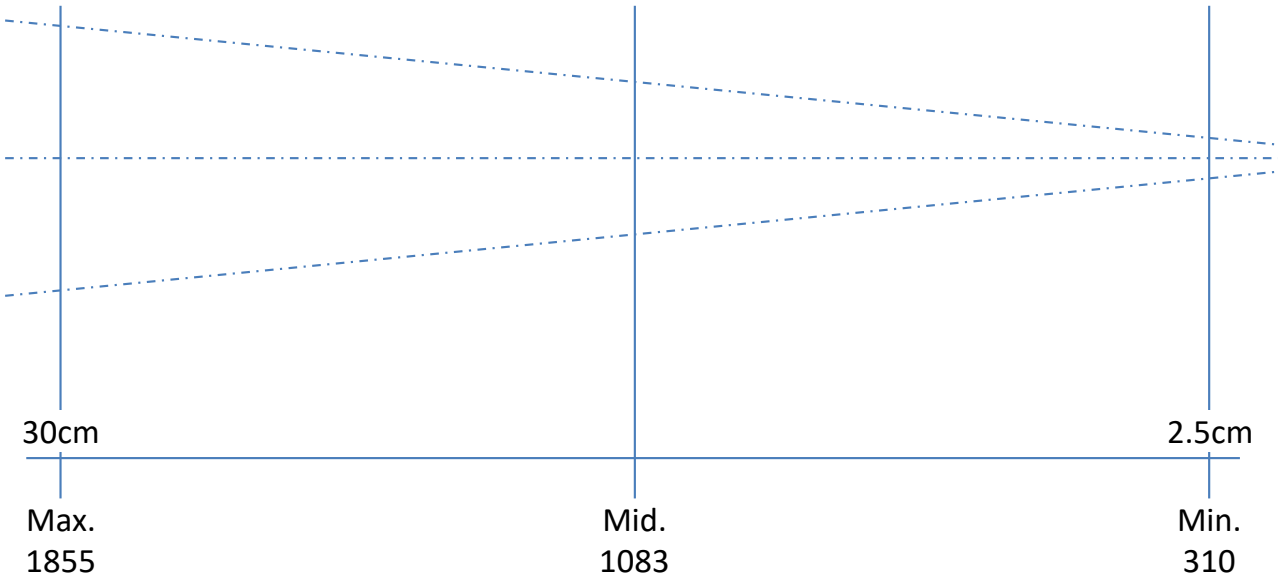
Leg angles

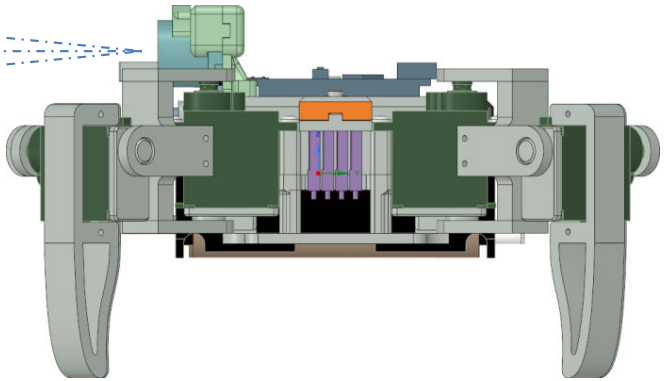


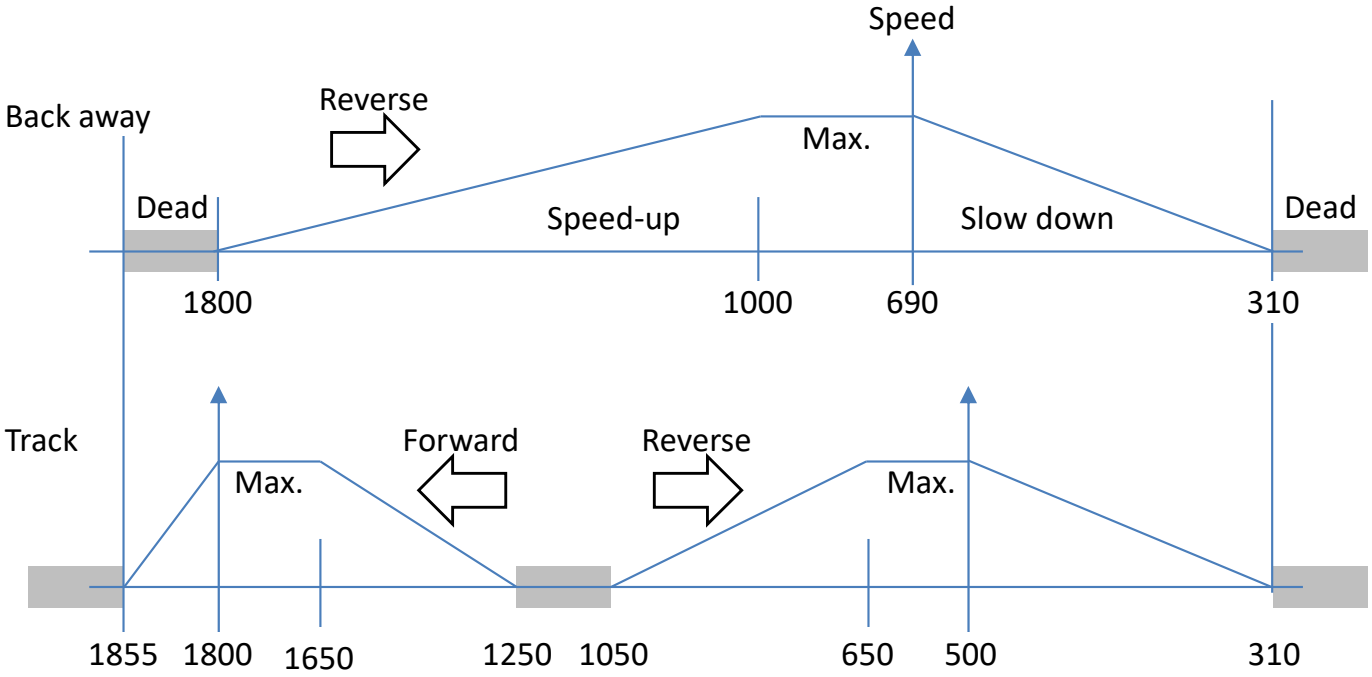
This diagram gives an indication of how high the legs lift from the ground for a given servo angle; where 90° is calibrated as the vertical angle.

When walking on a hard floor you will only need a small amount of lift on the freely moving legs for it to walk; where as in thick piled carpet the robot will sink in and a higher leg lift will be needed to avoid unnecessary drag. You could change the code to use options to set different heights from the infrared controller.

Ultrasonic Sensor







These are the values I determined from my ultrasonic sensor, and the speed maps I developed for the back away and wall tracking functions.

You should be able to find these values in the code and if necessary substitute the values you have determined from your sensor. Use the serial monitor in the IDE along with Serial.print() functions to display your values.