

MeArm v0.4

Robot Arm



**Build documentation and usage
with a Raspberry Pi Maker Kit**

version 1.1

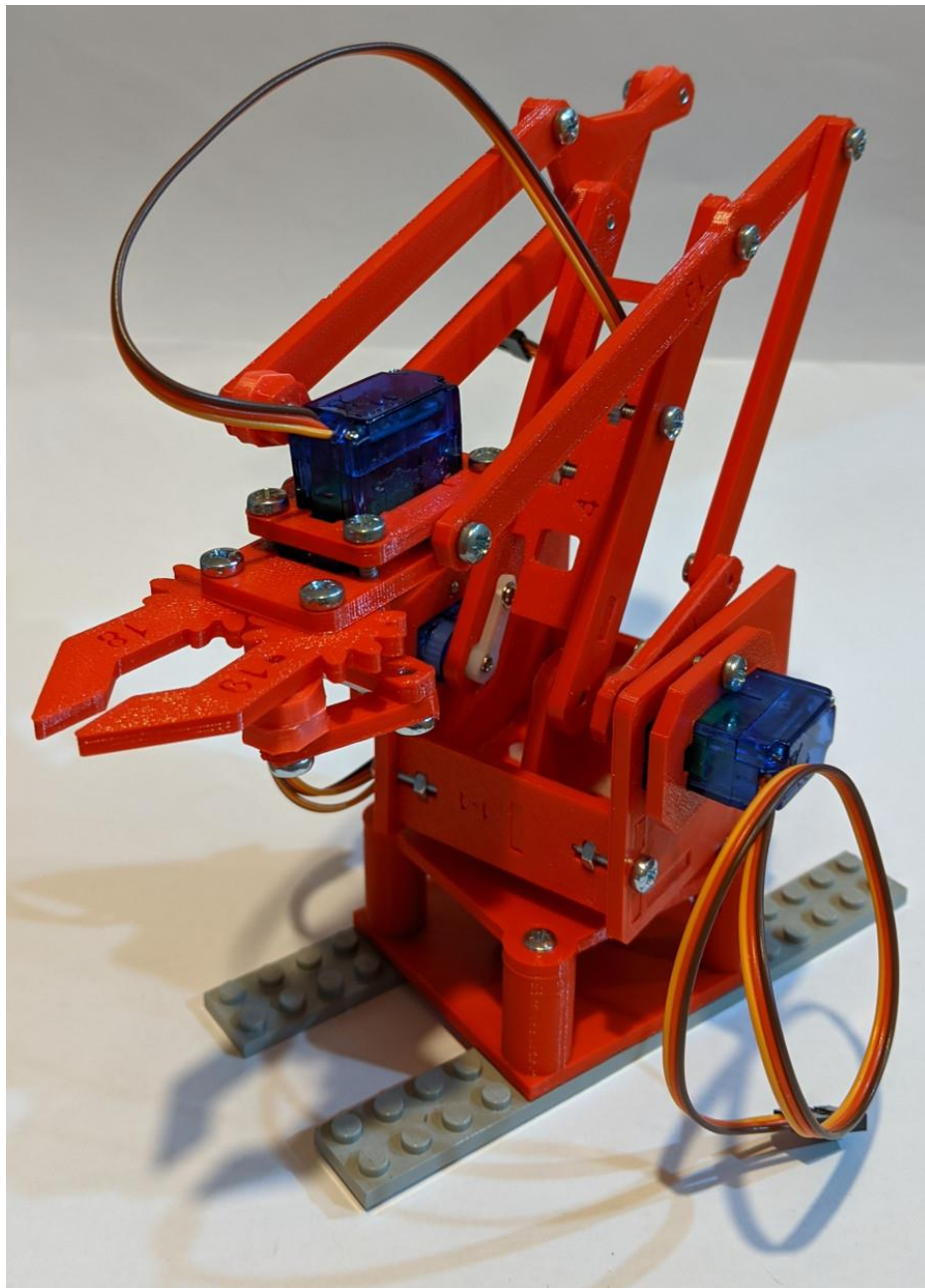
Table of contents

Introduction.....	3
3D printed components.....	4
All the components for the build.....	6
Robot Arm build steps	7
Step 1: Pre-build 3D component checks.....	7
Step 2a: Prepare the robot arm Lego base	8
Step 2b: Prepare the robot arm plain base	8
Step 3: Sandwich the 'base' servo between a bracket and part 27	9
Step 4: Parts to build the left-hand side of the robot arm.....	10
Step 5: Fit the left-hand 'elbow' servo to the robot arm	10
Step 6: Build the left-hand side drive lever	11
Step 7: Complete the build of the left-hand lever.....	11
Step 8: Parts to build the right-hand side of the robot arm.....	12
Step 9: Fit the right-hand 'shoulder' servo to the robot arm	13
Step 10: Build the right-hand side levers	13
Step 11: The parts for bringing the two sides together	14
Step 12: Prepare and pre-assemble all the middle parts	15
Step 13: Attaching the two robot arm halves.....	16
Step 14: Attach the assembled cradle to the base.....	17
Step 15: Add left and right 'forearms'.....	17
Step 16: The gripper components and servo assembly build.....	19
Step 17: the gripper jaws	20
Step 18: Connecting the robot arm to a Raspberry Pi Maker Kit	21
Raspberry Pi software.....	22
Raspberry Pi operating system:	22
Python robot arm management software:	23
Underlying library software:	23
MeArm motion control 'driver' functions:	24
Set up/demonstration software:	24
Appendix A: additional reference data	26
Appendix B: SG90 servo control with PWM.....	28
Appendix C: pulse width modulation (PWM)	29
Appendix D: I ² C connection to a Raspberry Pi.....	30
Appendix E: I ² C control and the PCA9685 board	31

Introduction

This Raspberry Pi Maker Kit controlled robot arm project, with an online description [here](#) and software + documentation [here](#) for v5.0 of the Maker Kit, is based upon the open hardware design by *phenoptix* in Technology Robots, first published 29th July'14.

This document, which provides detailed build and usage instructions for digital makers, will be updated periodically as further revisions are made to this project which was first started in July '18.

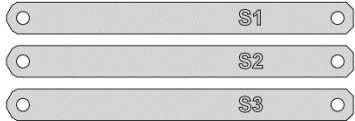
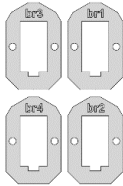
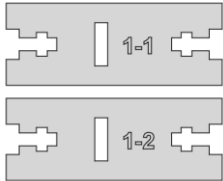
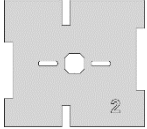
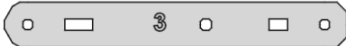
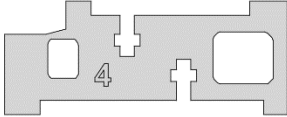

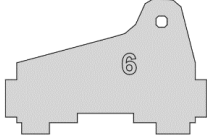

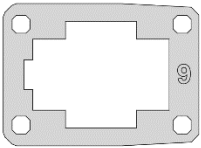
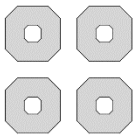
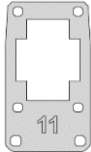







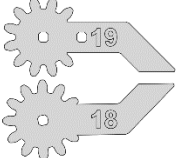
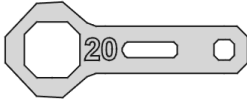

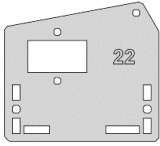
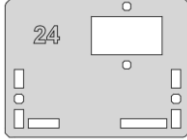
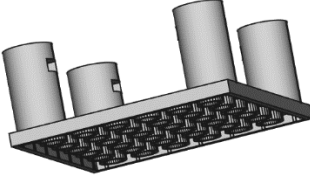
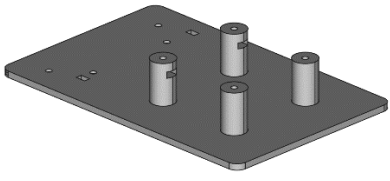
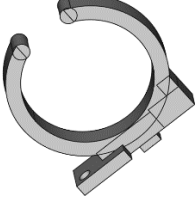
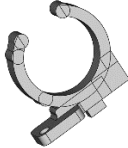
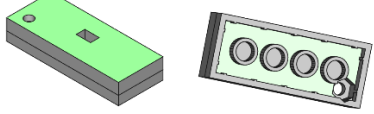
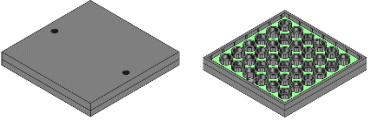
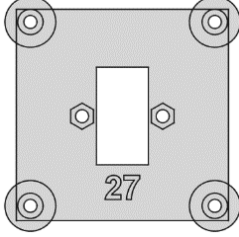
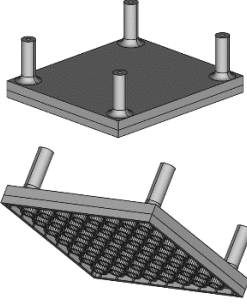
3D printed components

The original *phenoptix* design was for a set of laser cut components, so each item was inevitably a simple flat element made from 3mm thick sheet. These laser cut components were however converted to 3D printable components by *Dingo_aus* and posted to the 'thingiverse' web site in December '14 (full details [here](#)) but retained their 'flat' nature without really taking any of the advantages that 3D printing can bring to a design.

Starting from the .stl files created by *Dingo_aus*, further updates and additions have been made for this build as well as some files being 'repaired'. Many more refinements are certainly possible with a full 3D printed approach now made possible, and further design iteration may be made at a future date, but the series of labelled thumbnail images below provide a reference and easy identification list of all the added and updated 3D printed components which now includes an optional LEGO compatible base which allows the robot arm to be set into an overall LEGO construct.

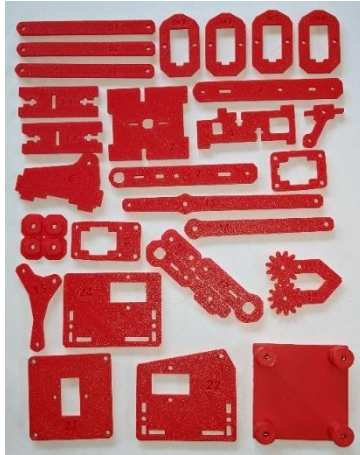
The part numbering, which has been 'embedded' into each 3D printed component, still maintains the original system approach - and full details of all the updated 3D print designs are available at the Prusa web site [here](#).

		
3x servo arms	4x servo brackets	2x part 01
		
part 02	part 03	part 04
		
part 05	part 06	part 07
		
part 09	parts 10, 12, 23 & 25	part 11

		
part 13	part 14	part 15
		
part 16	part 17	parts 18 and 19
		
part 20	part 21	part 22
		
part 24	part 26a – LEGO tile base	part 26b – plain base
		
2x 32mm battery clamps	2x 22mm battery clamps	2x battery clamp LEGO tile
		
4AA battery holder LEGO tile	part 27	RPi Maker Kit LEGO tile

All the components for the build

The components for both the build of the robot arm and its 'control' connection to a Raspberry Pi Maker Kit are described below.



Robot arm main build components:

- One set of PLA 3D printed components, as shown above left and described in the previous section (part 26b and other optional components like the battery clamps are not shown in the image above).
- 4 x SG90 servos, each with a set of 'horns' and 'horn' fixing screws.
- M3 (metric 3mm) nuts and 'pozi' pan head screws as follows (excludes the nuts and screws for the optional components listed next):
 - 6mm x 9 ○ 8mm x 12
 - 10mm x 3 ○ 12mm x 11
 - nuts x 12

Optional 3D printed parts and additional fixings:

- 4x self-adhesive, 12mm dia/8mm high, rubber feet for use with the plain base, part 26b.
- 2x 32mm battery clamps for optional use with a 5000mAh 5V/2A battery bank fixed to the plain base, part 26b, with 2x M3 nuts and 2x M3 8mm 'pozi' pan head screws.
- 2x 22mm battery clamps for optional use with a 3350mAh 5V/1A battery bank fixed to the plain base, part 26b, with 2x M3 nuts and 2x M3 8mm 'pozi' pan head screws.
- 2x M3 nuts and 2x M3 6mm CSK head screws for fixing a 4x AA battery holder to the plain base part 26b.
- individual battery clamp LEGO tiles, with a M3 nut and M3 8mm 'pozi' pan head screw to fix a clamp to a tile – a pair of tiles with attached clamps are used to hold a battery bank.
- 4AA battery holder LEGO tile with 2x M3 nuts and 2x M3 6mm CSK head screws for fixing the battery holder to the tile.
- Raspberry Pi Maker Kit LEGO tile with 4x M2 self-tap 6mm long flanged pan head screws

Computer control components:

- PCA9685 PWM servo controller board.
- 4 off of 3x20cm male-to-female jumper leads to extend the reach of the SG90 connector leads to the PCA9685 inserted into the Maker Kit PCB.

The only other items needed are a Raspberry Pi (any of the single board computer versions that can connect to a Raspberry Pi Maker Kit), and either a 5V rechargeable battery bank or a battery holder for four AA batteries – nickel metal hydride (Ni-Mh) rechargeable batteries are recommended.

Before starting your build, you should check that you have all the required parts, and whilst no additional soldering is required if an already assembled Maker Kit PCB is being used, the use of a general range of 'making' tools is needed.

Robot Arm build steps

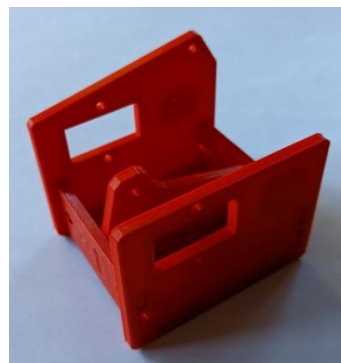
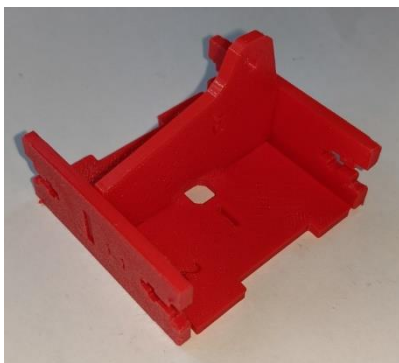
Step 1: Pre-build 3D component checks

Before starting the build, you need to check that several of the 3D printed parts can be 'smoothly' slotted together. You may need to gently sand the edges of the tabs of the various parts just a small amount to get a good fit and this is best done at the start of the build since, whilst in the final build the connections are reinforced by bolting them together, you will need to be able to easily insert them stage by stage through the various steps of the build.

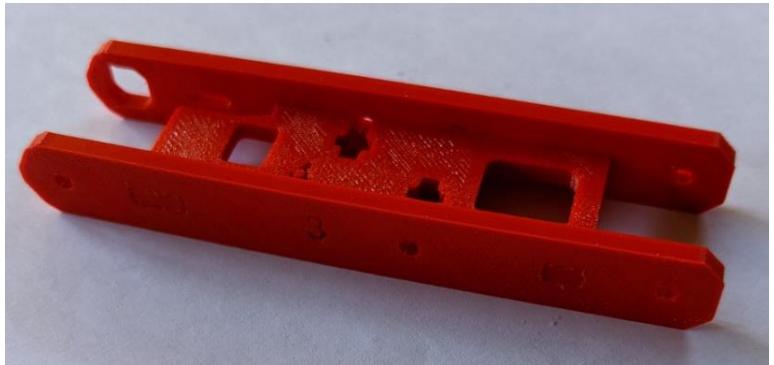
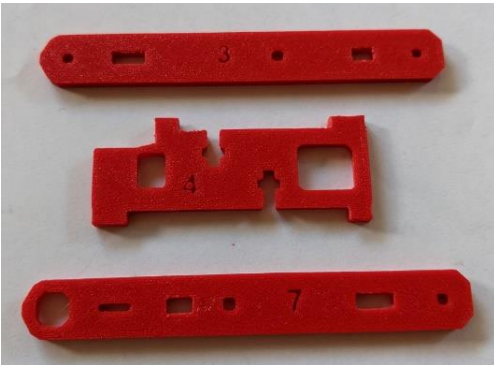
The first set of parts to be pre-checked are parts 24, 22, 2x 01 (i.e., 1-1 + 1-2), 02 and 06, as shown in the image below left. As a first step check that part 06 will slot into part 02 as shown in the image below right. You need to take care that this is fitted the right way around and using the image as a reference, the corner of part 02 with the '2' marking is at the bottom left of the image.



Then, as shown in the image below left the two parts 01 slot onto part 06. Parts 24 and 22 are fitted as shown in the image below right. Take care to make sure the side plates, parts 22 and 24, are fitted the right way around.



Finally, in a similar way you must also ensure that part 04 'smoothly' connects to parts 03 and 07 as illustrated in the images below.

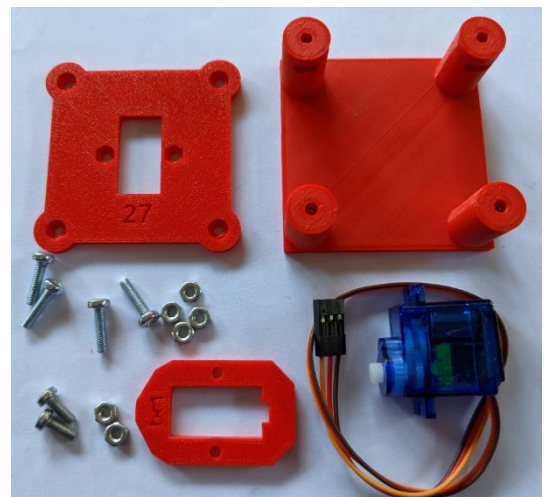


Step 2a: Prepare the robot arm Lego base

If the LEGO tile base (part 26a) is to be used, then for this step, and the subsequent step 3, you will need the following components:

- 3D printed:
 - part 26a (LEGO tile base)
 - part 27
 - servo bracket br1
- 1x SG90 servo
- 6x M3 nuts
- 4x 12mm 'pozi' pan head screws
- 2x 8mm 'pozi' pan head screws

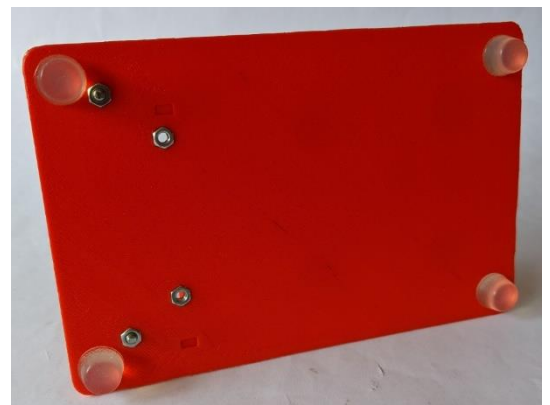
These parts are shown in the image on the right.



Step 2b: Prepare the robot arm plain base

However, if the plain base (part 26b) is to be used, then for this step and the subsequent step 3, you simply use part 26b instead of 26a with all the other parts as shown above remaining the same.

But as shown on the right, the 4 x self-adhesive rubber feet are used in addition with one adhered in each corner of the underside of the plain base. In addition, M3 nuts are inserted into hexagonal openings on the underside to support the fixing of either the 32mm or 22mm battery clamps, or the 4AA battery holder.

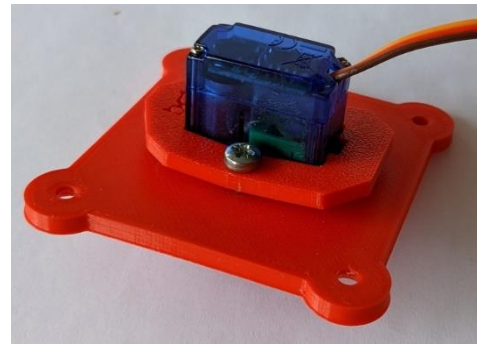
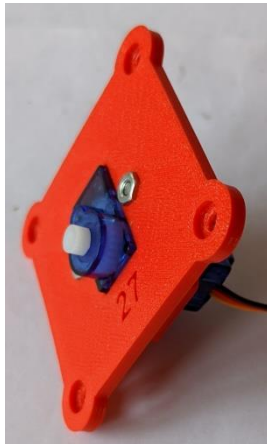
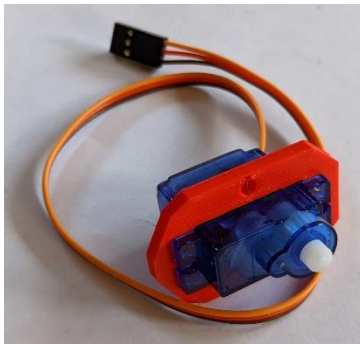


Step 3: Sandwich the 'base' servo between a bracket and part 27

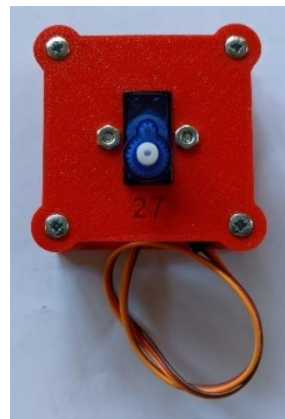
The servo brackets are used in a similar way to attach three of the servos to the robot arm – there is a spare fourth bracket in the set of 3D printed parts since it is possible to 'crack' the bracket if you overtighten the fixing screws.

For the servo to be fitted to either type of base plate, which will (obviously!) be called the 'base' servo:

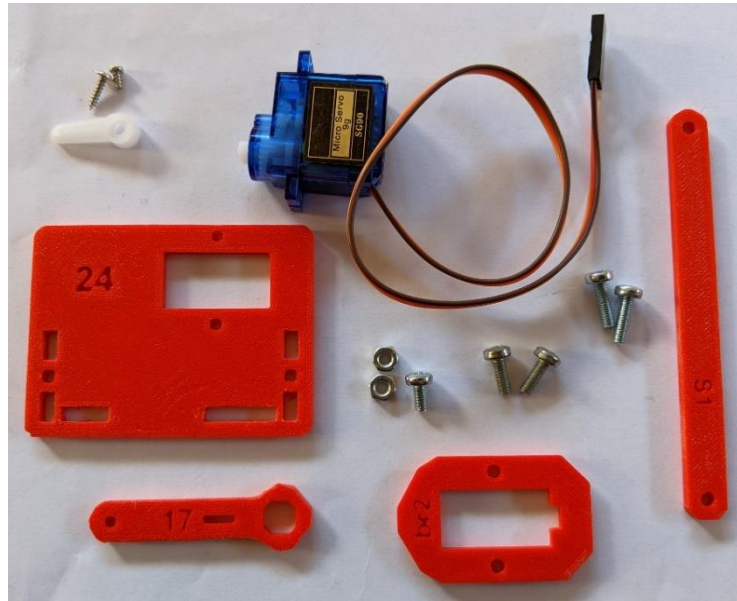
- Thread the connection wire of the servo through the bracket, as shown below left.
- Line the cut out on the bracket up with the end of the servo where the wires attach.
- Bring the servo bracket over the bottom of the servo.
- Push home so it's flat with the flange on the servo - this will be a tight fit and you may have to remove the sticky labels on either side of the servo to get the bracket to fit.
- Now complete the 'servo sandwich' with part 27, as shown below right, using 2x 8mm 'pozi' pan head screws that insert through to 2x M3 nuts that are inserted into the side of part 27 which is labelled.



- Finally insert 4x M3 nuts into the cylindrical stands on either the plain or LEGO tile base as shown on the right and then use 4x 12mm 'pozi' pan head screws to fix the 'servo sandwich' to either the plain base (part 26b) or the LEGO tile base (part 26a) as shown below.



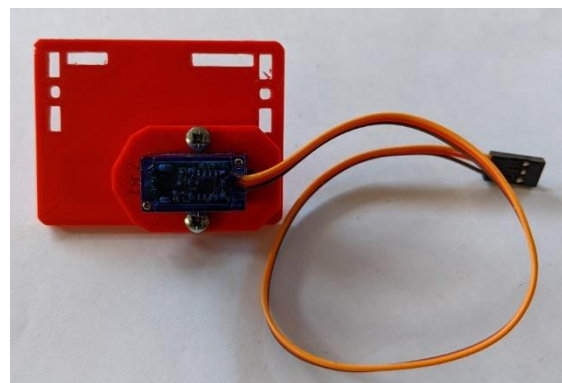
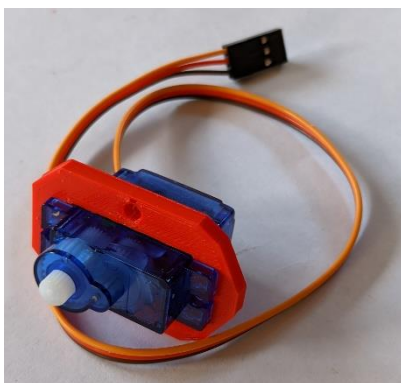
Step 4: Parts to build the left-hand side of the robot arm



The components for this step and the subsequent steps 5-7 are shown above and are listed below:

- 3D printed parts 24, 17, one of the servo brackets (br2), and one of the three servo arms (S1).
- 1 x SG90 servo along with a single arm 'horn' plus one long and one short horn fixing screw.
- 2 x 8mm M3 'pozi' pan head screws.
- 2 x 12mm M3 'pozi' pan head screws.
- 2 x M3 nuts.
- 1 x 6mm M3 'pozi' pan head screw.

Step 5: Fit the left-hand 'elbow' servo to the robot arm



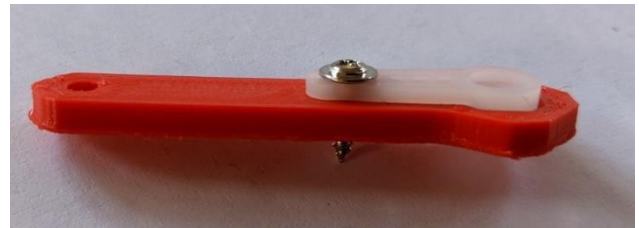
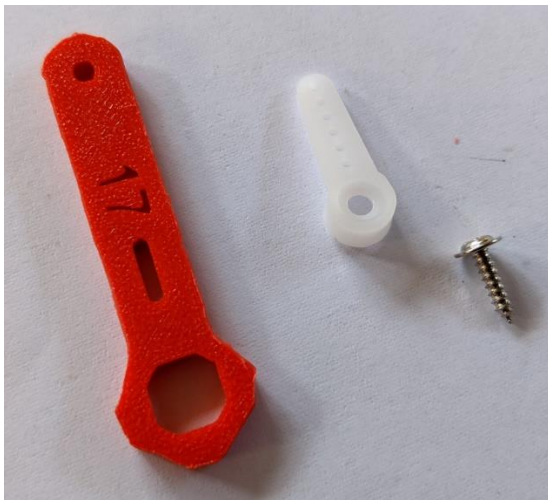
The servo fitted to the left-hand side of the robot arm is known as the 'elbow' servo. To first fit the servo to a servo bracket, as shown in the image above left, you thread the servo wires through the servo bracket (just as you did for the 'base' servo) and then

screw the bracket onto the side plate, part 24, using the two 8mm screws, which for this servo simply self-tap into the hole in part 24.

Pay careful attention to the orientation here. Note the direction of the wire and the way the servo pokes out from the side plate.

Now thread the 12mm screws through the round holes that are between the small slots on the side plate, with the screws' heads on the side of the plate where the servo is attached and put the nuts on the other side of the plate just a few turns to keep the screws in place.

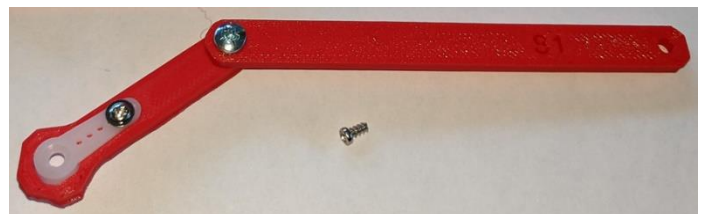
Step 6: Build the left-hand side drive lever



Attach the single-arm servo horn to the servo lever part 17 (as pictured) using the long servo 'horn' screw. The screw will poke out the back of the resulting combined drive lever, and as this a bit 'spikey' you should trim/file the pointed end off once you are confident the combined part is put together correctly.

Step 7: Complete the build of the left-hand lever

Now with the 6mm screw first attach the servo arm to the combined drive lever with the screw first going through the servo arm and then self-tapping into drive lever with the servo arm and pan head on the same side as the servo horn on the drive lever. The image on the right shows you how this should look.

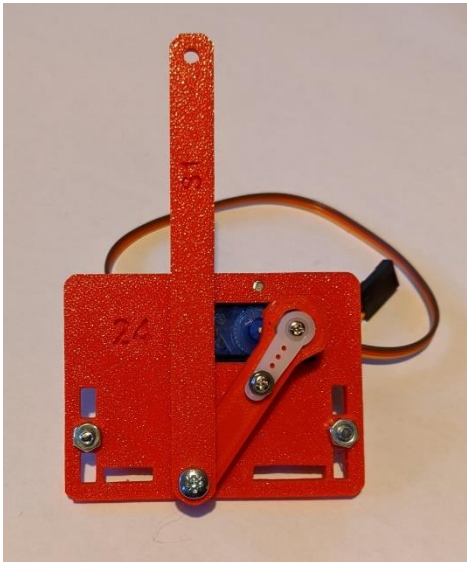
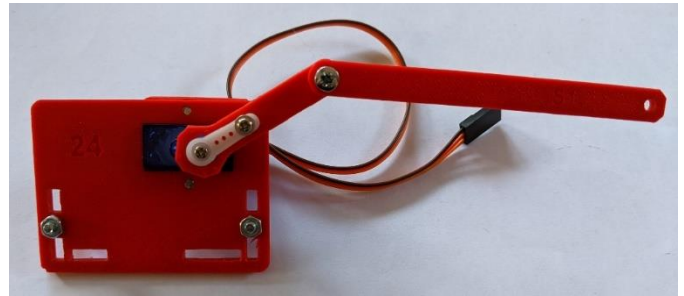


This is the first of many more moving parts. It can be adjusted later but it is important that the servo arm can rotate with very little force. A balance needs to be struck between the servo arm not moving sideways but still moving freely. Remember, every bit of force used to overcome friction in these joints is less force available to lift objects.

If it is hard to get the lever to move even with the screw slackened off, try rotating the joint a couple of times. The 3D printed hole on the servo arm may be a bit 'snug' but it should loosen up with some extra movement. You should also make sure that the screw has been inserted completely perpendicular into the drive lever and it has not threaded itself off centre and the joint is angled. If this has happened remove the screw, hold the parts together and screw in again. The tolerance on the 3D printed hole in the drive lever should allow rethreading 2 or 3 times.

Getting the drive lever/servo arm assembly attached to the servo in a 'calibrated' way is important so take care with this next step.

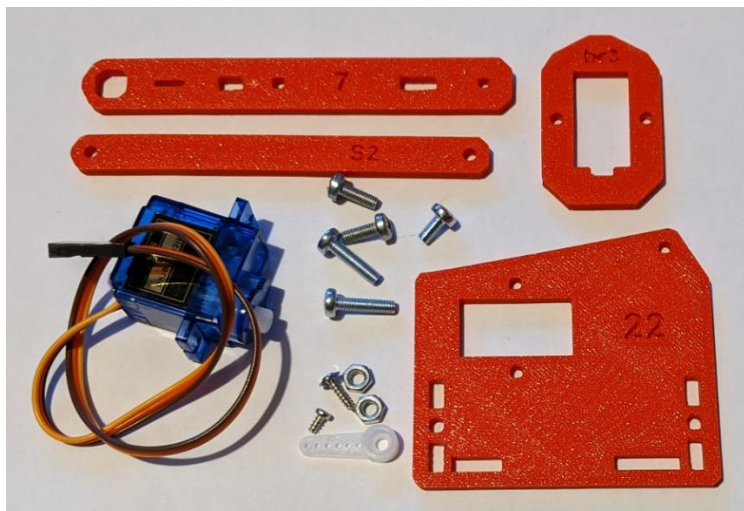
First gently push the drive lever onto the servo's splined draft shaft. The small SG90 servos will turn by hand, so use the drive lever to *gently* turn the servo all the way clockwise until it stops. When it stops, pull the drive lever off the splined shaft and put it back on so it matches the image shown on the right.



Then put the small servo horn screw through the middle of the horn and screw it gently into the hole in the centre of the servo drive shaft so it just grabs - don't over tighten - it is possible for this screw to lock the servo so check that the servo still rotates once you have fitted the horn screw.

As a last check gently turn the servo counter-clockwise and it should go all the way to how it is shown in the image on the right. If it does not, then something is not quite right, so go back and repeat all the 'calibration' steps above.

Step 8: Parts to build the right-hand side of the robot arm



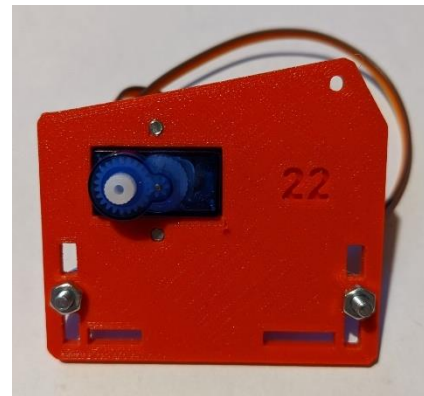
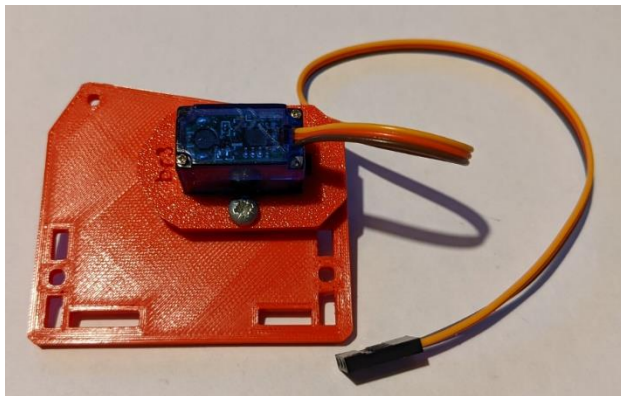
The components for this step and subsequent steps 9 and 10, are shown above and are listed below:

- 3D printed parts 22, 07, one of the servo brackets, and one of the servo arms
- 1 x SG90 servo along with a single-arm 'horn' plus one long and one short horn fixing screw
- 2 x 8mm M3 'pozi' pan head screws
- 2 x 12mm M3 'pozi' pan head screws
- 2 x M3 nuts
- 1 x 6mm M3 'pozi' pan head screw

Step 9: Fit the right-hand 'shoulder' servo to the robot arm

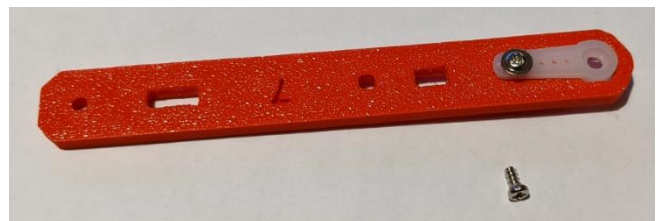
The servo fitted to the right-hand side of the robot arm is known as the 'shoulder' servo and the build of the right-hand side of the robot arm is very similar to the left-hand side.

The servo is inserted into a servo bracket and then screwed to its side plate, part 22, with the two 8mm screws self-tapped into the holes in part 22. Take great care to put the servo on the correct side of the part 22 plate as shown in the images below, and as carried out for the left-hand side assembly, insert the 12mm screws on the correct side of the plate and half turn the nuts to keep the screws in place.



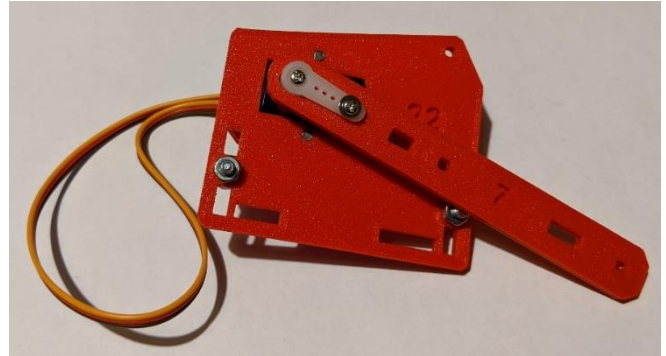
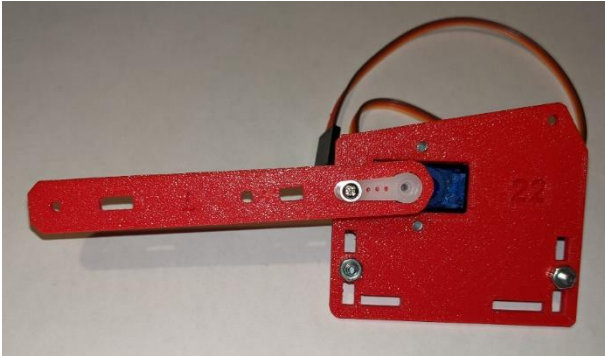
Step 10: Build the right-hand side levers

Now take the single-arm servo 'horn' and attach it to the part 07 lever arm with the long horn fixing screw as shown below.



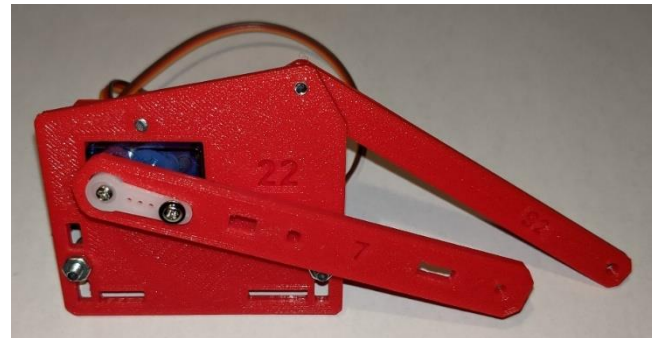
As was done with the drive lever for the left-hand side you should trim/file the 'spikey' point of the fixing screw that will protrude from the other side of the 'horn'.

Now push this combined lever and 'horn' onto the servo splined shaft and turn the servo gently all the way counter clockwise.



Then remove the lever assembly and put it back on so that it matches the image above left. Now fix the lever assembly in place by inserting and tightening the small servo 'horn' fixing screw through the 'horn' into the servo drive shaft (not too tightly again!). Then gently wind the lever arm clockwise and it should then match the image shown above right.

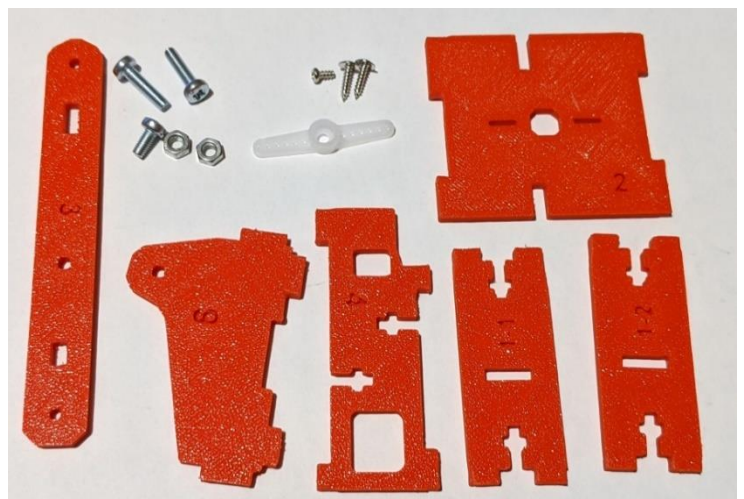
Finally, with the 6mm screw self-tap the 3D printed servo arm part to the outside corner of the side plate, part 22, as shown in the image on the right. This servo arm part is another component that needs to move freely, so as before only tighten the 6mm screw sufficiently to hold the arm firmly in place whilst it can still rotate freely - if it is stiff even with the screw loosened, rotate it backwards and forwards a couple of times.



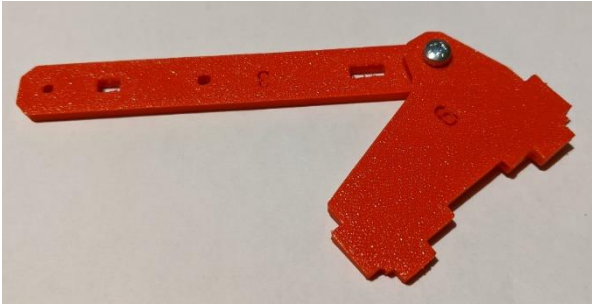
Step 11: The parts for bringing the two sides together

Now we're going to join the left and right-hand sides together with the set of central parts shown in the image on the right and listed below.

- 3D printed parts 03, 06, 02, 04 and 2x part 01
- a two-armed 'horn' plus two long and one short horn fixing screws
- 2 x 12mm M3 'pozi' pan head screws
- 2 x M3 nuts
- 1 x 6mm M3 'pozi' pan head screws



Step 12: Prepare and pre-assemble all the middle parts



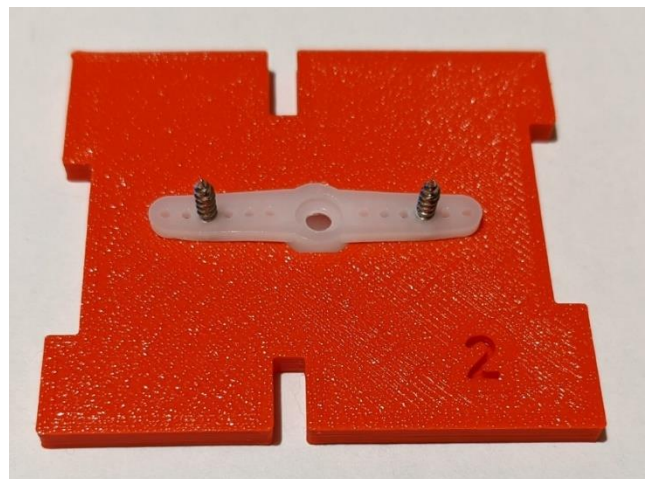
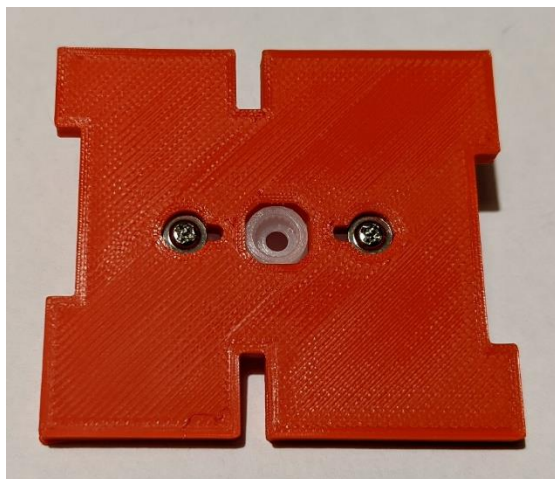
The first preparation is to use a 6mm screw to join the lever arm part 03, to part 06, making sure that as shown in the image on the left, that the hole offset in the middle of the lever arm is furthest away from part 06 and that the lever arm is underneath part 06.

Once again this is another rotational part so make sure the lever arm rotates freely whilst

it is still firmly attached to part 06 with the 6mm screw.

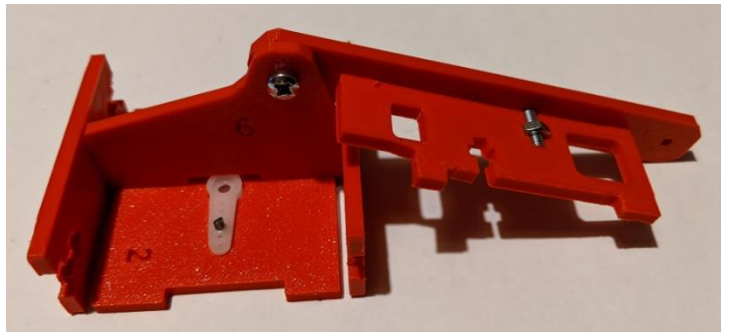
Now use the two long servo 'horn' screws to attach the two-armed 'horn' to part 02 as shown in the two images below.

Make sure that the cut-out slots are as shown in the images below. If you temporarily marry part 03 up to one of the side pieces and think about how that horn will attach to the base servo it will help you get this part right.



As was done with other 'horn' fixings you should trim/file the 'spikey' points of the fixing screws that will protrude from the other side of the 'horn'.

Now you can pre-assemble part 02 (with the horn fitted), part 06 with the lever arm part 03 fitted, and the 2x parts 01, along with part 04 which is connected to the lever arm, part 03 using one of the 12mm M3 screws and one M3 nut. The image on the right shows in detail how everything is connected.



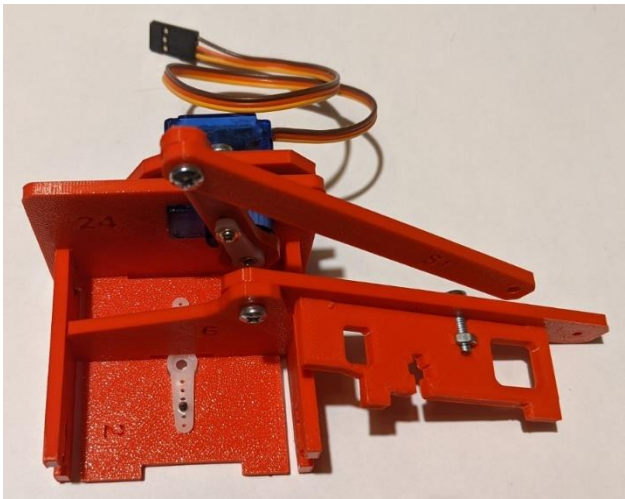
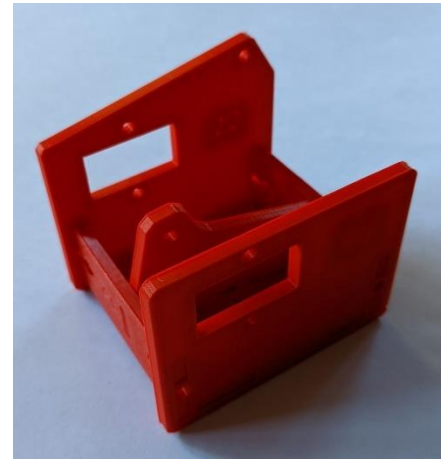
Do not tighten the 12mm M3 screw, just leave it quite loose for now until both the previously built left and right-hand halves of the robot arm have been connected to this central 'cradle'. This cradle assembly will involve a number of these 'bolted together' connections and they should all only be fully tightened once the attachment of both the robot arm halves is complete.

Step 13: Attaching the two robot arm halves

In Step 1, the pre-build check stage, you have already seen how to connect both part 24 (the left-hand side plate) and part 22 (the right-hand side plate) to the rest of the middle parts. The image on the right shows this earlier stage again as a reminder.

Both these side plates are now complete sub-assemblies with their servos and other components attached, however they are connected to the middle components in the same way.

Start by attaching the left-hand side sub-assembly to the middle components, as shown below left, as this will simply click onto the 2x parts 01 and part 02.

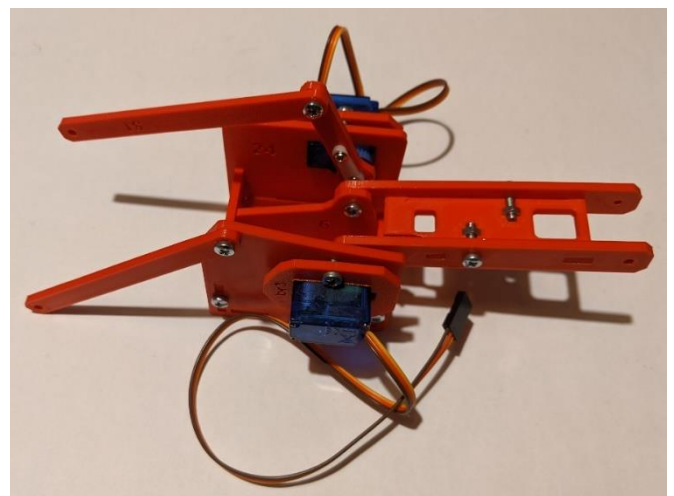


The 12mm M3 screws that were previously loosely inserted into the holes in the side plate, part 24, are now used to more securely connect the side plate to the 2x parts 01 in a similar way to that used above to connect parts 03 and 04, but again do not tighten the screws at this stage.

Then attach the right-hand side sub-assembly, which will not only have to click onto the 2x parts 01 and part 02, but you will also need to connect the lever arm, part 07, that connects to the servo on the right-hand side sub-assembly, to part 04.

Use the remaining 12mm M3 screw and M3 nut to secure these two parts together.

The completed assembly should now look like the image on the right and if you are satisfied that all the various click-together parts are fully home, and the overall assembly is both 'square' and rigid, then you should gently tighten all 6 of the 12mm M3 screws that lock together the various components.



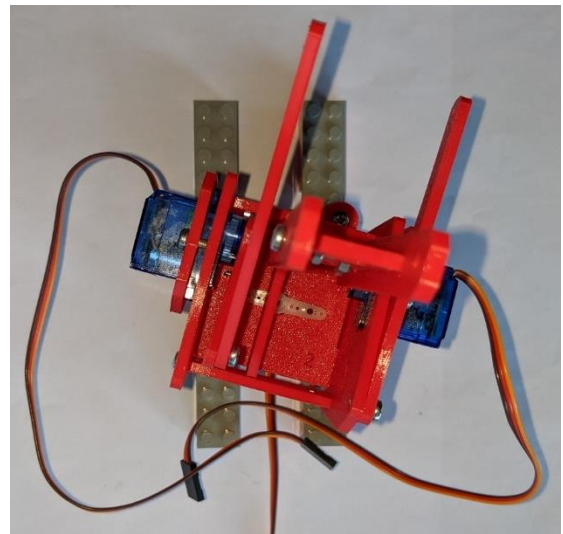
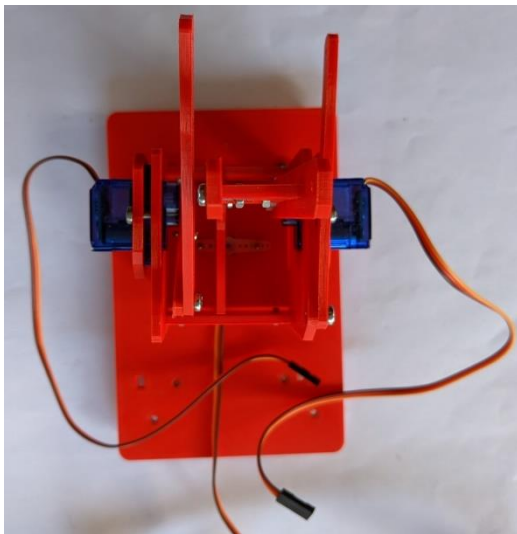
Step 14: Attach the assembled cradle to the base

Push the currently completed cradle onto the splined servo drive on whichever base type is being used, and check that it can swivel the same amount in either direction when centred on the base – removing and repositioning the cradle as required.

If the plain base is used the orientation of the cradle in the centre position should be as shown below left so that left and right servos on the sides of the cradle are furthest away from where the batteries will be fitted to the base.

If the LEGO tile base is used the cradle should simply be oriented so that when it is in a 'central' position it is 'square' with the LEGO tile.

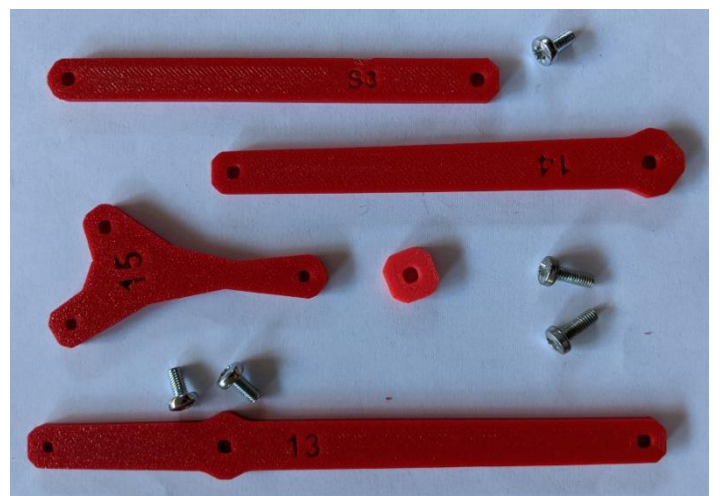
Once a correct positioning is achieved with either base, the cradle is fixed in place by using the small horn fixing screw, but after tightening the screw check that the cradle will still turn in either direction.



Step 15: Add left and right 'forearms'

The 'forearms' are the components that will connect the robot arm's gripper to the cradle assembly that has already been constructed. This step uses the components shown in the image on the right and listed below.

- 3D printed part 13 + 2x 6mm M3 'pozi' pan head screws
- 3D printed parts 15, 10, and 14 + 2x 10mm M3 'pozi' pan head screws
- 3D printed servo arm + 1x 6mm M3 'pozi' pan head screw



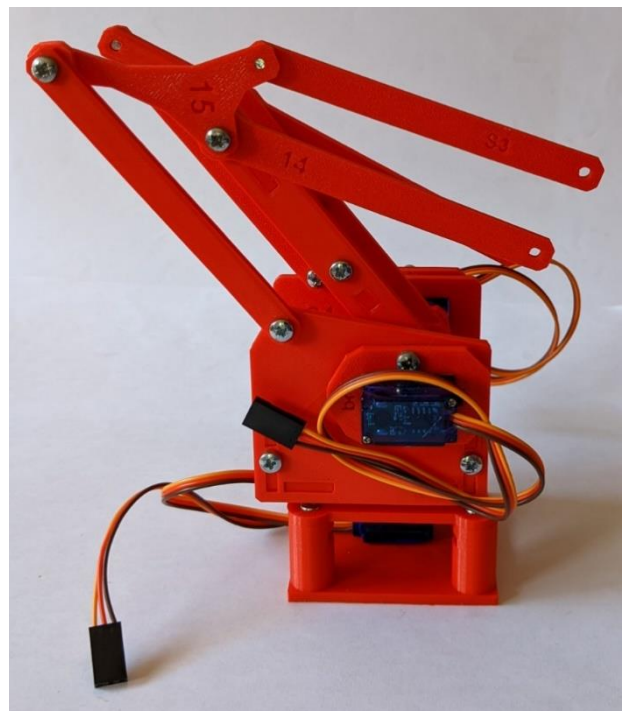
Attaching the left 'forearm' just needs part 13 and the 2x 6mm M3 screws. Now screw part 13 to the servo arm and part 03 that are already on the left-hand side of the cradle assembly. The added part 13 will look like the image on the right with part 13 attached behind the servo arm and in front of part 03.

As with all the other rotating joints make sure that the screws are not so tight that they restrict rotation but do ensure that the parts are not slack.



The right 'forearm' uses the remaining parts for this build step as is shown in the two images below, both viewed from the right-hand side of the robot arm. Part 15 is used to interconnect the servo arm that is already attached to the right-hand side of the cradle to the additional servo arm and part 14. The 'order' in which the parts are positioned around part 15 is important and as can be seen in the detailed image below left, one of the 3D Printed 'spacers' (parts 10, 12, 23 or 25) is placed between the existing servo arm and part 15 and a 10mm screw is used to connect them.

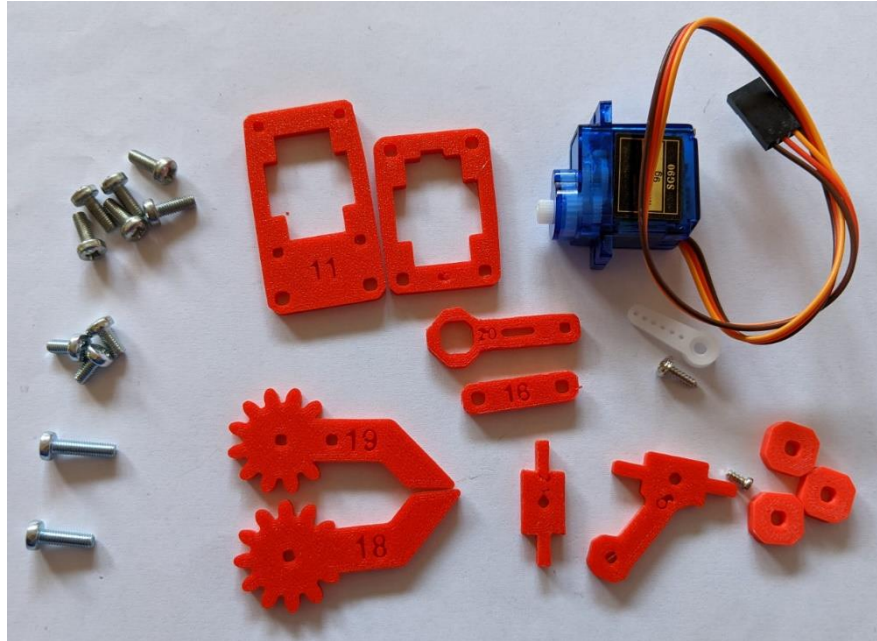
A 6mm screw is used to connect the additional servo arm to the top hole of part 15, and a 10mm screw is then used to connect part 15 through its bottom hole, through part 14 and into the top hole of the existing part 03 i.e., the servo arm connected to the 'shoulder' servo.



Once again check that the various rotational parts can spin freely!

Step 16: The gripper components and servo assembly build

The components for this final stage of the build are shown in the image and list below.

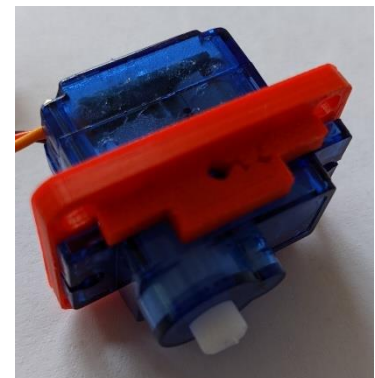
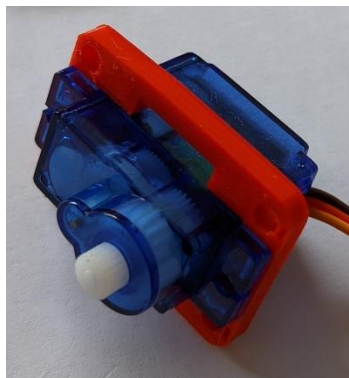
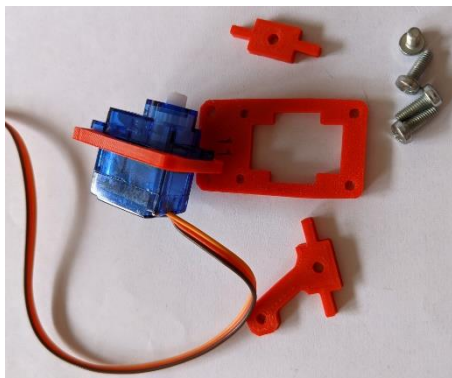


- 3D printed parts 11, 09, 18, 19, 21, 05, 20, 16 and the 3 remaining spacers (parts 12, 23 & 25)
- 1 x SG90 servo along with a single arm 'horn' plus one long and one short horn fixing screw
- 6 x 8mm M3 'pozi' pan head screws
- 1 x 10mm M3 'pozi' pan head screws
- 1 x 12mm M3 'pozi' pan head screws
- 3 x 6mm M3 'pozi' pan head screw

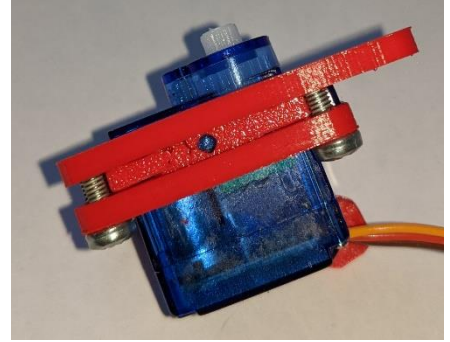
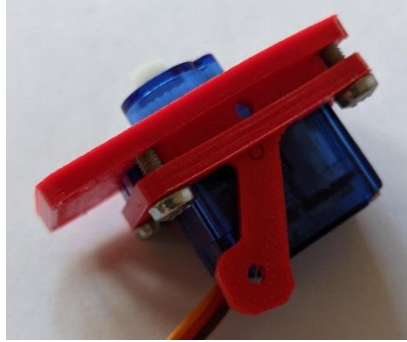
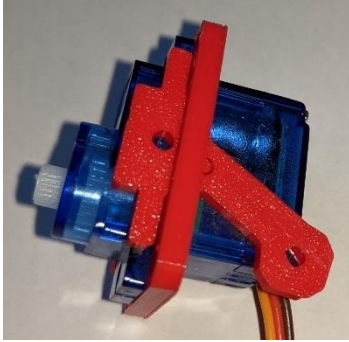
[Just to note: the only remaining component at this stage should be a spare servo bracket]

To build the gripper servo assembly you will need parts 11, 09, 05 and 16 plus four of the 8mm M3 screws.

The images below provide a step by step guide to how this assembly is put together.



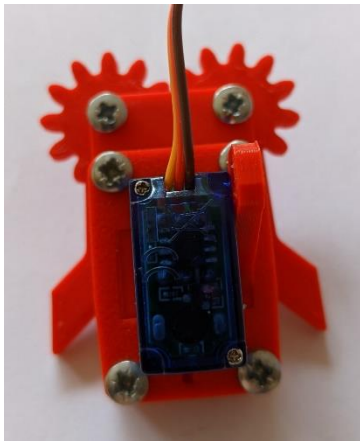
The image above left shows all the components for this step arranged in the way they will be put together. The image above centre then shows how the servo is first inserted into part 09 which is a more specialised servo bracket. The image above right then shows how part 21 is slotted into the space between the servo and part 09 – make sure you put this in the right way around.



The image above left then shows how part 05 is slotted into the space on the other side of the servo and part 09. Now part 11 is placed over the servo, as shown above centre, to 'sandwich' the parts together and the four 8mm M3 screws fix the two sides of the 'sandwich' together. Part 05 then provides two fixing holes, and the image above right shows the other side of the assembly where part 21 provides another fixing hole.

Step 17: the gripper jaws

The images below detail the series of steps to now assemble the jaws of the gripper.



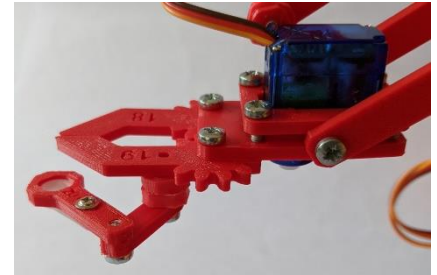
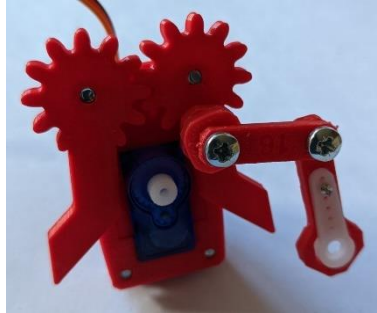
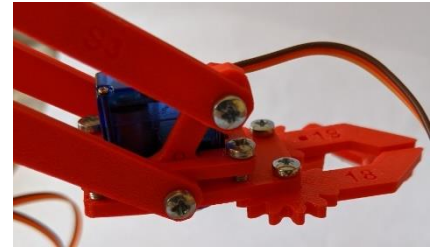
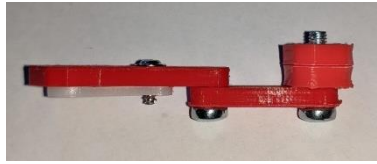
With a 6mm M3 screw attach parts 19 and 18 to the already completed gripper servo assembly as shown in the image above left. Make sure that part 19 (with the extra hole) is on the left as viewed in the image above left.

Make sure the two 'jaws' mesh cleanly and that they both rotate smoothly.

Now add the single-arm servo 'horn' to part 17, as shown in the image above centre and then connect part 17 to part 16 with the last 6mm M3 screw, as shown in the image above right. As with all rotational joints make sure it can rotate smoothly whilst still

being held firmly, and you should trim/file the 'spikey' point of the 'horn' fixing screw that will protrude from the other side of the 'horn'.

The combined parts 16 and 17 now use the last 12mm M3 screw and two spacers, as shown in the image immediately on the right, to be attached to main gripper, as shown in the lower image on the immediate right.



The completed gripper is then attached to the main robot arm as shown in the two far right images. The top image of these two images on the far right shows how the bottom right-hand 'forearm' server arm (part 14) is attached with an 8mm M3 screw to the bottom fixing hole of part 05 on the gripper. The top right-hand 'forearm' servo arm is then connected to the top fixing hole of part 05 on the gripper with the last 10mm M3 screw with the last spacer in between.

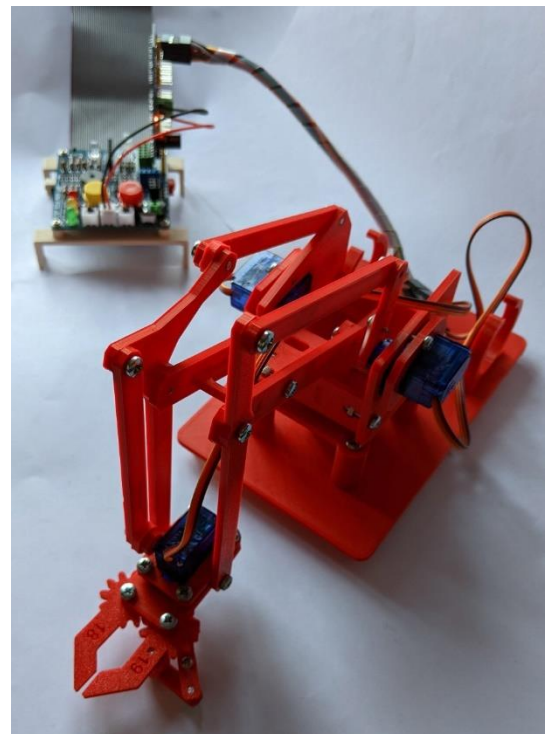
Finally, the image above, bottom right, shows how the last 8mm M3 screw is used to attach the left-hand 'forearm' servo arm to the fixing hole in part 21.

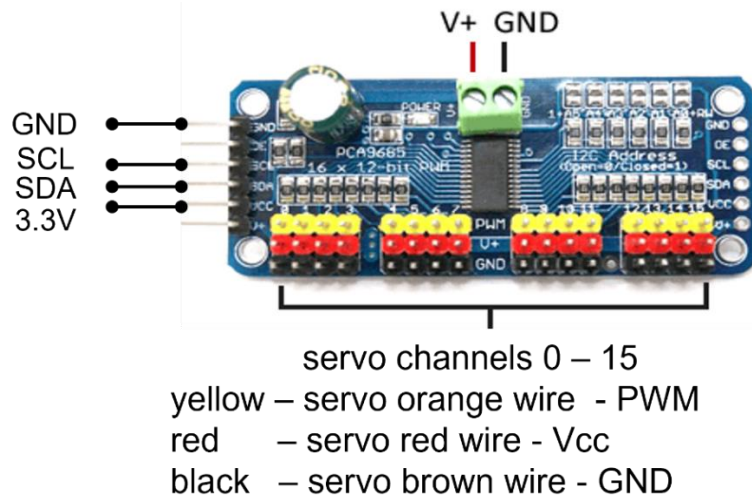
At this stage do not attach the horn on part 17 to the gripper servo since this needs to be done after the Raspberry Pi is set up so that the jaws open/closed positions are properly calibrated, but you might want to temporarily screw the small horn screw into the servo shaft so that it doesn't go astray.

Step 18: Connecting the robot arm to a Raspberry Pi Maker Kit

The four servos on the robot arm can be controlled through a PCA9685 PWM servo controller board that is managed via an I²C connection to a Raspberry Pi. The controller board also allows a separate power source to be connected to provide power to the servos.

All of these connections are however simplified, as shown in the image on the right, by using the Raspberry Pi Maker Kit with the PCA9685 inserted into the dedicated 6-pin (yellow) female connector on the Maker Kit PCB and the required power source using the Maker Kit's separate power bus. The Maker Kit's power bus can then be 'fed' by either a 5V rechargeable battery bank or a 4AA battery holder (neither of which are shown in the image on the right).





The controller board, illustrated above, will allow up to 16 servos (0-15) to be separately managed and these are arranged in blocks of 4, with the robot arm software currently configured to use channels 12-15.

As the PCA9685 module will be positioned some way from the robot arm assembly, the servo connection cables should be extended using four of 3x 20cm male-to-female jumper leads. The jumper leads may be any colours, so you should take care to ensure the right connections are made from the extended servo cables to the PCA9685 board.

Once you have connected the robot arm to a Raspberry Pi via the Maker Kit PCB and carried out any necessary software calibrations (details of the available software described next), as well as finally attaching the drive levers to the gripper servo, you may need to make some adjustments to other parts of the physical build, checking whether any of the rotational links are over tightened and may need to be loosened.

Raspberry Pi software

Raspberry Pi operating system:

Only the newer 'Buster' and 'Bullseye' versions of the Raspberry Pi operating system have been used in the development and testing of this Maker Kit robot arm project, which has mainly been carried out and shown to operate successfully, with a Raspberry Pi 4B, Pi 3B+ and a Pi 400, but has also been shown to operate successfully with a Raspberry Pi 2B (v1.1) and a PiZero W (v1.1).

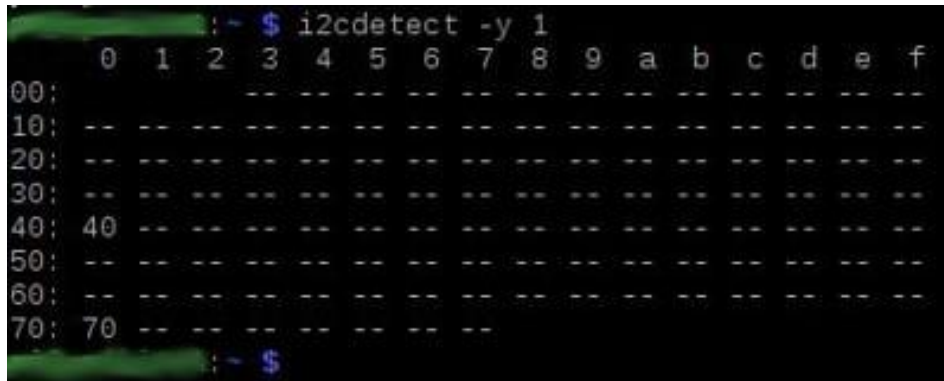
Earlier Raspberry Pi operating system versions e.g. 'Stretch' or 'Jessie' may or may not have operational problems, so users should probably upgrade to at least 'Buster' in order to use the robot arm, with the use of 'Bullseye' is highly recommended.

The I²C interface on the Raspberry Pi must be enabled and the following specialised Python libraries should be installed if they are not already present:

- i2c-tools
- python-smbus

Then, with the PCA9685 module connected to a Maker Kit PCB – and hence onward to a Raspberry Pi, the unique I²C address for the device can be checked by using the following command in a Terminal window:

```
i2cdetect -y 1
```



```
~$ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40: 40  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70: 70  --  --  --  --  --  --  --  --  --  --  --  --  --  --
~$
```

The screen shot above shows that the two possible 'detected' addresses for this example PCA9685 are either 40 or 70. The 40 value is used in the main motion control Python 'driver' software, *meArm.py*, that, as described next, is made available along with some demonstration software and all the documentation from [here](#).

Python robot arm management software:

Python 3 software downloadable [here](#) is made available to help with the initial set up of the robot arm and then to provide demonstrations of various movement cycles.

The software, which after downloading should all be installed at:

```
/home/YOURUSERNAME/RPi_maker_kit5/complete_projects/MeArm_v0-4/
```

.. can be considered at three different layers:

- underlying libraries that provide I²C and PWM functionality;
- MeArm 0.4 specific motion control 'driver' functions that use the underlying libraries, and
- Individual set up/demonstration programs that use the generic MeArm functions

The various Python scripts can then either be run from the Thonny IDE or from a CLI window using the suggested command shown near the top of the code for each script, which is typically something like:

```
python3 ./RPi_maker_kit5/complete_projects/MeArm_v0-4/DemoIK.py
```

.. where this generic ./ command format assumes that the user is currently 'in' their home folder.

Each of the three 'layers' of software will now be briefly described.

Underlying library software:

PCA9685_I2C.py – software that provides the PCA9685_I2C class with a range of functions to manage the I²C 'messaging' between the Raspberry Pi and the PCA9685.

Not edited/adapted for use here, other than minor changes to make the code Python 3 compliant and to note that the *getPiRevision()* function no longer works with the much wider range of Pi's that are now available, so its use downstream in the overall code base is avoided.

PCA9685_PWM_Servo_Driver.py – driver software specifically for the PCA9685.

Not edited/adapted for use here other than minor changes to make it Python 3 compliant.

kinematics.py – inverse kinematics code, i.e. the mathematics that converts the requested x, y, z positioning to the required individual servo angular motion - adapted for Python by Bob Stone from C++ original by Nick Moriarty May 2014. Further adapted here to make it Python 3 compliant and to add further debug output options. Debug output is triggered by a False/True *debug* parameter in the various functions, that defaults to False but can be optionally set to True when an individual function is called.

MeArm motion control 'driver' functions:

meArm.py – provides a MeArm robot arm specific class with a range of functions to initialise and control the arm and gripper. The code has been edited/adapted to make it Python 3 compliant, to add extensive annotation/comments throughout the code to make its use more understandable, and to add further debug options in a similar manner to the *kinematics.py* code.

The main set up function allows what are called the servo 'sweep' values to be varied for each usage instance via the function's parameters, but the code has been edited so that the defaults for these values can be easily adjusted in the code. The 'sweep' values control what the resultant PWM signal will be for executing the beginning and end positions, i.e. the 'sweep', for each of the servos. The various set up/demonstration programs described next allow various tests to be carried out so that these default 'sweep' values can be fine tuned for each specific servo since each one will have slightly different manufacturing tolerances.

In addition, a 'sleep' period is used as a 'damping' interval for the speed of the arm movement, and the I²C address can be modified should the PCA9685 have a different address to the usual value of 40.

Set up/demonstration software:

When using, for example, a Raspberry Pi 4, all of the following programs can be loaded into Thonny IDE and run from the IDE, but whilst the previously described programs will be 'called' by these set up/demonstration programs the underlying library and MeArm function programs do not need to be loaded into Thonny.

[You may however experience 'Internal Errors' when trying to run the code within the IDE when using either a Raspberry Pi 2 or a Raspberry Pi Zero (which may be due to memory limitations) – but all the code can still be successfully run with these 'smaller' Pi's in a Terminal window by simply using the command shown towards the top of the listing of each program.]

show_servo_info.py – allows a simple overall system check to be carried out by setting a single arm position, the so-called 'home' position, whilst the *debug* parameter has been set to True. This then causes many different system/code derived values to be displayed.

Robot arm movement demos: each individual movement demo in the list below is simply a cut down/custom version of the **DemoIK.py** code with the Maker Kit's RGB LED and passive buzzer used to indicate the completion of an individual movement.

It should be noted the x, y and z axes are as shown in the schematic on the right.

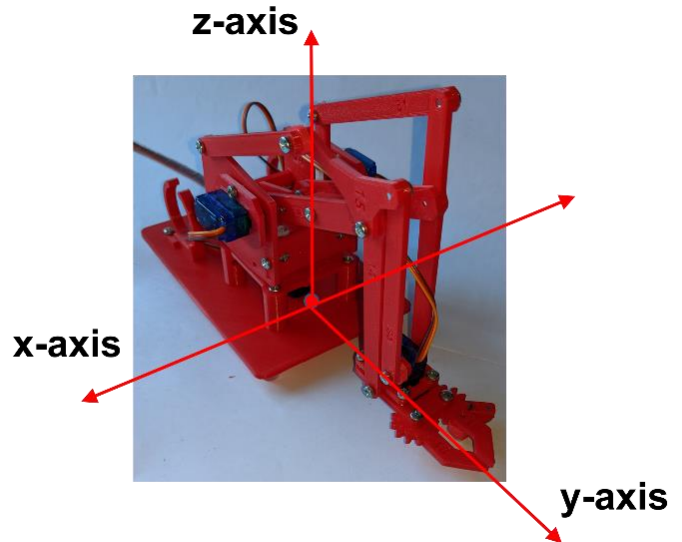
Each demo can be run from the CLI window using the command shown near the top of the code text e.g., typically:

```
python3 ./RPi_maker_kit5/complete_projects/MeArm_v0-4/Demo_x-axis.py
```

.. which assumes the code is stored in:

```
/home/YOURUSERNAME/RPi_maker_kit5/complete_projects/MeArm_v0-4
```

or each demo program can be run from the Thonny IDE.

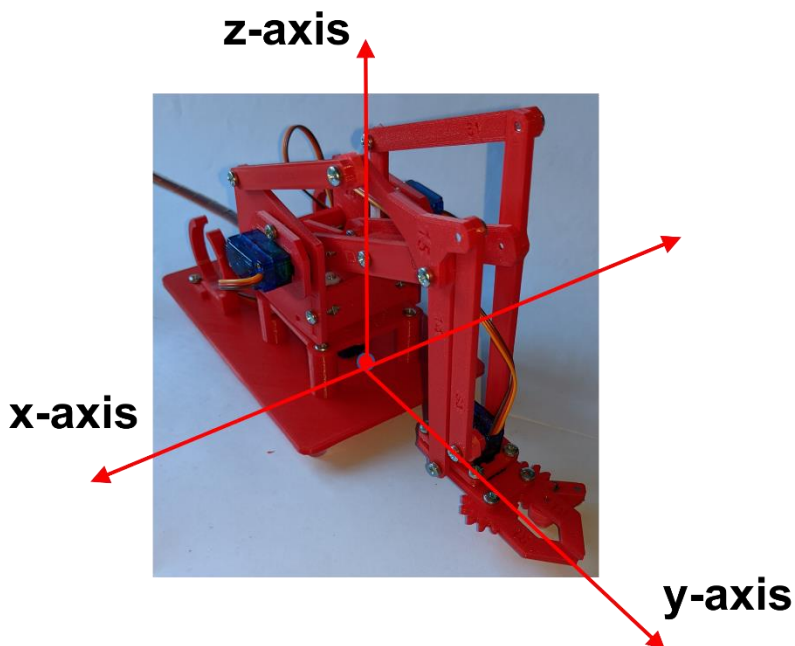
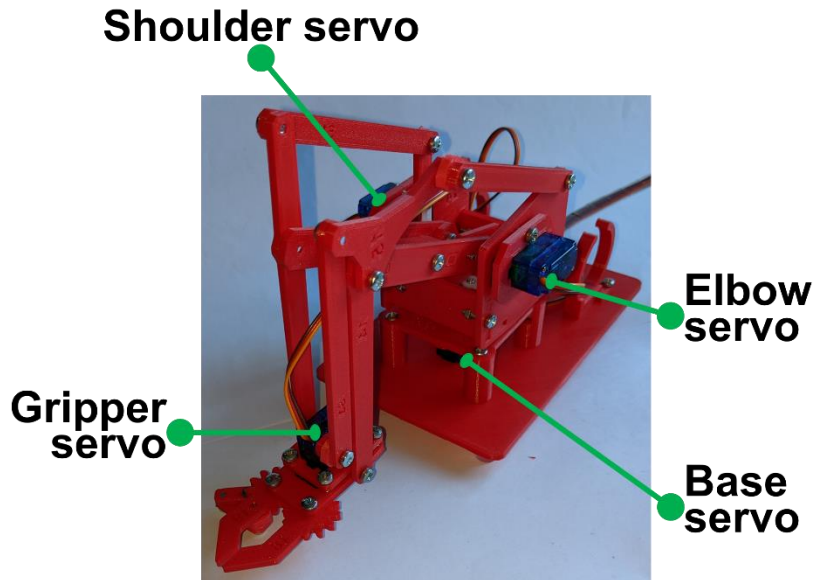


- **Demo_gripper_servo.py** – simply closes and opens the gripper for a single cycle.
- **Demo_home_position.py** – checks that the so-called 'home' position is achieved reasonably accurately. Some tweaking of the base, shoulder and elbow servo 'sweep' values can be carried out to try and fine tune this positioning.
- **Demo_x-axis.py** – starting from the 'home' position, this code moves the gripper through a series of x-axis negative and positive positions with the y and z-axis values held constant. This means that the base, shoulder and elbow servos all operate so some careful tweaking of their 'sweep' values can be made to fine tune the positioning.
- **Demo_y-axis.py** - starting from the 'home' position, this code moves the gripper through a series of y-axis positive positions with the x and z-axis values held constant. This means that the shoulder and elbow servos both operate so some careful tweaking of their 'sweep' values can be made to fine tune the positioning.
- **Demo_z-axis.py** - starting from the 'home' position, this code moves the gripper through a series of z-axis positive positions with the x and y-axis values held constant. This means that the shoulder and elbow servos both operate so some careful tweaking of their 'sweep' values can be made to fine tune the positioning.
- **DemoIK.py** – based upon the code from the York Hack Space in May 2014, this Python code puts the robot arm through a series of positions and is set to run continuously until stopped. The 'IK' in the program title refers to "Inverse Kinematics" which is what the calculations/math used for deriving the robot arm 'joint angles' and hence the servo 'instructions' in order to achieve a specific (x, y, z) cartesian co-ordinate position, is called.

Appendix A: additional reference data

The four servos are often referred to as a Base, Shoulder, Elbow, or Gripper servo, as indicated in the annotated image below right, and are typically connected to the servo channel pins on the PCA9685 PWM servo I²C control module as shown in the table below.

Base servo	channel 12
Shoulder servo	channel 13
Elbow servo	channel 14
Gripper servo	channel 15



The position of the gripper is defined in x , y , z cartesian coordinates, as shown on the left, where the pivot point is centred on the Base servo axis and measured (very approximately!) in mm.

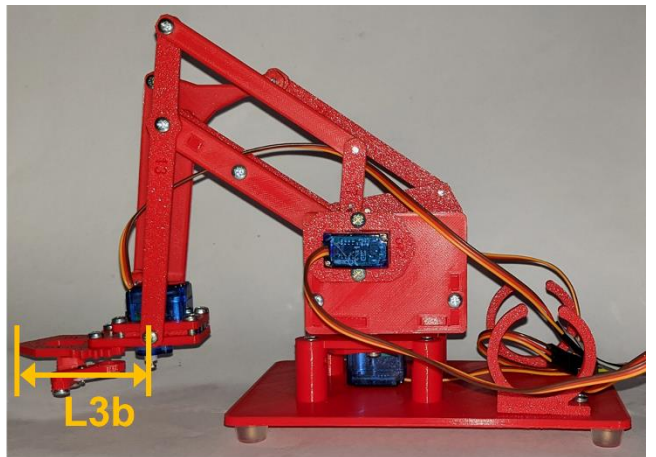
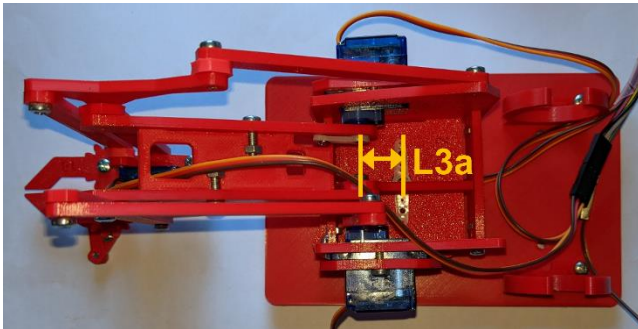
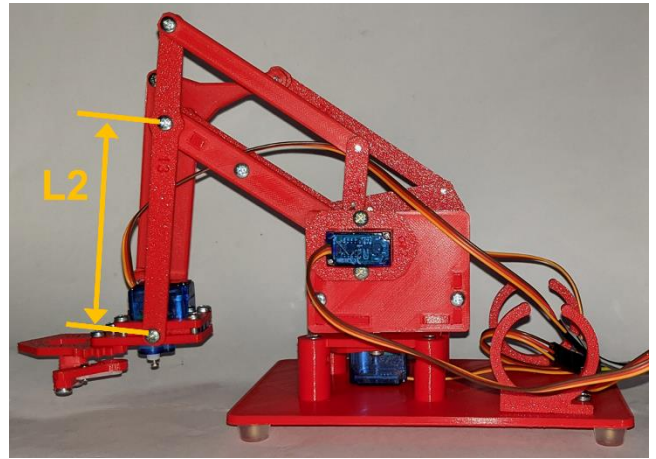
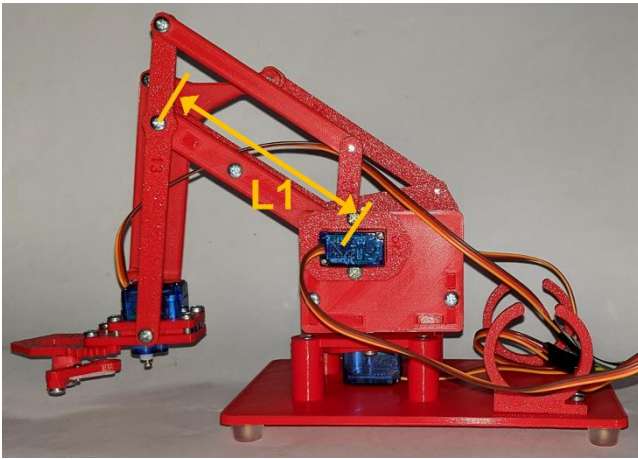
For example, the usual 'home' position is at (0, 100, 50) i.e., 100mm forward of the base and 50mm off the ground.

For the various geometric calculations (in the kinematics.py software) the annotated images below show the key distances for the assembly where:

L1 = the shoulder to elbow length = 80mm

L2 = the elbow to wrist length = 80mm

L3 = the wrist to hand plus base centre to shoulder lengths = L3a + L3b
= 68mm (54 + 14)

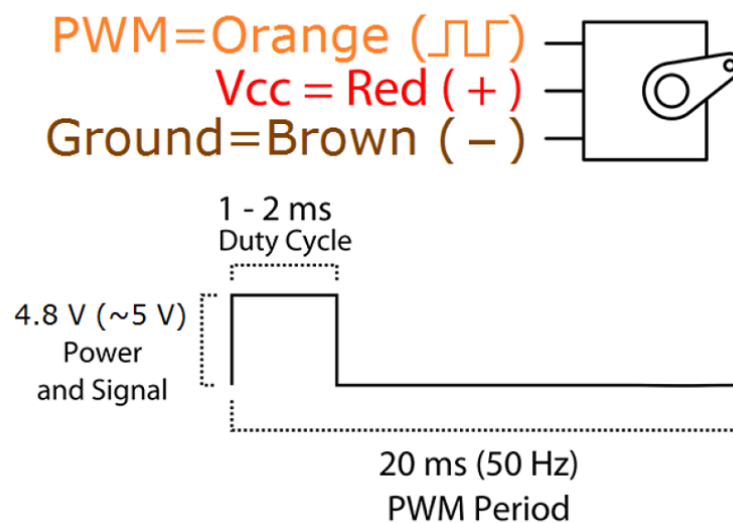


Appendix B: SG90 servo control with PWM

The robot arm's SG90 servos can rotate approximately 180° (90° in either direction), and during the build one of the SG90's plastic 'horns' connected to a robot arm lever will have been fitted on to the splined gear box shaft of each servo.

Control of a servo is provided by applying a Pulse Width Modulation (PWM) signal to the servo's orange signal wire, where the PWM Frequency is set to 50Hz (i.e., 20ms cycle duration or period) and the middle, full left and full right positions are set by the PWM Duty Cycle being set between 1 and 2 milliseconds (ms). A fuller description of Pulse Width Modulation and its various parameters is provided next in Appendix C.

The schematic below provides a summary of the wiring connections to the servo and the applied PWM control signalling.

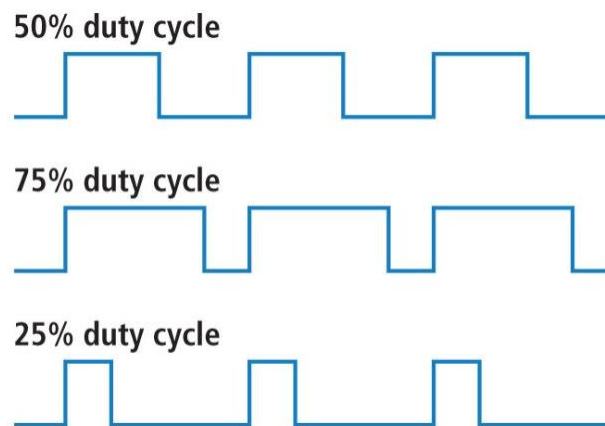


Appendix C: pulse width modulation (PWM)

Pulse-width modulation (PWM) simply means a digital signal that is on/off (pulsed) for a period of time, where the **frequency** at which the on/off switching occurs can be set, and where the amount of time spent on, versus off, known as the **duty cycle** and expressed either as a time or as a percentage, can also be set.

PWM can be used to encode a message into a pulsing signal, but it is mainly used to control the amount of power supplied to electrical devices, such as electric motors (drive, servo or stepper) and LEDs.

By switching the applied voltage on/off, the average voltage (and therefore current) that is applied to the electrical device is controlled by using different duty cycles, e.g., a duty cycle of 0% means no power and a duty cycle of 100% means full power. The schematic below illustrates some other duty cycles where the frequency is exactly the same for all three examples.



The frequency at which the switching occurs needs to be appropriate for the electrical device being powered, so that there are no negative effects i.e., the resultant wave form that is applied to the device is 'perceived' as something that is very smooth. For example, if the frequency used to control a LED was too low you might see the light varying in intensity, or for a motor it might not rotate smoothly. A typical frequency for use with a servo motor might be 50Hz, whereas you would need about 500Hz for a LED to avoid visible 'flicker'.

Appendix D: I²C connection to a Raspberry Pi

The I²C bus and its signalling protocol provide a simple but very functional way for a Raspberry Pi to control the PCA9685 and to 'tell it' what PWM signals to apply to each servo.

A fuller description of the I²C protocol and its various parameters is provided next in Appendix E, but there are just four connections to be made to a Raspberry Pi no matter how many servos are being controlled: two 'signal' wires (SDA and SCL) plus a power and ground connection. These connections are simplified by using a Raspberry Pi Maker Kit which has a dedicated 6-pin (yellow) female connector on the PCB into which the PCA9685 module can be directly inserted.

The following table describes the six main input pins on the PCA9685 board and their routing through to the four I²C connections on the Raspberry Pi via the Maker Kit:

PCA9685 pin	Raspberry Pi GPIO pin
GND	Any GPIO GND pin
OE (Output Enable)	Not used
SCL (serial clock line)	SCL GPIO pin (GPIO pin#03)
SDA (serial data line)	SDA GPIO pin (GPIO pin#02)
VCC (3.3V supply for the control board)	3.3V GPIO pin
V+ (5V supply for the devices from the bus)	Not used because the Pi's power supply is not sufficient, and an external battery supply is used instead

Appendix E: I²C control and the PCA9685 board

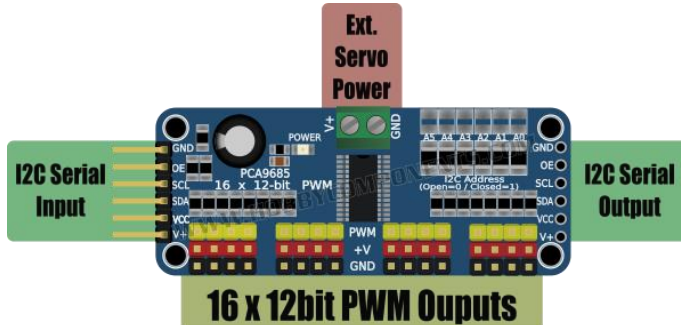
The I²C bus¹ was originally designed by Philips in the early 1980s to allow easy communication between components which reside on the same circuit board, and the name stood for "Inter IC" i.e., Inter Integrated Circuit, sometimes shown as IIC but more commonly as I²C.

I²C is now, not only used on single boards, but also to connect components which are linked via cable, and it is its simplicity and flexibility that make this bus attractive for many applications.

The most significant features of I²C include:

- Only two bus lines are required for the data transport and signaling:
 - SDA: serial data (I²C data line).
 - SCL: serial clock (I²C clock line).
- A simple master/slave relationship can be defined between all components on the bus with multiple masters being possible.
- Each device connected to the bus is software-addressable by a unique address.
- There are no strict data exchange speeds (baud rate) requirements, as for instance is needed with RS232, as the master generates a bus clock speed.
- I²C is a true multi-master bus providing arbitration and collision detection.

The PCA9685 board is a I²C controlled device that can apply Pulse Width Modulated power to 16 devices such as LEDs or servo motors.



As shown in the schematic on the left, there are six input pins, GND, OE, SCL, SDA, VCC and V+, and these are mirrored as outputs (but typically, as supplied, no pins are soldered on the output connections) so that additional devices could be added downstream on the bus.

GND and VCC are a 5V DC input to power the board and V+ is power input for the PWM devices if this can be supplied down the bus. The SCL and SDA pins are the two main I²C control lines and the OE (Output Enable) pin is effectively an overall on/off switch which has a default LOW setting for 'on'. For use with a Maker Kit, neither the V+ is used, as the Raspberry Pi cannot internally support enough power for multiple servos, nor the OE, as this is unnecessary.

There is also a pair of screw terminals to provide a separate power source for the 16 PWM controlled devices which can be 'fed' from the Maker Kit's power bus, and 16 sets of 3 pins for the PWM devices, designated as channels 0 to 15 (left to right on the diagram above).

END OF DOCUMENT

1. *a bus is a shared digital pathway between multiple resources and devices in an electronic system where signals and data can be exchanged between everything connected to the bus using defined protocols*