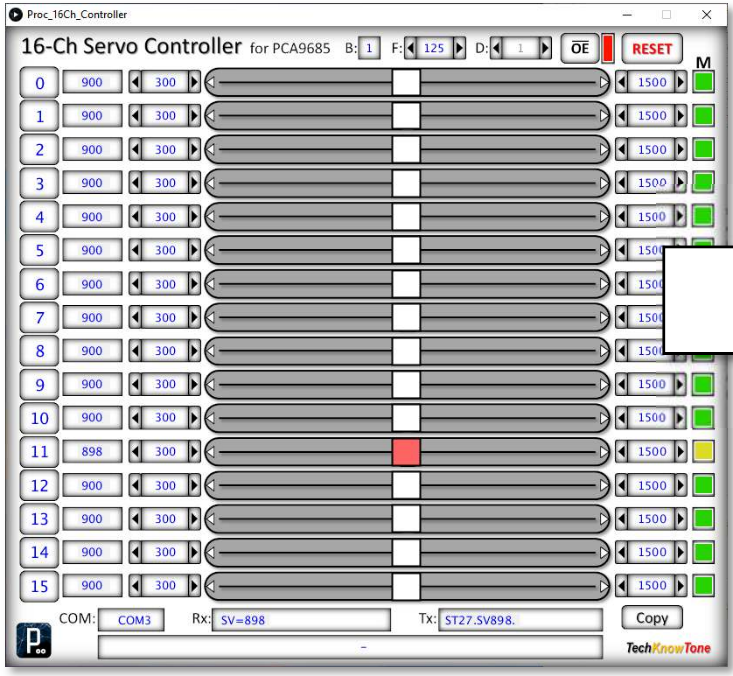
PIXAR Lamp

Servo Calibration



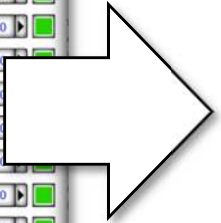
Proc_16Ch_Controller

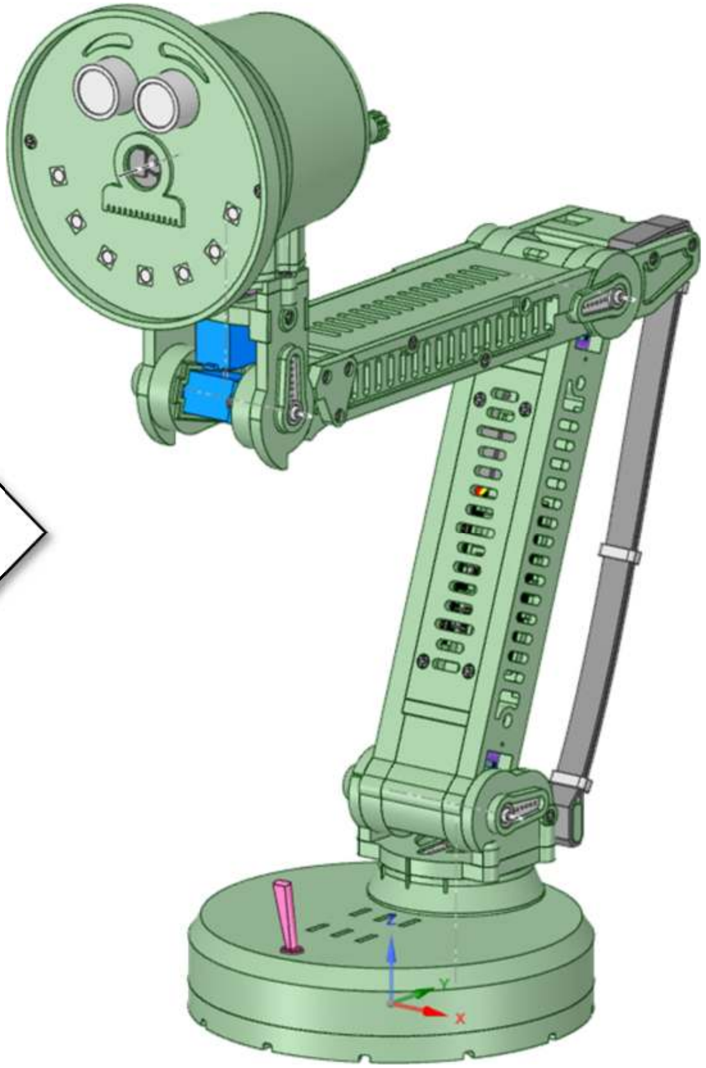
16-Ch Servo Controller for PCA9685 B: 1 F: 125 D: 1 OE RESET

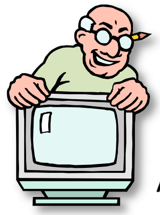
				M
0	900	300	1500	🟩
1	900	300	1500	🟩
2	900	300	1500	🟩
3	900	300	1500	🟩
4	900	300	1500	🟩
5	900	300	150	
6	900	300	150	
7	900	300	150	
8	900	300	150	
9	900	300	1500	🟩
10	900	300	1500	🟩
11	898	300	1500	🟨
12	900	300	1500	🟩
13	900	300	1500	🟩
14	900	300	1500	🟩
15	900	300	1500	🟩

COM: COM3 Rx: SV=898 Tx: ST27.SV898. Copy

TechKnowTone







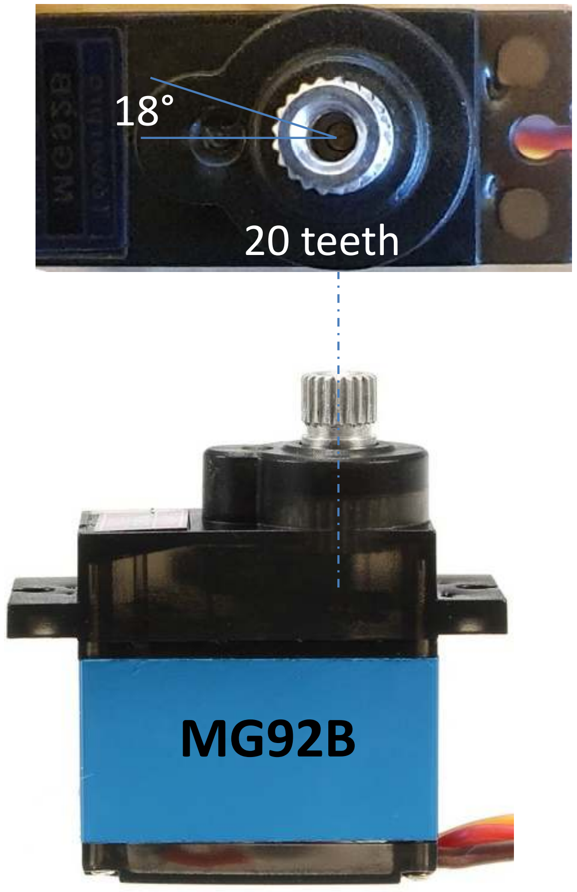
+ ESP32

An important process, that will govern your robots performance.

Issue: 1.0 Released: 02/05/2024 TechKnowTone

Why do we need to calibrate the servos?

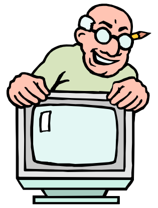
- No two servos are the same.
- Servos can overheat and be damaged if not setup correctly.
- Course calibration must be performed during the assembly process.
- This sets approximate positions for the leaver arms.
- A servo drive shaft has 20 splined teeth == 18° (best fit +/-9°)
- Course calibration ensures servos are within mechanical range/limits.
- Fine calibration sets angles, and min/max robot physical limits.
- The ESP32 C++ code needs limit values in order to work accurately.
- Hence, all PIXAR robots have a unique set of calibrated PWM code values.
- No two PIXAR's would ever be exactly the same.
- Code limits. and default values would be different for each one.
- Once calibrated we can use angles, as common values, not PWM.



Servo calibration is performed in three stages:

- Course, ensures mechanical parts are assembled correctly.
- Fine calibration, performed during testing, for accurate movement.

- Repeat this process for a given servo, if it is ever replaced.



Only use genuine Tower Pro MG92B servos, for best performance.

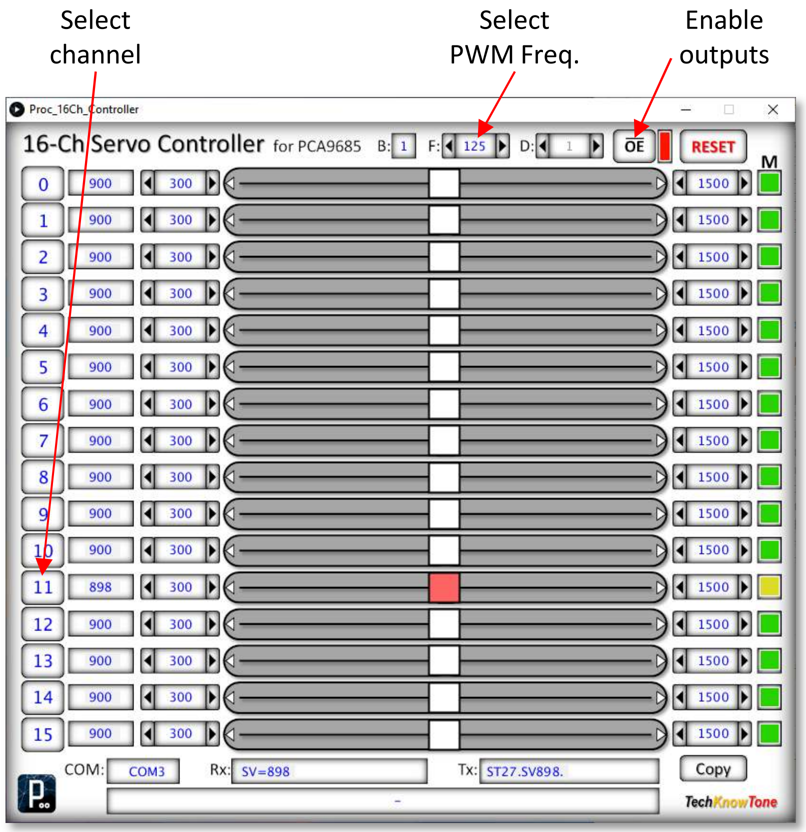
Servo Testing

The PIXAR lamp employs 5 powerful MG92B servo motors, providing the high torques needed to move it efficiently. Course calibration, during the construction phase, is most easily performed using a HJ servo tester. These relatively low cost devices, explained on the next page, provide an effective means of determining servo PWM values, and can be used on all of your projects that employ servos using PWM as the control signal.

I have created a Windows app which, when used with an ESP32 micro running the PIXAR code, enables you to control up to 16 servos. Selected servos will respond to the on-screen slider settings, so you can determine what PWM values are required to set the servo control arms to a particular position.

This app is used for fine calibration, after the build process. It is mainly used to determine limits of travel, at the extremes of movement. These limits also correspond to specific angles, which then allow the robot to be controlled by mapped angle values, making them independent of the servos PWM values.

This app can be used with other projects employing servos, provided that you use the C++ code in PIXAR lamp, which enables the micro to decode and implement the serial commands. It Is designed to work with a range of servo types, not only conventional servos like the MG92B. See the page which provides an overview of this app.

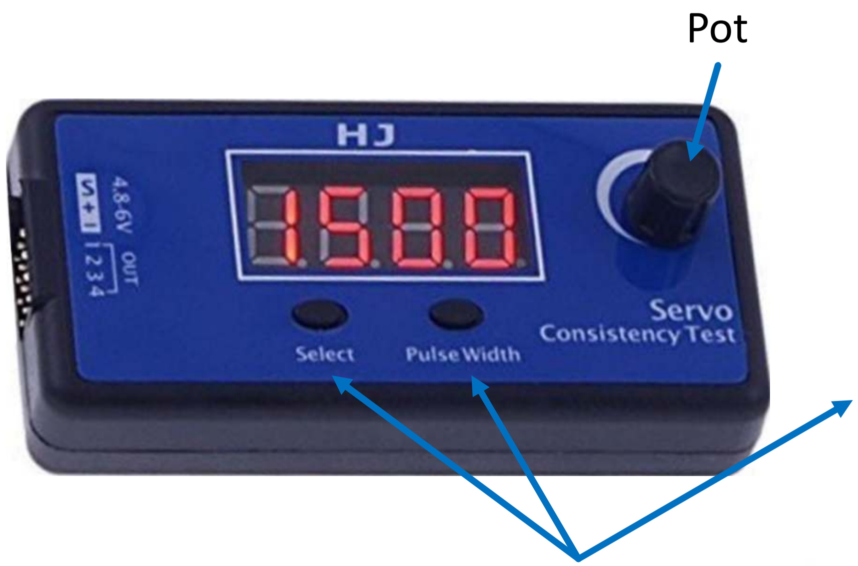


16-Ch Servo Controller for PCA9685 B: 1 F: 125 D: 1

				M
0	900	300	1500	🟩
1	900	300	1500	🟩
2	900	300	1500	🟩
3	900	300	1500	🟩
4	900	300	1500	🟩
5	900	300	1500	🟩
6	900	300	1500	🟩
7	900	300	1500	🟩
8	900	300	1500	🟩
9	900	300	1500	🟩
10	900	300	1500	🟩
11	898	300	1500	🟨
12	900	300	1500	🟩
13	900	300	1500	🟩
14	900	300	1500	🟩
15	900	300	1500	🟩

COM: COM3 Rx: SV=898 Tx: ST27.SV898.

HJ Servo Consistency Tester



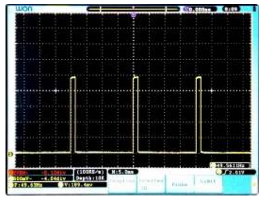
Select Button functions:

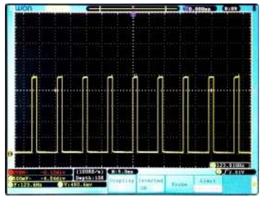
- Variable (Pot) pulse width 800 – 2200 µs (default mode)
- Fixed constant pulse width 1500 µs, centre position
- Sweep (Pot) pulse width from 800 -> 2200 -> 800 µs

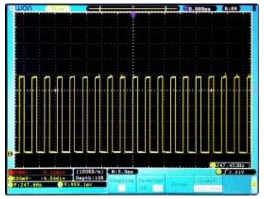
Pulse Width Button functions:

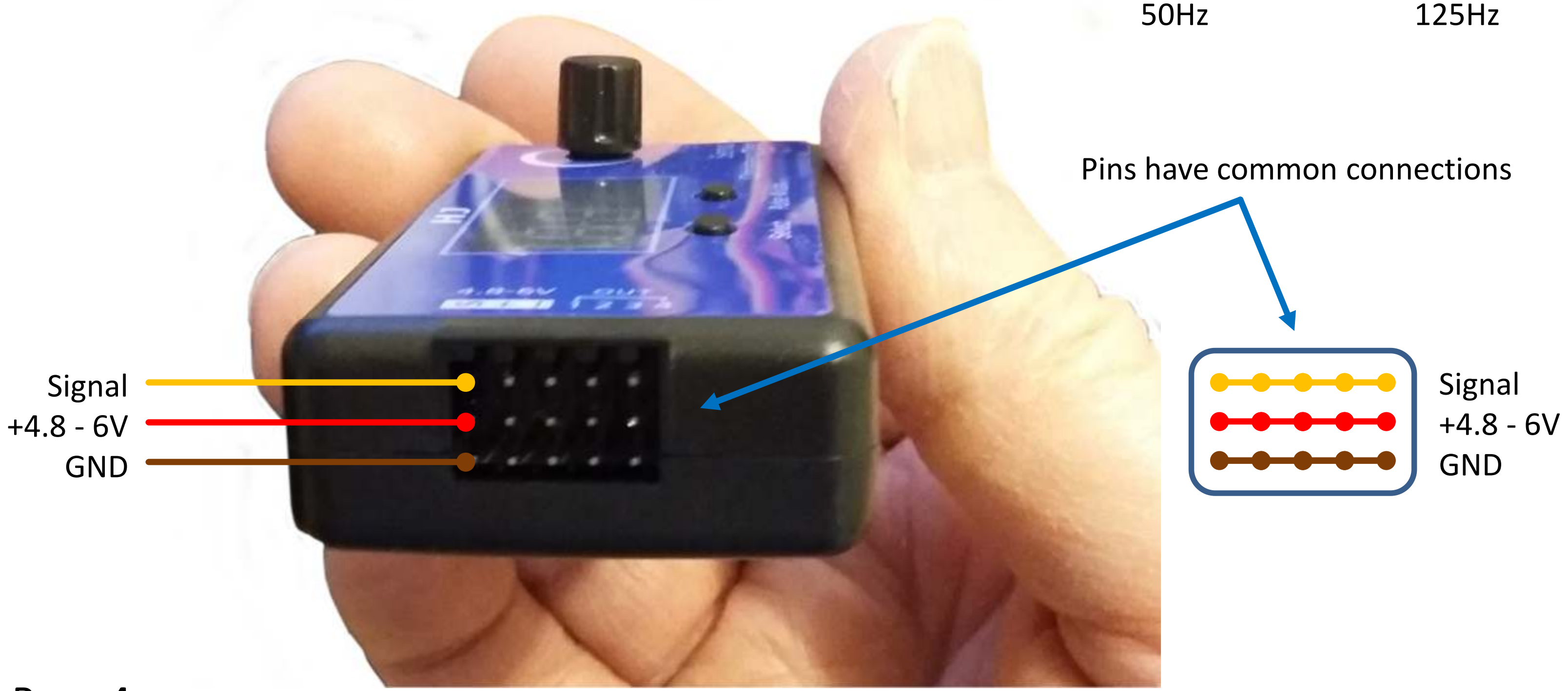
This is actually pulse frequency (Hz)

- 50H = 50 Hz - run MG92B at this frequency (default)
- 125H = 125 Hz
- 250H = 250 Hz


50Hz


125Hz


250Hz

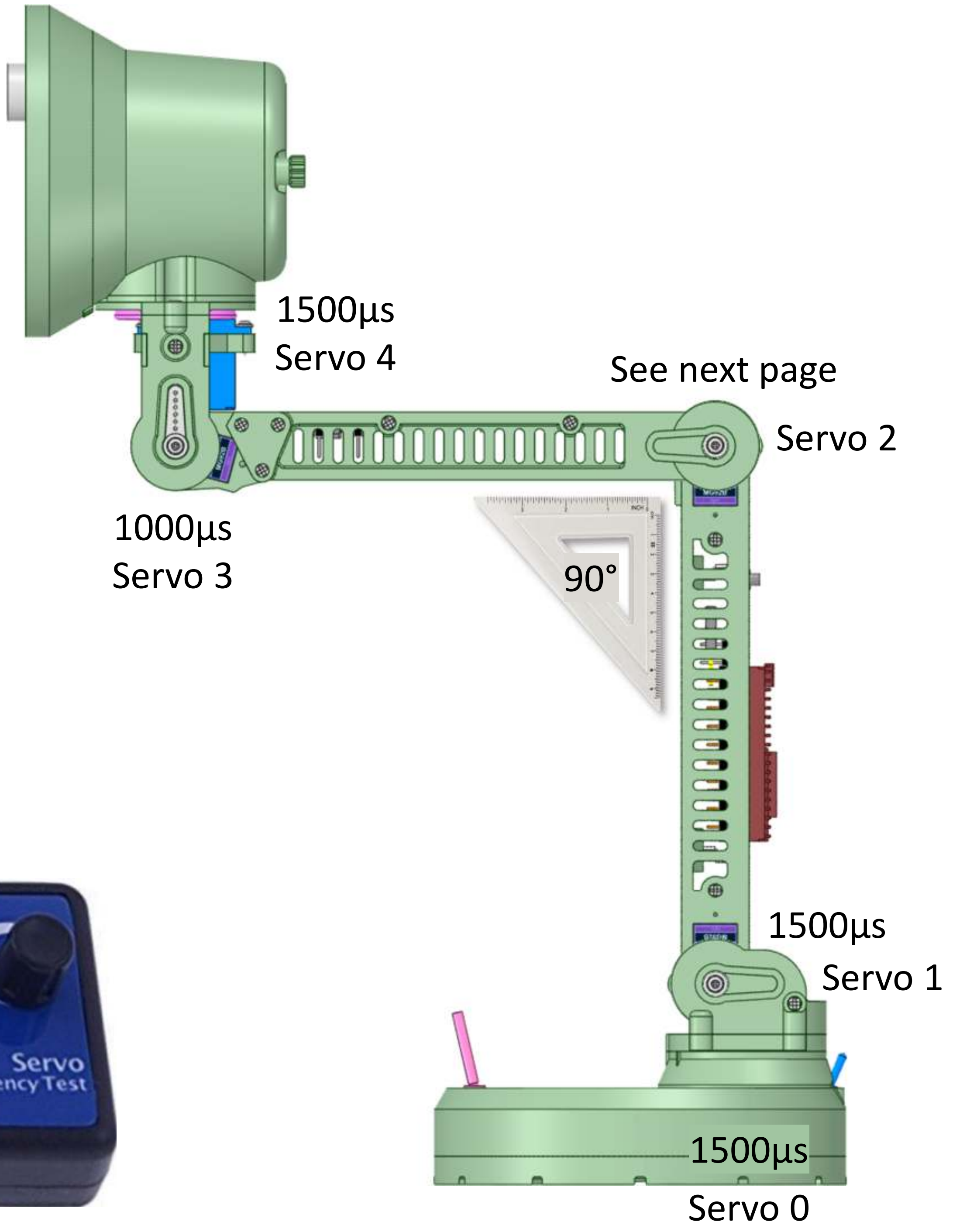


Course Calibration

At this stage you are in the process of building the robot and attaching leaver arms to the servos. These diagrams show you what PWM values to use, with a Servo Consistency Test box, to position each servo for the fitting of its leaver.

Note that all of the joints are at right-angles, when fitting the servo leavers.

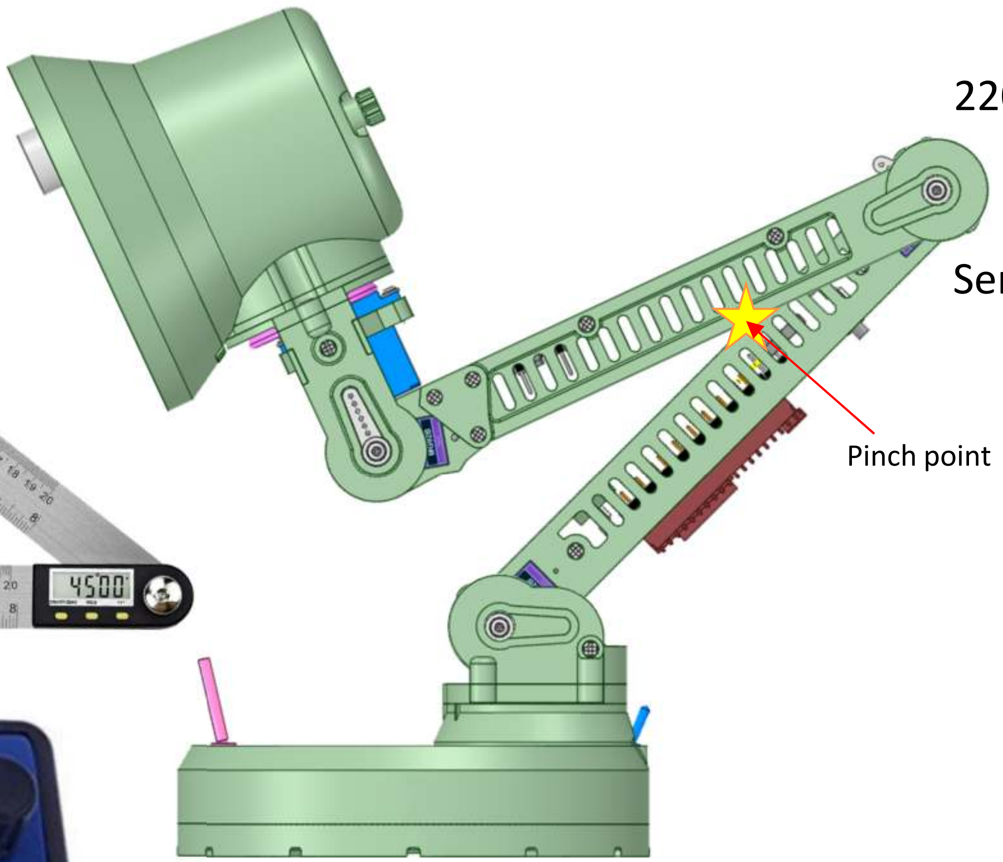
If you don't have a Servo Consistency Test box, then you can flash code into the ESP32 micro, and use the supplied Windows 16-channel app to set the PWM values. Then attach the leaver arm to the servo as shown, to best achieve the centre position. The splines on the drive shaft may compromise your ability to achieve this exactly (remember the +/-9°), so find the best position.



Course Calibration

For servo 2 we adopt the same method; but in this case we close up the joint, so that it is at its minimum angle, resting on the cross brace. There is a pinch point where the cross member touches the lower arm. Use this position.

Out of interest, for angles you can use a digital protractor, if you have access to one. They aren't very dear, and can be very useful for servo projects like this.



2200µs

16-channel Controller App

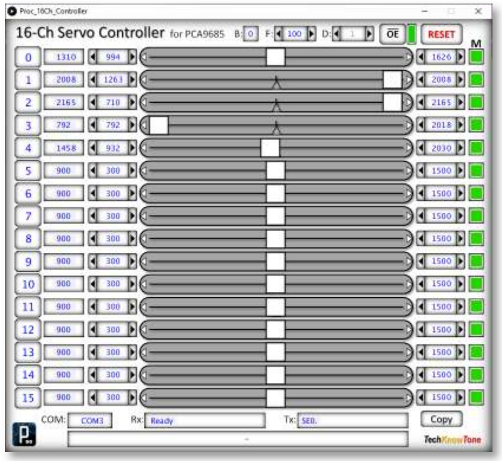
This Windows app communicates with your ESP32 micro, using the USB serial port, or over Wi-Fi. When you first connect it to the micro, it will request default values from it, which are then used to position the first five sliders (for PIX), to represent those default values.

The app can be used in several modes. By default it is set up for use with PCA9685 16-channel I2C controller boards; but for the PIX project you can switch to PWM mode by simply clicking on the bottom left 'Processing' logo. Initially you will want to use the app in 'PWM Mode', but later it may be useful to use 'Angle Mode'.

From left to right, the first column is 16 ON/OFF buttons. When a channel is ON the slider will be coloured red, and values will then be sent to the micro as the slider is moved. The 2nd column represents the values being sent; which will change as you move the sliders. The 3rd column represents the lower limit of the slider. Clicking in the field will set this to the current value, and is a useful way of limiting the slider range. Then you have the click-n-drag sliders themselves. In the 5th column we have the slider upper limits. With the lower and upper limits set, you can then freely move the slider, to swing a servo through its desired range of movement.

At the bottom of the app you will see fields for the USB COM port, receive (Rx) and transmit (Tx) fields, and below them a useful Help field.

If a COM port connection has not been established, the COM field will be in red. To force a connection attempt, simply click-left in the COM field. If a connection is established, the text will turn blue and the COM port number will be displayed. Click-right to cancel a connection, as you may want to download code to your micro over the same USB link.

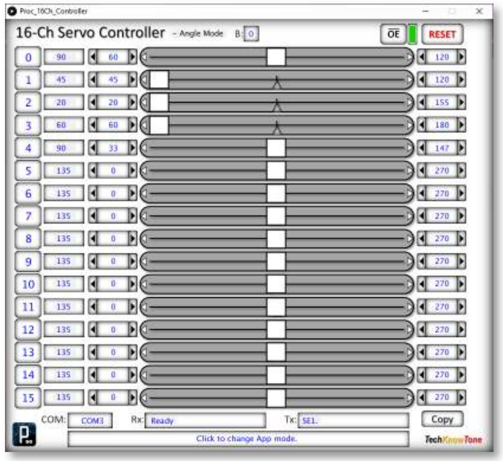


Proc_16Ch_Controller

16-Ch Servo Controller for PCA9685 B: 0 F: 100 D: 1 OE RESET

0	1310	994	1626	🟩
1	2008	1263	2008	🟩
2	2165	710	2165	🟩
3	792	792	2018	🟩
4	1458	932	2030	🟩
5	900	300	1500	🟩
6	900	300	1500	🟩
7	900	300	1500	🟩
8	900	300	1500	🟩
9	900	300	1500	🟩
10	900	300	1500	🟩
11	900	300	1500	🟩
12	900	300	1500	🟩
13	900	300	1500	🟩
14	900	300	1500	🟩
15	900	300	1500	🟩

COM: COM3 Rx: Ready Tx: SE0. Copy



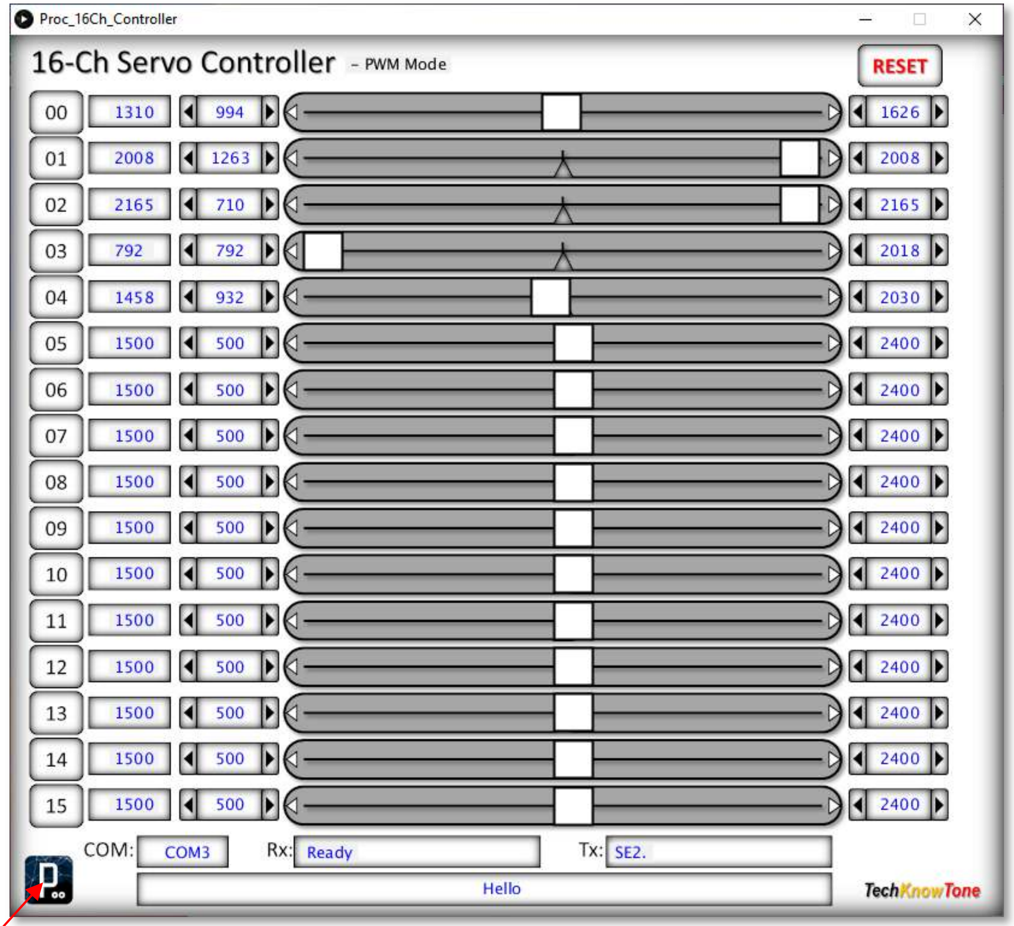
Proc_16Ch_Controller

16-Ch Servo Controller - Angle Mode B: 0 OE RESET

0	90	60	120
1	45	45	120
2	20	20	155
3	60	60	180
4	90	33	147
5	135	0	270
6	135	0	270
7	135	0	270
8	135	0	270
9	135	0	270
10	135	0	270
11	135	0	270
12	135	0	270
13	135	0	270
14	135	0	270
15	135	0	270

COM: COM3 Rx: Ready Tx: SE1. Copy

Click to change App mode.



Proc_16Ch_Controller

16-Ch Servo Controller - PWM Mode RESET

00	1310	994	1626
01	2008	1263	2008
02	2165	710	2165
03	792	792	2018
04	1458	932	2030
05	1500	500	2400
06	1500	500	2400
07	1500	500	2400
08	1500	500	2400
09	1500	500	2400
10	1500	500	2400
11	1500	500	2400
12	1500	500	2400
13	1500	500	2400
14	1500	500	2400
15	1500	500	2400

COM: COM3 Rx: Ready Tx: SE2.

Hello

Click to change mode

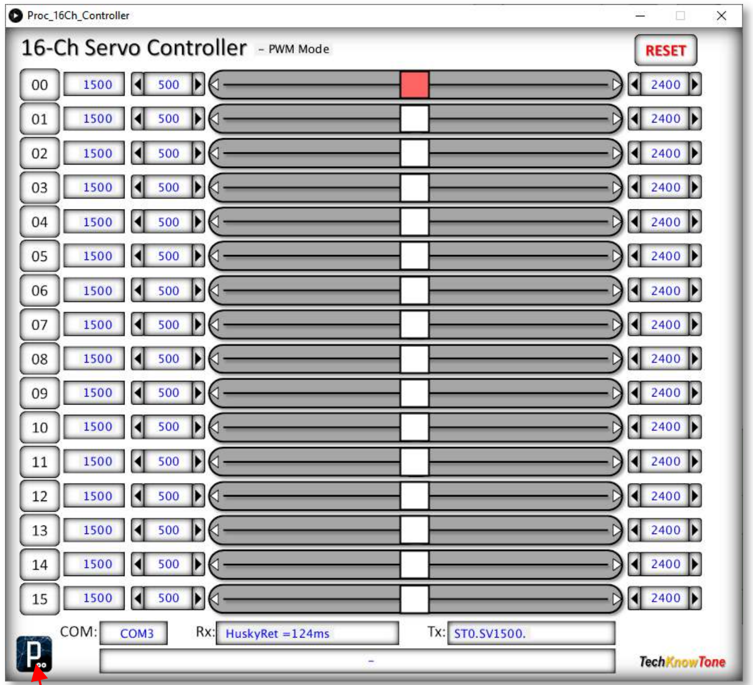
Fine Calibration – Servo 0

Servo 0, mounted in the base, swings the arm from side to side. The 3-D models include three calibration marks; one for forward (90°), one 30° to the left (60°), and one 30° to the right (120°).

By convention, in the PIXAR code their corresponding PWM values are defined as follows:

S0_60	1626	30° to the left
S0_90	1310	forward
S0_120	994	30° to the right

Use the 16-channel Servo Controller app, in PWM mode, to move channel 0 and determine these PWM values for your servos. I have included mine here for reference, and included them in the ESP32 code.


Proc_16Ch_Controller

16-Ch Servo Controller – PWM Mode

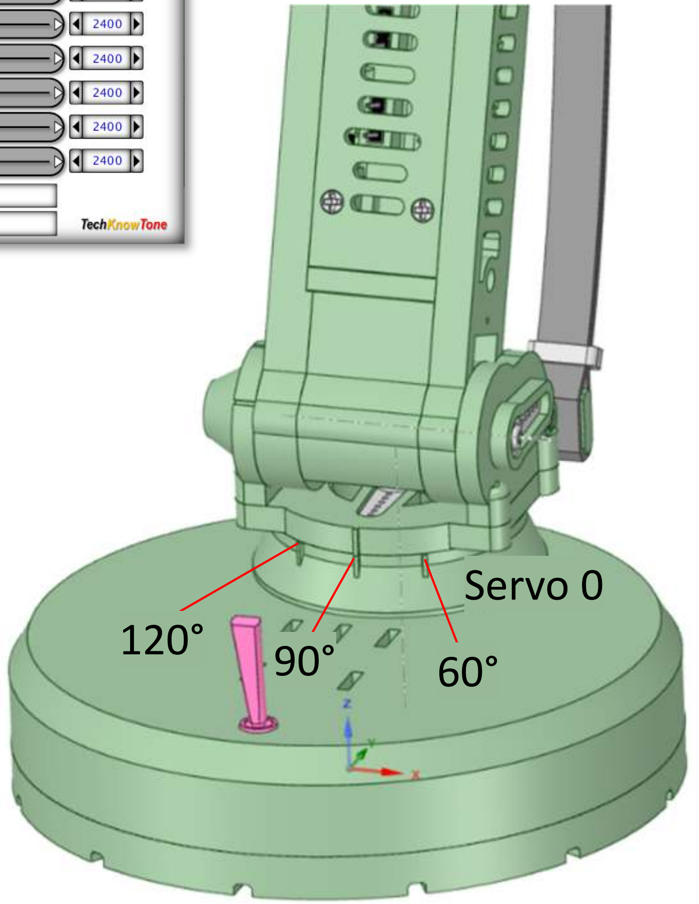
RESET

00	1500	500	2400
01	1500	500	2400
02	1500	500	2400
03	1500	500	2400
04	1500	500	2400
05	1500	500	2400
06	1500	500	2400
07	1500	500	2400
08	1500	500	2400
09	1500	500	2400
10	1500	500	2400
11	1500	500	2400
12	1500	500	2400
13	1500	500	2400
14	1500	500	2400
15	1500	500	2400

COM: COM3 Rx: HuskyRet =124ms Tx: ST0.SV1500.

TechKnowTone

Note that when you invoke the 16-channel app it will start up in PCA9685 mode, by default. You need to switch it to PWM mode by clicking on the Processing logo button, bottom left. Successive clicks on this logo will change the mode of the app; there are 4 modes in total; which can be used in other projects too.



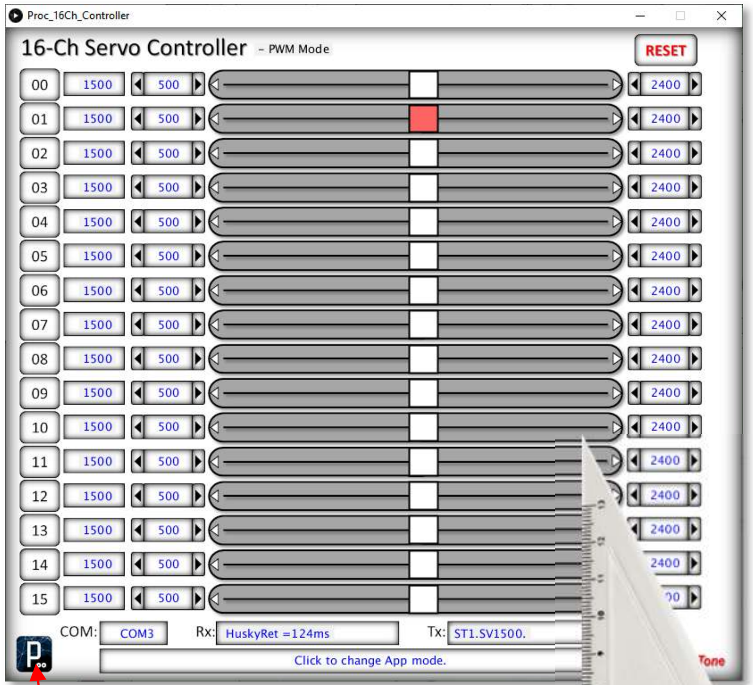
Fine Calibration – Servo 1

Servo 1, mounted at the bottom of the lower arm, swings the arm forwards and backwards. The 3-D model includes physical stops, at the rear, to prevent the arm from going backwards more than 45°. This position can be used for one of the measurements. The other two angles are set and measured using a 45° set square, as shown.

In the code their corresponding PWM values are defined for servo 1 as follows:

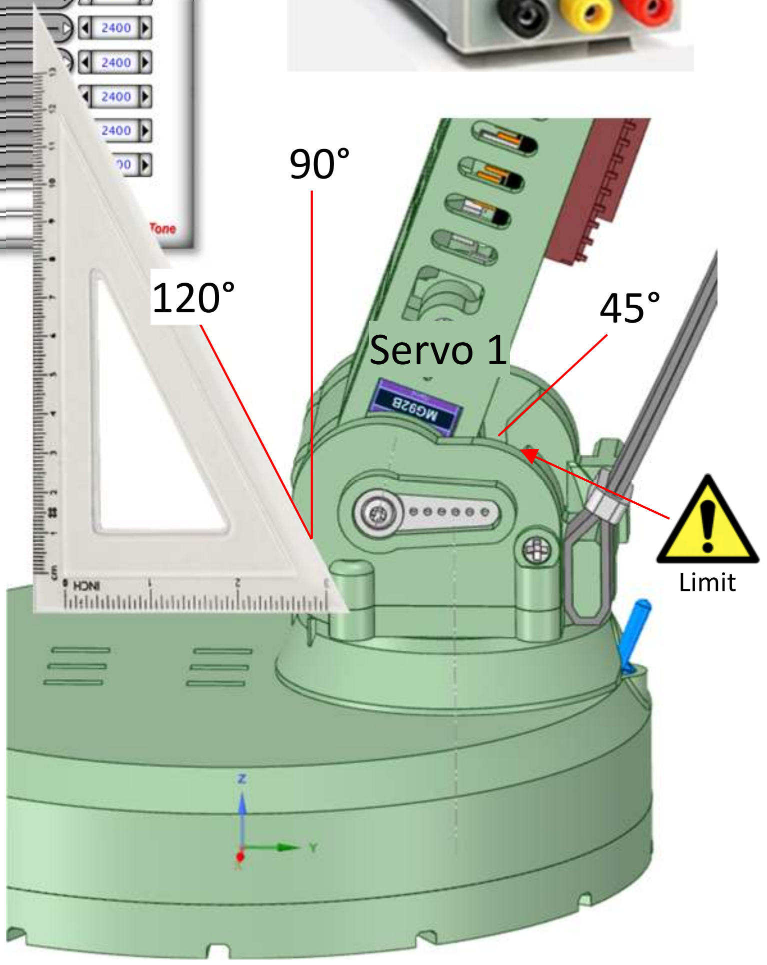
S1_45	2008	45° leaning backwards
S1_90	1592	upright, vertical
S1_120	1263	30° leaning forwards

Use the 16-channel Servo Controller app to move channel 1, to determine the PWM values for your servo. I have included mine here for reference, and included them in the code.



00	1500	500	2400
01	1500	500	2400
02	1500	500	2400
03	1500	500	2400
04	1500	500	2400
05	1500	500	2400
06	1500	500	2400
07	1500	500	2400
08	1500	500	2400
09	1500	500	2400
10	1500	500	2400
11	1500	500	2400
12	1500	500	2400
13	1500	500	2400
14	1500	500	2400
15	1500	500	00

COM: COM3 Rx: HuskyRet =124ms Tx: ST1.SV1500.



Note: that when you invoke the 16-channel app it will start up in PCA9685 mode, by default. You need to switch it to PWM mode by clicking on the Processing logo button, bottom left. Successive clicks on this logo will change the mode of the app; there are 4 modes in total. Which can be used in other projects.

At the 45° angle, the servo PWM must not be set so high a value that it is driving the arm into the physical stop limit; but just very close to it. By using an external power supply, with current sensing, you will see the current rise rapidly when the servo effectively hits the end stop, so you can back it off slightly from that position.

Fine Calibration – Servo 2

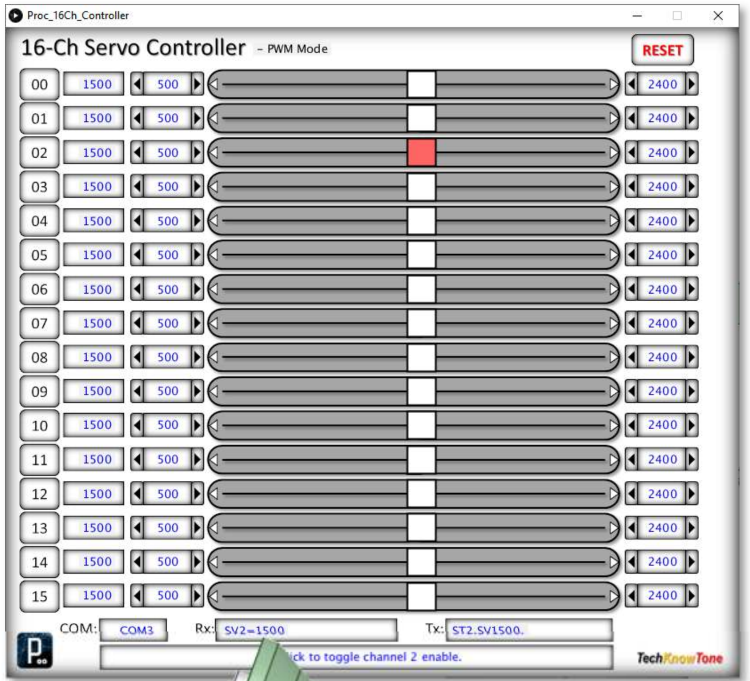
Servo 2, mounted at the top of the lower arm, swings the arm upwards and downwards. The 3-D model includes two physical stops, one at the rear of the forearm, and one below it, to limit the arm movement. These positions can be used for two of the measurements. The other 90° angle is set and measured using a set square, as shown.

In the code their corresponding PWM values are defined for servo 2 as follows:

S2_20	2165	20° leaning downwards
S2_90	1428	horizontal
S2_155	710	65° leaning upwards

Use the 16-channel Servo Controller app to move channel 2, to determine the PWM values for your servo. I have included mine here for reference, and included them in the code.

Use the external power supply, current sensing, to detect and avoid exceeding physical limits.


Proc_16Ch_Controller

16-Ch Servo Controller - PWM Mode

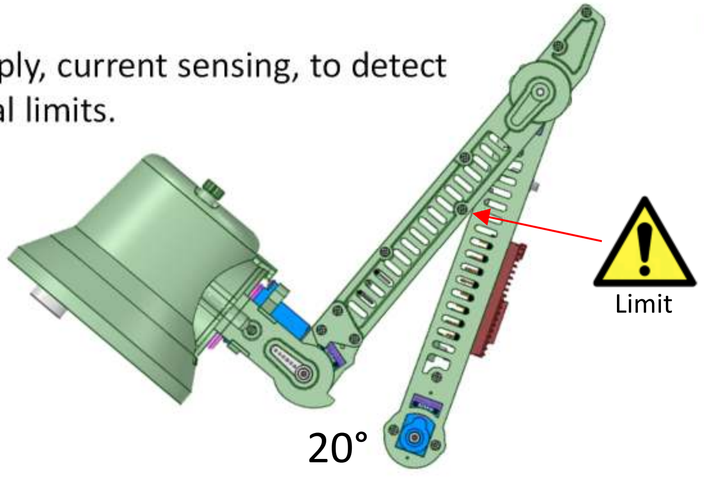
RESET

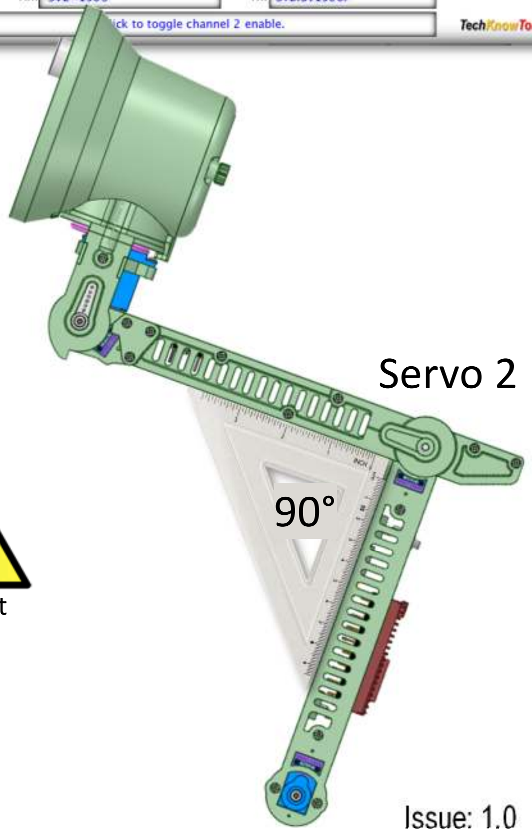
00	1500	500	2400
01	1500	500	2400
02	1500	500	2400
03	1500	500	2400
04	1500	500	2400
05	1500	500	2400
06	1500	500	2400
07	1500	500	2400
08	1500	500	2400
09	1500	500	2400
10	1500	500	2400
11	1500	500	2400
12	1500	500	2400
13	1500	500	2400
14	1500	500	2400
15	1500	500	2400

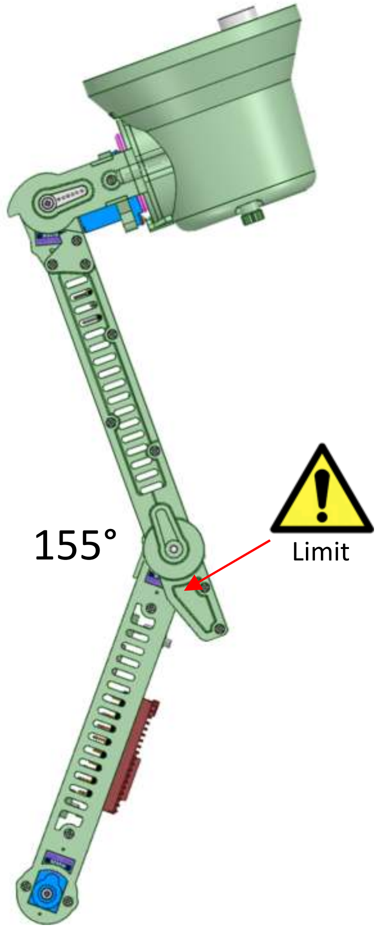
COM: COM3 Rx: SV2=1500 Tx: ST2.SV1500.

ick to toggle channel 2 enable.

TechKnowTone







Fine Calibration – Servo 3

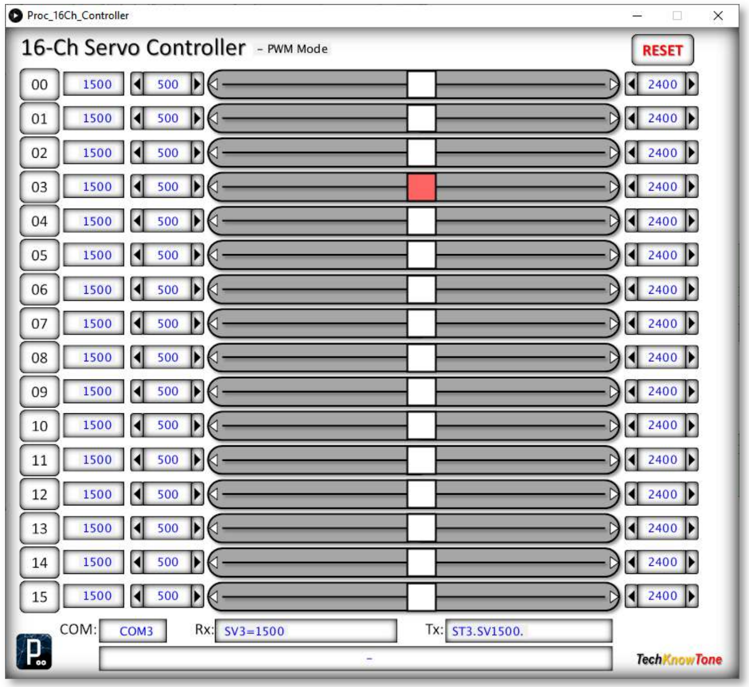
Servo 3, mounted at the end of the forearm, swings the lamp upwards and downwards. The 3-D model includes two physical stops, one at the rear, and one below it, to limit the tilt movement. These positions can be used for two of the measurements. The other 90° angle is set and measured using a set square, as shown.

In the code their corresponding PWM values are defined for servo 3 as follows:

S3_60	792	30° leaning backwards
S3_90	1120	vertical, looking forwards
S3_180	2018	90° looking downwards

Use the 16-channel Servo Controller app to move channel 2, to determine the PWM values for your servo. I have included mine here for reference, and included them in the code.

Use the external power supply, current sensing, to detect and avoid exceeding physical limits.


Proc_16Ch_Controller

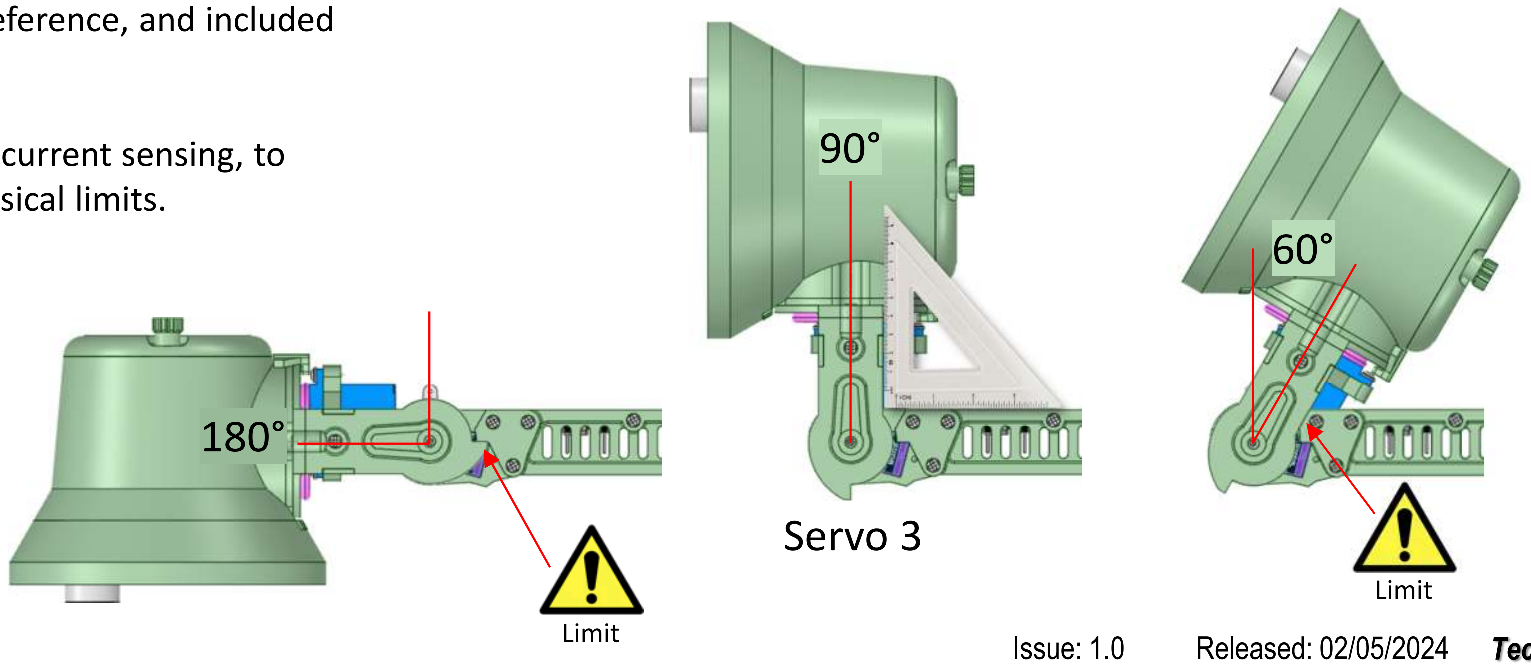
16-Ch Servo Controller - PWM Mode

RESET

00	1500	500	2400
01	1500	500	2400
02	1500	500	2400
03	1500	500	2400
04	1500	500	2400
05	1500	500	2400
06	1500	500	2400
07	1500	500	2400
08	1500	500	2400
09	1500	500	2400
10	1500	500	2400
11	1500	500	2400
12	1500	500	2400
13	1500	500	2400
14	1500	500	2400
15	1500	500	2400

COM: COM3 Rx: SV3=1500 Tx: ST3.SV1500.

TechKnowTone

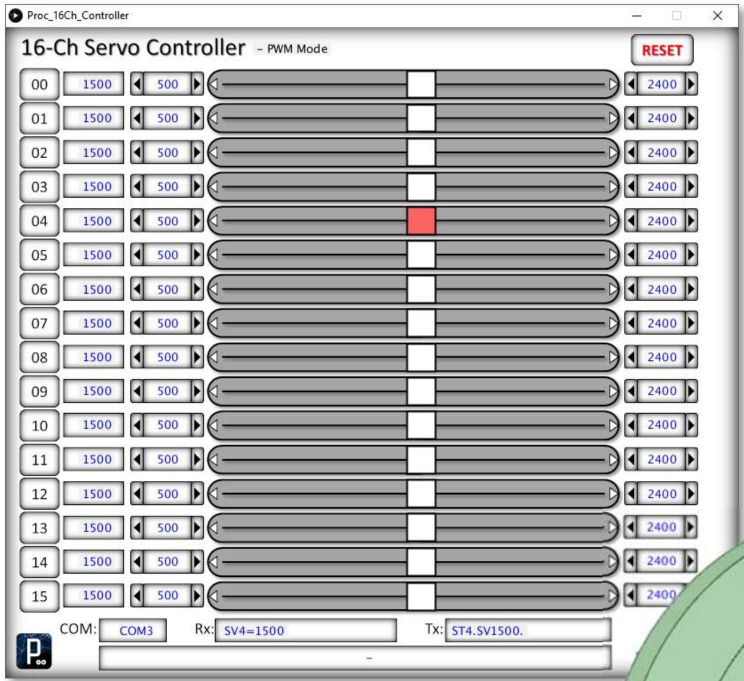


Fine Calibration – Servo 4

Servo 4, mounted at the end of the forearm, above servo 3, turns the lamp side to side, left to right. The 3-D model includes a physical stop, which rotates with the lamp, to limit its movement. These positions can be used for two of the measurements. The other 90° angle is set using visual judgement.

In the code their corresponding PWM values are defined for servo 4 as follows:

S4_33	2030	57° turned to the left
S4_90	1458	central, looking forwards
S4_147	932	57° turned to the right



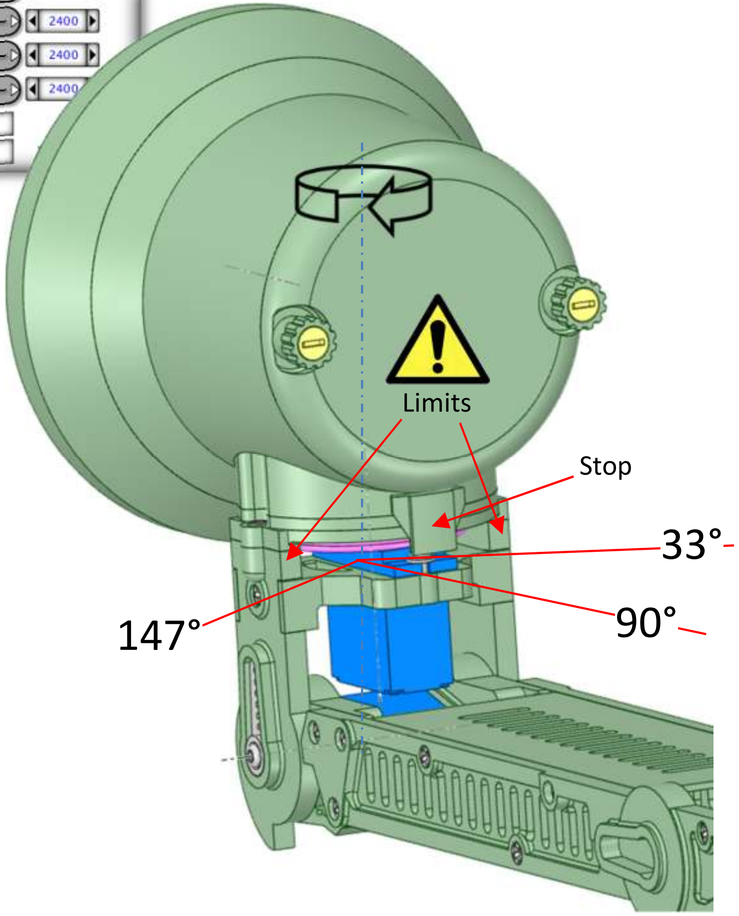
Proc_16Ch_Controller

16-Ch Servo Controller – PWM Mode

RESET

00	1500	500	2400
01	1500	500	2400
02	1500	500	2400
03	1500	500	2400
04	1500	500	2400
05	1500	500	2400
06	1500	500	2400
07	1500	500	2400
08	1500	500	2400
09	1500	500	2400
10	1500	500	2400
11	1500	500	2400
12	1500	500	2400
13	1500	500	2400
14	1500	500	2400
15	1500	500	2400

COM: COM3 Rx: SV4=1500 Tx: ST4.SV1500.

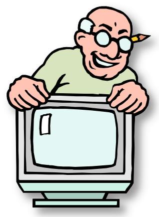


Use the 16-channel Servo Controller app to move channel 4, to determine the PWM values for your servo. I have included mine here for reference, and included them in the code.

Note - you can turn on servo 3 and set it to the vertical position, to steady the lamp whilst taking these PWM readings.

Use the external power supply, current sensing, to detect and avoid exceeding physical limits.

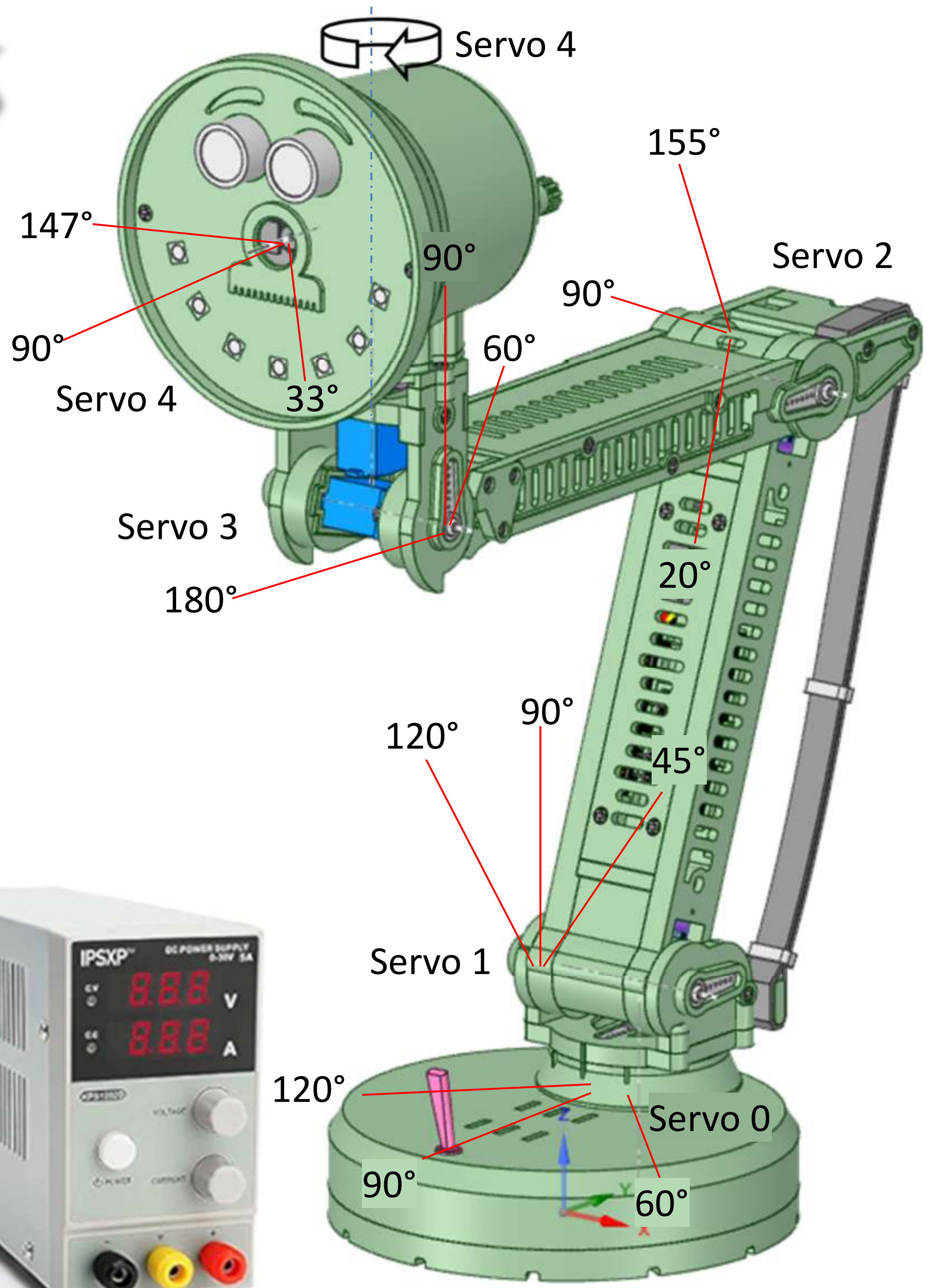
Angle Summary

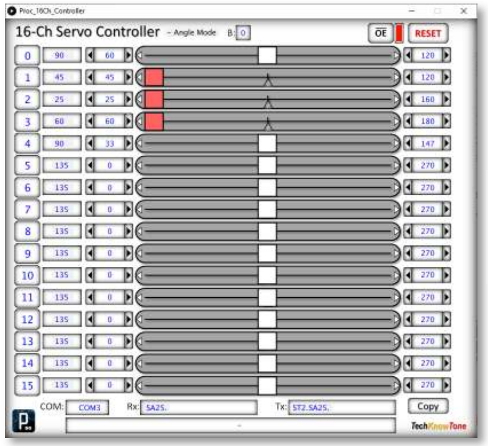


This diagram summarises the range of angles defined for each of the five servos, and acts as a useful reference, when working out moves for PIX.

Note that the elastic tensioner works to not only assist with the lifting of the forearm, but also has a tendency to pull the lower arm backwards, This makes it necessary to raise the forearm, to reduce the tension, when wanting to move the lower arm forward. Otherwise it will fail to do so.

I used the Windows app in angle mode, along with an external DC power supply, to determine the best range of angle for ease of moving the forearm and lower arm servos. These ranges were then built into the code as limit ranges.





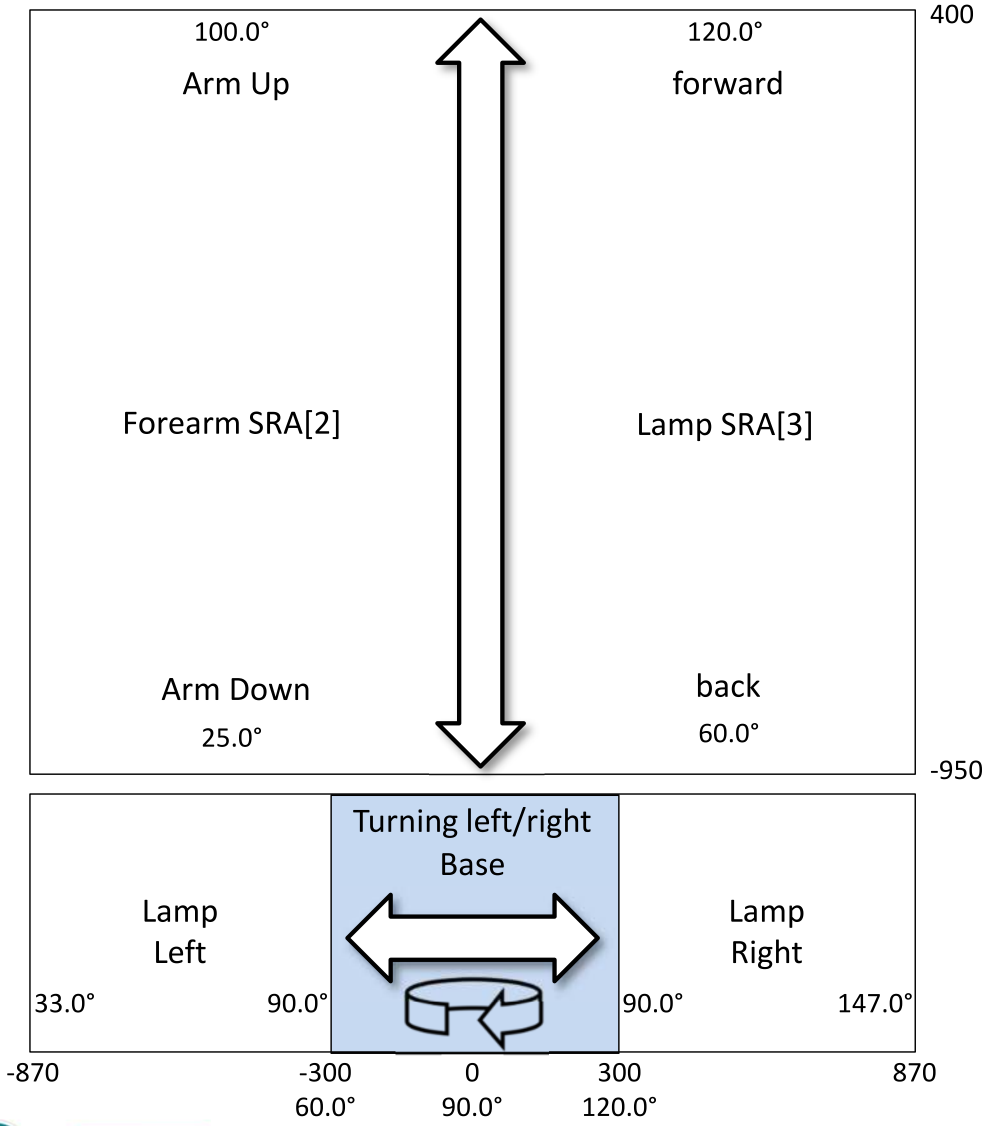
16-Ch Servo Controller – Angle Mode B: 0 OE RESET

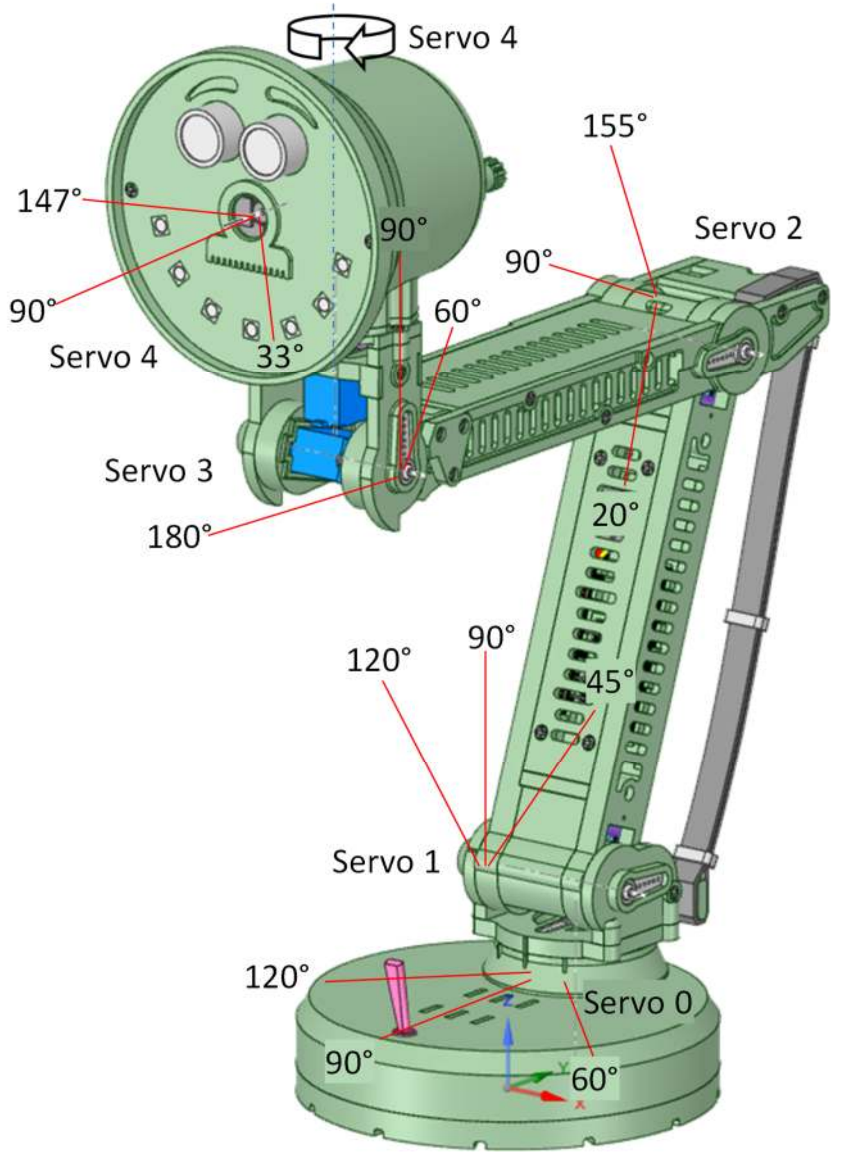
0	90	60	120
1	45	45	120
2	25	25	160
3	60	60	180
4	90	33	147
5	135	0	270
6	135	0	270
7	135	0	270
8	135	0	270
9	135	0	270
10	135	0	270
11	135	0	270
12	135	0	270
13	135	0	270
14	135	0	270
15	135	0	270

COM: COM3 Rx: SA25. Tx: ST2.SA25. Copy

View Space

The view space map can be used to determine angles for the robot, where it has detected a face. Servo 0 and 4 map the turning of the base and head into a horizontal angle. Servo 2 and 3 map the vertical angle. Servo 1 is ignored, as its contribution to height is low. The robot control system aims to keep the lamp looking horizontally. The location of faces fades over time, but are refreshed. All internal angles are in x10 integers, to improve speed, whilst not compromising accuracy.

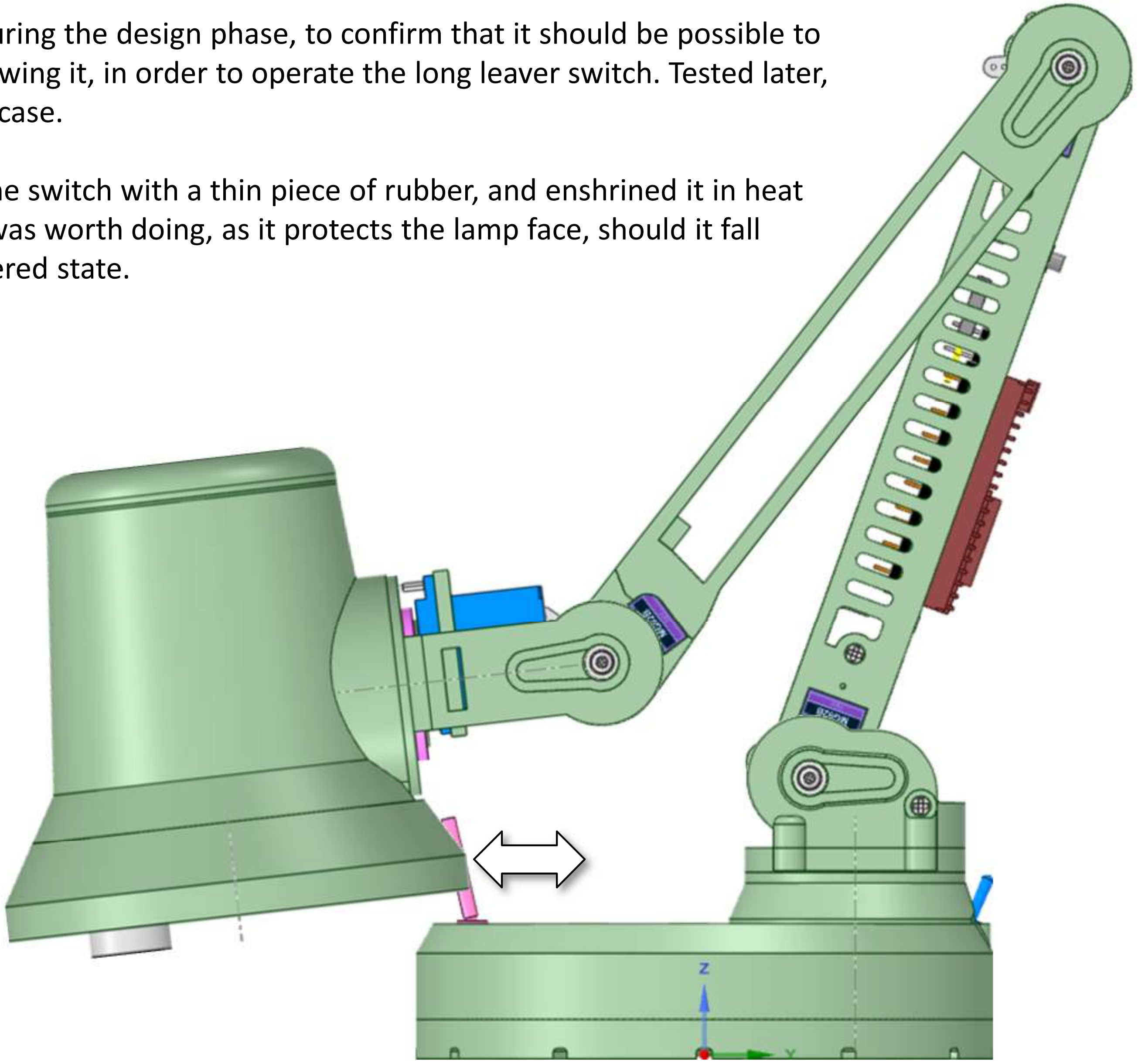




Leaver switch operation

I used this diagram during the design phase, to confirm that it should be possible to move the lamp, and swing it, in order to operate the long leaver switch. Tested later, this proved to be the case.

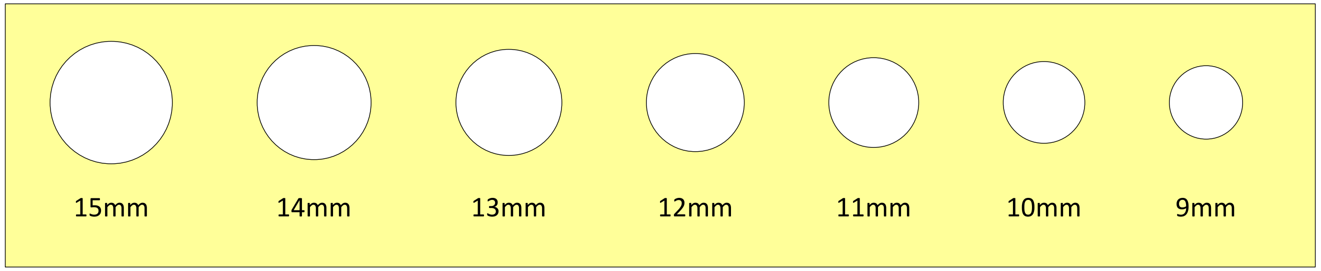
Note that I covered the switch with a thin piece of rubber, and enshrined it in heat shrink sleeving. This was worth doing, as it protects the lamp face, should it fall forward in an unpowered state.

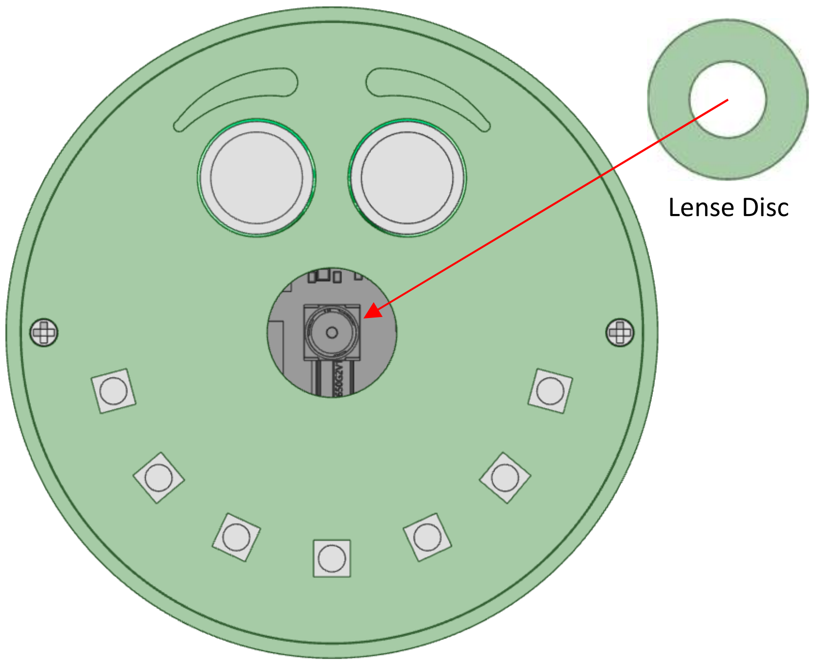


HuskyLens Camera field of view

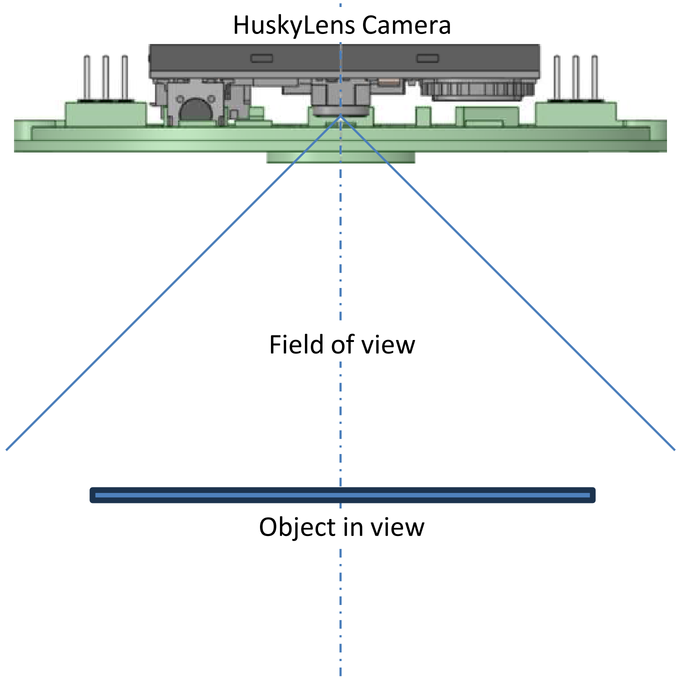
I wanted the camera to see the whole image being presented to it, and not be obscured by the size of the whole in the face plate. Therefore there is a Lense Disc which was sized accordingly, and then glued into the face plate.

The method I used to get this centre hole as small as possible, was to print strips of paper with different sized holes in them, and place them over the opening. There by reducing the size of the hole until it was seen to obscure parts of the cameras display image.





You need to have the HuskyLens camera powered up for this, and the rear cover of the lamp housing removed, in order to view the screen on the rear of the camera. The hole size I came up with was 12 mm, which suggests that the view angle of the camera is in the region of 60°.



In the end I included a moustache shape for the Lense Disc, and, as it is a glued insert, you could include whatever design you wanted for this.