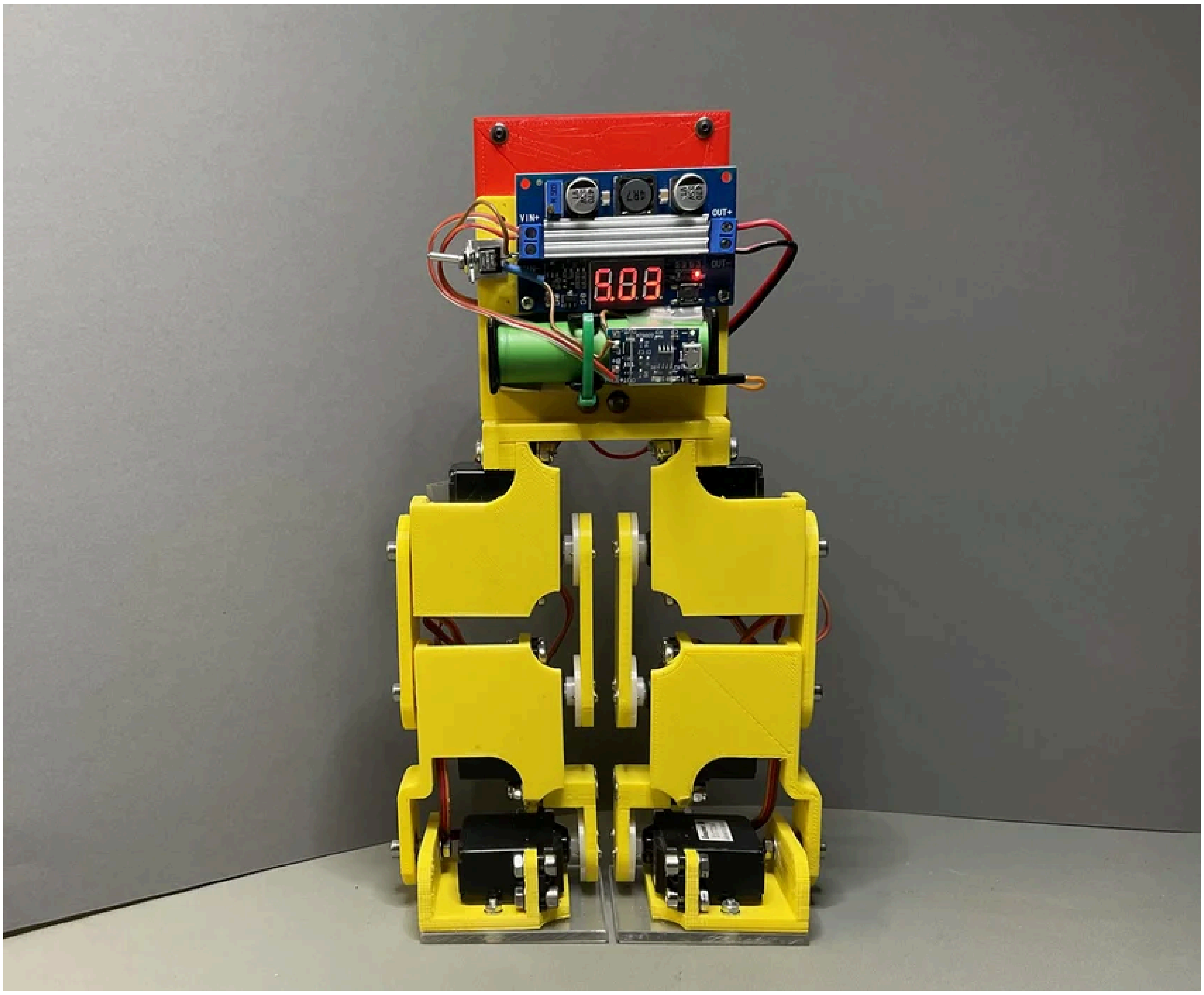
AUTODESK
Instructables

Poor Man's Bi-ped Robot Hardware - Using RP2040 and Micropython

By wolf2018 in CircuitsRobots

Introduction: Poor Man's Bi-ped Robot Hardware - Using RP2040 and Micropython



Since people build their own little robots, I wanted to build one of these robots myself - and of course, he needed a name: **Robi** will be my new toy.

Thinking of the project, I quickly came up with the following requirements for Robi:

- Leveraging a bi-ped hardware, as there are many good designs out there
- Use readily available components, to make it easy to implement
- For software implementation use python / micropython
- Build a prototype first and then tweak/optimize, which means:
- the hardware should be 3-D printable
- the firmware/software should allow for optimization
- easy to play with - a user interface to remote control the robot
- Optional: when Robi started walking longer distances, the cables (USB and power) "got longer and longer", so I ended up adding a battery and making Robi controllable over WLan.

While writing this instructable with the focus to make it easy to reproduce (as all of my "poor man's" instructables), I decided to split into 3 parts (three instructables):

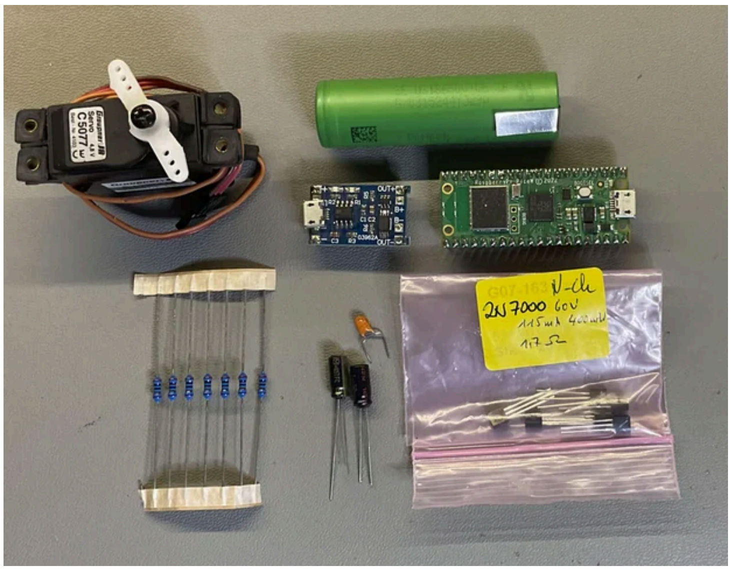
- *Poor man's bi-ped* ***Robot Hardware*****(this instructable)** describes how to build the robot hardware and document the electronics to run the servos which move the robot.
- *Poor man's bi-ped* ***Robot Controller*** describes the software which is running on the robot electronics to make the robot walk.
- *Poor man's Robot* ***Remote Control*** describes the User interface to program the robot to walk a certain distance and control the parameters, which influence the way the robot walks.

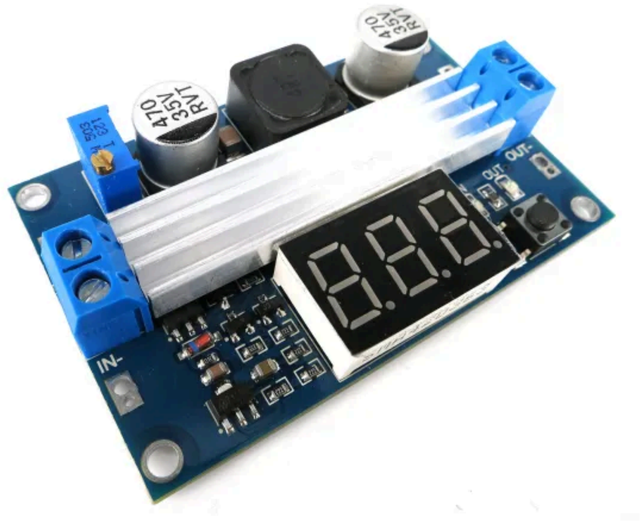
The three parts are documented in a way that you can use them independently. If you consider building your own Robi the robot, you can just follow through all three instructables or you can pick one or two parts of it and leverage for your own project.

One remark: These three instructables document the project to make it easy to reproduce, but will not go deeply into all the knowledge areas, which are involved building and controlling a bi-ped robot. Basic knowledge will be provided and I will provide links to resources, which I found very helpful.

Have fun reading and building! - and if you have questions or improvement ideas - feel free to add a comment below.

Supplies





To build your own bi-ped robot you should own -or have access to- a 3D printer and some experience in soldering components. I recommend to build the electronics on a prototype board or even a printed circuit board.

Choosing a robot hardware: The first decision you need to take is governed by the servos you want to use to move the robot. This determines the size of the robot hardware. Or vice versa, you decide on the size of the robot, which determines the servos to use.

I wanted to use standard RC servos I had left over from an RC airplane. These measure typically 40mm by 20mm and are readily available. With that decision taken and some internet research, I settled for building the hardware of the instructable Robotic Biped by Technovation.

The electronics parts:

- 1x RP2040 picoW board
- 6x 2N7000 Fets
- 6x 2.7 kOhm resistors
- 1x 1uF Tantal capacitor, to buffer the micro-controller power supply
- 2x 10 uF electrolyte capacitors to buffer the servo power supply
- 6x standard RC servos - I used Graupner C5077 and C507, but any other 40x20mm servo will do
- crimp connectors for the servo cables
- cables for power supply and USB

For a first start you are probably good using a 5V 3A power supply. USB power from e.g. a PC is definitely not enough to power six servos in movement.

Optional: to make Robi independent of a power cable:

- 1x 18560 size Lithium-Ion battery (or a Lithium-Polymer battery)
- 1x battery holder for the battery chosen (which I 3D printed, there are plenty on thingiverse)
- 1x Lipo battery charger board (e.g.TP4056 or equivalent)
- 1x DC-DC converter to boost the battery voltage of 3.5V -4.2V to 5V needed for the servos.

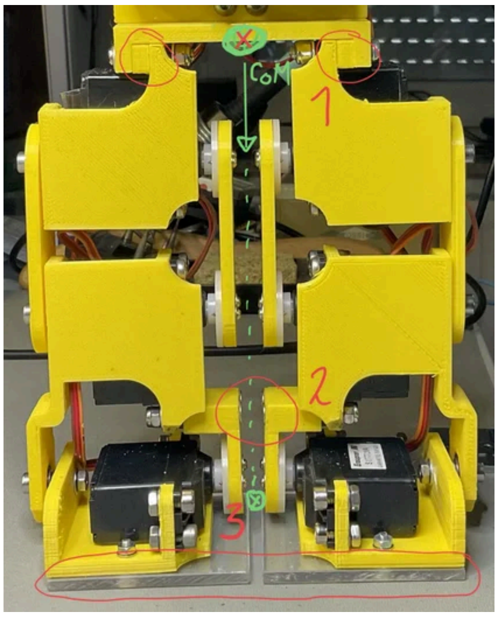
Few comments on the battery powering:

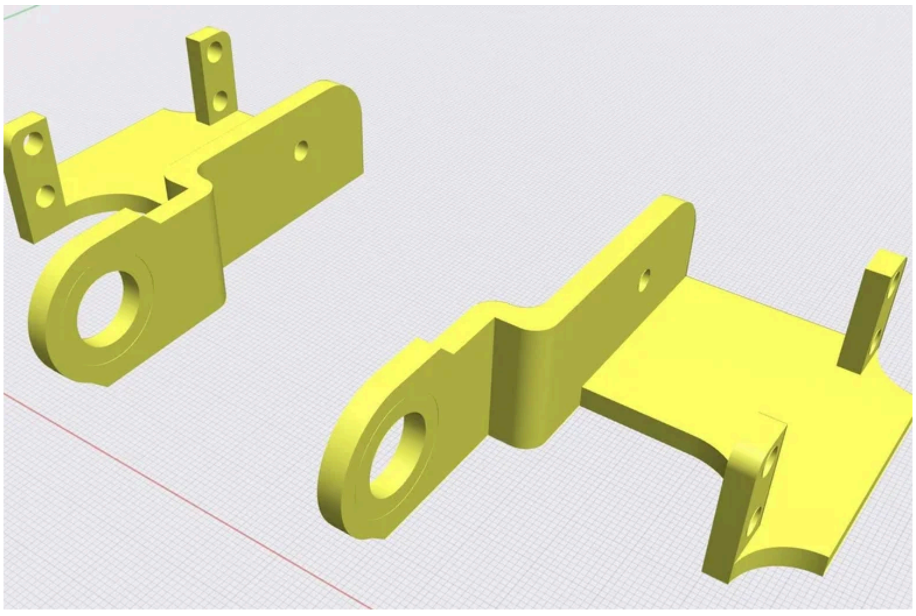
- I decided to use one cell to keep the weight as low as possible
- Eventually I used a Li-Ion battery in a metal case. They are more rugged and I guarantee the robot will fall to the floor sooner or later ;-)
- The DC-DC converter should support input voltages down to 3.5V. It should be capable to supply a minimum output current of 3 Ampere (more is better, dependent on the power requirements of your servos).
- I use the 100W DC-DC converter CP07052 of manufacturer QITA (see picture). This is the generic part number you need to see who is supplying this part - or a similar module.

For building instructions of the robot hardware, please refer to the Robotic Biped by Technovation , which is very detailed. A big thank you to Kousheek Chakraborty and Satya Schiavina for this great instructable.

Finally I want to disclose my second favorite robot: Ottis. It is smaller and it uses SG90 micro servos. The servo size is the reason why I did not choose that design, though it is a very nice design, too! I like the idea of visual and audio gestures. May be I will add gestures to Robi later :-) .

Step 1: Changes and Improvements to the Robot Hardware





Building the robot hardware:

I was building the Robotic Biped by Technovation as described in their instructable with one initial change. As I had ball bearings of different dimensions I modified the STL files of the robot to fit my bearings prior to 3D printing. With that initial change the robot hardware was working fine.

Remark: if you are using an rp2040 picoW board and micropython, follow the Technovation instructable to assemble the Robot except for the Initial Setup (Step 10 in the Technovation instructable). Please follow Step 3 of this instructable instead.

Improvements:

Better support of the Center of Mass (CoM):

While I was experimenting with different walking patterns, e.g. walking by lifting the moving leg ("like humans do"), Robi tilted sideways, when lifting a leg, because the legs have been too far apart, better, too far away from the Center of Mass (CoM).

I did three improvements to get the supporting leg -the leg standing on the ground- closer to the CoM:

The numbers of the bullets below refer to the numbers in attached picture.

1. Add two 3D printed spacers of 5mm to move the legs closer together by 10mm
2. Use of countersunk screws to fix the feet, to avoid collision of the screws while the robot walks
3. Add two (acrylic) foot plates of 60x60mm to enlarge the support area of each foot.

If you start to building the robot, you could modify the STL files before you 3D print.

These improvements decreased the gap between the feet to approx. 4mm, which means the CoM is only 2mm off the area of the supporting (standing) leg while lifting the other leg.

With this change the robot is still not stable and starts to tilt slowly to the side of the lifted leg, but this is manageable by using small steps.

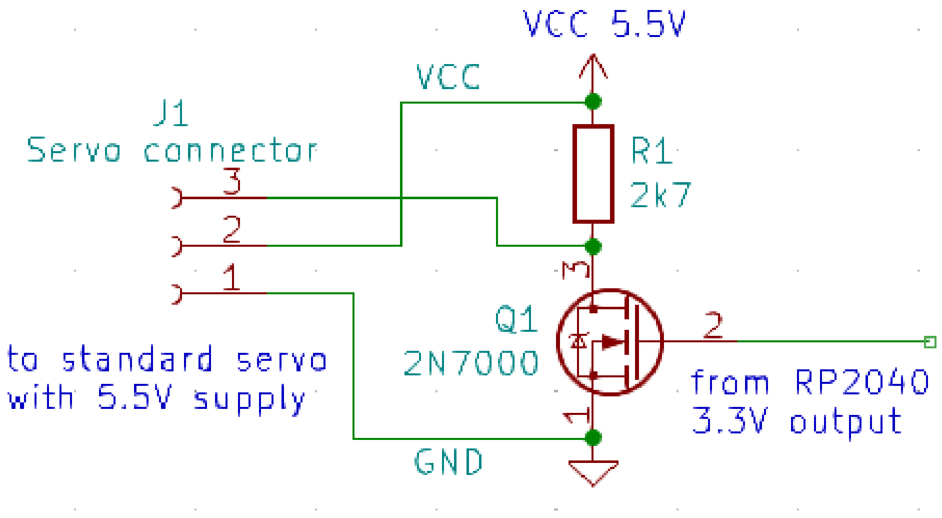
The servos:

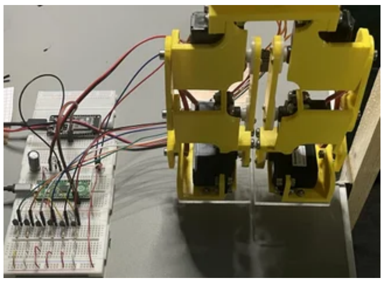
Although standard RC servos with plastic gears work fine and sofar I had no issues with broken gears, for durability probably servos with metal gears are a better choice.

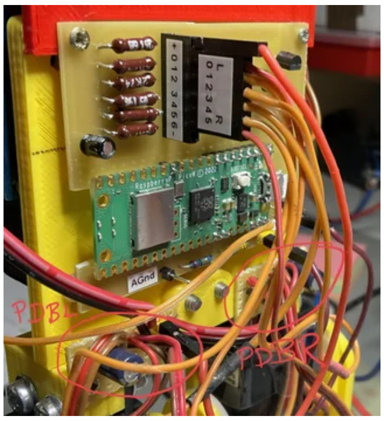
Credits:

Many of the ideas I got from an brilliant article by Stefane Caron Prototyping a walking pattern generator. This article goes beyond what I have implemented and beyond the capabilities of the chosen robot hardware, but the basics described in that article are helpful for any bi-ped robot project.

Step 2: Robi Electronics







I like to do projects with the RP2040 picoW board. It has plenty of IOs, RAM, Flash storage, WLan and most important the PIO with the programmable state machines -AND- the board can be programmed using Micropython. For me a perfect choice to run a robot using PWM controlled servos.

Few considerations, please refer to the attached schematics and pictures:

- The RP2040's IO output voltage range is 0V...3.3V. Servos run at 0V... 5V, so a level shifter is needed to boost the 3.3V to the 5V range. I am using a simple one Fet one resistor level shifter to drive the servos (see picture). In the schematics this is Q1 to Q6 and R1 to R6. The caveat, that this level shifter inverts the PWM signal, can be fixed by the software, which generates the PWM.
- The servos use much more power than a USB port can supply. They need a separate power supply, either a 5V power outlet or a battery and a DC-DC converter. In the schematics 5V power is supplied for the servos via J2.
- If you use standard RC servos a 5V 3 Ampere power supply should do to start the project, but consult your servos's specification to be sure.
- I added a battery and opted to use the USB power of the pico board to charge the battery, therefore I added Mosfet Q7. This is according to the Raspberry Pi Pico W datasheet (section 3.6 using a battery charger).
- The DC-DC converter's input is connected to the battery. I added a simple on/off switch. The output of the DC-DC converter is connected to J2 to power the servos.
- With Q7 in place the battery is charged when the USB cable is connected to the pico board. The pico board is powered by the battery, when USB is not connected.
- R7 and R8 are used as a voltage divider connected to ADC2 (Pin34). With ADC2 the battery voltage can be monitored. Lithium batteries do not survive a deep discharge, older batteries might actually burn. Please see your batterie's manual for specifications and warnings.

Wiring and connections:

- To check all hardware and basic servo controls, I built a bread board, using jumper cables to connect the servos. This is not very usful once the robot starts walking, but good to test the mechanics.
- Next I created a PCB with all electronic parts and two tiny power distribution boards.
- I split the 3 wire servo cables.
- All signal cables (orange color in my case) go to a 6 pin crimp connector and are connected the PCB.
- The servo power cables (brown for ground and red for +5V) of the left leg are soldered to the small power distribution board (PDB) on the left side.
- The servo power cables of the right leg are soldered to a small PDB on the right side.
- The 5V and ground power cables for the servos are connected to the left and right PDBs

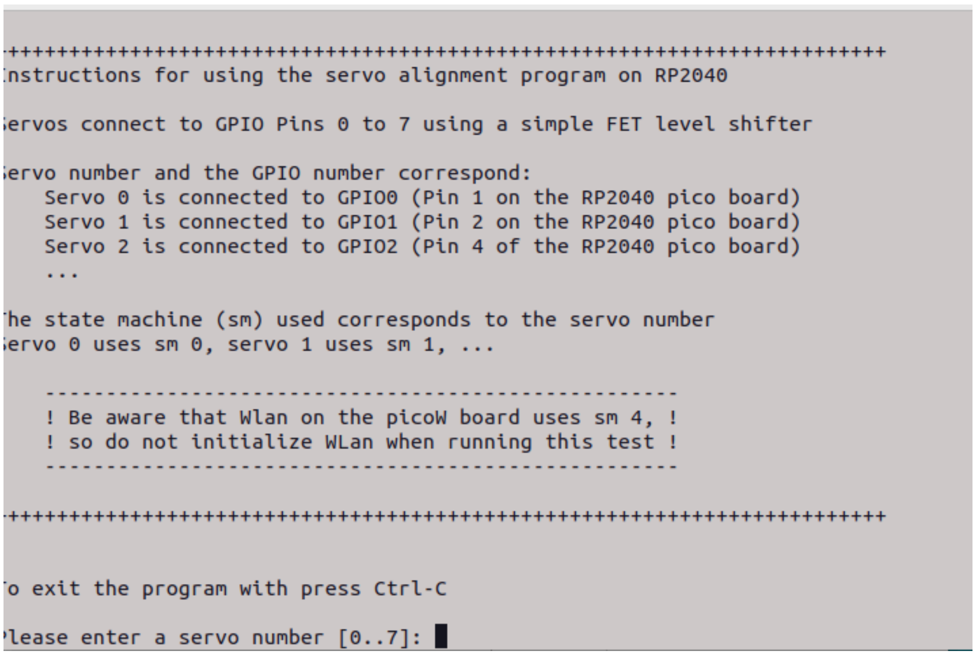
The last two pictures shows Robi's hardware fully assembled and servos connected to the PCBs.

For first tests it is a good idea to attach Robi to a fixture, so that the legs do not touch the bench. If something goes wrong, this avoids Robi falling off the table or servos getting mechanically blocked.

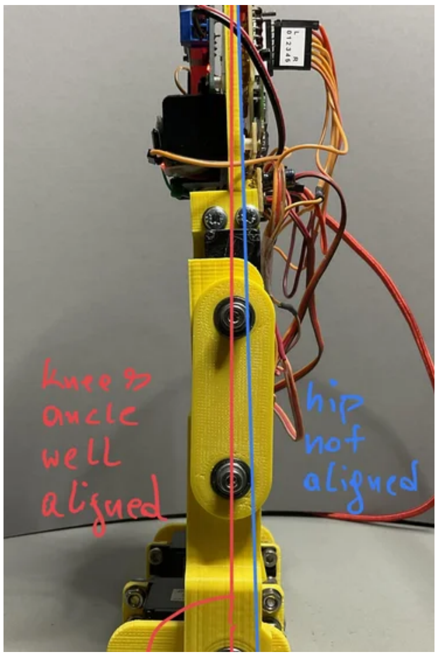
If you made it that far, congratulate yourself! You have a bi-ped robot hardware ready to rumble.

Next is servo alignment.

Step 3: Hardware Test and Servo Alignment



```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
nstructions for using the servo alignment program on RP2040

ervos connect to GPIO Pins 0 to 7 using a simple FET level shifter

ervo number and the GPIO number correspond:
    Servo 0 is connected to GPIO0 (Pin 1 on the RP2040 pico board)
    Servo 1 is connected to GPIO1 (Pin 2 on the RP2040 pico board)
    Servo 2 is connected to GPIO2 (Pin 4 of the RP2040 pico board)
    ...

he state machine (sm) used corresponds to the servo number
ervo 0 uses sm 0, servo 1 uses sm 1, ...

    -----------------------------------------------------
    ! Be aware that Wlan on the picoW board uses sm 4, !
    ! so do not initialize WLan when running this test !
    -----------------------------------------------------

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


o exit the program with press Ctrl-C

lease enter a servo number [0..7]: █
```



I am using Micropython version 1.19.1 as "operating system".

This instructable assumes you have some basic knowledge on how to use a Pico board, load micro-python and python modules to the Pico. In case you need help, please find the step by step guide getting started with micropython on raspberry pi pico.

Load the software to the rp2040 PicoW:

To load **Micropython version 1.19.1 or higher** onto the RP2040 PicoW board, please follow the instructions on the Micropython download Website.

Use your favorite IDE (Thonny, MU Editor, mpremote, mpfshell...) to download the attached servo alignment program.

When you start micropython on the rp2040 picoW board you get the message:

MicroPython v1.19.1 on 2022-09-14; Raspberry Pi Pico W with RP2040

Type "help()" for more information.

>>>

At the micropython >>> prompt type: *import servo_alignment* , which will start the program and a welcome screen will be displayed (see picture).

Servo alignment:

The goal of the servo alignment procedure is twofold:

1. to find **neutral position** of the servo, which is the correct angle of each servo so that the robot stands straight with both feet aligned (see picture). This is also called the servo offset.
2. to find the **endpoints** of servo movement, this is the maximum and minimum angle for each servo before it mechanically blocks. Not all servos can be moved the full 0 to 90 degree

range. E.g. the ankle servos can only move approx. between 10 and 75 degrees, otherwise they mechanically block.

Write down the angle for the neutral position and the end points for each servo. This data is needed later for the Robot controller. E.g. when Robi is initialized all servos move to their neutral position and Robi stands straight.

Servo alignment procedure:

Connect and do the alignment procedure for **one servo at a time**, keep the other servos disconnected

Preparation:

- at the first time mechanically disconnect the servo arm from the servo, as we do not know the servo's position at this time
- connect the servo power supply
- Hint: you can do the alignment for all servos with the same GPIO output e.g. Servo 0 @ GPIO0 at Pin1 of the pico board and connect the servos to their respective GPIOs when the alignment is completed.

Alignment Neutral position:

1. Start the servo alignment program and enter the servo number e.g. Servo 0
2. Enter the angle for the neutral position of the Servo:
3. left ankle: 45 degree
4. left knee: 0 degree
5. left hip: 45 degree
6. right hip: 45 degree
7. right knee: 0 degree
8. right ankle: 45 degree
9. The servo should move to the neutral position. Then you can connect the servo arm so that:

- hip and knee - the legs- align in a straight line
- the ankles align 90 degree to the legs
- Hint: depending on servo direction the neutral position for the knee servos could be 0 or 90 degree
- Remark: with the given angles the alignment will likely not be perfect. If there is a small misalignment of a few degrees, it is ok at this time. The misalignment can be corrected with a fine adjustment later.
- Leave the servo connected and continue with the alignment of the end points.

The picture shows the final result after the neutral position alignment of all servos. It shows as well a small mis alignment of the hip servo.

Alignment of end points:

1. Start the servo alignment program (if not already running) and enter the angle for the neutral position.
2. If the **neutral position is 45 degrees**:
3. move the servo towards the zero degree position in small steps, e.g. 45, 40, 35 ...
4. when the servo does not move anymore, this could be at 0 degree or a larger number, you are at the minimum end point. If the servo starts humming loud it blocks, then move back a bit by increasing the angle. Make note of the minimum angle.

5. move the servo to the neutral position (45 degree) and then move it towards the 90 degree position in small steps, e.g. 45, 50, 55...
6. when the servo does not move anymore, this could be at 90 degree or a smaller number you are at the maximum end point. If the servo starts humming loud it blocks, then move back a bit by decreasing the angle. Make note of the maximum angle .
7. Remark: the ankle servos will likely only move between 10 and 75 degree.
8. If the **neutral position is 0 degrees**:
9. move the servo towards the 90 degree position in small steps, e.g. 0, 5, 10, 15...
10. when the servo does not move anymore, this could be at 90 degree or a smaller number you are at the maximum end point. If the servo starts humming loud it blocks, then move back a bit by decreasing the angle. Make note of the maximum angle .

<u>Neutral position fine adjustment procedure:</u>

The initial alignment for the neutral position may not result in perfectly aligned legs and 90 degrees for the ankles. This can be achieved by tweaking the neutral positions of the servos.

1. Start the servo alignment program and move the servo to the neutral position.
2. Increase or decrease the angle by small steps, e.g. 1 degree, and watch the movement. You can do even smaller steps if needed. Increase/decrease the angle for each servo until the alignemnt is perfect: legs straight,left and right leg exactly parallel and both ankles 90 degree to the leg.
3. Once done, create a little table containing the neutral angle and the end points for all servos. Safe it, these numbers will be needed by the robot controller. E.g. when the robot is initialized all servos are moved to their neutral position and Robi stands straight.

Remark: If you play with the servo alignment program, the pico state machines might not properly de-initialize. If that happens they keep their memory allocated and you may run out of memory at a certain point. If you get an "*OSError: [Errno 12] ENOMEM*" do a soft reset (Ctrl-D). If a soft reset does not fix the problem, switch power for the pico off and on for a clean restart micropython.

Congratulations, now you have a ready to action and well aligned bi-ped robot hardware.
Stay tuned for the next instructable, the Robot Controller program, which will make the robot walk. It's in progress and coming soon!