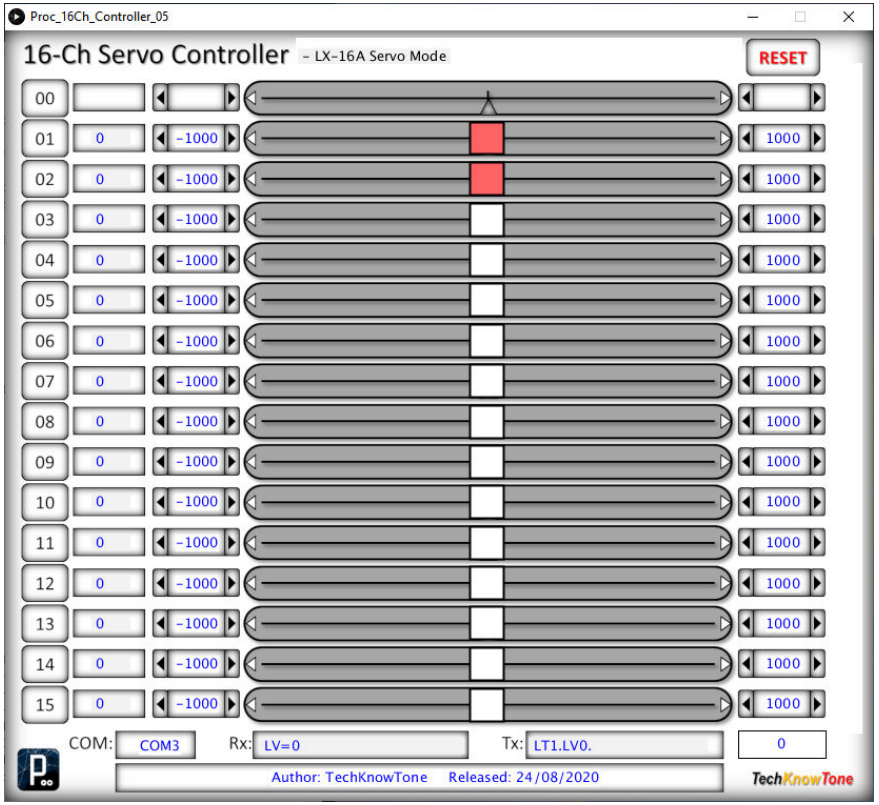
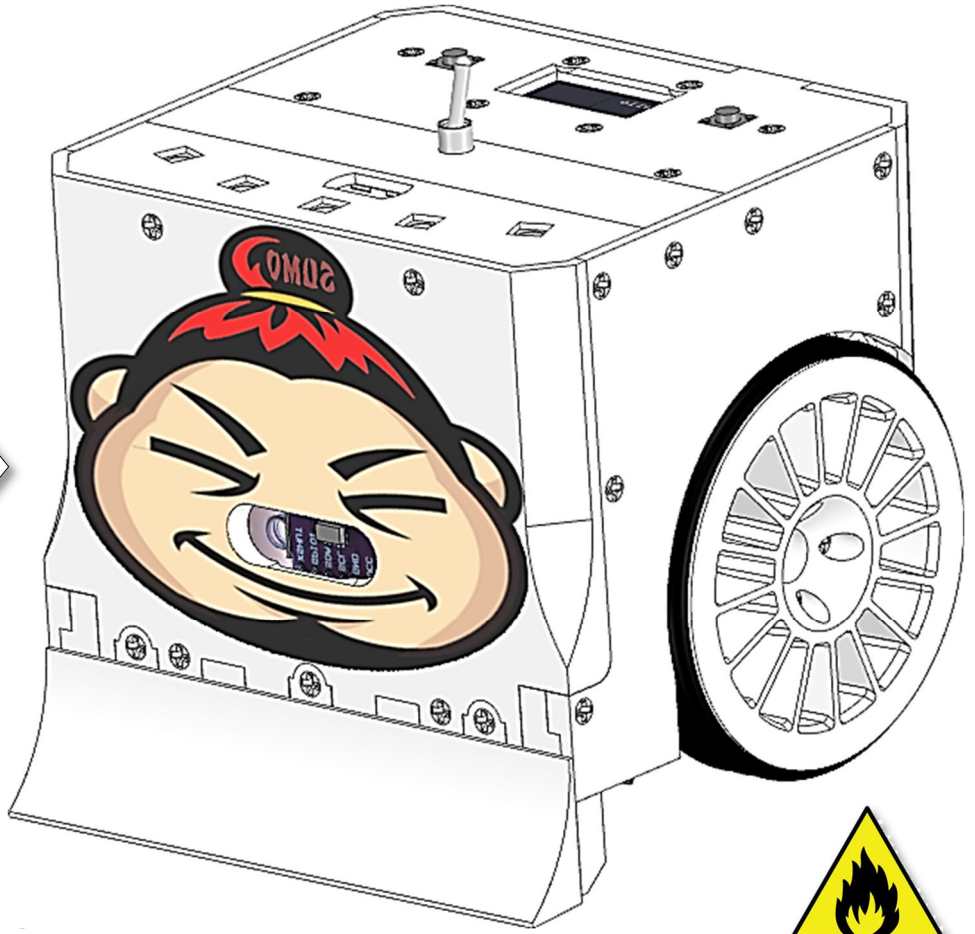
Sumo Robot

Servo Calibration

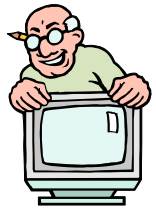




+ESP32

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 Voltage Health Monitoring

See 18650 discharge curve obtained from the internet. In this analysis both batteries are identical and connected in series,
Assume fully charged batteries max voltage is V_{BM} >= 8.2v max
I measured my rechargeable PP3 at 8.65v when connected and ON.
Set battery warning point at V_B = 7.00v
Set battery critical point at V_{BC} = 6.60v

ESP32 is powered from batteries connected to V_{in}.
3.3v at VADC == 4095 on 12-bit converter (4095 max).
If we use a 6k8 resistor feeding A0 and a 3k3 resistor to GND, we get a conversion factor of 10.1v == 4095 or 2.47mV/bit or 404.85
Using a Multimeter I determined the conversion factor needed to be reduced to 372.0 to display voltage correctly.

MAX: V_M = 8.2v, gives A14 = 3048 on ADC (V_M*404.85 * 0.918)

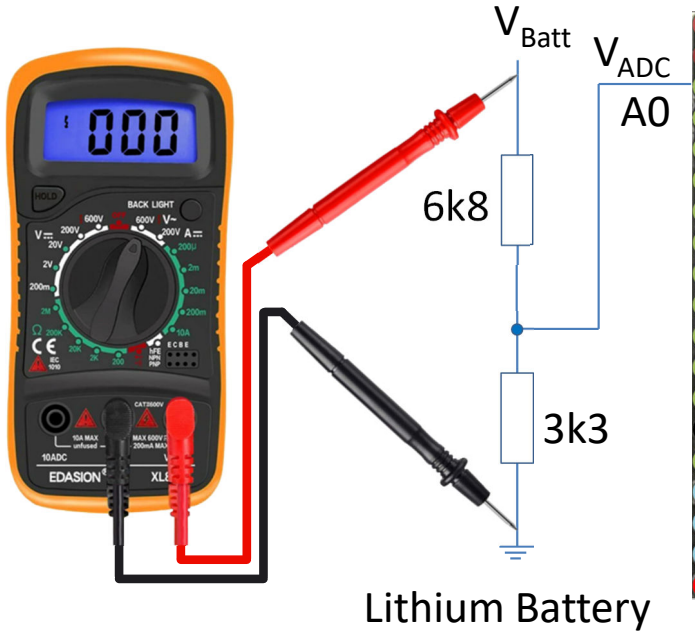
WARNING: V_B = 7.0v, gives A14 = 2602on ADC (V_B* 404.85 * 0.918)

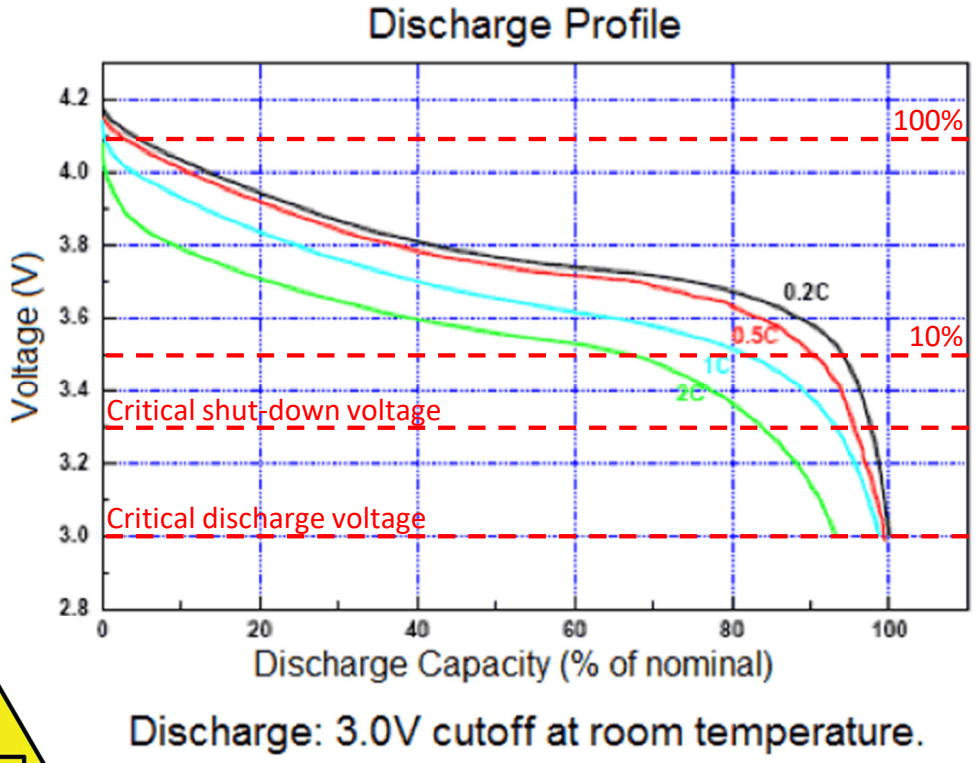
CRITICAL: V_{BC} = 6.6v, gives A14 = 2453 on ADC (V_{BC}* 404.85 * 0.918)

The code will sample the battery voltage on power-up to ensure it is sufficient, then at every 40ms interval, calculating an average (1/20) to remove noise.

Given the relatively light current drawn I have assumed a linear discharge curve ranging from 8.2v (100%) to 6.6v (0%) capacity. The rate of discharge is monitored and used to actively predict the life of the battery in use.

Note: If connected to USB port with internal battery switched OFF the ADC will read a value 5 volts (A14 = 1858) or less. So if the micro starts with such a low reading it knows that it is on USB power.



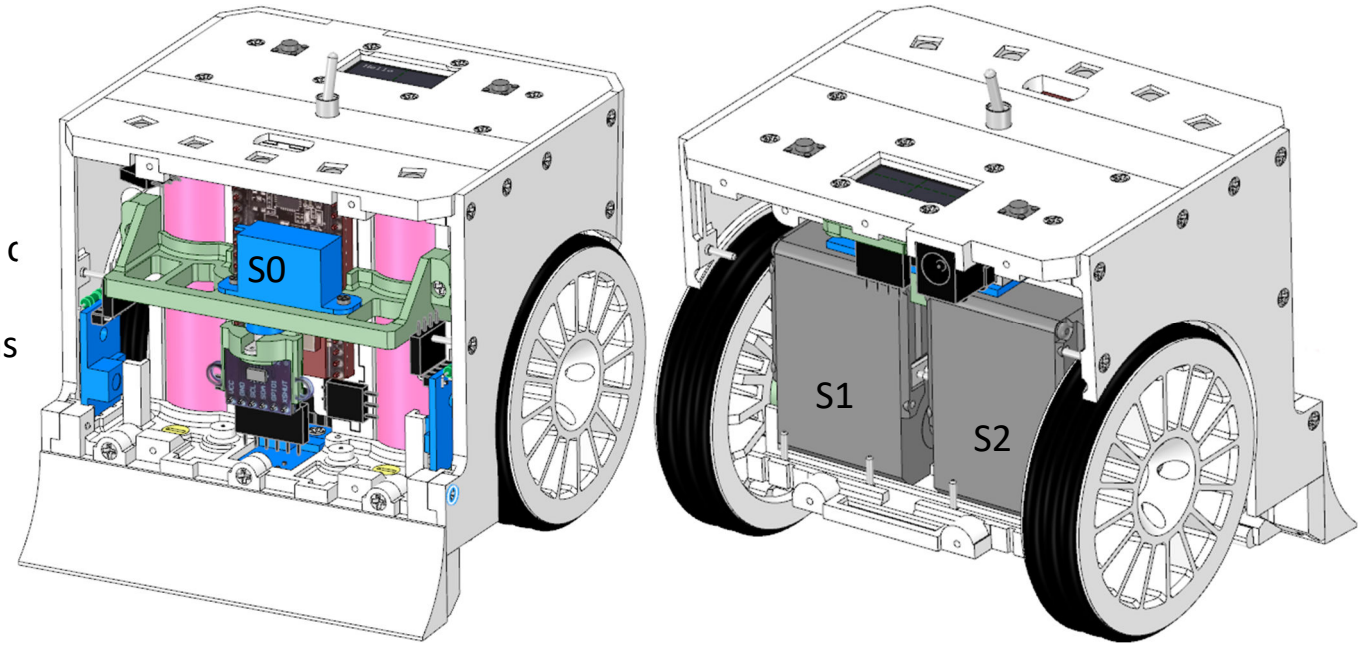


Discharge: 3.0V cutoff at room temperature.

Issue: 1.1 Released: 08/12/2025 TechKnowTone

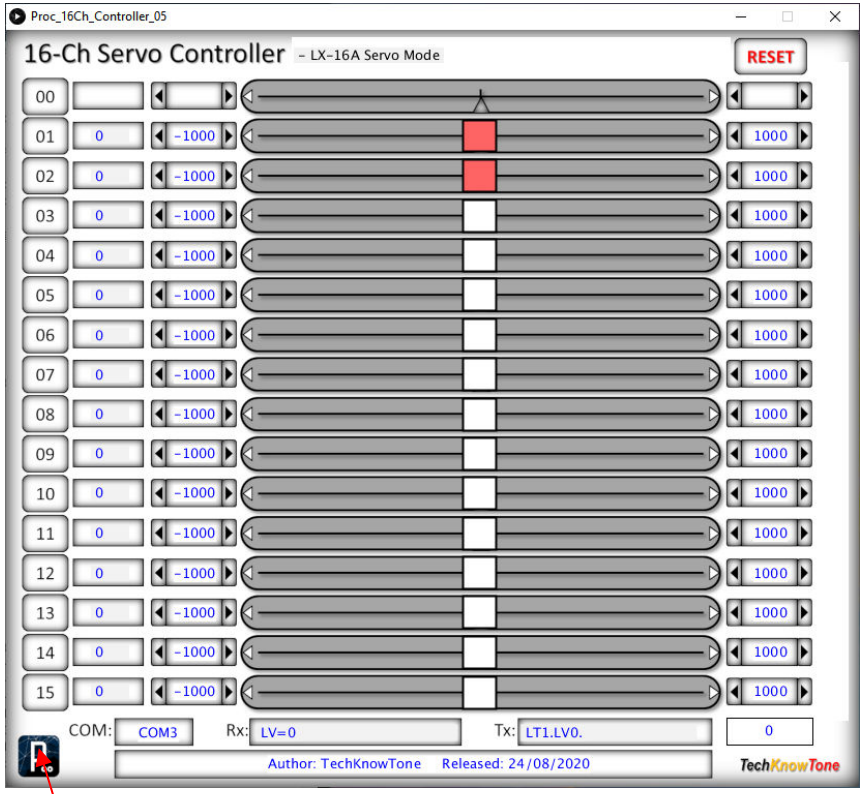
Servo Calibration

The Sumobot has three servo motors, one for each wheel, and one to rotate the laser range finder left and right. The two wheel servos are c type LX-16A, connected to a digital serial bus, and are assigned as S1 and S2. Where as the las pan servo is assigned as S0, and this is a conventional PWM driven servo.



As the ESP32 microcontroller can provide both types of control signals, I decided to develop a custom Windows app to aid with the set up process. The app is capable of controlling up to 16 independent channels, but here we use three.

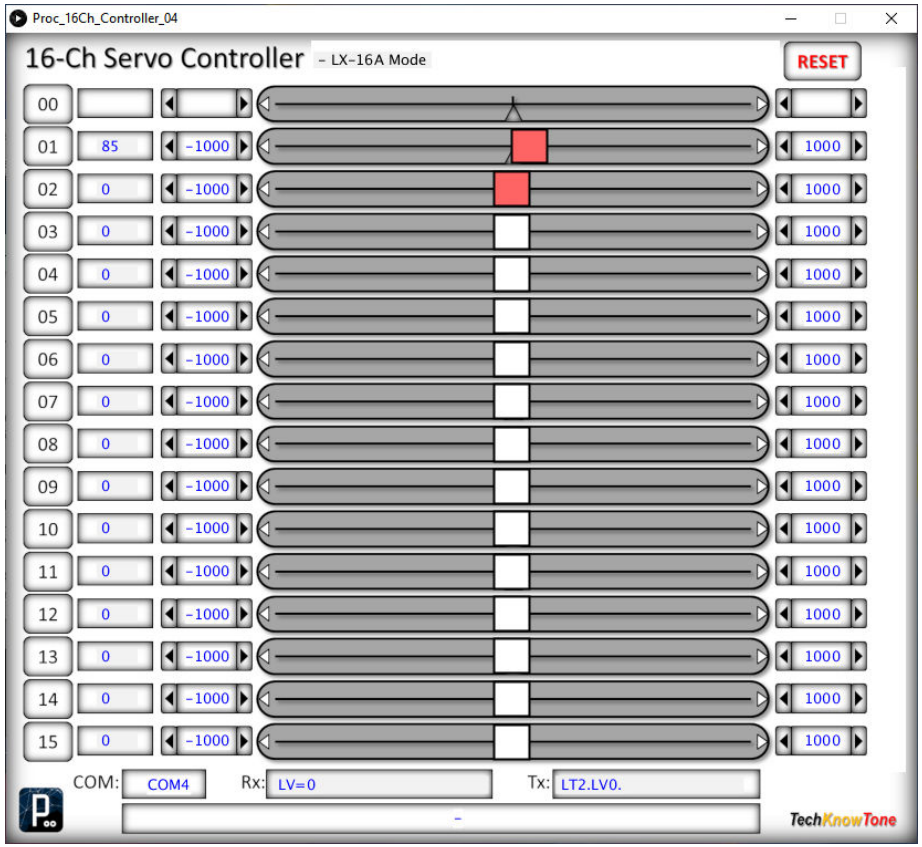
The code in the ESP32 micro responds to commands sent over the serial interface from the app, and the app can be configured to work in different ways. By default the app starts in LX-16A mode, as shown here, where the '00' channel is disabled, as it will be used in PWM mode. So when you click on channels '00' or '01' the respective sliders become active and you can adjust then between -1000 to +1000. Doing this with the Sumobot connected to the serial port should cause the two wheels to turn.



Click to change app mode

The app can be switched to PWM mode, by clicking on the Processing logo in the bottom left corner. This enables channel 00 for servo S0 and changes the slider range 600 to 2400 µs. Use this to adjust S0 and enter calibration values into your code, for centre and left/right laser cut-off points.

LX-16A Motor Calibration



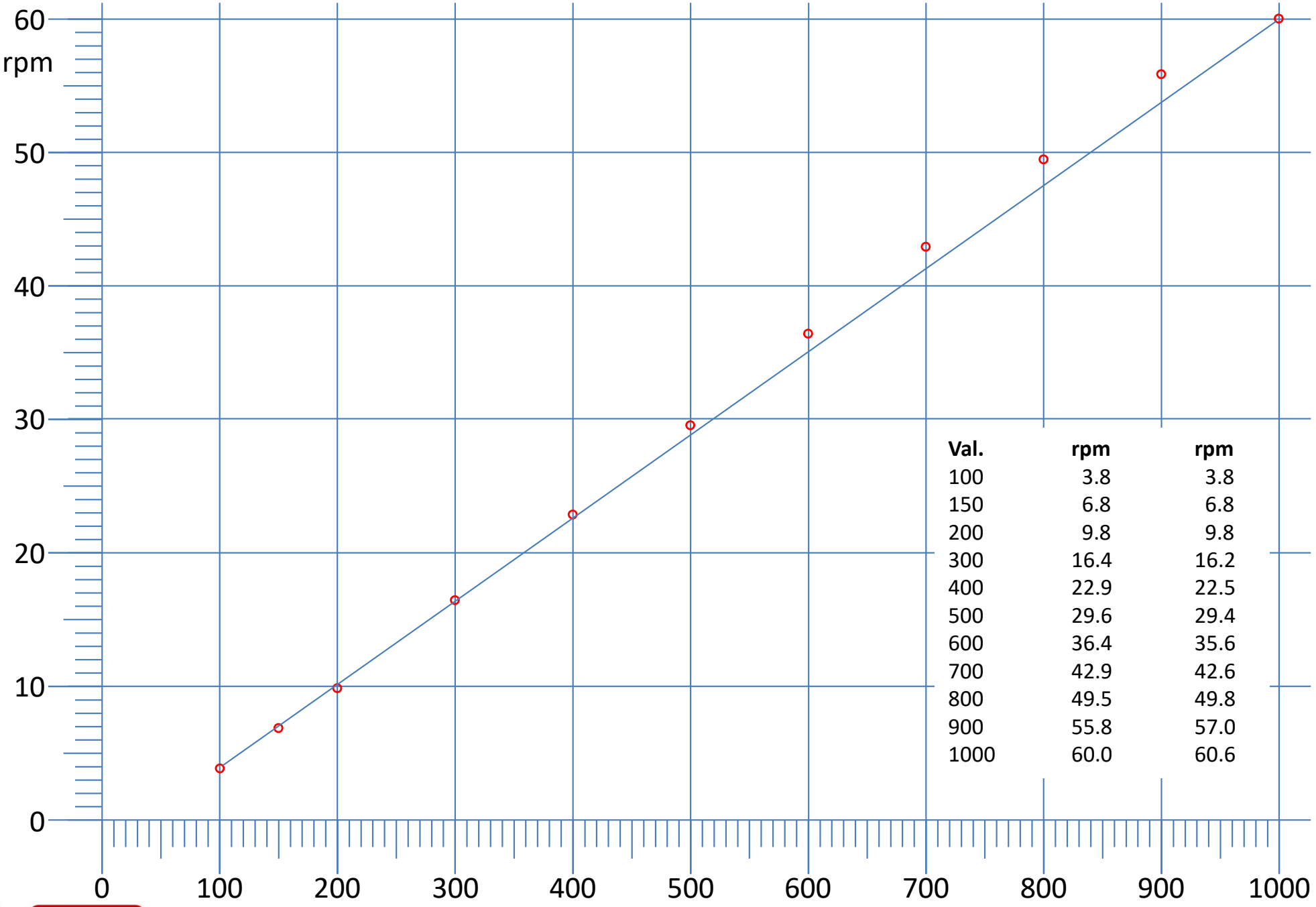
	Sumo1		Sumo2	
	Left	Right	Left	Right
Start Fwd	90	-100		
Rev	- 90	- 90		
Val.	rpm	rpm		
100	3.8	3.8		
150	6.8	6.8		
200	9.8	9.8		
300	16.4	16.2		
400	22.9	22.5		
500	29.6	29.4		
600	36.4	35.6		
700	42.9	42.6		
800	49.5	49.8		
900	55.8	57.0		
1000	60.0	60.6		

Working range

Note: from the measurements taken we can conclude that the servos are design calibrated to 1000 : 60 rpm, or 1 revolution per second. With a wheel diameter of 62mm one revolution covers 194.8mm. This is the distance travelled per second at 1000 demand. Equivalent to the robot travelling 0.1948mm in 1 millisecond.

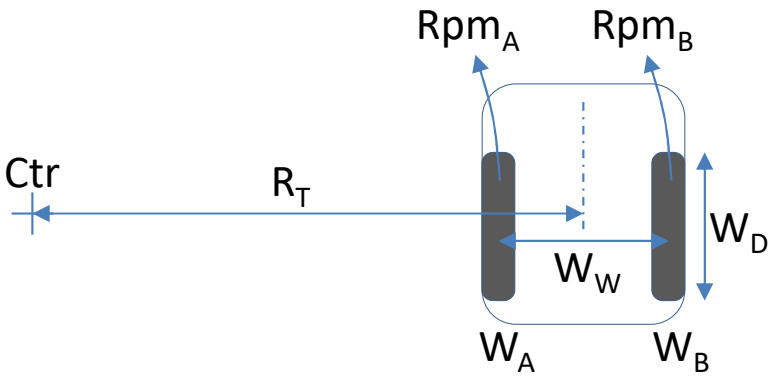
At a speed of 6.8 rpm it will travel 1324.64mm in one minute, which is 22.077mm per second, or .022077mm in 1 millisecond. With this data we can use a simple mapping function to determine the distance travelled at a particular speed, and sub-divide that by the sampling rate, say 25 Hz (40ms) to give incremental distance travelled. Or we can integrate the time between speed changes, which is more appropriate to this application.

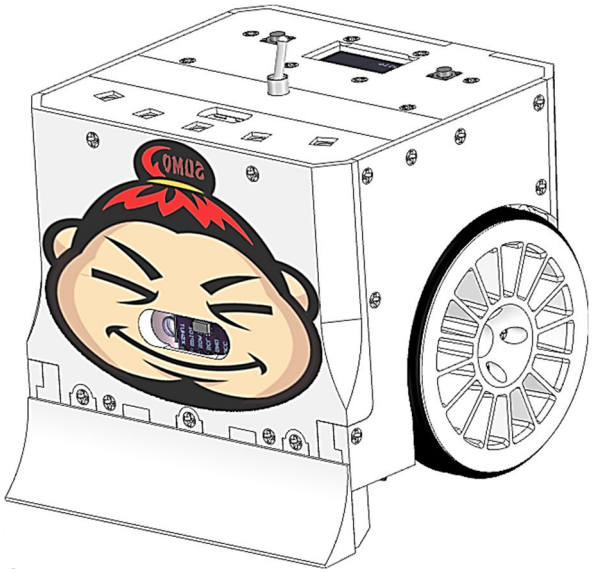
LX-16A Motor Calibration



Val.	rpm	rpm
100	3.8	3.8
150	6.8	6.8
200	9.8	9.8
300	16.4	16.2
400	22.9	22.5
500	29.6	29.4
600	36.4	35.6
700	42.9	42.6
800	49.5	49.8
900	55.8	57.0
1000	60.0	60.6

Sumo Robot Turning radius





A two wheeled robot can be steered by adjusting the relative speeds of its wheels. If, for example, the right-hand wheel is driven faster than the left-hand wheel, then the robot turns towards the left in a circular motion, following a path where the radius of travel is R_T from the centre point Ctr.

When the speeds of the two wheels W_A and W_B are equal, then $Rpm_A == Rpm_B$ the turn radius R_T == infinity

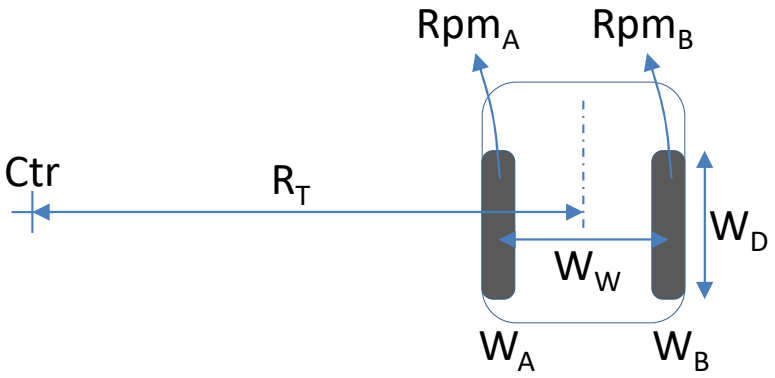
When speed $Rpm_A < Rpm_B$ the robot turns towards centre Ctr.

When wheel W_A stops, $Rpm_A = 0$ then wheel W_B revolves around wheel W_A at a radius of W_W where $R_T = W_W/2$

In this design, wheel separation W_W = 86mm, and the wheel diameters are W_D = 62mm.

Is it possible to develop a general equation which would enable you to predict the turning radius based on known wheel speeds?

Turning radius



When Rpm$_A$ == Rpm$_B$ then turn radius R_T == infinity

When Rpm$_A$ < Rpm$_B$ robot turns towards centre Ctr

When Rpm$_A$ = 0 then wheel W_B revolves around wheel W_A at a radius of W_W where $R_T = W_W/2$

Wheel separation W_W = 86mm
Wheel diameter W_D = 62mm

For turning radius $R_A = R_T - W_W/2$
$R_B = R_T + W_W/2$

Turning circumferences are $2\pi R_A$ and $2\pi R_B$

Distance travelled/min = πW_D * Rpm$_A$ for wheel W_A

= πW_D * Rpm$_B$ for wheel W_B

Time to complete 1 revolution $= \frac{2\pi R_A}{\pi W_D * Rpm_A} = \frac{2\pi R_B}{\pi W_D * Rpm_B}$

This becomes $R_T = \frac{W_W * (Rpm_A + Rpm_B)}{2 * (Rpm_B - Rpm_A)}$

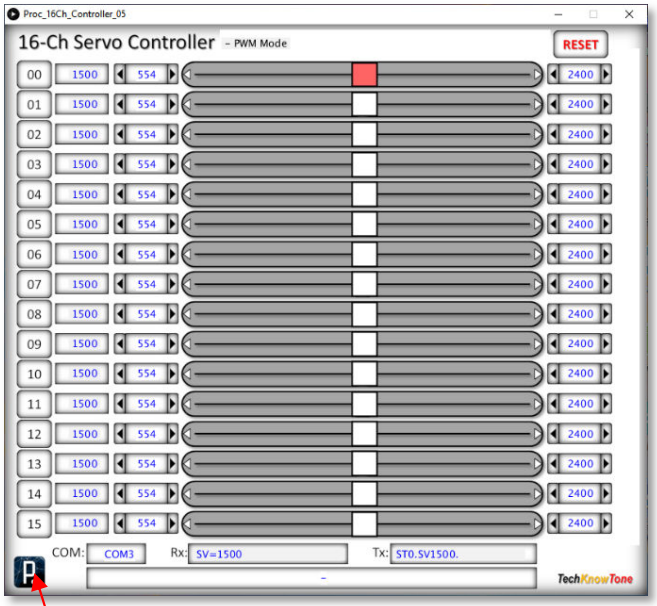
Note: wheel diameter W_D affects speed, but has no impact on turning radius. Robot width W_W and relative rpms are the controlling factors.

A/B	RpmA	RpmB		Rad. RT (mm)		A/B	RpmA	RpmB		Rad. RT (mm)
1.00	60	60		inf.		0.48	29	60		123
0.98	59	60		5117		0.47	28	60		118
0.97	58	60		2537		0.45	27	60		113
0.95	57	60		1677		0.43	26	60		109
0.93	56	60		1247		0.42	25	60		104
0.92	55	60		989		0.40	24	60		100
0.90	54	60		817		0.38	23	60		96
0.88	53	60		694		0.37	22	60		93
0.87	52	60		602		0.35	21	60		89
0.85	51	60		530		0.33	20	60		86
0.83	50	60		473		0.32	19	60		83
0.82	49	60		426		0.30	18	60		80
0.80	48	60		387		0.28	17	60		77
0.78	47	60		354	Dohyo	0.27	16	60		74
0.77	46	60		326	rad.	0.25	15	60		72
0.75	45	60		301		0.23	14	60		69
0.73	44	60		280		0.22	13	60		67
0.72	43	60		261		0.20	12	60		65
0.70	42	60		244		0.18	11	60		62
0.68	41	60		229		0.17	10	60		60
0.67	40	60		215		0.15	9	60		58
0.65	39	60		203		0.13	8	60		56
0.63	38	60		192		0.12	7	60		54
0.62	37	60		181		0.10	6	60		53
0.60	36	60		172		0.08	5	60		51
0.58	35	60		163		0.07	4	60		49
0.57	34	60		155		0.05	3	60		48
0.55	33	60		148		0.03	2	60		46
0.53	32	60		141		0.02	1	60		44
0.52	31	60		135		0.00	0	60		43
0.50	30	60		129			Turning on one wheel			

Speed ratio A/B @60 rpm

LTOF Calibration

As mentioned earlier you can use the custom Windows app I have provided to set up the micro servo used to position the laser range finder. By default the app loads in LX-16A mode, but easily switched to PWM mode by clicking on the Processing logo bottom left. This action enables channel 00 and sets the slider ranges for PWM adjustment.

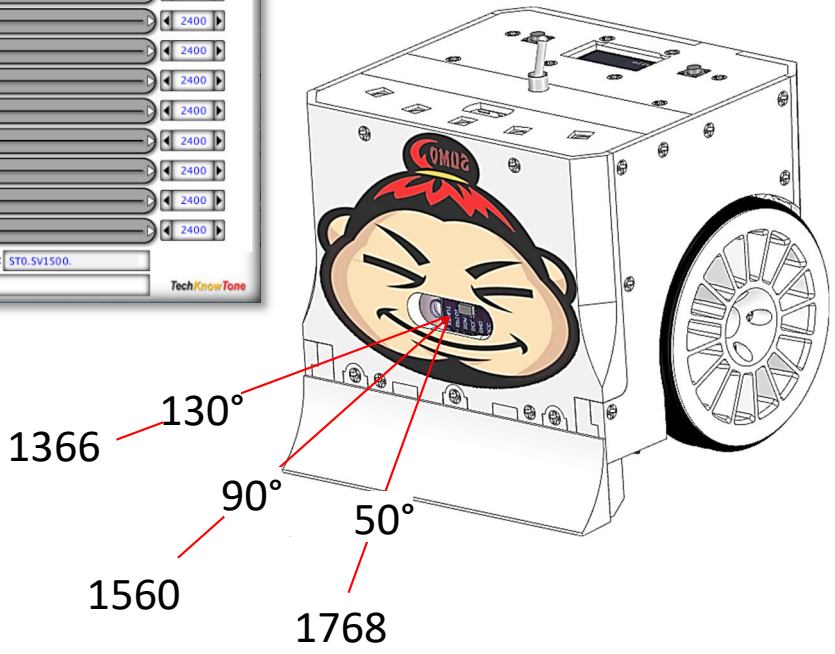


Click to change app mode

Clicking on the '00' channel button will turn the slider button red and adjustments will send values to the ESP32 micro in the Sumbot, which in turn will send the PWM pulses to the micro servo.

Fit the laser range finder mount to the servo spindle with the servo PWM set to 1500 µs, so that it is pointing straight ahead. Note that the splines on the servo shaft will only allow an approximate position to be set. You then use the slider on the app to determine the PWM values for the centre, lower and upper limits, and then enter them into your code as defined constants. You can see here the values I used in one of my Sumobots.

Note that all servos vary, and all will require different values for them. So if you ever have a need to change the micro servo, you will need to calibrate the new one and replace the PWM values in your ESP32 code.



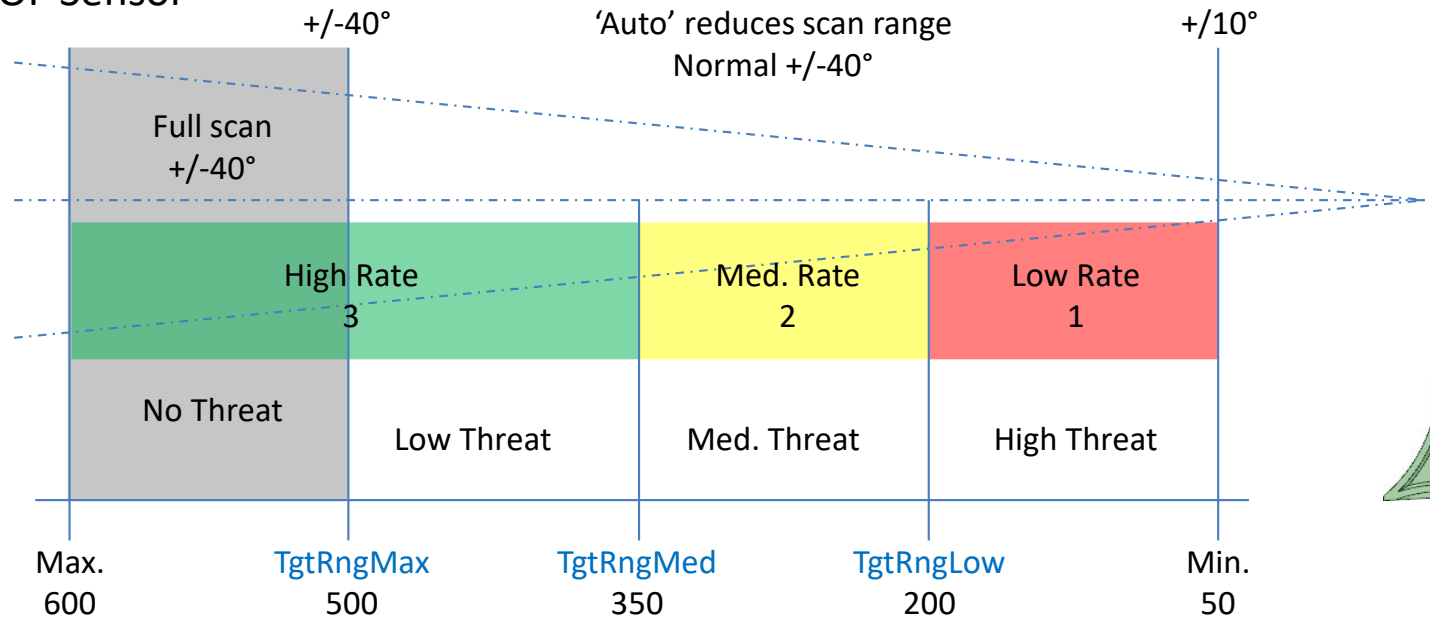
Angle	PWM	
50°	1754	UL
90°	1560	Centre
130°	1366	LL

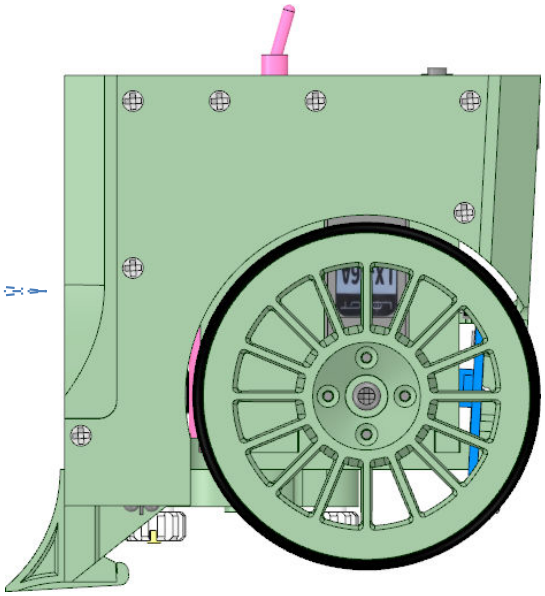
Centre PWM = LL + ((UL – LL)/2)
= 1366 + ((1758 – 1366)/2)
= 1567

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

Issue: 1.1 Released: 08/12/2025 TechKnowTone

LTOF Sensor

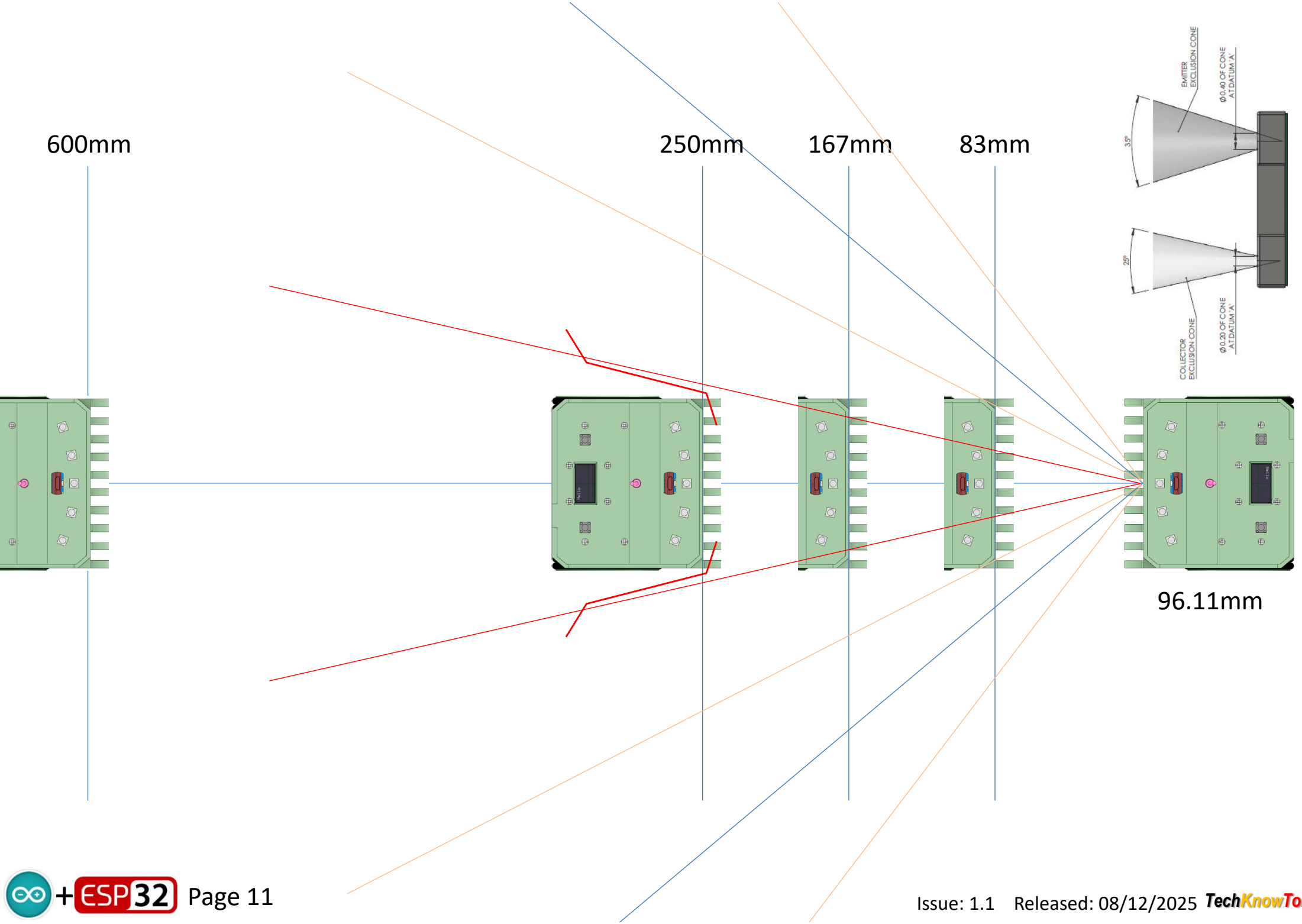




The LTOF range finder returns an accurate range below 1m, albeit sometimes noisy at that limit. Here we use a max range of 600mm, in a ring competition that is 770mm in diameter, giving good coverage.

We scan the target looking for a minimum range, and for points either side of that which exceed the minimum by 10mm. In normal mode we perform a full range scan at high rate, irrespective of range. In 'auto' mode, below the max target range, we reduce the scan range, centred on the minimum range angle.

The scanner changes angles once a new range has been received (LTOF ON) or at 40ms intervals. The ranging code flags when 'raw' and 'filtered' range values are available. IT is the filtered range that triggers the scanner.



Issue: 1.1 Released: 08/12/2025 TechKnowTone

SUMO ROBOT LOGIC