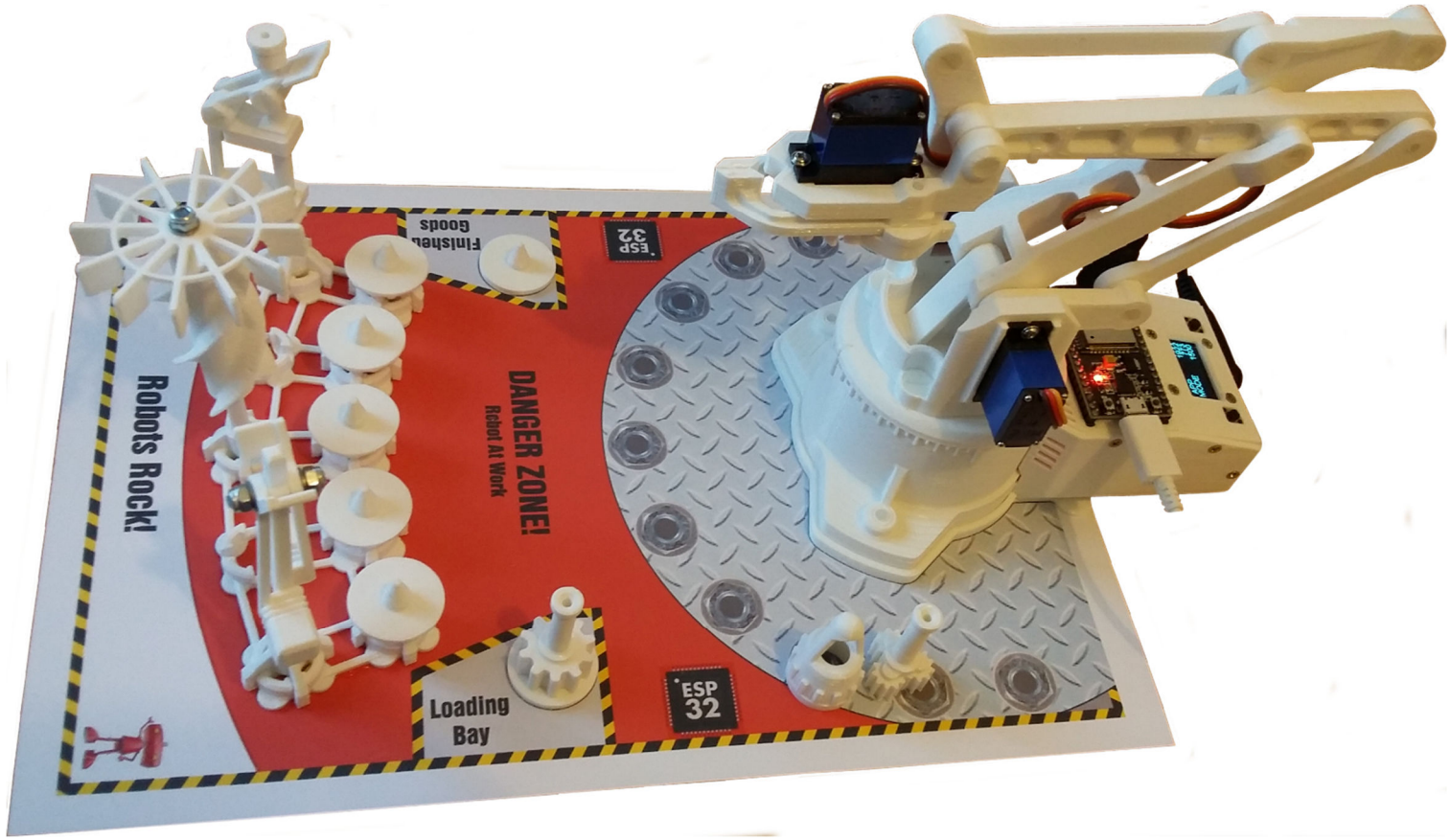
# Reach Robot Mk1

## Servo Calibration



## Why do we need to calibrate the servos?


- No two servos are the same
- Servos can be damaged if not setup correctly
- Course calibration must be performed prior to the assembly process
- This sets approximate positions for the leaver arms
- Course calibration ensure servos are within mechanical limits
- Fine calibration determines min/max robot physical limits
- The ESP32 C++ code needs limit values in order to work accurately
- Hence, each robot has a unique set of calibrated values


## Servo calibration is performed in three stages:

- Pre-set ensures mechanical parts are assembled correctly
- Fine calibration, performed during testing

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

# HJ Servo Consistency Tester

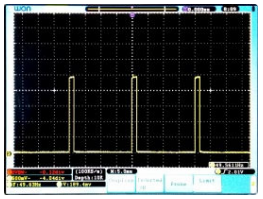

**Select Button:**

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

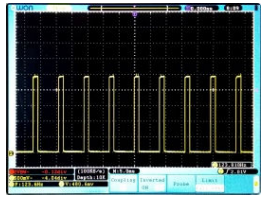
**Pulse Width Button:**

This is actually pulse frequency (Hz)

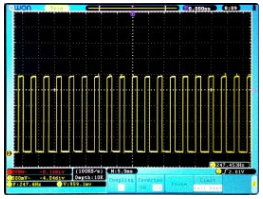
- 50H = 50 Hz - run MG90D at this frequency (default)
- 125H = 125 Hz
- 250H = 250 Hz



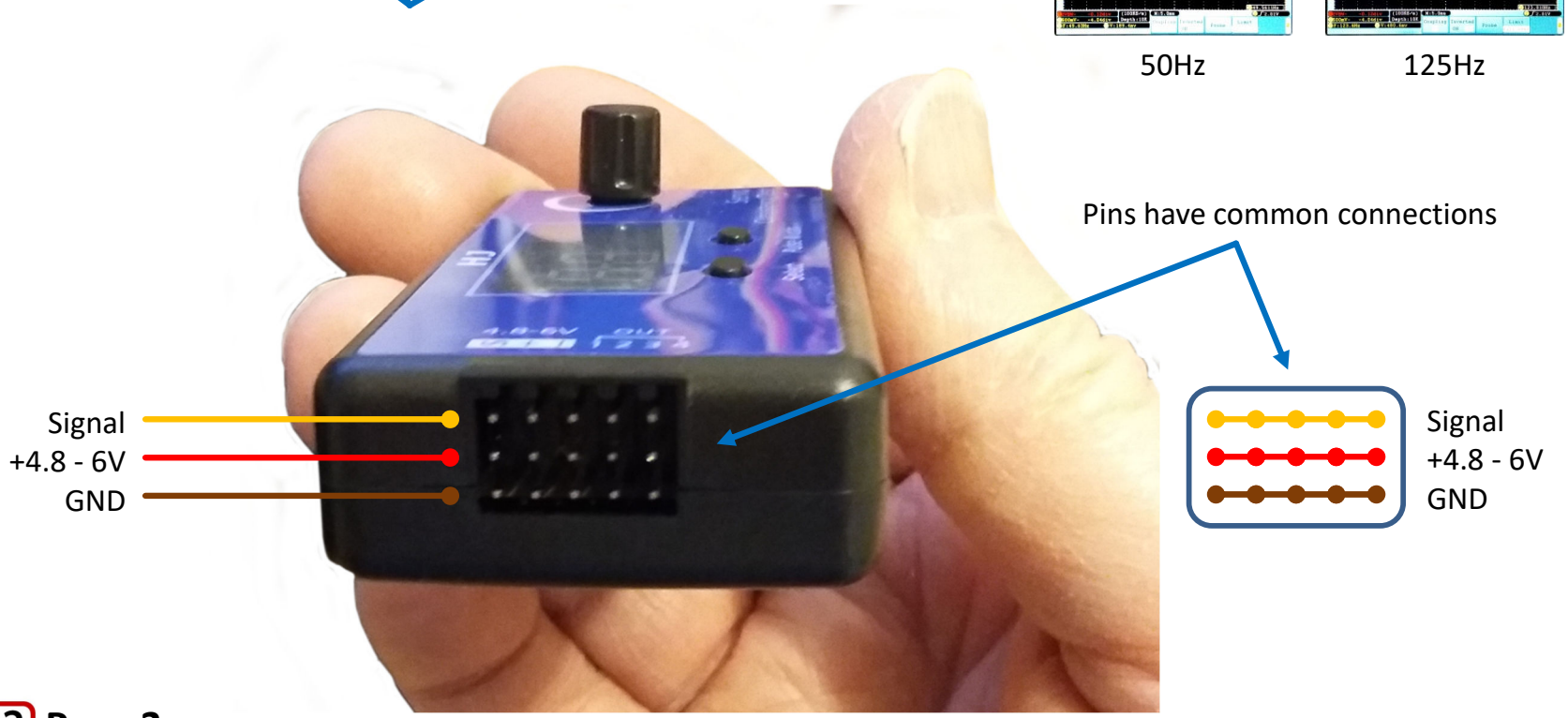
50Hz



125Hz



250Hz






Issue: 1.0 Released: 18/03/2021

## Servo Pre-set For Assembly:

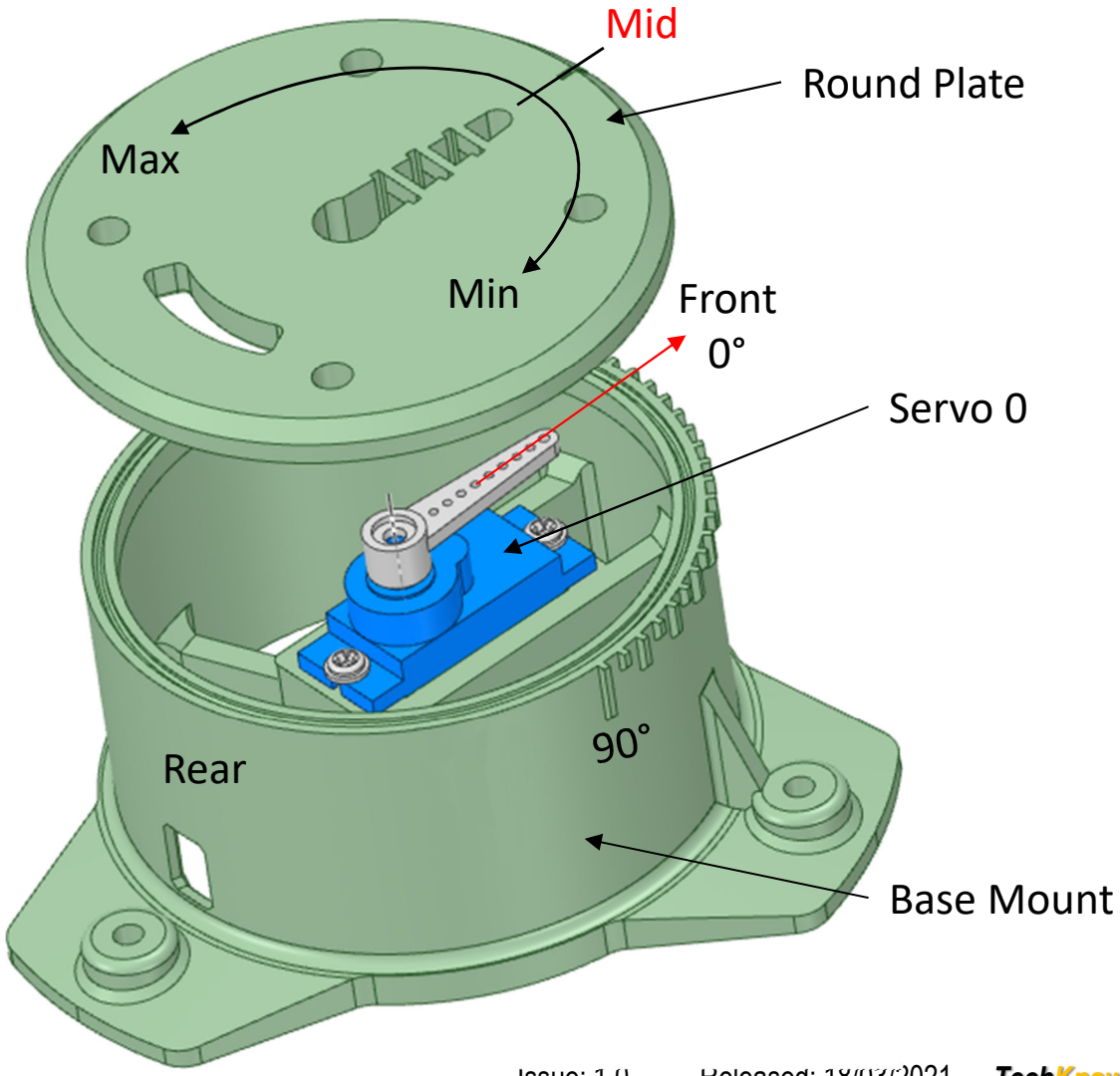
This ensures that attached mechanical part will have sufficient range.

- Select the correct servo. ie. Servo 0 for turntable
- Connect servo to consistency tester and set appropriate value
- Attach mechanical part to the servo in the position indicated


## Servo 0 - Pre-set For Assembly:

- Attach Servo 0 to the Base Mount
- Set Servo 0 pulse width to 1500µs (Mid)
- Attach servo leaver arm facing the front, 0°
- Attach round plate in forward facing position
- Attach pulse width 800 – 2200 µs to ensure equal swing

Note: Servo splined shaft has 20 teeth. So the arm can only be attached in 18° intervals. The closest compromise position for 0° has to be found for the servo arm.

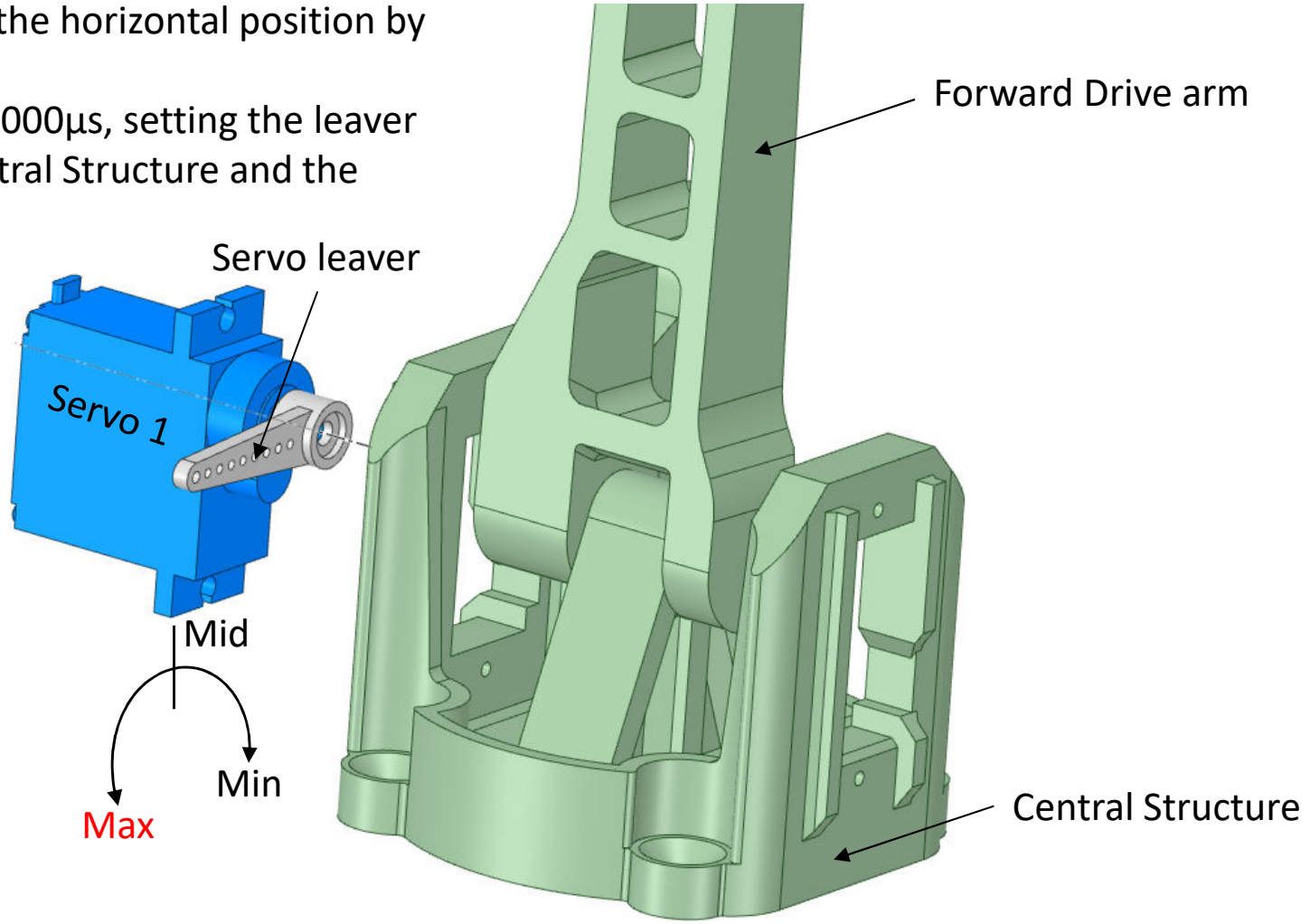





Issue: 1.0 Released: 18/03/2021 TechKnowTone

# Servo 1 - Pre-set For Assembly:


- Ensure Servo 1 with leaver attached vertically will engage snuggly with Forward Drive arm, but not tight.
- Then withdraw it from assembly.
- Set Servo 1 pulse width to 2200µs(Max)
- Firmly attach servo leaver in the horizontal position by its centre screw.
- Set Servo 1 pulse width to ~1000µs, setting the leaver vertical, for fitting to the Central Structure and the Forward Drive arm aperture.
- Attach with fixing screws.

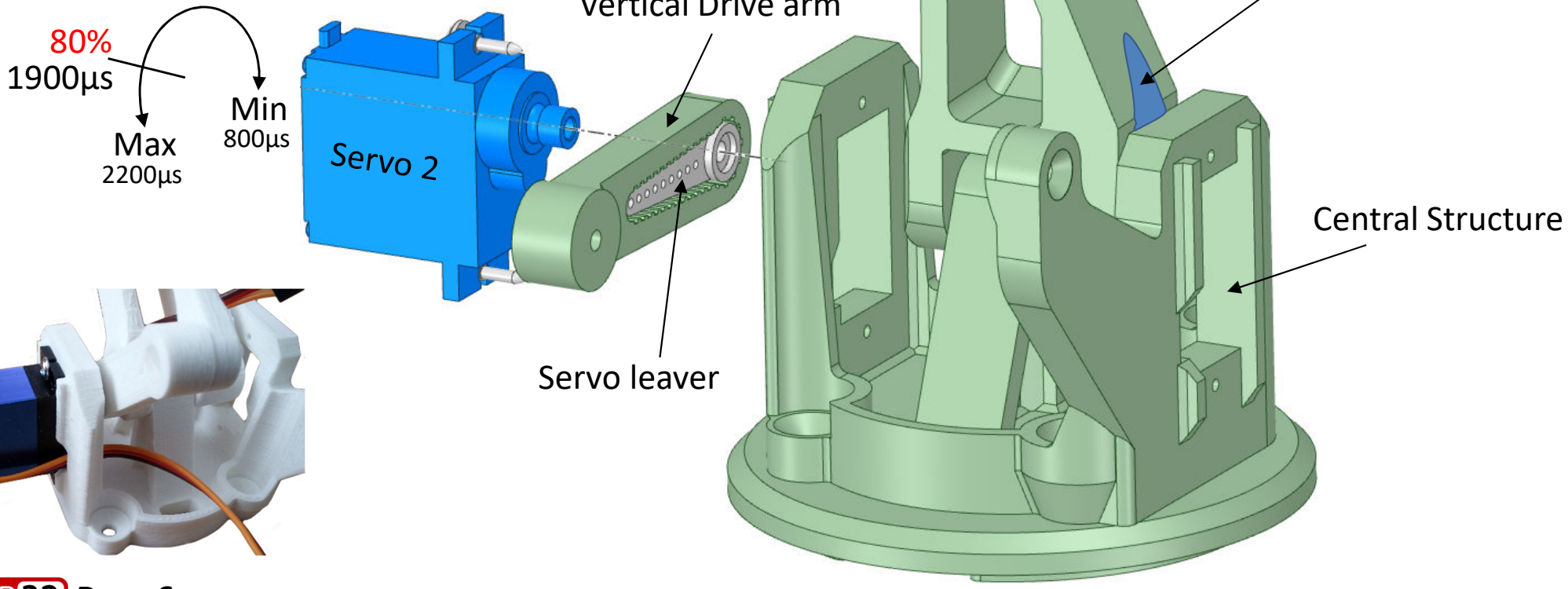





Issue: 1.0 Released: 18/03/2021 TechKnowTone

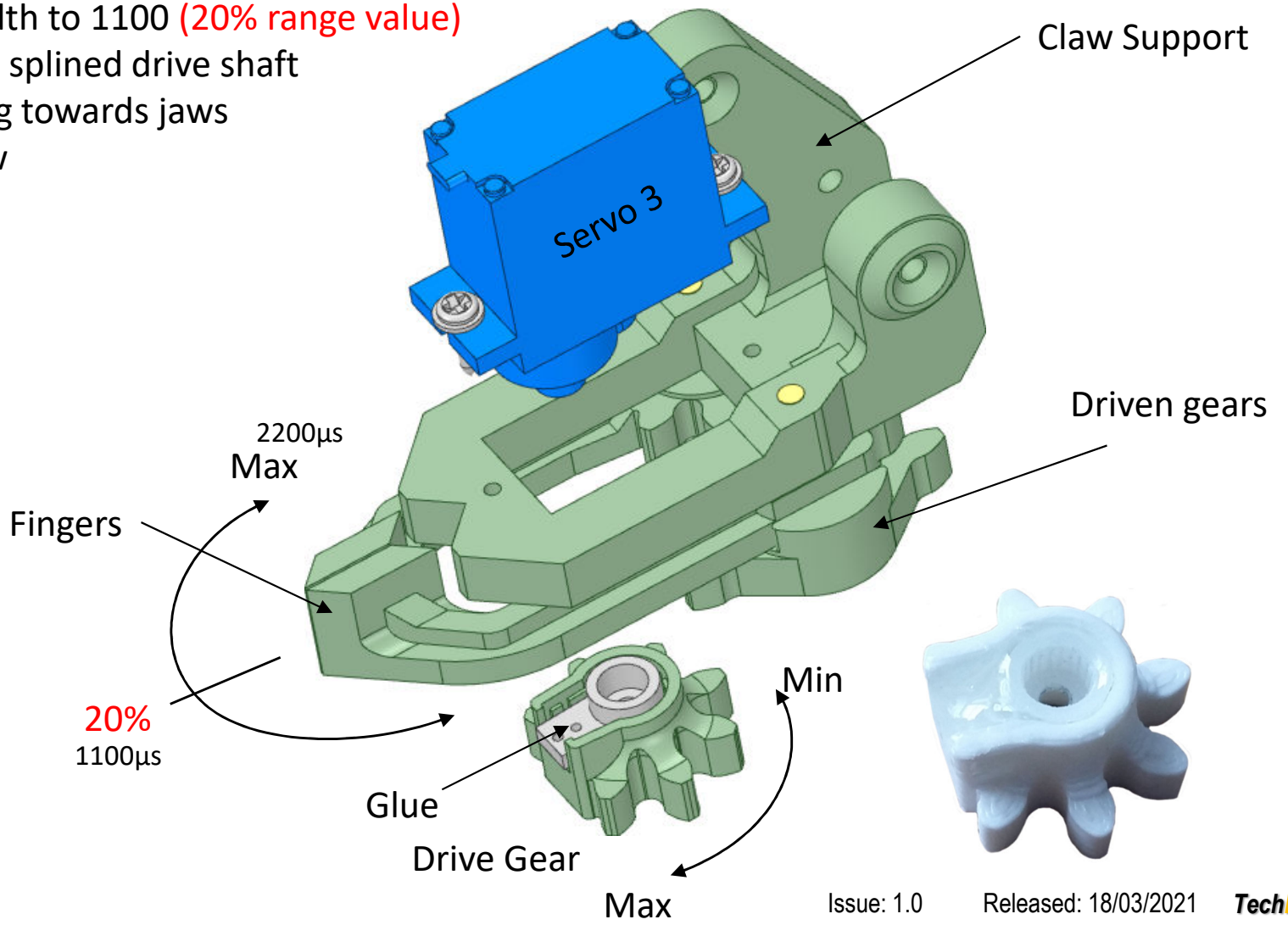
# Servo 2 - Pre-set For Assembly:

- Glue servo leaver into Vertical Drive arm with epoxy
- Allow time for the glue to fully harden (12 hrs)
- Set PWM pulse width to 1900µs (80% range value)
- Locate arm on splined shaft at a right angle as shown
- Tighten servo centre screw
- Set PWM pulse width to 1000µs to rotate the arm
- Feed cable and servo into the Central Structure
- Attach the servo with fixing screws provided

# Servo 3 - Pre-set For Assembly:


- Glue leaver into drive gear with epoxy
- Allow glue to fully harden
- Mount Servo 3 and fingers onto Claw Support
- Set Servo 3 pulse width to 1100 (20% range value)
- Locate Drive Gear on splined drive shaft
- Cropped leaver facing towards jaws
- Tighten locking screw






Issue: 1.0 Released: 18/03/2021 TechKnowTone

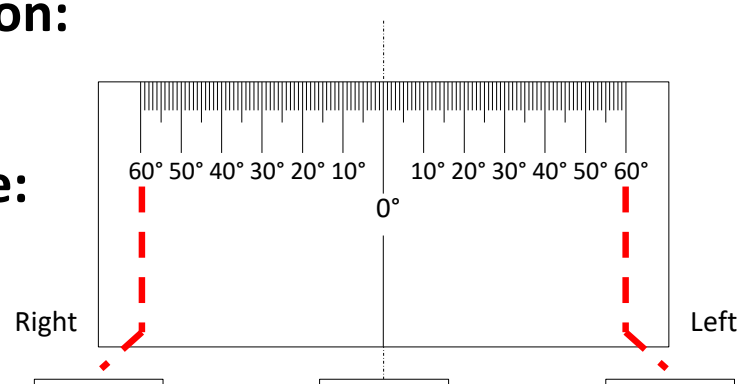
# Servos Fine Calibration:

Servo Calibration Record Sheet

**EXAMPLE**

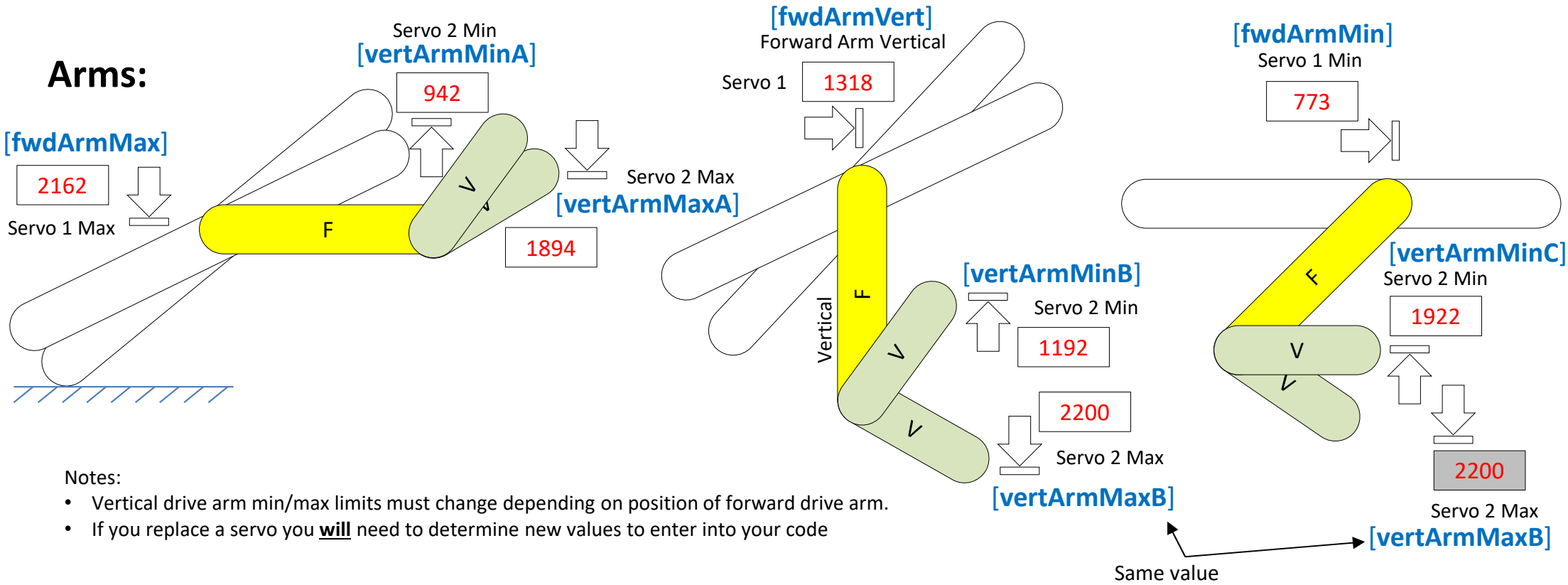
These are the numbers I obtained from my servos and used in the ESP32 code provided. You will determine different values for your servos.

## Turntable:




Servo 0: 838 [turntableMin] | 1500 Centre [turntableCtr] | 2062 [turntableMax]

## Arms:




Notes:

- Vertical drive arm min/max limits must change depending on position of forward drive arm.
- If you replace a servo you **will** need to determine new values to enter into your code



Issue: 1.0 Released: 18/03/2021 TechKnowTone

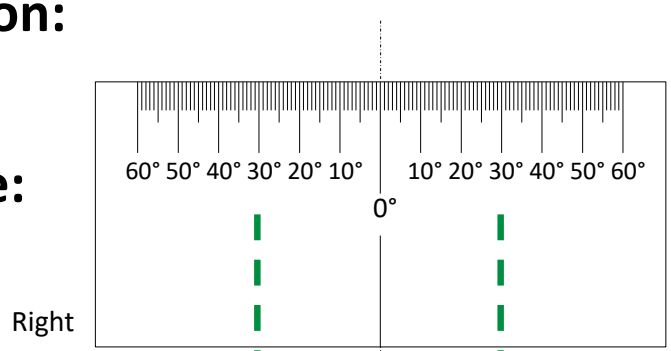
# Servos Fine Calibration:

Servo Calibration Record Sheet

## DEFAULT

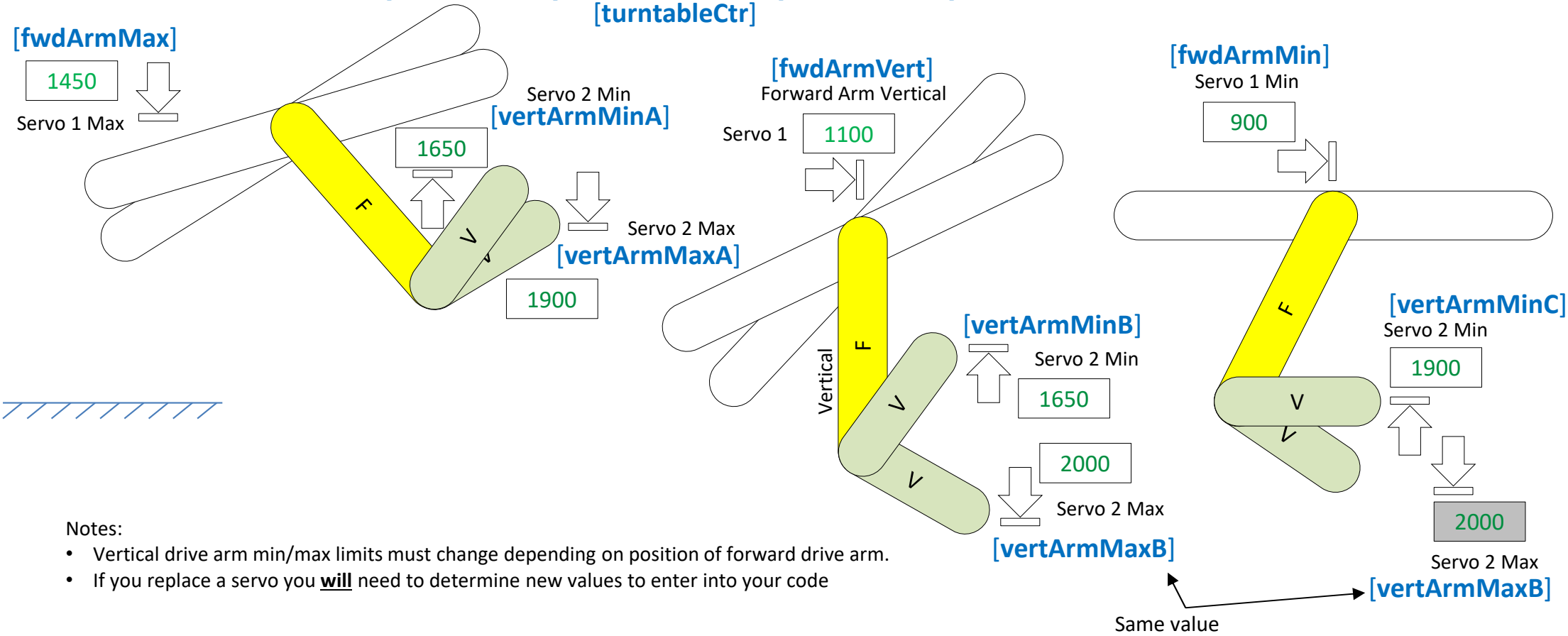
These are the numbers listed as default values. They should be your starting point, and should be 'safe' values. You will determine the correct values for your servos as you calibrate your robot.

## Turntable:




Servo 0: 1200 [turntableMin] | 1500 Centre [turntableCtr] | 1800 [turntableMax]

## Arms:




Notes:

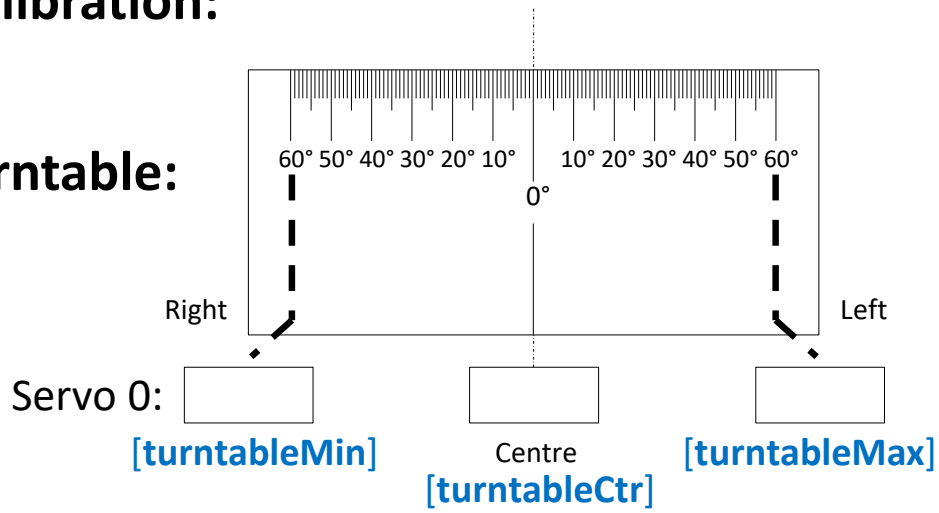
- Vertical drive arm min/max limits must change depending on position of forward drive arm.
- If you replace a servo you **will** need to determine new values to enter into your code



Issue: 1.0 Released: 18/03/2021 TechKnowTone

# Servos Fine Calibration:

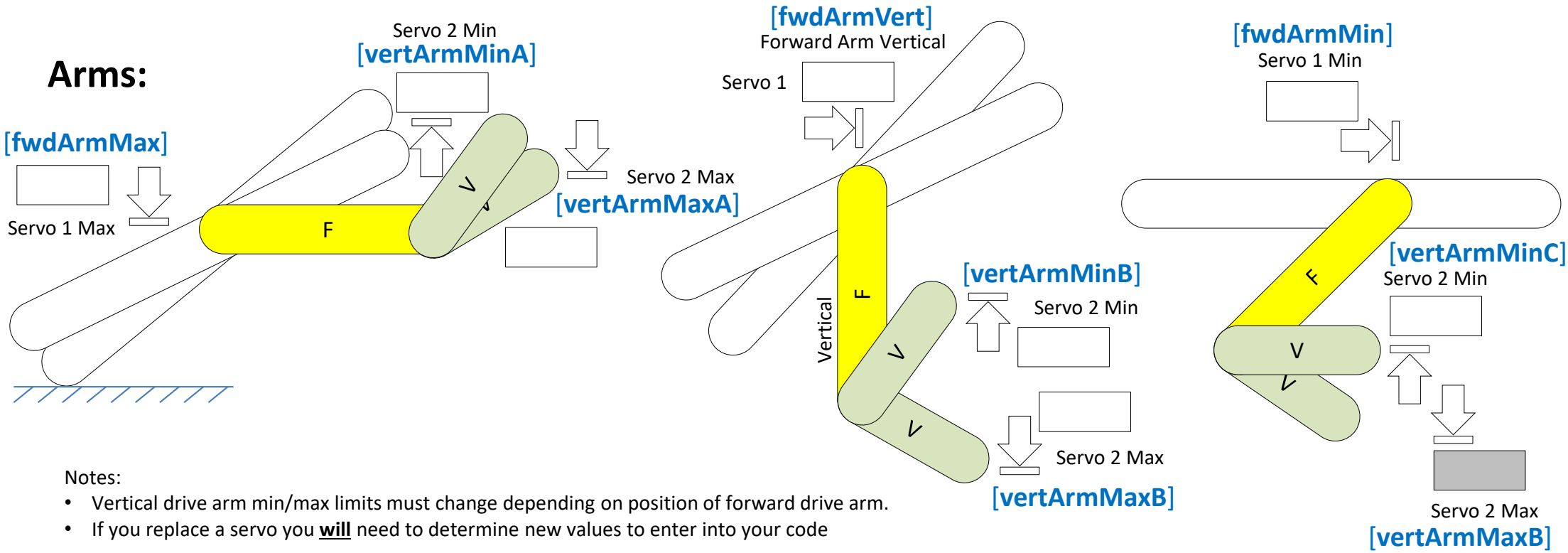
## Turntable:




## Servo Calibration Record Sheet

Use this sheet to record the values determined for your robot, or enter them directly into the code.

## Arms:




Notes:

- Vertical drive arm min/max limits must change depending on position of forward drive arm.
- If you replace a servo you **will** need to determine new values to enter into your code

# ESP32 Code Values:

- We use servo PWM values as constants in the Arduino code:

```
// Define servo calibration constants
#define fwdArmMax 2162    // forward arm Max servo value, default = 1450
#define fwdArmMin 773     // forward arm Min servo value, default = 900
#define fwdArmVert 1318   // forward arm vertical servo value, default = 1100
#define gripClose 1109    // jaws closed servo value, default = 1000
#define gripOpen 1500     // jaws moderately open value (23%), default = 1500
#define gripWide 2011     // jaws wide open values, default = 2000
#define turntableCtr 1500 // turntable servo centre value, default = 1500
#define turntableMax 2062 // turntable servo Max value, default = 1800
#define turntableMin 838  // turntable servo Min value, default = 1200
#define vertArmMaxA 1894  // vertical arm Max 'A' servo value, default = 1900
#define vertArmMaxB 2200  // vertical arm Max 'B' servo value, default = 2000
#define vertArmMinA  942  // vertical arm Min 'A' servo value, default = 1650
#define vertArmMinB 1192  // vertical arm Min 'B' servo value, default = 1650
#define vertArmMinC 1922  // vertical arm Min 'C' servo value, default = 1900
#define servoMinPWM  500  // minimum servo limit when uncalibrated
#define servoMaxPWM 2400  // maximum servo limit when uncalibrated

#define Floor0 turntableCtr   // floor position for servo 0
#define Floor1 1725           // floor position for servo 1
#define Floor2 1135           // floor position for servo 2
#define Floor3 gripOpen       // floor position for servo 3
#define Home0 turntableCtr    // home position for servo 0
#define Home1 fwdArmVert      // home position for servo 1
#define Home2 vertArmMinC     // home position for servo 2
#define Home3 gripClose       // home position for servo 3
#define PWR_tMax 3000         // 3 second time-out for auto-power off
#define PWR_tPing 500         // 500ms time-out on receipt of a ping character
#define Reset0 turntableCtr   // RESET position for servo 0
#define Reset1 fwdArmMin      // RESET position for servo 1
#define Reset2 vertArmMinC    // RESET position for servo 2
#define Reset3 gripClose      // RESET position for servo 3
#define servoOffMax 0         // sets maximum thermal drift offset for servo 0
#define servoOffRmpDwn 100000 // sets thermal offset ramp down time in miliseconds
#define servoOffRmpUp 10000   // sets thermal offset ramps up time in miliseconds
```

Servo Calibration Values

## EXAMPLE

These are the numbers I obtained from my servos and used in the ESP32 C++ code provided. You will determine different values for your servos.

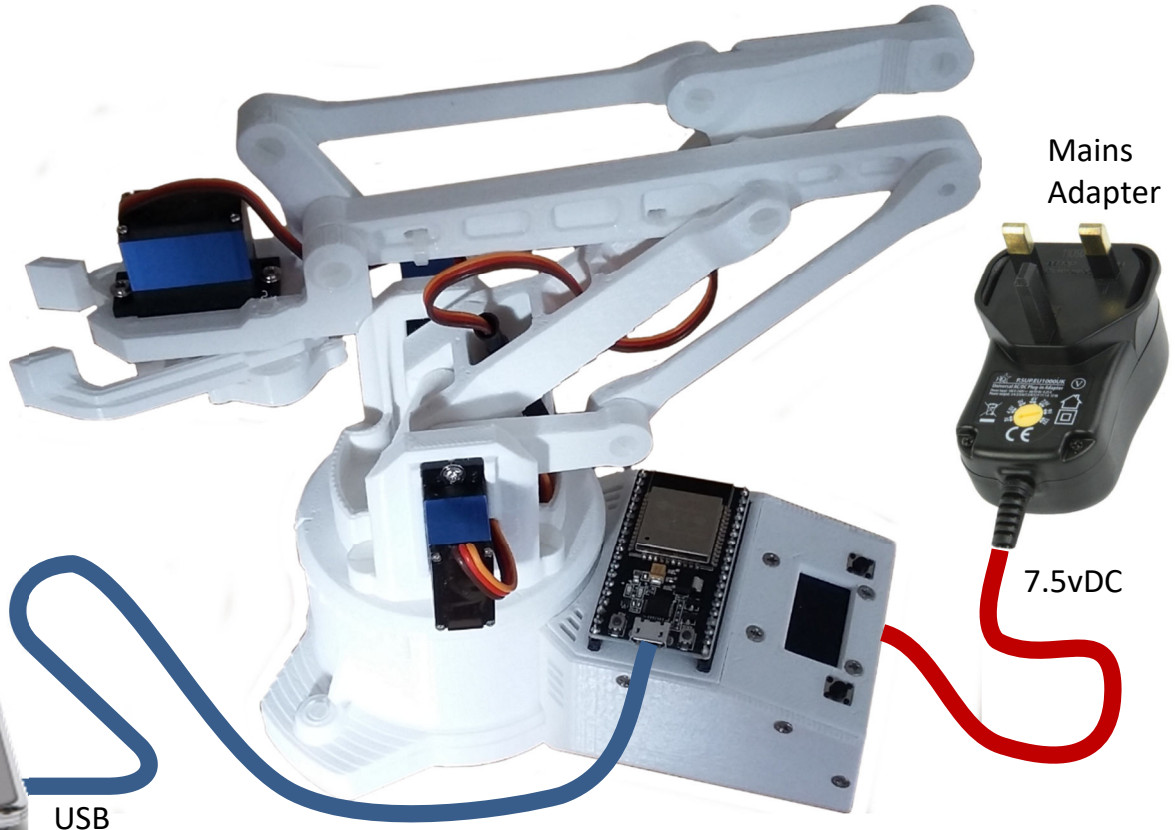
These values are determined during the ‘Fine’ calibration process and entered into the code as constants. Follow the instructions on the following pages to determine unique servo values for your robot.

These values are determined after the calibration process. You can determine your own ‘Home’ and ‘Reset’ co-ordinates using a Windows app.


‘Reset’


‘Home’

# Servo Fine Calibration:


Tools needed for Servo fine calibration:

- Windows PC, with Java runtime installed
- Arduino IDE; latest version with necessary libraries
- Download Software Code .zip file containing:
  - Servo Keyboard app: Servo_KeyBrd_Cntrl.exe
  - ESP32 .ino files: ESP32_Wii_Classic_Reach_Robot_00
- Extract files from .zip to respective folders
- Load the ESP32_Wii_Classic_Reach_Robot_00 file into the IDE
- This has my servo value, which you will replace with the ones you determine from this procedure.

**Note:** These instructions assume that you know how to use the Arduino IDE and have a basic understanding of writing C++ sketches, and transferring the compiled code into an ESP32. If you do not, you will need to learn this prior to completing these calibration tasks.

Your robot will need to be powered from a 7.5v mains adapter for the servo motors to function.

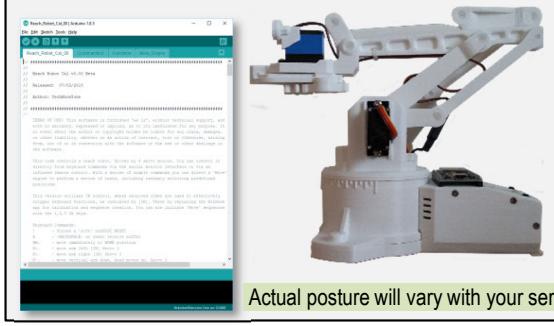
You can't use the IDE serial monitor or program the ESP32 whilst the keyboard app is running, as it hogs the USB interface.

# Servo Fine Calibration:

Note: some images are used from the original Reach Robot

01

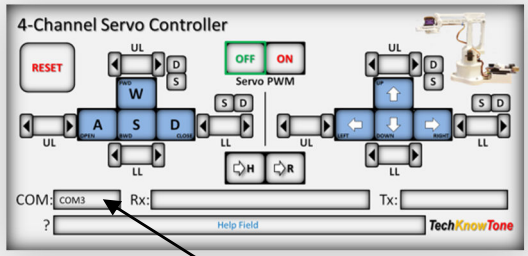
With the .ino file loaded the robot will adopt this posture…



Actual posture will vary with your servos

02

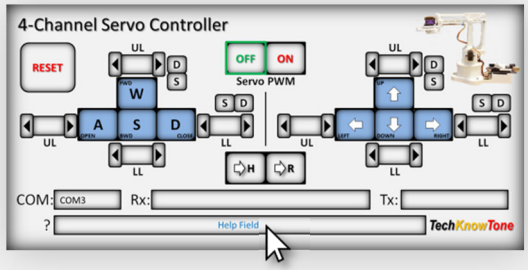
Now launch the keyboard Controller app…




This should connect to the ESP32 over the USB COM port

03

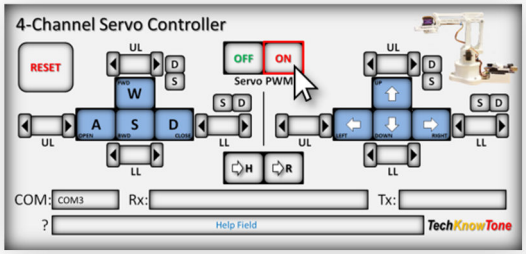
Move the mouse pointer to learn what buttons are for…




If the text is **red** then you are **not** connected to the Arduino…

04

To moved servos you must enable the PWM signals…




Once enabled PWM will auto-switch OFF after 2 seconds

05

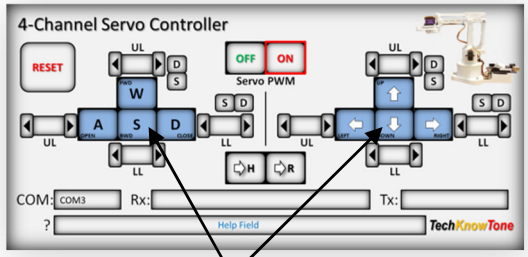
PWM auto-switch OFF is a feature of the ESP32 sketch code.

It serves to reduce overall power consumption, when the robot is not moving, prevents servos from over heating and ensures longer life.

The PWM signals will be turned back on again automatically, whenever a servo movement is required.

06

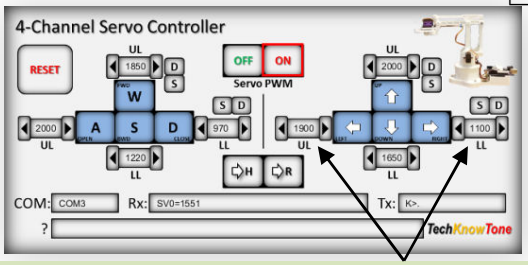
The blue coloured buttons correspond to keyboard keys…




You press keyboard keys or click on the app with the mouse

07

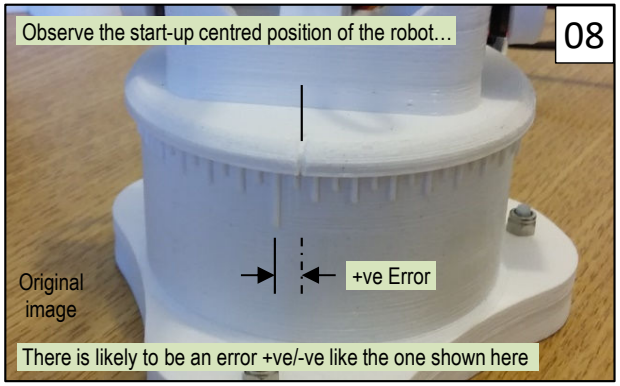
The robots servo movements are restricted to start with…




Upper & lower servo limits are loaded into this app from the ESP32
Click on **RESET** if this does not happen to force a reload.

08

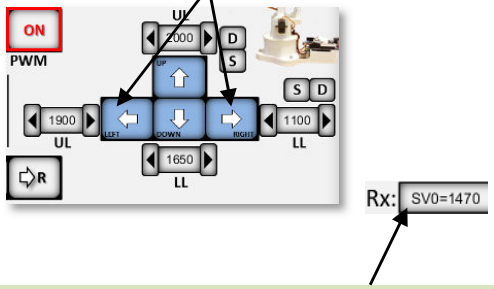
Observe the start-up centred position of the robot…




There is likely to be an error +ve/-ve like the one shown here

09

Use the left or right arrow keys to remove this error…

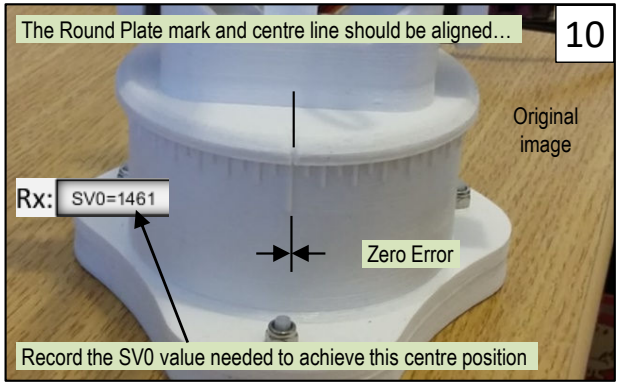



As you do so the Arduino will report the Servo 0 PWM value

# Servo Fine Calibration:

10

The Round Plate mark and centre line should be aligned…




Record the SV0 value needed to achieve this centre position

11

Find the default #define value in the .ino file…

```
#define turntableCtr 1500
```

Change this number to the one you have determined

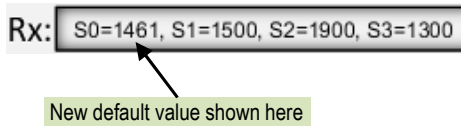
```
#define turntableCtr 1461
```

Compile and send the new file to your ESP32

You will need to stop/disconnect the keyboard app from USB COM!

12

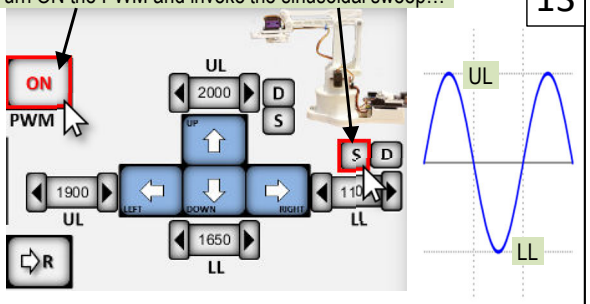
Relaunch/re-connect the keyboard app…




After Resetting, press the **SPACEBAR** to see the servo values

13

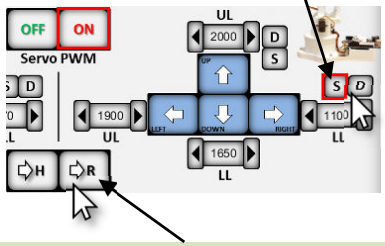
Turn ON the PWM and invoke the sinusoidal sweep…




S-Box highlight will be blue until sinusoidal sweep is in synch, Red

14

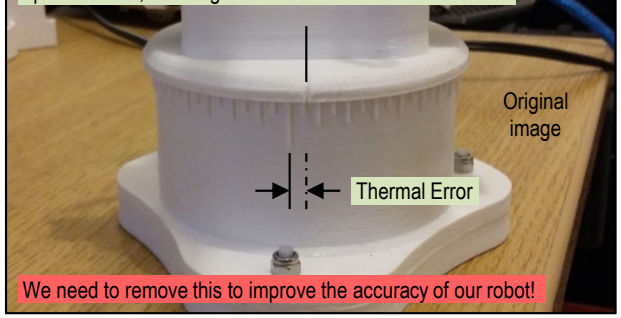
After 3 minutes turn OFF the sinusoidal sweep as the robot passes the centre position…




Then quickly click on the go to RESET position button

15

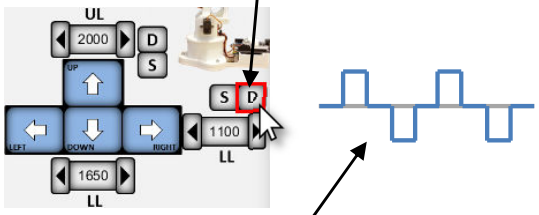
You are likely to see an offset, caused by Servo 0 heating up as it works, resulting in a thermal offset …




We need to remove this to improve the accuracy of our robot!

16

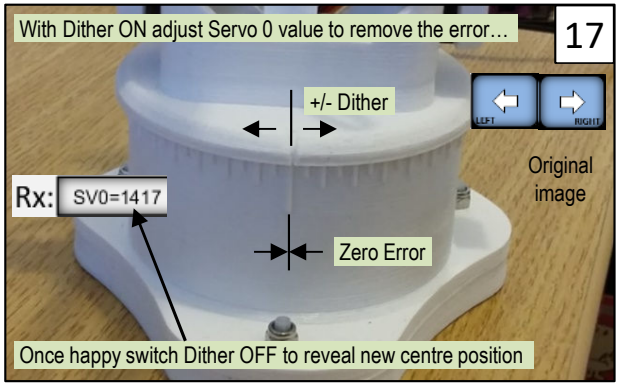
Quickly switch ON the Servo 0 Dither function…




Dither applies a small shifting function to the Servo 0 value

17

With Dither ON adjust Servo 0 value to remove the error…




Once happy switch Dither OFF to reveal new centre position

18

Find the default #define value in the .ino file…

```
#define servoOffMax 0
```

Subtract the latest value from the previous, 1461 – 1417 = 44

Change the default servoOffMax number to this value…

```
#define servoOffMax 44
```

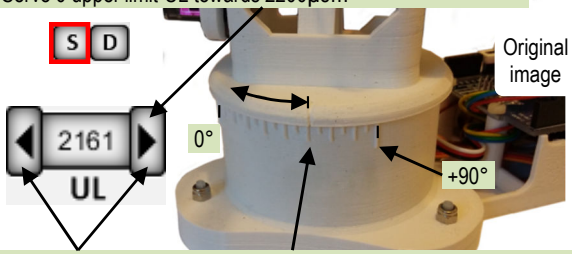
Compile and send the modified sketch file to your Arduino UNO

You will need to stop/disconnect the keyboard app from USB COM!

# Servo Fine Calibration:

19

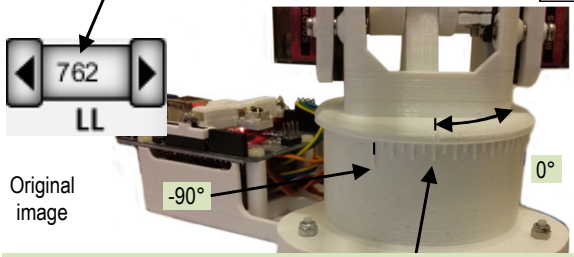
Turn ON PWM and Servo 0 sweep function, then Increase Servo 0 upper limit UL towards 2200µs…




Adjust Servo 0 UL value until it just reaches the +60° rotational limit at the extreme of travel. Note this value.

20

Decrease Servo 0 lower limit LL towards 750µs…




Adjust Servo 0 LL value until it just reaches the -60° rotational limit at the extreme of travel. Note this value.

21

Find these two default #define values in the .ino file…

```
#define turntableMax 1900
#define turntableMin 1100
```

Change the default numbers to your values like this…

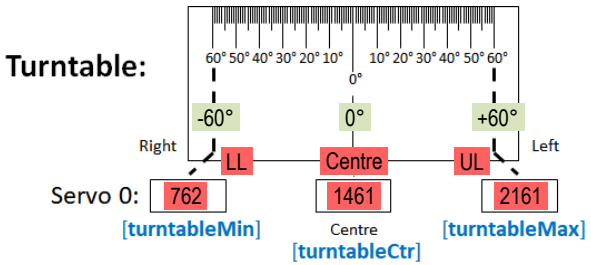
```
#define turntableMax 2161
#define turntableMin 762
```

Compile and send the new file to your Arduino UNO

Remeber to stop/disconnect the keyboard app from USB COM!

22

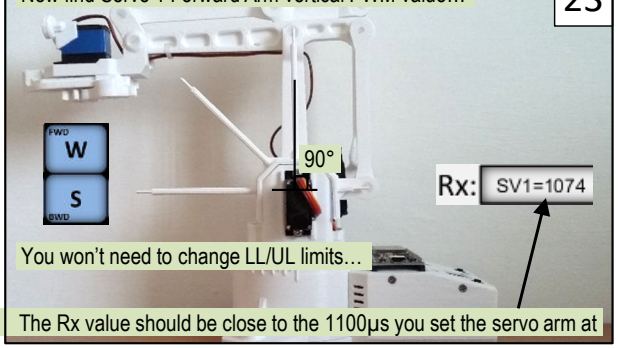
You have now determined the turntable values…




These three Servo 0 values are unique to your robot!

23

Now find Servo 1 Forward Arm vertical PWM value…

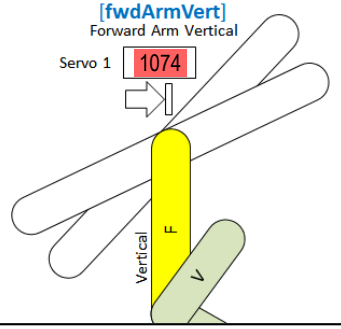



You won't need to change LL/UL limits…

The Rx value should be close to the 1100µs you set the servo arm at

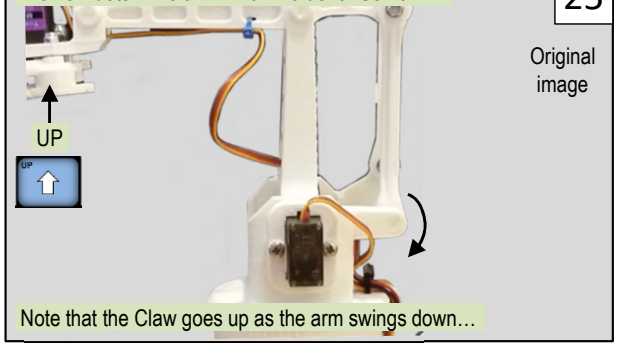
24

Record the Servo 1 value you have determined like this…




25

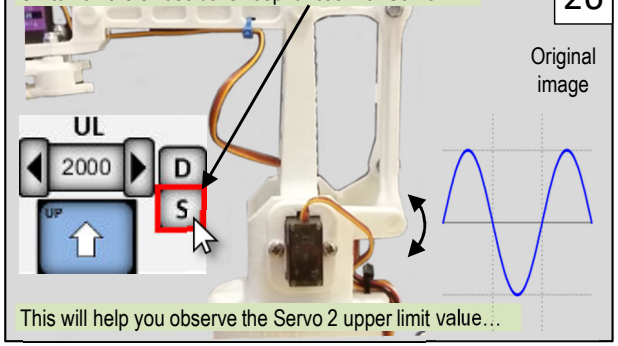
We now determine a minimum value for Servo 2…




Note that the Claw goes up as the arm swings down…

26

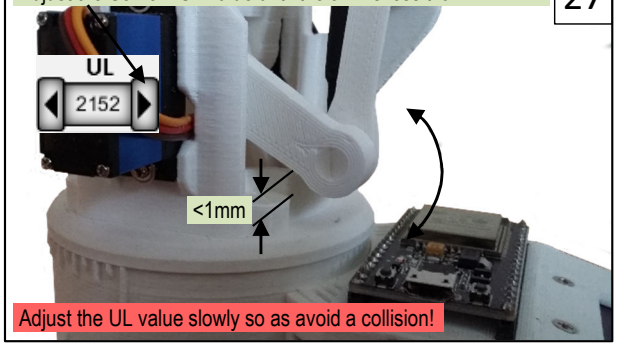
Switch on the sinusoidal sweep function for Servo 2…




This will help you observe the Servo 2 upper limit value…

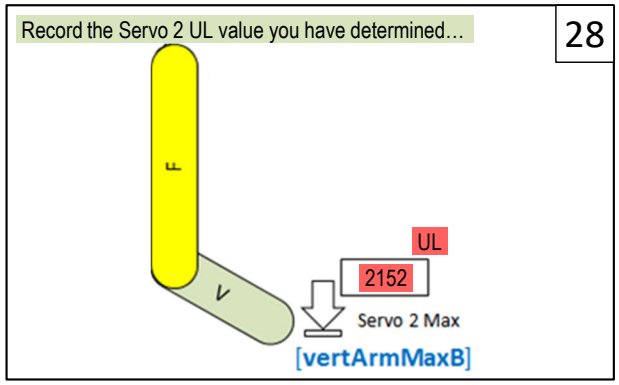
27

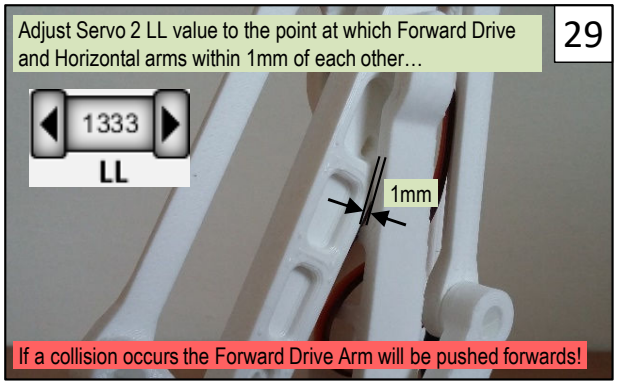
Adjust the Servo 2 UL value until the arm is less than 1mm…

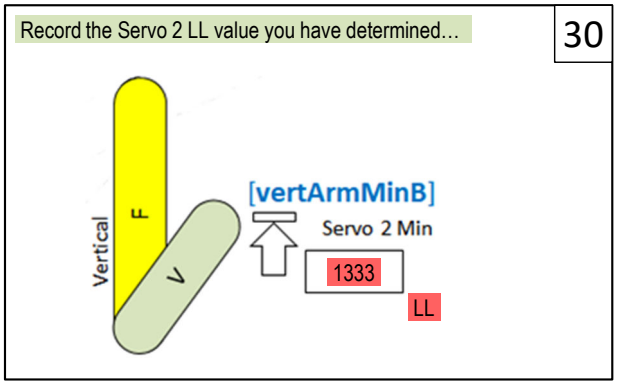


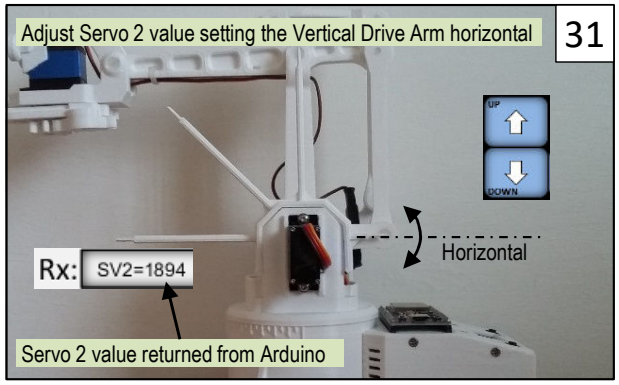

Adjust the UL value slowly so as avoid a collision!

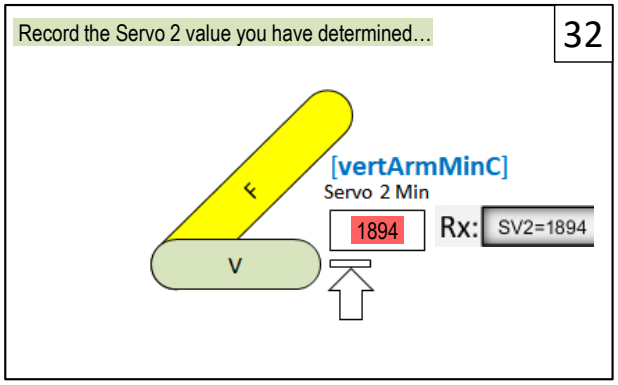
# Servo Fine Calibration:

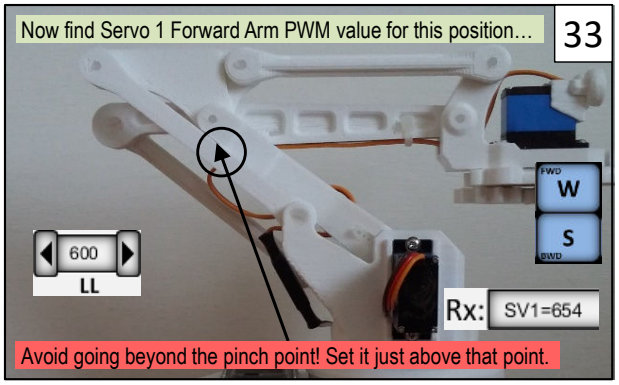


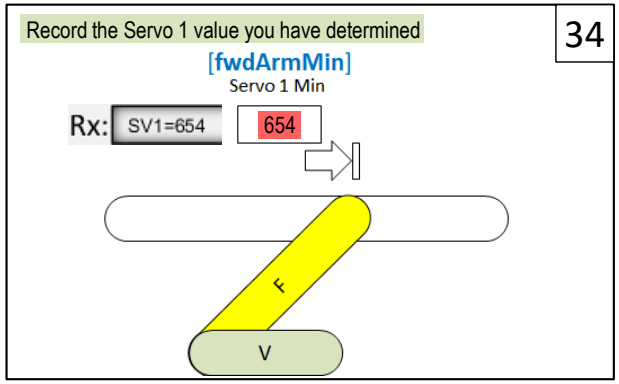



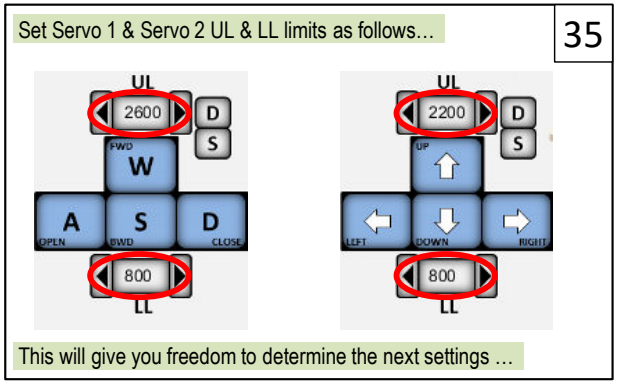



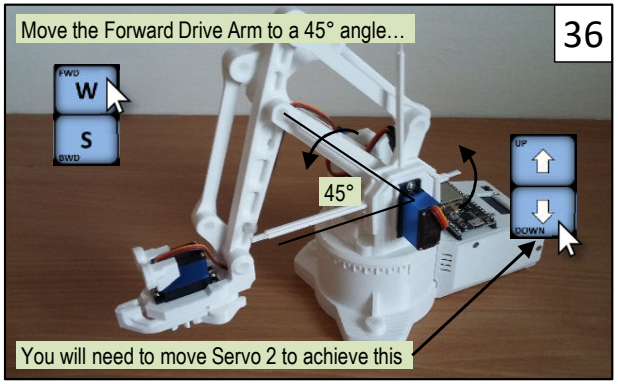



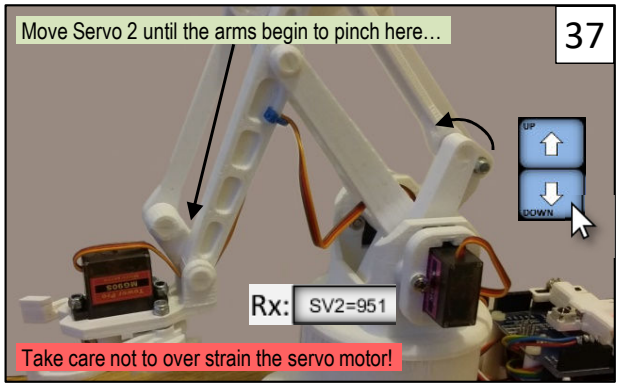



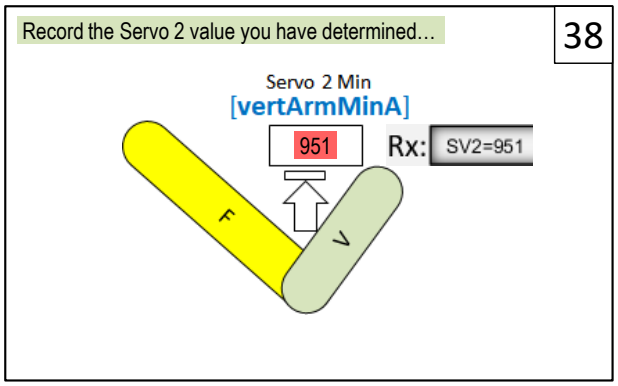



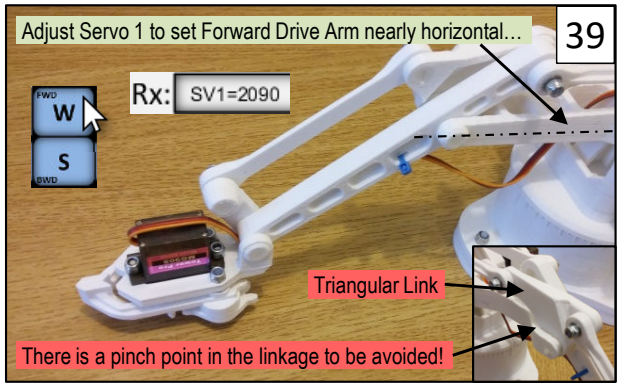



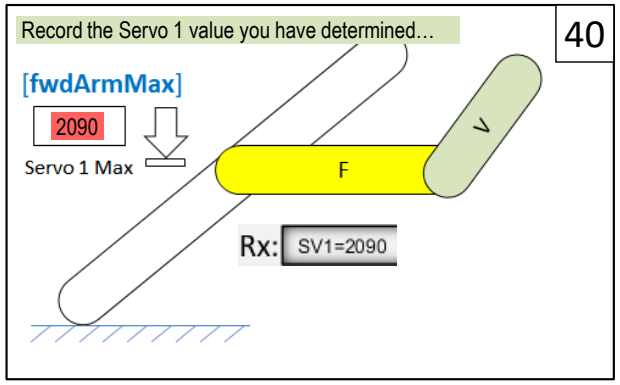



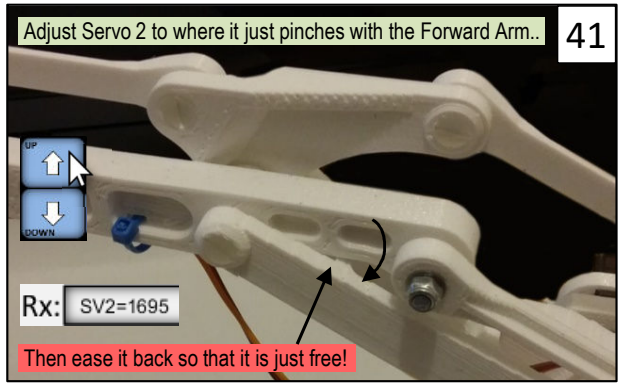



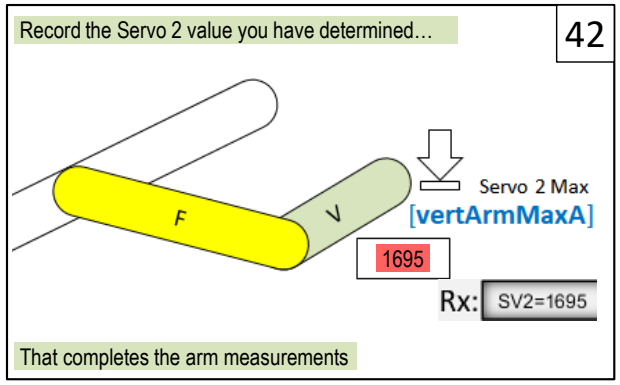

Move Servo 2 until the arms begin to pinch here…
37
UP
DOWN
Rx: SV2=951
Take care not to over strain the servo motor!

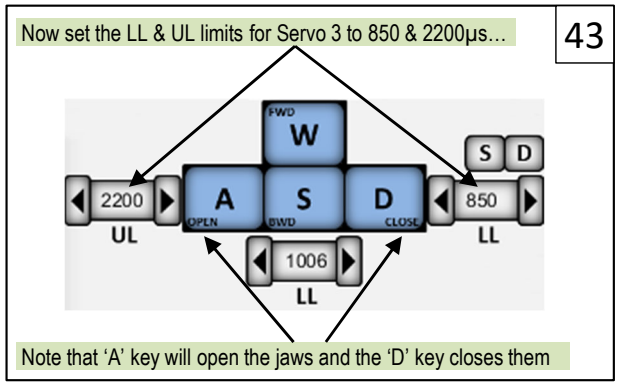

Record the Servo 2 value you have determined…
38
Servo 2 Min
[vertArmMinA]
951
Rx: SV2=951
F
V

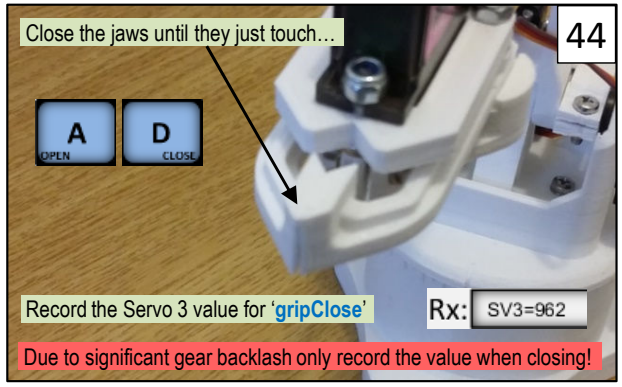

Adjust Servo 1 to set Forward Drive Arm nearly horizontal…
39
Rx: SV1=2090
FWD
W
S
BWD
Triangular Link
There is a pinch point in the linkage to be avoided!

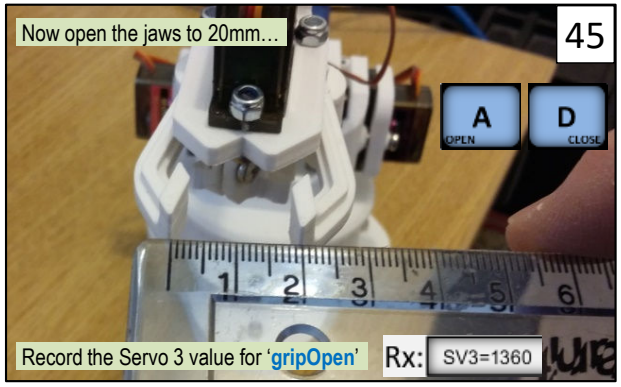

Record the Servo 1 value you have determined…
40
[fwdArmMax]
2090
Servo 1 Max
F
V
Rx: SV1=2090


Adjust Servo 2 to where it just pinches with the Forward Arm..
41
UP
DOWN
Rx: SV2=1695
Then ease it back so that it is just free!


Record the Servo 2 value you have determined…
42
F
V
Servo 2 Max
[vertArmMaxA]
1695
Rx: SV2=1695
That completes the arm measurements

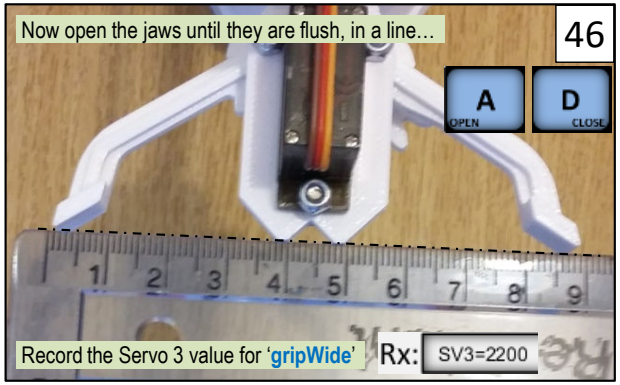

Now set the LL & UL limits for Servo 3 to 850 & 2200µs…
43
FWD
W
S
D
2200
UL
A
OPEN
S
BWD
D
CLOSE
850
LL
1006
LL
Note that 'A' key will open the jaws and the 'D' key closes them


Close the jaws until they just touch…
44
A
OPEN
D
CLOSE
Record the Servo 3 value for 'gripClose'
Rx: SV3=962
Due to significant gear backlash only record the value when closing!


Now open the jaws to 20mm…
45
A
OPEN
D
CLOSE
Record the Servo 3 value for 'gripOpen'
Rx: SV3=1360

Now open the jaws until they are flush, in a line…

46




Record the Servo 3 value for '**gripWide**'

Rx: SV3=2200

47

You must now enter all of your recorded values in your ESP32 sketch, replacing the values provided in the code.

This provides your robot with much greater freedom of movement, and important limits which are used by the code to ensure that the linkages work correctly.

See the code examples below comparing the default values with those determined in these measurements.

Finally change a flag value in the ESP32 sketch…

49

```
int Calibrated = false;
```

The variable 'Calibrated' can now be set to true…

```
int Calibrated = true;
```

This changes the behaviour of your code for the 'Move Engine'.

48

## Default Values

```
#define fwdArmMax 1450
#define fwdArmMin 900
#define fwdArmVert 1100
#define gripClose 970
#define gripOpen 1400
#define gripWide 2000
#define turntableCtr 1500
#define turntableMax 1900
#define turntableMin 1100
#define vertArmMaxA 1900
#define vertArmMaxB 2000
#define vertArmMinA 1650
#define vertArmMinB 1650
#define vertArmMinC 1900

#define Home0 turntableCtr
#define Home1 1200
#define Home2 1900
#define Home3 gripOpen
#define Reset0 turntableCtr
#define Reset1 1200
#define Reset2 1900
#define Reset3 gripOpen
#define servoOffMax 0
#define servoOffRmpDwn 60000
#define servoOffRmpUp 10000
```

## Calibrated Values

```
// Define servo calibration constants
#define fwdArmMax 2090 // forward arm Max servo value
#define fwdArmMin 654 // forward arm Min servo value
#define fwdArmVert 1074 // forward arm vertical servo value
#define gripClose 942 // jaws closed servo value
#define gripOpen 1360 // jaws moderately open value (23%)
#define gripWide 2200 // jaws wide open value
#define turntableCtr 1461 // turntable servo centre value
#define turntableMax 2161 // turntable servo Max value
#define turntableMin 762 // turntable servo Min value
#define vertArmMaxA 1695 // vertical arm Max 'A' servo value
#define vertArmMaxB 2152 // vertical arm Max 'B' servo value
#define vertArmMinA 951 // vertical arm Max 'A' servo value
#define vertArmMinB 1333 // vertical arm Max 'B' servo value
#define vertArmMinC 1894 // vertical arm Max 'C' servo value

#define Home0 turntableCtr // home position for servo 0
#define Home1 1200 // home position for servo 1
#define Home2 1900 // home position for servo 2
#define Home3 gripOpen // home position for servo 3
#define Reset0 turntableCtr // RESET position for servo 0
#define Reset1 1200 // RESET position for servo 1
#define Reset2 1900 // RESET position for servo 2
#define Reset3 gripClose // RESET position for servo 3
#define servoOffMax 44 // sets maximum thermal drift offset for servo 0
#define servoOffRmpDwn 60000 // sets thermal offset ramp down time in miliseconds
#define servoOffRmpUp 10000 // sets thermal offset ramps up time in miliseconds
```

Given what you have learnt you can now determine and enter servo values for the 'Home' and 'Reset' positions…

50


Reset is normally the rest or powered down position…

51