Robot Dog Experience

James H Phelan

2022.09.11

Post to HBRC mailing list:

On Sun, Sep 11, 2022 at 8:47 AM James H Phelan <jhph...@hal-pc.org> wrote:

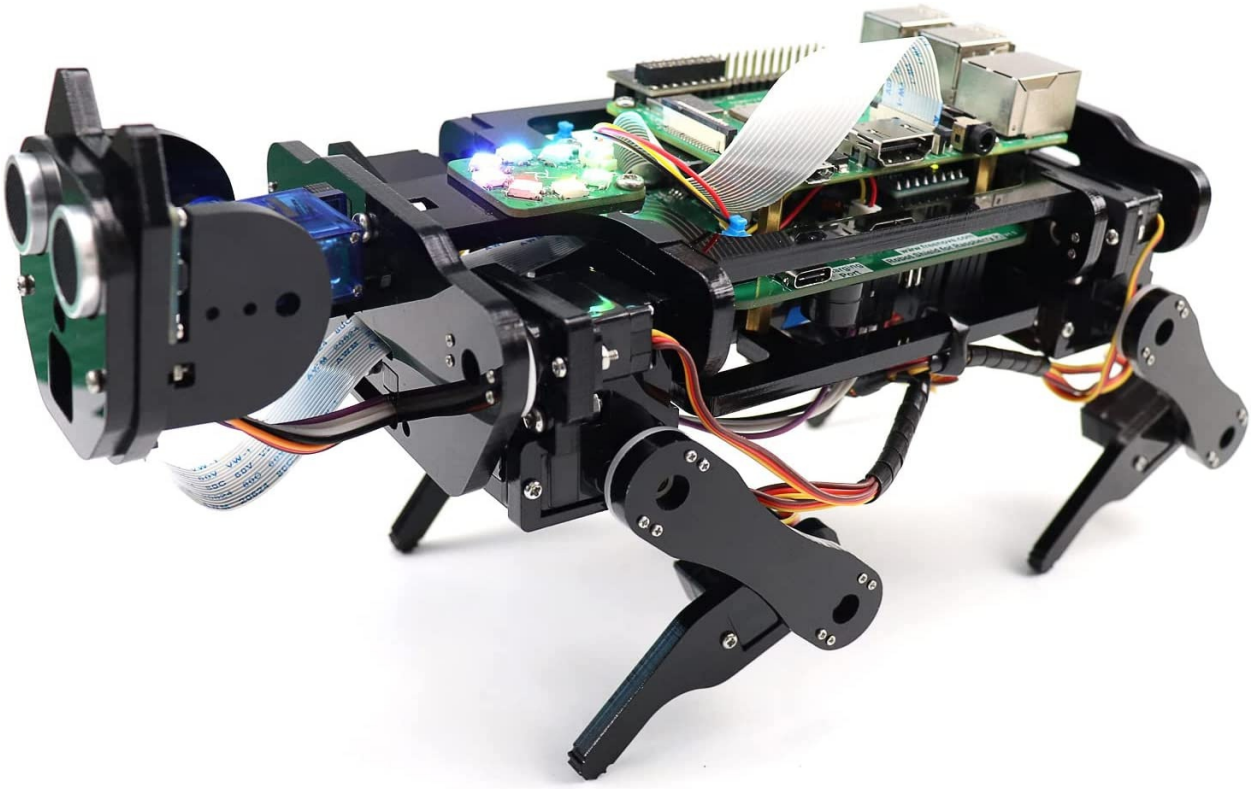
Team,

Stumbled upon this on Amazon. *[$130]* Since I already have a Pi4 for the Rover, decided to go for it!

https://www.amazon.com/gp/product/B08C254F73/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1

Robodoc

James H Phelan



On Sep 11, 2022, at 9:50 AM, Chris Albertson <alberts...@gmail.com> wrote:

I've been working on something like this, but larger. I think I spent more on plastic for my printer and screws than their asking price. I like that this robot comes with open-source Python code.

Could you give us a link to the Python code? They say it is printed on the box. I'd like to see how it works.

On Sep 11, 2022, at 1:35 PM, Dave Everett <daveev...@gmail.com> wrote:

I got a virus alert when unzipping the files.

On Sunday, September 11, 2022 at 2:41:34 PM UTC-7 Alan Downing wrote:

You can look at the online python files without cloning at:

https://github.com/Freenove/Freenove_Robot_Dog_Kit_for_Raspberry_Pi

On 9/11/2022 5:25 PM, Mark Johnston wrote:

I'm glad it runs on the 3B as well, I have those comming out of my ... um ... bench!

4B is a precious thing still and to get one is about the price of the robot kit!

On Sun, Sep 11, 2022 at 4:24 PM James H Phelan <jhph...@hal-pc.org> wrote:
Mark,
Good catch! I have a similar affliction with 3B's!

On Sunday, September 11, 2022 at 5:17:12 PM UTC-7 alberts...@gmail.com wrote:
I'm beginning to understand the software better. I could not find a step generator on the robot because that is not how it works.

There are two parts
(1) server that runs on the Raspberry Pi on the robot. It gets commands from the client and (2) the client. This can run on a Windows PC. The client sends commands like "take one step forward" or sets the lean angle or the height of the body over the ground.

So to walk continuously the GUI must send a series of "take forward step" commands

This is how the robot is able to walk to a red ball. The video data is sent to the PC and it runs openCV to find the ball and then sends step forward, or step left or other movement commands to the server. The server reads commands then calls the function to do the command, then reads the next command.

So I was able to read and understand all the kinematics and how the PCA9685 and the IMU were used but was frustrated because there was no "cycle" that made it walk. There is none. The cycle is done on the PC. The walk forward 8 steps it send 8 "forward" commands. This is very flexible and greatly offloads the Pi. No wonder any Pi can work. The PC is doing the heavy lifting.

That said. I bet $1 my software would work on this robot after a hour or so and even better the cvmp/champ ROS based stuff would work too.

One more idea, maybe the best... The server runs as a Linux process on the Pi. Write your own "client" that also runs on the same Pi and have it send "step forward" commands. You can also run a voice recognizer in the Pi and have the robot do simple voice commands. Or maybe just read from a game pad controller.

On 9/11/2022 6:26 PM, Chris Albertson wrote:
I have been reading the code, some observations

This robot uses the same chips I use in my robo-dog. I am not surprised because every robo-dog I've seen that uses hobby servos uses the same chips in the controller, That would be the PCA9685 for generating the pulses and the MPU6050 as the IMU. It seems that everyone who designs this uses the same solutions for the same reasons.

Movement is done in (x,y,z) space and then inverse kinematics is applied using what textbooks call the "analytic technique" where you just use a bunch of trig functions in custom code. Again just about everyone does this if the legs are 3-DOF or less.

The step generation is hard to figure out. In fact I don't see it (yet). I'm beginning to suspect they don't bother to do this and use a different method. Here is what I'm beginning to suspect: For forward movement, they translate the body forward by moving the legs backward then as a leg gets too far back and crosses a limit it is instantly zipped back to a neutral position where it again starts to translate backward.

This is actually an easy and fully generalized method but can't be extended to non-flat floors. The body moves by rotation and or translation as commanded from the GUI on the client system (windows, mac, Iphone,..) then as leg angles reach their limit they are reset to a standard neutral. This one method then does spins, crab walking and even normal forward. What I have not figured out is why all four legs don't get moved back at the same time. But I could be totally wrong. If someone figure it out please post. But I can't see anything that looks like what I would call a "gait generator"

About the mechanical design. It is completely conventional (but called down) as robo-dogs go. I try to look at every one of them I can find. This one is clearly optimized for low-cost and light weight. Mine is optimized for very high strength and repeated assembly/disassembly. They are polar opposites but logically share the same geometry. This one on Amazon is going to "eat servos" as all the weight is on the output shafts with high bending loads on the shafts. But it uses really cheap servos so you can afford to replace them. I used $20 servos so I placed the loads on external ball bearings. But my dog-bot might cost over $500 to build and this one is under $200. Functionally they are quite a lot alike.

If someone wants to take this robot as far as it can go. Try running the chvmp/champ software on it. https://github.com/chvmp/champ
This is a ROS based quadruped and is very sophisticated. I think this cheap dog-bot could run this software. With this you get the full SLAM and obstacle avoidance, Gazebo simulation, and everything else you'd expect.

On Mon, 12 Sept 2022 at 05:41, Alan Downing <downi...@gmail.com> wrote:
The code appears to be available via:
git clone https://github.com/Freenove/Freenove Robot Dog Kit for Raspberry Pi.git

On 9/13/2022 1:56 AM, Mark Johnston wrote:
Very helpful Chris and Alan. Thank you for your insightful and informative comments.

I get my 'Fido' tomorrow. I would be interested in something Chris has suggested which is form some scripts to do the commands to the on-board server for 'The Pooch'. Sort of open loop but still it would allow standalone minimal operations.

Thanks to others who have commented as well as James of course for another cool find. Looking forward to a whole lot of fun!

On 9/13/2022 8:28 PM, James H Phelan wrote:
Dear Freenove,

On unzipping fnk0050 I get an error on two of the .dll files that the "name is too long".
I'm assuming if I rename them, the main application won't be able to find them.
Can you shorten the names?

Thanks!
Jim Phelan

On 9/13/2022 9:40 PM, support@freenove.com wrote:
Dear customer,

Please try this one:
https://drive.google.com/file/d/1rRrmtcX6ePmtjQjCpr01ftllK7U5382x/view?usp=sharing

We will wait your feedback.

Denzel
Freenove Support Team
---
www.freenove.com

On 9/13/2022 10:12 PM, James H Phelan wrote:
RoboBuds,

My "dog" arrived this evening & unboxed. Nicely packaged and laid out.
Scanned the Tutorial.pdf and was VERY impressed with the detail, clarity, and examples throughout!
Looking forward to assembly and programming!

RoboDoc

On 9/14/2022 12:24 AM, Dave Everett wrote:
On Monday, 12 September 2022 at 7:41:34 am UTC+10 Alan Downing wrote:

> You can look at the online python files without cloning at:
> https://github.com/Freenove/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/tree/master/Code

Ahh yes but I couldn’t pass up the opportunity to buy yet another robot dog so I will need to run those files :)

Dave

On 9/14/2022 2:57 AM, Dave Everett wrote:
Good to hear of your progress. I'll be keen to see any changes you make to the walking. As with most robot dogs I've seen the legs seem way too far apart in the X axis, that is the 2 front legs are too far apart and the 2 rear legs similarly. It makes smooth walking impossible. I suspect I will redesign and cut the body to put them closer together.

I have seen to date no real dogs with legs so far apart, I suspect because any examples failed to walk to a suitable mate to pass on their genes.

Dave

On 9/14/2022 2:42 AM, Mark Johnston wrote:
Having completed all servo assemblies (about half way through the build). I feel that the statement 'Some assembly required' is a very safe and true statement.
The only part of all of it that was not so fun was removing all the dang paper that was on both sides of the (very nicely) laser cut parts. They do include a little tool to help but it was almost easier to use fingernail to get a corner going then peal it away. Again, THAT was anoying but the parts are shiny black and are very nicely cut.

The next step is getting the Pi to 'zero' all the servos then attach the servo assemblies. Its quite a build but I must say, rather fun.

I agree with Chris when he said this thing will eat servos. They are not beefy and are just the MG90S (ish) sort of servos. When I run this thing I am going to be very careful to not let them overheat. At least the servos have brass gears so that makes me feel a tiny bit better.

The last half of the build will be power up the Pi (I'm going with Pi 3B for now as my Pi 4 units ... 'They are precious to me'. One Pi to rule them all!

With some luck I may have this puppy at least assembled with zeroed servos tonight (note the use of 'puppy'!)

Mark

On 9/14/2022 2:57 AM, Dave Everett wrote:
Good to hear of your progress. I'll be keen to see any changes you make to the walking. As with most robot dogs I've seen the legs seem way too far apart in the X axis, that is the 2 front legs are too far apart and the 2 rear legs similarly. It makes smooth walking impossible. I suspect I will redesign and cut the body to put them closer together.

I have seen to date no real dogs with legs so far apart, I suspect because any examples failed to walk to a suitable mate to pass on their genes.

Dave

On 9/14/2022 5:52 AM, James H Phelan wrote:
FWD: Freenove response to my filename too long question.

Puppy pals:

Freenove is VERY responsive to problems!

RoboDoc

On 9/14/2022 11:04 AM, WILL fill wrote:
If you just want to get your zip working you can enable long paths on windows. Go to Registry Editor->HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem and change longpaths to a 1.
you can see in this article:
https://helpdeskgeek.com/how-to/how-to-fix-filename-is-too-long-issue-in-windows/

On 9/14/2022 11:27 AM, Chris Albertson wrote:
As an experiment, prior to rebuilding the chassis, a simpler approach is to modify the software to move the feet closer to the center. Angle them in a little. In the code supplied, they define a "neutral" stance and this would be easy to redefine.

I've experimented with walking with a wide and narrow stance. In seems intuitive that a wider stance would be more stable. It is until the robot tries to walk and lifts a foot. If the stance is more narrow, then lifting a foot creates less imbalance. The real problem is the servos are not precision devices and the feet on the ground don't have equal pressure and a narrow robot falls over. There is an optimal width should I say a "least bad width" that you can find by experiment. As said, the way the code works is by reference to a neutral stance that is defined by some constants. Just change the constants.

The other thing I notice in their software is that the IMU is not used except when running the

"balance trick". It seems not to be used for normal walking. So if the stance is very narrow keeping the feet under the "zero moment point" might be hard. It is easy to keep the feet under the geometric center of mass on a flat floor, but if the robot moves, momentum matters. But my experience is with my MUCH heavier and much taller robo-dog.

My conclusion from experiment is that if the robot can actively balance then the best stance width would be what my real dog does. She moves her feet inward as close as she can, maybe to half her shoulder width when at a fast trot. But with no ability to actively balance (the IMU is of little use if using hoby servos) then you need a wide stance for better static stability. The problem with servos is (1) They are slow to move, any balance correction has lag and (2) the positioning is imprecise. Using BLDC motors and digital shaft encoders would be ideal, but they are 25 times more expensive.

For a given speed, the product of step-length times step-rate is constant. So you get to choose a trade-off. With no active balance, short steps "win" but it looks very unnatural.

I think you get the point. Putting that IMU to work is very important for dog-like walking. If you ignore the IMU you end up with very robot-like walking with short, fast steps and wide stance.

Look at the chvmp/champ software if you want a near state of the art dog-bot. I'm trying to create a more flexible software in pure Python from scratch and it is slow-going. This Amazon robotuses the same PCA9586 cip at the same I2C address as every other sevo-controlled dog-bot so software is somewter portable
github.com/chvmp/champ

On 9/14/2022 11:30 PM, Mark Johnston wrote:
WARNING ON FIX FOR THE Servo.py that sets servos to 90 for proper assembly:

I found a couple bugs.

1) The print statements that were printing 90 for 90 deg (on two different lines) were failing so I just made them print 90 , plain and simple, no fancy trick for the 'degree' character

2) The while for 16 servos needed a delay between servo settings or it bombs out of the script.

So make your __main__ look like this for the fixes I had to use for the script used to set servos to 90 so proper angles can be had on legs

```
if __name__ == '__main__':$
    print("Now servos will rotate to 90") $
    print("If they have already been at 90, nothing will be observed.")$
    print("Please keep the program running when installing the servos.")$
    print("After that, you can press ctrl-C to end the program.")$
    S=Servo()$
    while True:$
        try:$
            for i in range(16):$
                S.setServoAngle(i,90)$
                time.sleep(0.02)$
        except KeyboardInterrupt:$
            print ("\nEnd of program")$
            break
```

On 9/14/2022 12:46 PM, Chris Albertson wrote:
Is this a filename that is too long or a pathname that is too long? I did not see a long filename in the Git repository. If the path length is too long, then place the file on the root directory on Windows that is directly on the "c" drive.

Why use the zip file? The files or all on Github, just clone the repository. This is best, as Git will maintain a change history for you as you make edits. And merge your edits with any future updates from the developers.

On 9/14/2022 11:51 PM, Chris Albertson wrote:
I think the problem is writing data too fast to the I2C interface. Did you set the bit rate to 400,000? (The default is 100,000.) If not, this is a big performance hit.

For a setup program, this is ok, but some improvements are

    1. Just write to the PSA9658 control registers once and set all PWM channels to 1000 uSec at the same time. No need to loop over range(16)
    2. Why loop forever? The controller chip does not forget its settings until you kill the power.

But I'd not waste time on fixing something that works and is only a test/setup program.

On 9/15/2022 1:48 AM, Mark Johnston wrote:
Status: Robo-Pooch is fully assembled and all tests in the test.py all run from servos, sonar, camera, led, buzzer.

I am using a Pi3 B with the Ubiquity Robotics most recent Kinetic image and it all works so far.
 I did that so I can have ROS (I) completely present already and I am very used to that image.

Next will be to connect up this 'puppy'.

One more suggestion: The head uses some M2 x 14mm screws.   They 'may' work but are too close for comfort for me so I attached the blue servo with some M2 x 12s I had in my stash.

The build is done but cable dressing I may do next or wait till this pup does some more 'doggie-like' things.

To Chris: Yes I set i2c 400000 clock.   I2C is working and sees the required 3 devices and does all the servos.
Maybe it is really not at fast clock but I'm ok for now and don't want to get distracted.

On 9/15/2022 3:11 AM, Mark Johnston wrote:
It was not bad to just make automated sequences in python starting from their   test.py script in the 'Servo' command execution.
Was with a little bit of sorting out direction of rotation vs what side of robot able to do simple push-ups on front legs.

This will be a ton of fun but I have to watch the servos and not burn them out. That is my prime fear here. They are 'whimpy'

Mark

On 9/15/2022 5:49 AM, James H Phelan wrote:
"Why use the zip file? "
Because the box directions said so to get the tutorial.
The actual code is later downloaded via git clone per tutorial directions.
"I am only an egg" when it comes to git, but I'm improving.

RoboDoc

On 9/15/2022 6:04 PM, Chris Albertson wrote:
If anyone wants, I can point you to some STL and CAD files. My "dog" is literally plug-compatible with the Amazon robot. (every servo powered dogbot seems to be a copy of every other servo powered dogbot)   Mine will cost you two spools of printer plastic and the cost of 12 of the best servos you can afford. plus $125 in stainless steel nuts and bolts and ball bearings and some high grade batteries.   The robot I have is very robust.

But really, It would be easier to just buy a dozen spare 90g servos. They are only $5 each.
This Amazon dogbot seems the ideal entry point for walking robots.

I've been mining the software for good ideas.

On 9/16/2022 2:24 AM, Mark Johnston wrote:
I will be now looking about for self contained 'walk a little ways' or 'walk in a circle' sequences. I am hoping there are some in the python scripts in their github. Time to look!

This is a fun diversion to be sure!

On 9/16/2022 2:35 AM, Mark Johnston wrote:
Ok, found great little sequences in Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server in Action.py Once you get going, check that module out.
It seems RIPE for building upon!

On 9/16/2022 9:31 AM, Sampsa Ranta wrote:
I liked this toy fellows open dog.
https://www.youtube.com/watch?v=eKZIJwJBjEs
In the videos he also explains the inverse kinematics and has open source code.
Not exactly same ecosystem stuff.

Enjoy,
 Sampsa

On 9/16/2022 12:25 PM, Chris Albertson wrote:
But it basically never worked well and he seems to have given up. His problem was the size and weight of this robot and the inflexible software. Don't under estimate the software. Most robot dogs use very simple software and fixed leg motions, These only work because there is human-in-the-loop control or they go for chvmp/champ. Pretty much the two extremes.

One of my tasks, coming up, is to see if my robot can stand on two diagonal feet. The feet are spherical balls so some balance is required. I doubt it will work well with hobby servos but it will have to work "well enough" for realistic dog-like walking to work.

On 9/18/2022 9:02 PM, Mark Johnston wrote:
Tips On RoboDog code fragments.

I had only one bad servo but am going to put in an order.

I printed some much taller stand pieces so I can dry run things. The stock ones are 80mm and good for 'training wheels' but do not fully isolate for your in-place tests. I attach an STL if you want to print some taller stands.

Here are some nice pieces to help other get going faster as I had to RTFS to find some/all of this.
Get into Code/Server as that is where the good stuff is located if you are going to try to make a self contained doggie (no windows PC embilical)

python test.py <command> This is very first thing to run and instructions discuss this. This at the end has the command interpriter and it a good 'back to test mode' tool

python Action.py This has some more complex movements but no 'interpriter' so go to end and uncomment the sequences you want to play with.

Control.py This is mostly imported by other tools and has a great many really nice starter primitives' for movement

python myCode.py This will go through many sequences and this is the 'gold'. For your own stuff take a look at all the things it does at the end calling methods.

myCode.py seems like a great tool to cut-paste out your own things that are perhaps done in a sequence you want as you develop your robot full standalone app. It does not use all the interfaces in Control.py but does use forWard() and there is in Control.py many other primatives such as backWard() Control.py has the primatives but also a big case statement for all the CMD_ commands.
so Control.py methods will be invaluable to make your doggie do things like walk around in some pattern or smething you want .
Later Control.py will be valuable so you sense things and then do different movements on how you want doggie to do stuff from external inputs.

Now it's time for some fun!
Mark

On Sunday, September 18, 2022 at 7:24:00 PM UTC-7 jhph...@hal-pc.org wrote:

Mark,

Just tonight got mine to the point of fully assembled, up on calibration stand and calibrated the best I could.
Looks like one servo isn't responding at all and another is slow and stiff.
I replaced the unresponsive one with a spare in the kit but will have to wait til tomorrow to re-calibrate.
What do you recommend for replacement servos? Got a link?
Would like a stand that would be easy to mount and dismount instead of 4 bolts with hard to reach nuts.
Will have a look at yours.
Explored the iPhone app, pretty good.

RoboDoc

On 9/18/2022 10:14 PM, Mark Johnston wrote:

Servos: I started reaching for my box of 9g style servos. Here is what I have found: I have none of my favorites as a 'match'

They use a brass head (and hopefully metal gear inside is my guess) 9g sort of servo but I have not measured if they are 9g or beefier.
Main trouble is these servos use a 15 tooth main gear that is 3.9 mm across and are totally void of any label.

My fav metal gear one is the MG90S but it is aluminum gear of 20 teeth and 4.8mm across so total non-fit.
Then I started looking at my plastic gear ones like Tower Pro Micro Server SG90. It is at least 20 teeth and about 4.8mm dia so it was also no-go.

I am joining a facebook 'Freenove Robot User group' but cannot post yet till I am 'in'. I have been browsing for type of servo, no joy yet.

On the mount easily, yes, I get that it would be nice for push-pull pin to mount those, have not done that 'yet'. Screws are a PITA for this sort of stand to swap them.

I'll know more after more research

Mark

On 9/18/2022 10:19 PM, Dave Everett wrote:
My dog has apparently arrived. I won't pick it up until tomorrow night. I have already been thinking about redesigning the laser cut parts if need be, and using different servos. I will post a link if I redesign to use better servos.

I also found they do an ESP32 version, so of course I bought that as well.

Dave

On 9/18/2022 10:55 PM, Mark Johnston wrote:

Found these servos as candidates and am going get some on amazon as they would be of use for other servo needs anyway.
All my servos are ok with only 1 was bad but note somebody else also has 2 servos DOA.   So count on some servos going to servo heaven (or hell ... not sure)
These servos they ship rotate 90 deg with near 10% duty cycle change (roughly measured today)

Anyway, what I found look like 15 tooth brass gear and are 12G so this would make perfect sense a bit more beef than the 9g models AND even internal gears look brass.
Link below has a bit of a spec for the ones I am buying but I do not have a spec for what is really in the dog ... still looking for that.
What I am going to try:
https://www.amazon.com/gp/product/B07RRWYXL9/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

Also: Yes, I saw they were doing an Esp32
Also the facebook group I discussed is mostly on the hexapod

On 9/20/2022 8:59 PM, Mark Johnston wrote:

Well forget about that push pin, it seemed a nice idea but due to stresses of a rather heavy robot dog above it they are not as strong at all as a nice metal screw. So scratch that pin!

Next: I now got today 12g servos that look identical to what is required. Have not tried one yet but seems very good. I'll try a 'knee' tonight.
They are the link earlier posted:
https://www.amazon.com/gp/product/B07RRWYXL9/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

Also I fully got my RoboDoggie to now have interface board for this RF controller I have used many time and I can go 'forward, reverse, right, left' and it all fully starts up from power-on so it is in a crude way fully standalone and operational! A Milestone for my goals so far.

I will show this thing at the hbrc meeting on 28th is the plan, hopefully with some modest ROS integration which is doable if all goes well.

Now it's off to try to make 'rubber boots' for the feet because the hard Acrylic feet slip too much. I will make feet using 3d printed PolyFlex or maybe NinjaFlex, both of which are 'rubber-like'.

Take Care you Scurvy Dawgs! (yesterday was ITLAP day, International Talk Like A Pirate Day)
Mark

On 9/23/2022 9:34 PM, Chris Albertson wrote:

Has anyone thought about how they would control a quadruped "dog-bot" using a game pad? It is easy enough to handle events from the gamepad, but what I mean is how should the controls work? For a simple differential-drive car a single joystick controls forward and reverse and rate of turn. And that is all that a simple two-wheel robot can do.

But the dog seems like it can do more things than there are buttons on the controller. While standing still, with all feet stationary, it can:

- translate its body in all three directions (x, y, z)
- rotate its body in three directions (roll pitch and yaw)
- modify its stance (width of the legs and length in fore/aft direction)

Then, in addition while walking it could also

- Make turns where the robot walks in the direction its head faces
- walk in other directions the head is not facing
- rotate in place by shuffling feet. Note that the center of rotation need not be directly under the robot's mid point.
- Pick up the feet higher.
- trade faster the longer steps to go the same speed. (step length vs step frequency)
- change gait from creep to trot or others.

Then there are poses, like lay down, sit up and "whatever" And then movements like "kick a ball" and eventually climbing a stair step

So far I have an X11 based interface using a mouse and a set of sliders. While the robot can do 9 motions all at the same time. I can only change one parameter at a time with my mouse.

I'm going to have to do something as controlling all this with a mouse is awkward. Anyone have ideas on how a dog-bot hand controller should operate?

On 9/24/2022 9:21 PM, A J wrote:
I have only used the microsoft controller with the quad copter in ros.
But that only has yaw, roll and pitch compared to a multi-legged robot.
I suspect that a higher level abstraction is easier because there many
patterns of walking and running.

https://support.bostondynamics.com/s/article/Spot-controller

https://www.sciencedirect.com/science/article/pii/S2405844018326938

On 9/24/2022 10:31 PM, Chris Albertson wrote:

This is my thinking too now. A lot of the robo-dog is going to have to run on autopilot. But I need to gain some experience driving by hand to know how to program the autopilot.

This means I'll control some things and the others will have to be computed.

The best solution would be a custom controller box but I don't want to take the time to build one. I thought of using a MIDI control surface like the link below. It has 16 analog controls and about 3 dozen buttons for $80. It is a USB device, and every OS now has MIDI drivers already installed.
https://www.sweetwater.com/store/detail/nanoKON2bk--korg-nanokontrol2-black?main_web_category_rollup=2/21/726/808&utm_source=google&utm_medium=organicpla

I'd like it to be self-driving but it will take lots of experiments to figure out the best ways to move.

My worst problem is the servos, they get sloppy quickly as the metal gears wear in.

On 9/25/2022 9:53 AM, James H Phelan wrote:

Robobuds,

After assembly discovered that some of the servos weren't responding.

Roberto loaned me a servo tester and a handful of identical replacement servos he had from Freenove.
I'll have to ask Freenove for some replacements. They've been responsive so far.
Replaced 2 servos and was able to calibrate FIDO and take it for a successful short walk before it demanded a rest.
It's not as graceful as Spot, but at $130 vs $74,500 I'm satisfied.
I'd like to try some of the other software mentioned previously in this thread.
I tried not bolting the calibration supports to the dog but instead just fixing the screws to the supports then poking them into the dog. Worked good enough and easy to remove.
Would like to find some rubber ball feet for traction. Perhaps these:

https://www.amazon.com/Umarex-Paintball-Count-50-Caliber-Rubber/dp/B08J62CBW7/ref=sr_1_1?keywords=50%2Bcal%2Brubber%2Bballs&qid=1664116112&sprefix=%2C50%2Bcal%2Bru%2Caps%2C302&sr=8-1&th=1&psc=1

RoboDoc

Dear Freenove,

After assembly I discovered that some of the servos weren't responding properly.
A friend who had bought the same dog in the past loaned me a servo tester and some identical replacement servos.
I replaced 2 servos which had responded slowly, stiffly, and incompletely in testing.
After reassembly I was able to calibrate FIDO and take it for a successful short walk before it demanded a rest.
It's not as agile as Spot, but at $130 vs $74,500 I'm satisfied!
I tried not bolting the calibration supports to the dog but instead just fixing the screws to the supports then inserting them into the dog. Worked good enough and much easier to mount and remove.
I would suggest modifying the instructions for this purpose.
I found the Windows client.exe would open a terminal window, then disappear. It worked better when I unzipped the master.zip into Program Files (x86) and ran as administrator.
The iPhone client works well too.
Would like to find some rubber ball feet for better traction. Perhaps these:

https://www.amazon.com/Umarex-Paintball-Count-50-Caliber-Rubber/dp/B08J62CBW7/ref=sr_1_1?keywords=50%2Bcal%2Brubber%2Bballs&qid=1664116112&sprefix=%2C50%2Bcal%2Bru%2Caps%2C302&sr=8-1&th=1&psc=1

Would it be possible to get at least 2 replacement servos so I may repay my friend?
If you wish (doubtful!) I can return the defective servos to you for quality control.

Thank you!

Jim Phelan

On 9/25/2022 2:54 PM, Chris Albertson wrote:
That is my current project, making rubber feet for my dog-bot.

What I did and I'd suggest it, is to make some improvised feet first with foam and tape. Figure out how big and how squishy or hard they need to by using whatever foam and tape and rubber bits you have around.

I plan to 3D print the feet from TPU. I can control the stiffness by setting the infill percent, adjusting the skin thickness and making the tread pattern more of less deep

These will be the first generation feet. The second generation will have pressure sensors. Foot pressure is really need if you ever want to walk on not-flat paths. The robot moves its foot down until it "feels" something like 1/2 the robot's body weight on the foot. The robot can also have a sense of balance by adjusting its stance so the weight on each foot is the same.

You can buy resistive pressure sensors for about $10. They change resistance with pressure and come in all different sizes and shapes.

In any case, try "gaffer tape" or even white cloth "sports tape" or bits of bicycle inner tube superglued to the feet. and then you get a better sense of what kind of ball to look for.

So people have used silicone seal window caulking for feet. This can actually look really good if you 3d print a mold (prep the mold with some car wax to help release.)

On 9/26/2022 12:52 AM, Mark Johnston wrote:

Have been away on a 3-night deep sky astronomy star party. Back on this now but time is tight for Wed.

James: I really think the ES08MA II 12g Mini Metal Gear Analog Servo is completely fine and perhaps exact replacement.
Do NOT use other gear teeth. It is not as simple as 'drilling out'. You really gotta match all those gears or you are in for a nightmare of flakey mechanical issues.

Chris: I at this time use a simple RF DIYhack for forward, reverse, right, left. My goal next but likely AFTER this HBRC meeting is to use the joystick node and use a joystick. It will always be non-precise and just hand operated but I would use Logitech or even X-box joystick with RF dongle.

Feet: I have now a 3D 'boot' and the disclaimer is this is first one that 'fits'. It may not be optimal. This has a flat side so it can be printed on an FDM printer using 'rubbery' stuff where I use PolyFlex and will do some in softer NinjaFlex if required. To print these soft FDM materials is an acquired 'art'. Too deep to try to explain it here but I offer STL if you have a nice process already worked out. My 'push goal' is to bring some to the HBRC meeting but it is not a 'promise'. Plenty of room to improve on this but its going to be better (Higher static coefficient of friction) than hard feet the robot uses from laser cut acrylic.

Mark

On 9/26/2022 5:59 AM, James H Phelan wrote:
Mark:

< Do NOT use other gear teeth. It is not as simple as 'drilling out'. You really gotta match all those gears or you are in for a nightmare of flakey mechanical issues. >

Pardon my ignorance but I don't see why.
There is an exact replacement so the question is moot but - assuming
1) the body of the new (20 tooth) servo is an exact fit
2) all the servos are replaced, not just some (although just opposite pairs might work?)
3) the servo teeth don't interact with any other gears, just their own servo horn, which is true.
The issue is whether the new servo horn could be made to fit where the old one was which might require new screw mounting holes.
The greater number of teeth might even make mechanical centering of the joints a little more (20/15ths) precise?

RoboDoc

On 9/26/2022 7:35 PM, Mark Johnston wrote:

On 20 tooth servos: When you said drilling out I thought you meant the existing horns. To replace all servos and all horns is of course okd. It of course would be a a massive undertaking. If you replace all the mating servo horns, sure. But wow, nasty bit of rework effort.

I think I lost track of why this is a good thing. Maybe because the 20 tooth ones have much better stall torque or reliability? Those would be the only reason to think about such an effort or perhaps other reason.

This may be a good 'segway' into something I must still do. I thought there was some sort of 'soft' cal where each servo had in a .txt file some sort of pwm offset and scaling and there was this gui I saw with lots of controls. I thought there was something to put soft servo cal data in a file for their drivers to use. So far I have thought this was in the gui of the 'client' in some way that is seen on page 97 of the turorial.

In the turorial on Step 15 of their stuff called 'Verify Assembly' is what I would call a rough 'physical' alignment, NOT a 'soft calibration'.

If there is no soft calibration then I could also see the need for finer teeth than 15.

Anybody read enough to find and do the 'soft cal' if it exists is documented? I'm a bit tight for time so 'throw me a bone' if ya got one.

Thanks, mark

On 9/26/2022 8:47 PM, James H Phelan wrote:

Mark,

Yeah, once you get the dog assembled and, when each servo is in "center" position which is how it's designed to start up, you can mechanically "center" the distal limb on the joint.

Then, once assembled and on the stand, you download the "client" onto your PC. I find it worked best to unzip the "-master.zip" into the C:/Program Files (x86) to avoid the

"long file name" problem. Run the client.exe As Administrator. This will pop up the GUI.

"Connect" to the IP address of the dog. Then "Calibrate".

This puts the dog into a standing "relaxed" pose, as opposed to laying down in "center" position.

You put the dog on the calibration drawing aligning the stand with the rectangles on the chart and the dog facing the head on the chart.

You then click on the GUI to move each foot forward/back, in/out, up/down til the foot rest on the indicated landing spot.

Once you've done all 4 feet, you click "Save" and it remembers the servo calibration (somehow). I'm sure you could dig into the code and figure where.

I used a PostIt note as a feeler gauge so that each foot would "almost" touch the chart to keep any foot from tipping the dog and throwing off the calibration.

Once done, you can take the dog off the stand and take it for a walk. Worked well on my rubber desk mat. Haven't challenged it with other surfaces, yet.

I've collected the correspondences in this thread and my step-by-step odyssey into the attached "Robot Dog Experience.pdf", so someone else might learn from my misadventures.

RoboDoc

On 9/26/2022 9:19 PM, Chris Albertson wrote:

This does have to do with both servo calibration and "boots". You can never get the servos calibrated 100% and this means that even on a level hardwood floor foot pressure will be uneven and it will change randomly. Some feet will have weight on them and one might even be in the air. The dog is directionally unstable. It seems to turn based on which ever foot happens to have to most friction. Flexible feet help.

Do watch the dimensions of the feet. They add length to the lower leg and mess of the inverse kinematics, You might have to measure and adjust constants in the code.

I think I found a way to steer a robodog. I got this working today. At first I tried "tractor turns" where to turn left, the right legs take longer steps and the left legs take shorter steps. This fails badly as it requires controlled sliding and which feet slip is random.

Then got this to work. To turn left, each front leg when it moves forward to take a step moves also a little to the left and then as it moves back, it moves back to the neutral position. The rear legs step to the right. This way the body rotates as it walks. The exact details depend on some trigonometry. But it roughly works. I command a 0.1 radian per second turn and the robot turns very roughly 0.1 radians per second to the left.

I can also now do roll, pitch and yaw rotations and x,y,z translations with stationary feet and also while walking. The software takes the rotation, translation, and gait and turning inputs and adds them together. Then computers the servo angles, writes to the controller chip then does it again, 20 times per second.

One more thing, I found my gait generator is not generalizable to curved walking paths. It assumes the feet move in straight lines when looked at from above. This is not true when walking around a corner. There is always more to do.

I'm controlling it all from an iPad now and this works well enough that I may postpone using the gamepad. No fancy ISO app, just an X11 GUI running on the robot, exported to the iPad screen.

On 9/26/2022 11:15 PM, Mark Johnston wrote:
My goal is not to make this thing navigate well. I will get around to responding to visual input so if feet slip it means keep doing it.

It is more a curiosity with a cheap-o toy-like doggie like this one.

The calibration I am seeking is for the legs (which use 3 servos each) to be 'about' in the same position when the code tells them to all go to some hip, upper leg and lower leg joint angles. Now I have some that are clearly off by quite a bit but less than one of the 15 gear teeth which is a very bad 24 deg per tooth.

On 9/26/2022 11:53 PM, Chris Albertson wrote:

They will never navigate well by open loop. I was only driving by remote control. So when the user presses the control left, what should the robot do? There are a dozen ways to steer a quadruped. With no user input the robot rotated 90 degree left after walking 5 feet. So I tried tractor turns and it only made it worse and more random. It turned out what worked was rotating the robot on the z-axis while it was walking.

At this point I have very low requirements. All it should do is follow the remote control. I'd like it to go from living room to kitchen and back while I control it with my thumbs on the controls.

Once the remote control is working, then I'll put a camera on it

On 9/27/2022 3:35 AM, Mark Johnston wrote:

Yes. Same goal. The human is the 'navigator'. That is 'Phase 1'.
While Phase 1 happens start to hook up ROS interfaces for sonars and camera.
When I get my nifty ArduCam TOF gizmo in Nov, play with that OR I may just put that gizmo on Droidbot that has a chance at navigation.
Robo Doggie may just use camera then send it back on Rviz. Just feedback to Rviz for sonars and camera for a while for Robo Doggie.

If I set my expectations low for El Cheapo Robo Doggie I will likely not be 'very' disappointed! Lol

On 9/28/2022 1:09 AM, Mark Johnston wrote:

Did the calibration with stands and gui. Had to have GUI on the main.py code on the Pi which IMHO was a pointless gui and it would have been lower 'baggage' if I could start python tool main.py from a ssh console screen. But I dorked around and got it running.

Sure enough, the cal data is saved to Code/Server/point.txt which is a flat ASCII tab separated file VERY easy to manually tweek too!
This is the cal I wanted so now my own app from my RF controller has the RoboDoggie standing nicely.

My polyflex boots work but I want even more grab so am now trying this with NinjaFlex which is the softest material I dare print at Shore hardness 85A which is 8-10 softer on the Shore scale of A (the softest scale).

On 9/28/2022 1:26 PM, Chris Albertson wrote:

To make softer feet, use less infill percent and thinner wall. Solid ninjaflex is actually pretty tough stuff, But the same stuff mixed as 20% ninja and 80% air is softer. I think a tread pattern in the bottom also helps, like knobby tires. I find that even with well calibrated servos there is always one shorter leg so what is needed is compliance.

But my "dog" weights in at about 10 pounds with larger feet so there is plenty room for detail design of the feet. Mybe you don't have room for these details in the tiny feet?

I've decided to start thinking about "software version 2". The current software is gathering many special cases and "if/then" checks. I think I can generalize it. The method described below uses some ROS specif language, but need not use ROS. So here is a description of "V 2.0"

The quadruped base controller subscribes to two input messages

1) Cmd_Vel, for "twist" messages. The following fields are populated, X_Velocity, Y_velocity and Z_rotation. These define the path over ground of the robot's "neutral center point" which is the center of mass when standing "straight". This is not much different from a simple differential drive robot except motion can be in any direction, not just forward.

2) Cmd_Pose, This in turn has two parts (a) translation in (x,y,z) and (b) an orientation quaternion (like roll, pitch and yaw.) This tells the robot how to move the body relative to the neutral stance

By using the above we can have the robot crab-walking East while facing North and then send a cmd_pose to have it yaw left by 15 degrees while continuing to walk East. It would twist its legs up by the 15 degrees and keep walking sideways.

The controller algorithm is conceptually simple. It generates leg motions to execute the Cmd_Vel motion and it generates leg angles to execute the Cmd_Pose, then literally adds those two together

Next it checks balance limits by calculating the support polygon and the Zero Moment Point (ZMP, is like the center of gravity but also considers acceleration of the object in addition to the acceleration of gravity) and limits leg position so the ZMP pass through the support polygon.

Robot leg motion is simple too. The leg lifts and then moves through the air at (say) 4x the robot's commanded velocity over ground and then moves to ground contact and then moves at exactly -1 times the robot's commanded velocity. The exact path is done with splines or Bezier curves.

I think this is it. My guess is that the above algorithm is 100% generalized and can control

any action the robot can physically perform. And it does this without treating yaws, body-rolls and crabwalks as special cases. There is only one case, not dozens.

Next steps. (1) Post this for comment in a few places and let people find any faults. (2) re-write it for clarity and add details (3) write some prototype code and run it on a PC. (4) write the real code on the robot

I see no good reason this can't run on the small "micro-size" bots you all are buying from Amazon.

On 9/30/2022 8:07 PM, Mark Johnston wrote:

HBRC meeting followup: Jim asked about the python raspberry Pi background monitor I used on RoboDoggie and have used in the past.

The start of the file will explain it in fair detail and I use it on many project both to auto-start things (even full ROS) or just python like for RoboDoggie.

https://github.com/mjstn/MarkWorldCodeModules/tree/master/RaspberryPiUtilities Look at file sys_monitor.py

It is very flexible but with flexibility comes complications. Assorted ways it run are done with files in config/bin and also for it looking for specially named files to exist and if it finds them it will operate differently. It is all explained in the start of the file.

Enjoy this script, another freebe from Mark-Toys.com!

Mark

On Tue, Oct 4, 2022 at 10:07 AM Chris Albertson <albertson.chris@gmail.com> wrote:

Graceful walking means that the paths the feet take are created accurately and the vertical accelerations of the feet are controlled.

I was thinking about what a quadruped could do simultaneously while walking and came up with

    There is a forward velocity in the direction of the head (normal walking)
    There is a sideways velocity in the left/right direction (crab walking)
    The robot can rotate around the z-axis. (in the same way as a different drive wheeled robot)

All of the above can be combined and typically are if you watch a real dog paying with a dog toy.

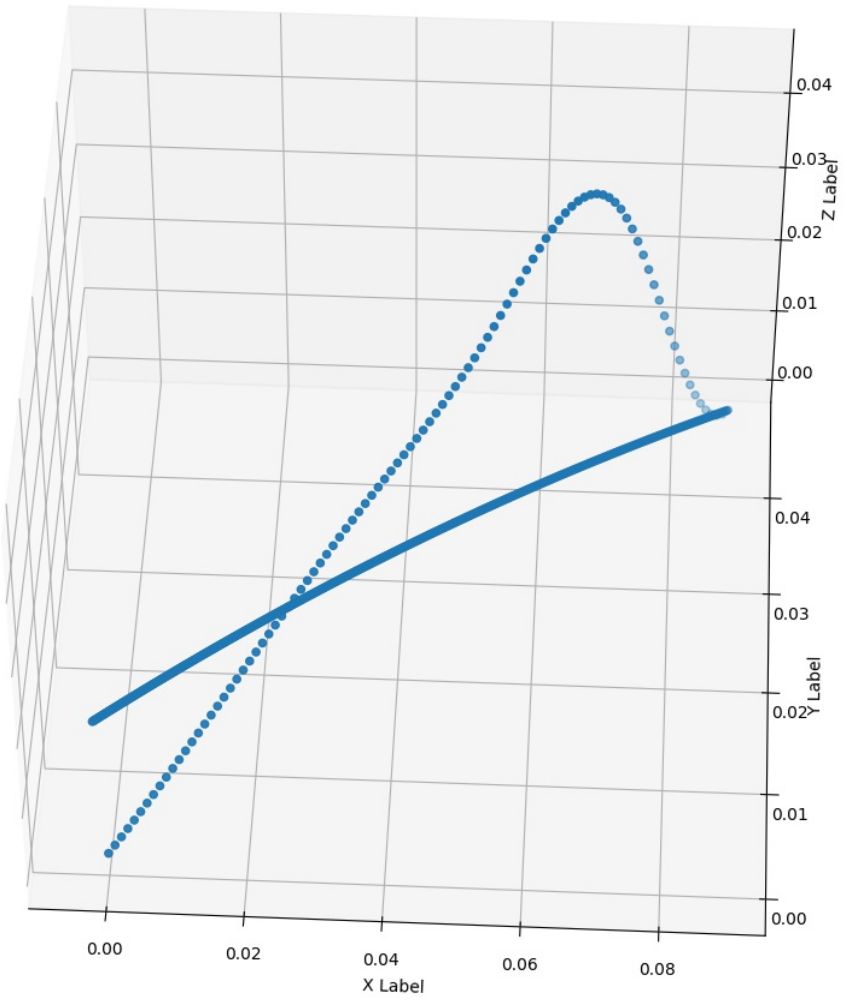
So how does a front, right foot move if the dog has

    0.5 meters per second forward speed
    0.2 meters per second sideways speed (effective 30 degree crab angle)
    1.0 radian per second rotation about the z-axis

Assume that you want the leg to be in the air 10% of the step cycle and on the ground 90% of the time with soft landing. Then you get the footpath shown in this diagram. The plot is taken from the robot body, so the view "moves with the robot". If you were looking at this it would look different because a foot that is in ground contact is stationary and it is the body that is moving. This plot is robot-centric. The trick it seems is to get this right and don't use shortcuts. shortcuts introduce error that cause the robot body to jump around 5mm or 10mm here and there several times per second. Shoot for sub-millimeter accuracy.

The dots are at equal time steps, so the part that is in the air is 10x faster than the arc on the floor where the dots overlap

Nothing to demo yet except for some plots. I will have existing code replaced with the new code used to make the plot in about a week.


0.04
0.03
0.02
0.01
0.00
Z Label
0.04
0.03
0.02
0.01
0.00
Y Label
0.00
0.02
0.04
0.06
0.08
X Label

On 10/4/2022 12:51 PM, Chris Albertson wrote:

For a walking robot base controller. That is the software that makes the robot move and walk the place to start is to define the inputs. Again this is just for locomotion. No "AI" or interaction with objects. What are the inputs. The goal is to be very general and define all the possible inputs.

In ROS' terminology we have two streams of commands.

(1) One controls the robots' path over the ground or to be more precise the path of the robot center point over the floor. There are three field used in the "twist" message Velocity in X, Velocity in Y and rotation about Z. Units are meters per secod and radians per second. (ROS topic cmd_vel)
(2) the second stream controls the orientation and it in turn has two parts. There are NOT velocities, just positions. (ROS topic cmd_pose)
    (a) rotation: roll, pitch and yaw of the body realive to some neutral stance (units are radians) and
    (b) shifts realive to some neutral stance. up/down, left/right and fore/back. These are "leans" relative the the feet, units are meters.

This means 9 numbers are input maybe 10 or 20 times per second. The controller must try and follow these commands as best it can, but the overriding constraints that can NEVER be violated are
1) motor acceleration and speed is a hard limit
2) joint angles have hard limits
3) legs shall not collide with each other.
4) the robot must maintain overall balance with as much reserve as possible.

I think this defines a controller for ANY walking robot no matter the number of legs, two, four or six legs are all the same. Only constraint #6 gets harder with fewer legs. Everything else is the same. The key point here is that constrains MUST be followed, inputs only receive best-effort attempts.

How to not "Stomp" I think I figured it out. Stomping is "high accelerations in the world coordinate system". Robot relative accelerations hardly mater. An example of the difference is that a foot when in ground contact has ZERO velocity and ZERO acceleration in the world system but can have any possible speed and acceleration in the robot's system. I think many controllers get this backward. What we care about is foot vs floor, not foot vs robot body. All systems that try to limit Servo motor speed and accelerations are counterproductive at best. Many times reaction force into the floor in increased if motor power is reduced. All these "smooth servo" libraries are addressing foot vs robot and ignore foot vs floor speeds.

Currently, writing new code with these ideas. I hope it will be usable by ANY walking robot.

On 10/4/2022 6:37 PM, 'Stephen Williams' via HomeBrew Robotics Club wrote:

I agree: Avoiding stomping is all about foot vs floor. This is the same problem as smooth braking: An experienced driver will often feather braking pressure off gradually so there is no point where there is a perceptible jerk to a stop. I didn't realize that I did this until I was teaching teenagers to drive and they asked me how I was doing that.

As a runner, who can run with different gaits, there are a couple ways to run silently, efficiently, and/or with low impact. We need motion control that can do that.

An interesting question is how the robot knows where the floor is relative to the robot & each foot, and how precise & dynamic that is. Should we have additional sensors to improve this and make it more reliable?

Chris, do you have a simulation of your robot running anywhere? It would be nice to collaborate on algorithms for this?

So, looks like a great topic to explore at the monthly meeting?

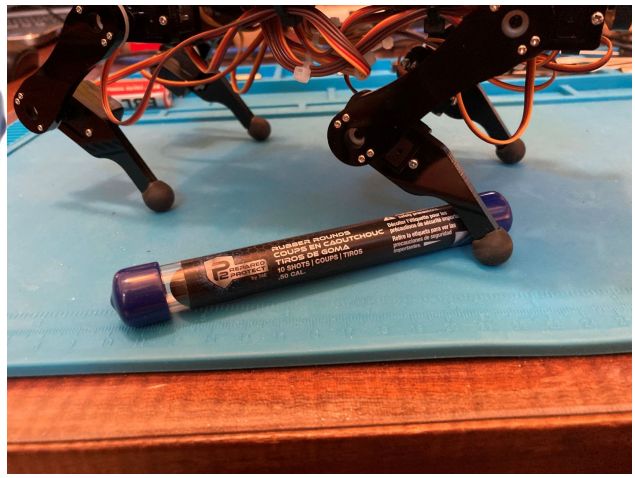
Thanks,
Stephen

On 10/6/2022 4:27 PM, James H Phelan wrote:
Re: [HBRobotics] robot dog - rubber feet

RoboBuds,

I got these from Amazon, drilled 1/2 way in with 1/4" bit then shoved them onto the feet.
Come 10/pkg
Seem to work good enough.

https://www.amazon.com/dp/B08J62CBW7?ref=ppx_yo2ov_dt_b_product_details&th=1&psc=1



On 10/9/2022 12:51 AM, Mark Johnston wrote:

James: Confirmation here on use of the paintball ammo for feet. They even look nicer than may printed ones.

They are a touch better in performance than my NinjaFlex printed boots and are certainly required over the hard plastic feet as shipped with the kit.

I think there is HUGE room for improvement in the walking code they are using. I'm going to put some effort in that as my RoboDoggie is a joke as far as how it walks. Perhaps just slower movements may help a lot as it moves so darn fast it cannot keep traction. Also it hardly pulls feet off the ground.

On 10/9/2022 1:39 AM, Chris Albertson wrote:

With quadrupeds that can't balance, there is an advantage to using a quick step. If you raise two feet and wait, the robot falls down. But if you pick the feet up and very quickly put them back down, the robot does not have time to fall. Walking slowly is harder than walking fast. There is also a best step height, that would be high enough and no higher.

The real solution is to track the center of balance and keep it over the support polygon. A walking root is not much different from a two wheel "balance bot".

One experiment I want to do is have the robot lift a pair of diagonal feet and balance on the other two for as long as possible. Then the robot should be able to translat along the line that connects the two feet that are on the ground. I doubt this can work perfectly are servo lack performance. Getting this to work "good as possible" is a first step to dynamically balanced walking

My latest success is what I call "fully generalized walk motion. It can walk sideways, forward and back, and can spin all at the same time while also applying roll, pitch and yaw and X, Y translations to the body. I think I will also be able to spin on an arbitrary vertical line, not just the Z-axis.

Next comes balance. I have the gyro/IMO connected and some tests work. But I don't fully now how fit balance into the motion. I think I will have to enforce a rule that the robot can never be commanded to move through an unbalanced pose. The balance would be the next to final check before sending position commands to the servos. (last check being joint angle limits) But I

don't like this. I think balance should be up higher in the command stack. A robot should think about balance BEFORE it moves. But maybe both is the right answer?

If you have an idea how the balance should work, let me know, or better make it work and post the code.

On 10/10/2022 11:15 AM, 'Stephen D. Williams' via HomeBrew Robotics Club wrote:

Sounds like great progress!

Most movement is about starting in balance (if stopped), transitioning into strategic imbalance then moving in a way that targets balance, or at least maintains a certain degree of imbalance. This is how we walk, run, etc. We tip forward, starting to fall, then continually catch ourselves by the next step. Walking & running is continuous falling. Ideally, motion planning should incorporate that dynamic planning.

Stephen

On 10/10/2022 7:36 PM, James H Phelan wrote:

RoboBuds,

An important part of animal locomotion (but clearly not critical as robots can stumble along without it) is PROPRIOCEPTION.

https://en.m.wikipedia.org/wiki/Proprioception

The sense that animals have of the position and forces on their parts.

This ties in with touch sensors on the feet and the balance mechanism of the brain (semicircular canals & the cerebellum - the brain's IMU).

A person with sensory neuropathy who can't feel their feet, have a real hard time walking and balancing.

Robots know where their limbs are 'supposed' to be, if the servos went where they're directed to.

But short of some feedback mechanism they're not certain. Nor what they've stepped onto - rock, mud or thin air.

On uneven ground it becomes more important to sense when you've touched the ground and how hard, especially relative to your other 3 (or whatever) legs.

Despite cameras, LIDAR, sonar, IR, etc., robots are largely walking around in the dark. When you do this at home, it's real important to sense where your feet are.

Some pressure sensors on the feet could beneficially inform the robot of its stance. How you integrate that into the rest of locomotion, I don't know.

I just hope for some kind of ROS node to do the magic.

RoboDoc

On 10/12/2022 1:35 PM, Chris Albertson wrote:

Yes, this is all true.   Clearly, a robot needs to know where all its legs and arms are and the forces currently on each of them.

The other thing it needs is a PREDICTIVE controller. It has to be able to plan ahead. At least a second or two ahead. If walking is falling then the robot needs to plan where to place it's foot and it has to know if that placement will be able to absorb the predicted momentum and if the motor is strong enough. Then failing that replan the footstep to be shorter, ut only if there is a secure place to land the foot.   It is like searching for the best chess move.

The other part is executing the plan. I think this can be done with a common PID-like controller.

Planning I think has to happen in layers.   Look at what a cow has to do if it wants to walk down to the creek to drink water. First it has to look and see a clear path and make a general plan.

Then as it walks it has to place each foot on a stable point of ground then it has to move the feet and use some kind of feedback to ensure the feet are going to the right place. This is "AI" but cow-level, not human-level AI.

Or robots can use human input or ROS' nav stack to plan paths and if walking on a flat hardwood floor can ignore the issue of footfall placements. So for now we can only work on plan execution. put the rest will need to be revisited or all you have is a remote-controlled toy that can walk on flat floors.

My plan is to make the robot work under remote control. But the remote will use ROS messages (cmd_vel and cmd_pose) so that at some point it will be easy to replace the hand controller with the ROS nav stack. The division is control will be that ROS nodes d path planning and the robot base does footfall planning and will ignore ROS messages that would cause the robot to fall over.

The OTHER thing that needs work is that a robot dog can and should do more then walk. There are other postures like sit and lay down and also the transitions for say walking to sitting.

On Thursday, October 13, 2022 at 1:30:29 AM UTC-7 Mark Johnston wrote:

Progress Report: Am using several more modes in Control.py.
I have up to 8 commands with my simple RF controller.
Here are the things I have running now (Plus I have tweeked bug not re-designed walking) so it is 'sort of ok' for forward and back now

Command	What it does
FORWARD	Walk forward and RoboDoggie is still rather spastic in walking but better than before
BACKWARD	Walk Backward. This is actually faster because of the way legs favor backward steps
NOD	I have the head very slowly on its own move up and down but a button makes it nod much faster
BARK	Ok, not really a 'BARK' but it has a buzzer it buzzes short on then little delay being off then on again
BOW	This uses a REALLY NICE Control.py member called .attitude that you set the roll, pitch, yaw. Very fine mode to learn about!
SIT	This also uses control.attitude() and varies pitch from 0 to 40 and back to 0 this is 'modestly impressive' action

For what it is worth, I offer up this script for folks to dork with as they like or use as examples of doing above basic things.
Trust me, this one script that resides in Code/Server is run like this:

See the attached for for a very nice 'head start' for your own doggie.

The ultrasonic is not working for my robot and must be debugged. I plan on having the detection of a close object show up on the led display as as a crude 'distance' guage once I get that to work.

I run this attached script like this:
ubuntu@robodoggie:~/robodog/Code/Server$ sudo python roboAppMj.py

So next I have to sort out sonar but it's ok as I am going to put on a ROS node from UbiquityRobotics and also I am next going to install camera node also from UbiquityRobotics (none of you should be surprised!)

So next 'milestone' will be run rviz against this 'puppy' and watch it in realtime with ROS (1)

On 10/17/2022 2:41 AM, Mark Johnston wrote:

Next step is voice commands and after that voice output (woff woff). So far I am installing both picovoice as well as vosk however both require minimum glibc of 2.27 to 2.28 and I only had 2.23 so I am doing the (MONSTER) install to get glibc up to 2.28 and as you may suspect is already a hour into the make but heck, it is after all glibc, the core to basically EVERYTHING! (May the force be with me!)

I believe Jim is happy with vosk on Orange so I will prefer that and try that first. picovoice seems to be in some way 'bound' to having an online dev account which I setup AND a 'key' so those two things make me sort of turned off from picovoice. I am hoping vosk is much more 'un-tethered'. I know at final runtime both can be disconnected from web which is my prime requirement.

On 10/17/2022 3:43 AM, <thomasfromla@gmail.com> wrote:

Robot Dog For Sale: 100% assembled and working.
It is a commercial product that I bought a few months back. It is Arduino-based. It has some sensors but is not autonomous. It is 15” tall and 22” long, close to the size of a pug.
I have the Arduino code. I was going to customize it. I wanted to add a head and tail, but I have too many projects so she must go.

Contact me for more info.

On 10/25/2022 3:35 AM, Mark Johnston wrote:
Update: Robodoggie is setting up assorted roadblocks that I am with some pain knocking down.
- Now have voice output through a USB plug-in stereo speaker for barks. Using HK-5002 with pulseaudio.
- Have standalone speach recognition. This has been a GREAT deal of pain due to my GCC lib is 2.23 and the following:
*   - Picovoice needs GLIBC_2.28*
*   - vosk needs GLIBC_2.27*
I will spare the intense detail and cussing at linux from everything from permissions to installs of GCC which is no 'simple' task NOR is it of low risk.

So I for speach had to use the greatly outdated and slow pocketsphinx and even that only worked for rev 0.1.15 (very old by todays standard)

Most all of the above is because I badly want ROS and I am most comfortable with the Ubiquity images which run GLIBC 2.23

So it has been 'make hell' and let me tell you, a GREAT many hours of 'make hell'. Thus back down to pocketsphinx

Next need to run the (very slow) pocketsphinx and have it figure out things then pass those on to another thread or ROS node.

I ran into one catch 22 here. To run this stuff I have to be in user mode but I cannot run the neopixel module unless root (due to some extreme timing requirements for neopixel it seems). So the fancy blinking and indications on RoboDoggie 7 neopixels is dead to gain speach.

On 10/25/2022 12:05 PM, Chris Albertson wrote:

Mark,

For things like glib issues and the like, just use Docker. It is like having a virtual computer with its own file system and OS but lighter weight.

On 11/6/2022 2:40 AM, Mark Johnston wrote:

RoboDoggie is back with a Howl! As there were just too many brick walls, I 'bit the bullet' and re-made my work on the UbiquityRobotics May 2022 Ubuntu 20.04 image that also has ROS Noetic.

Now installing things and moving forward is a breeze as I took one of my Pi4 boards from a different bot. We are ready to Rock!

Now RoboDoggie has speach output (dog speak of course!). Also had messed with pocketsphinx because I could not get vosk running but now on this Ubuntu 20.04 image vosk setup was a piece of cake. So in progress now is speach command recognition. I was encouraged by Jim D. and Orange to go for the gusto right to vosk, a 'big boy' speach system. HBRC is about encouraging others through sharing our work! (Right Camp???)

This new RoboDoggie platform has ROS working so the camera node for raspicam is also online.

Next I have my sys_monitor rc.local launch monitor I use on most all my Pi bots for pushbutton clean reset or shutdown as well as app startup.

FINALLY I have also made some fun 'dog like' anamatronic actions on random intervals for head and body shifting around to make RoboDoggie look more natural (although most dogs are not black acrylic ... Lol! Also RoboDoggie responds with different bark patterns based on what it sees with the sonar unit.

So next is vosk speach commands which is this weekends goal.

I am awaiting the Arducam ToF camera and if it gets here in time THAT will give RoboDoggie GREAT frontal perception. It could show up anytime.

Woof Woof! Mark

On 11/6/2022 9:27 AM, James H Phelan wrote:

Mark,

Do you have a github or web site w/ a tutorial?

Here in Houston Roberto and I would like to follow in your footsteps but we are "only an egg"!

BTW speach >> speech (Ya gotta love English! It often makes no sense)

RoboDoc

On 11/6/2022 6:15 PM, Mark Johnston wrote:
In working on RoboDoggie I have my 1st cut at a python3 script that uses vosk to output to a file so that my main script can read it as recognized speach input for commands.

Kudos to JimD for showing Orange and discussing vosk.

As RoboDoggie moves along here the plan is that this will be ported into a python ROS node and the speach output can then be on a ROS topic. A nodes such as that likely already exists but for now RoboDoggie is not on ROS although ROS is active on this Pi4.

So for your amusement get from my github:
https://github.com/mjstn/MarkWorldCodeModules/blob/master/speachRecognizer/speachRecognizer.py

Because it is 'free', I will add the obvious: 'Worth every penny! and Ya gets what Ya gets.

Havin some good fun with this mostly animatronic project

Mark

On 11/6/2022 6:19 PM, Mark Johnston wrote:

RoboDoggie will be presented at this month's HBRC meeting. It is at that time I will have a slide deck.
My goal will be give basic info and links as I show what RoboDoggie has for his tricks.
I am not going to plan on in depth explanations as IMHO that is part of the fun, learning how to do this stuff on one's own.

My github does have sys_monitor.py as well as a vosk speach recognizer component both in RoboDoggie so see these under my sort of 'Misc' catagory of stuff here:
https://github.com/mjstn/MarkWorldCodeModules

Film at 11, (This means at HBRC meeting by the way ... LOL)
Mark

On Thu, Nov 10, 2022, 4:12 PM Michael Wimble <mwimble@gmail.com> wrote:

Once your Arduino-like system goes beyond, say, monitoring a sonar sensor and driving a motor, you'll probably find that you should be organizing your software into individual modules, one module for each kind of thing you want to sense or control. And, because a moving robot needs to have frequent sensor input data so that the robot doesn't move very far before it acts on the data, it helps if you can measure the performance of each module. The performance statistics can also guide you as to where to next expend your coding effort to improve the system performance.

In my Teensy Monitor code, I wrote a module, TModule, the wraps the usual setup and loop calls to provide just such performance monitoring. Here is a short article about and the code for Tmodule.

https://wimblerobotics.wimble.org/wp/2022/11/10/replacing-the-arduino-setup-loop-mechanism/

On 11/10/2022 4:40 PM, Chris Albertson wrote:

I just learned how to profile a multi-tasking Python app and find what functions are using the most CPU time. The answer is "yappi" and it's simple to use.

So what did yappi find? I let my robo-dog walk for about 50 seconds and found:
1) All the matrix algebra used to compute where the feet should be and to combine the body roll, pitch and yaw is trivial and does not even so up in the stats. Numpy is so fast that to a first approximation it takes zero time. Lesson here. ALWAYS let numpy do operations on arrays and NEVER loop over your data using Python for loops.
2) The Inverse Kinematics took a total of about 3.5 seconds. This is all a bunch of trig functions using standard Python.
3) Sending the above results to the PCA8596 servo controller chip over I2C using the Adafruit library took about 10 seconds of CPU time.

In other words, the big majority of the time (20% of one CPU core) was just pushing bits over a wire with a horribly inefficient Servo Library.

I bet you all can guess my next project. Using Numpy to do all the calculations for a new Servo Library. What it must do is accept a desired angle, apply a calibration that corrects for servo installation error, then apply limit checks and then convert to a PWM value and finally find the counter values to load into the PCA9685 control registers. Then I write 16 bytes to the device.

To use yappi your Python code, three lines of code will do it.
to install, "pip install yappi"
to use it look at readme https://github.com/sumerc/yappi

On 11/11/2022 12:55 AM, Mark Johnston wrote:

yappi sounds like a great tool, thanks.

What I feel would be of interest is to break apart the prep time of the PWM values separated from the (perhaps expensive) I2C bus handling time.
Is your I2C done in hardware with interrupts back to the host or is it forcing a core to hang out and babysit the I2C traffic (rather expensive even at 400kbit)

Lastly I would assume you are able to do the I2C to your PCA9685 at 400khz? is your SDA line over-loaded with unexpected capacitance thus slowing down the bus due to rise times? I always look at the I2C bus with a scope just to be sure the signals look healthy and fast as is safe.

Thanks again on the tip on yappi. I will note that it is 'interesting' we are looking at robot dogs, mine rather small, and this tool is called 'yappi' ... makes ya think ... or not LOL

On 11/11/2022 8:41 AM, Sergei Grichine wrote:
This seems to be a great idea, and something everybody should be using to track down timing bugs.

I am wondering though, how it will work with scheduling libraries, like this one:
https://www.arduino.cc/reference/en/libraries/taskscheduler/

Basically, having a version of monitor to complement task scheduler would be great.

Also, not everybody is working with ROS, having a readable on-demand report on serial would be helpful.

Arduino IDE encourages use of tabs, and discourages h-files. I'm not sure how that would work with your code.

I usually dedicate a couple pins to debugging, and raise/reset those when entering and exiting code sections of interest. Then I can see the timing on a logic analyzer. Your code would add very valuable statistics to it, thanks for sharing!

On 11/11/2022 2:16 PM, Chris Albertson wrote:

Your micro-size Free-Nove robo-dog uses the same PCA9685 chip for servo control as I do. In the end the Linux I2C device driver is used. I used the Adafruit library because it was easy to use. I looked at the code the micro-dog uses and it is MUCH more simple and compact and you can read it all in a few minutes. I'm going to "borrow" it.

Even if I'm running the I2C bus at 100 kHz, we only write 32 bytes to the chip, and I do this 40 times per second, so 0.0128 seconds are used each second to write on the bus. This is fast enough even if the bus is slow. (And I think I set it to run faster) There is "something" in the Adafruit code that is non-optimal and I'm not about to look inside

My plan is write a new servo handler that accepts a 12 x 1 numpy array of angles in radians scales them to pulse widths and then uses the code borrowed from the micro-dog.

What yappi shows is that you REALLY need to code in Python as below and avoid for-loops. There is an order of magnitude difference in execution time. in some cases Numpy can use the CPU's vectorized instructions (NEON for Pi4) or even in some cases the GPU which pushes the loop into hardware. Yapi says that the matrix multiplies I'm doing use trivial amounts of time on a Raspberry Pi4. (I don't know if the example below uses NEON or not, but it might. You can see that I could do all the scaling with vectorized NEON instructions)

```
>>> x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
>>> y = 2*x + 3
>>> print(y)
[ 5  7  9 11 13 15 17 19 21 23 25 27]
```

About profiling..

the "standard Python profiler is "cProfile" but it only works on the main task. My robot's main task is boring and only starts other tasks, then waits for them to finish. If you run yappi in any task, it times all the other tasks too. And unlike cProfile, yappi has reasonable defaults from printing results

Mechanical....

The bigger problem is gear wear, Testing, even using my test stand as much as I can is killing the metal gears in the servos. It can cost $40 to $80 per month just to replace the servos. Torque is acceptable, but speed and longevity are not.

A group called "ODRI" has come up with a very elegant design for a quadruped that uses BDLC motors. Their cost is still over my budget. But work at the SimpleFOC project has found a way to further reduce costs. It may be that TI's DV8316 chip is the key to making a controller/driver for under $20 per joint. Then it is the cost of a motor and a handful of ball bearing units. The rest is printed.

Here is what I consider the best low-cost quadruped today.
Video https://youtu.be/6kY0sRYaXyI

main website: https://open-dynamic-robot-initiative.github.io/

Note the power cable. Performance would be lower if the weight of the battery had to be inside the body., but still, it would be impressive. Also, the design is very modular, the same parts work for a biped.

I had planned on making a second version of a servo-powered dog-bot with reduced moving mass but I just might jump to BLDC. But first comes software....

On 11/12/2022 11:57 AM, Chris Albertson wrote:

Wth Arduino, I can run any numer of tasks very simply. Below is the pseudocode I typically use. TaskA might be toggling a LED and task B reading serial input. "C" might be a PID loop for a motor.

The result is a hard-realtime scheduler where each task runs on a very fixed period.

Also I might have an interrupt handler but that is serportate

```
loop{
  if      system_time > taskA_time{ taskA(); taskA_time = system_time+taskA_period}
  else if system_time > taskB_time{ taskB(); taskB_time = system_time+taskB_period}
  else if system_time > taskC_time{ taskC(); taskC_time = system_time+taskC_period}
  else {idle_count++}
  loop_count++
  sleep(0.001)
}
```

Performance is the ratio idle_count / loop_count. When it gets close to zero, I need a faster CPU or smarter code.

So multi-tasing in real-time only adds the code you see above. You don't need more.

On Sat, Nov 12, 2022 at 5:11 PM James H Phelan <jhphelan@hal-pc.org> wrote:

Chris, et al.,

The ODRI Solo 12 performance is indeed impressive!

The best (only) estimate of "low-cost" I could find is from

https://odri.discourse.group/t/rough-cost-to-build-the-solo-12-robot/221

"the price of the robot is difficult to estimate since it will depend on the suppliers and machine shops that you use, the country you are in and the part quantities that you are ordering.

[a LOT of it is 3D printed and custom machined]

We have estimated the cost for our Solo12 robot at around 6300€ + tax. [+/-=US$6500]

That is not including the imu which is the most expensive part on the robot. [I don't see the specs for the IMU so don't have a price for that.]

That is also not including the tools that you might need to buy if you don’t have them in your lab."

RoboDoc

On 11/13/2022 12:06 AM, Chris Albertson wrote:

Yes, the current cost to build "Solo" is not cheap. A good source is the ODRI github repositoy. They have the bill of materials and links to vendors. One could spend $6K on this project.

What I'm looking at is RADICALLY reducing the price. Here is just one example: The encoders they place in each joint use a $55 optical sensor and a $25 encoder disk that is attached to the motor shaft This gives about 2K lines per revolution. I think this might be replaced with a new design that costs only $3.00 I would print out a barcode on photo paper and glue it around the outside of the spinning motor rotor housing and then use a pair of $1.50 reflective sensors

to scan the bars as they fly by. This saves $684. Next, they use motors that cost $85 each. I want to look at lower-performance motors. There are three motors in each leg and not all of them need to be the same size, Shulder joints need much less power and speed than do knee joints. It simplifies things, but is wasteful to use the same motor for each joint.

I can buy smaller motors for under $20. There is a whole range of price points available.

The next place to address cost is the driver electronics. They spent over $100 per motor on drivers. I think we can reduce the cost to $20, but with lower performance. Some experiments are needed. A really cheap drive unit can be made from a Raspberry Pi Pico microcontroller and TI DRV8316 chip (or DVR8332) Not much more is needed. But is 8 amps enough? I need to build a prototype knee joint and test it.

Don't worry about the IMU. Those are cheap as dirt and I have a half dozen of them. The MPU6050 is good enough and cost $3.50 on Amazon or $0.99 from China. You can use two of them and get better data. With the current chip shortage, the 6050 is the only one you can actually buy.

I am not worried at all about 3D printing. I pay $20 per Kg for plastic and have already printed a couple of dog-bots about this size. No issues. There is some minor machining. But I have some small machine tools, a lathe, and a small CNC mill.

It all needs to be re-engineered with a very careful eye to cost. I would not do it the same way exactly. This is NOT the kind of robot where you just start building with no plan to see what happens. LOTS of 3D CAD work is needed and lots of prototyping is needed and the entire design needs to run in simulation before it is built. The design needs to account for every gram of mass and every screw and nut.

There is another person doing this too. He was able to reduce the cost by a lot but I think I can go even lower. In fact, he just send me an email, I need to read it. Perhaps something above is already obsolete.

THEN COMES The HARD PART, software. I think I can use 90% of the software I'm developing for my current robot-dog on the Solo-like robot. I think these sevo-powered robots are good-enough software development platforms. I could use some existing software (cmvp/champ is good) but my goal is to make software that is (1) simple and easy to understand and (2) generalized enough to do any motion a real dog could do and do it in a way that looks like animal motion, not robot motion.

This last goal, animal-like motion, requires higher-performance motors than servos.

I think I could get some of the prototyping done in 2023. My goal would be a $50 knee or hip joint by the end of 2023. My current dog-bot will have presentable software by the end of 2022. I've been refactoring and doing unit tests and comments these last weeks. It is still a rat's nest but look at it here https://github.com/chrisalbertson/SpotMicroModular You run "main.py" and it mostly works

The Solo project is not worth doing if it costs $6K. But I bet it can get it down to $1,200.

...

Just read his email. The record low cost for a usable rotary encoder is now under $1 and has been well-tested and documented. So it was $80 for the ODRI people and now $0.80. It just takes a community of people thinking hard about cost. This is a 100x cost reduction.

The way the 80-cent method works is to place hall-effect sensors around the motor and measure the periodic rotating magnetic field and track it with analog pins on the microcontroller. Each motor needs to be measured and calibrated but only once. Then the resolution seems to be limited by the bits in the ADC and the sampling rate.

I claim that it actually cost mor then 80 cents because you now need a microcontroller with hardware floating point.

On 11/18/2022 2:41 PM, Alan Downing wrote:

UC Berkeley has recently enabled the Unitree robot to go up regular stairs. Looks like the low-stair limitation was more a problem of the COTS software. This quadruped looks pretty impressive to me.

Alan Downing

On 11/18/2022 6:56 PM, James H Phelan wrote:

They say they use the camera output to directly control the gait with a simulator trained machine learning function without using any kind of mapping or path planning.

Basically just visual memory to motor control: "I see stairs, this is how you climb. I see rocks, this is how you walk. I've not seen this before, just fake it."

A more detailed explanation was provided here: A Low-Cost Robot Ready for Any Obstacle

I wonder if something like this would work for RoboDoggie? Use simulated reinforcement learning to develop a more graceful and adaptable gait.

I wonder what their starting ML application was? ***I'll ask them****.*

I pondered whether it would help our rolling robots, but without any gait or climbing considerations, didn't see how it would.

RoboDoc

On 11/18/2022 7:44 PM, James H Phelan wrote:

To: dpathak@andrew.cmu.edu

Subject: A Low-Cost Robot Ready for Any Obstacle

Dear Dr. Pathak,

A fellow member of the Homebrew Robotics Club https://www.hbrobotics.org/

pointed out the Youtube demo of "CMU, Berkeley Researchers Design System Creating Robust Legged Robot"

and the accompanying article A Low-Cost Robot Ready for Any Obstacle

Several of us have assembled the Freenove Robot Dog Kit for Raspberry Pi

and a few of our robotics experts are working to expand its capabilities.

The discussion extends to quadruped robots in general and gait control.

I took the NVIDIA "Getting Started with AI on Jetson Nano" course as adapted by Houston Community College (the only Community College in the US to offer an AI curriculum)

so I am a little familiar with Machine Learning.

Since "I am only an egg" when it comes to robotics and Machine Learning, I have a few questions:

I assume you used reinforcement learning to train the robots -

1) what ML application did you use as the base to train your simulated robots?

2) how did you set up the simulation and the "reward"?

3) how do you combine the experience of 4000 cloned robots into one learning experience?

Would you, or one of your team, be interested in doing a 1 hour presentation / discussion at one of the HBRC meetings, virtually or in person?

With interest,

Jim "RoboDoc" Phelan

On 11/18/2022 8:53 PM, andy wrote:

The paper talks about training the robot outdoors with a 2070 rtx laptop.
Definitely would like to hear a talk about their RL training experience.

https://arxiv.org/abs/2208.07860

***[Which points to:** https://github.com/ikostrikov/walk_in_the_park]*

On 11/18/2022 9:13 PM, Chris Albertson wrote:

Yes! There is absolutely no reason why this can not work on a lower-cost robot. But you have to accept lower performance on the lower-performance platform. Our robots are good-enough for software development, but the mechanics are not powerful enough for fast real-time execution. But we can work on the same problems.

For example, while that Freenove "nano-size" robot physically could never walk up a flight of real stairs. It could walk up a series of 1-inch tall stairs. The problem is the same. Go to Home Depot and buy some 12" floor tiles and build a stack with stairsteps. My larger robot could walk up 3" stairs. We can even paint the tiles to make recognition easier. (green risers and orange treads would speed up training time.)

The other problem to overcome is mounting a reasonable sensor on the robot. This is likely impossible on the Freenove 'bot. But you could place a depth camera on a tripod in a place where the camera can see the stairs. This is good enough for algorithm development. My larger 'bot can fit an Oak-D lite camera inside its "head". But camera mounting is just cosmetics, only the coordinate transform is different.

The other thing you need is computing power. That is going to have to be via Wi-Fi. You would never get an Nvidia 2060 inside a robot.

My plan is to continue working on my Servo powered robot's software. The robot itself is basically finished and can basically walk. But I'm looking at a $1K version of Solo. And I suspect it will have the size, speed and power to walk up real stairs.

The bottleneck is software. But we have good-enough software development platforms.

Finally one more suggestion to someone who does not yet have their own robo-dog. "Use an ESP32." They cost $10 and can replace a Raspberry Pi. OK not quite replace. But the ESP32 can run micro-ROS and the rest of the software can run on some cheap notebook PC. Only the real-time control loop need run physically on the robot. Realistically a full-size servo-powed 'bot would cost $300 if an ESP32 were used.

Someday what we lean using quadrupeds can be used to build a biped. There is not much difference, many MUCH faster control loops and more DOF in the legs.

On 11/18/2022 9:25 PM, Chris Albertson wrote:

If you want to jump into this right away. Look at chvmp/champ at github. It can run on a PC in simulation or in any of the Servo-powered 'bots or on the Cheetah/Unitree 'bot. LOTS of active work on this. In fact, if you are serious about Quadrupeds and just want to make one work. Use chvmp/champ and be done with it. I'm nuts and just want to re-invent everything. Those wanting to be up and running, use the "standard" software. (Yes, people are running this on current version of ROS2 now, seems the README needs updating)
https://github.com/chvmp/champ

On 11/19/2022 9:14 AM, James H Phelan wrote:

I emailed dpathak@andrew.cmu.edu the guy mentioned in the article and asked if someone from his team would be interested in a presentation or discussion with us.

We'll see what kind of response I get.

RoboDoc

On 11/19/2022 11:06 AM, Chris Albertson wrote:
in response to: ***On Mon, Sep 26, 2022 at 5:35 PM Mark Johnston <mjstn2011@gmail.com>***

I'm working on a solution to this. I've copied the Freenove PCA9685 driver and added a layer to it.

Then there s a GUI app where the user sends commands to the servo and measures legs with a protractor and enters the values. He can make as many measurements as he likes, dozens if he has lots of free time to burn. The software then does a least-squares fit of a pulse width to angle function and writes the parameters to a place the layer above the pca9985 driver can see.

There is some graphics that show how good the fit is and the ability to edit measurements that might be outliers. I think this part is important, so the user (that would be me) can find and remove inaccurate measurements.

I'm thinking I will add a feature for adjusting the PWM frequency. The Freenove software sets this at 50Hz, but I've been buying better quality servos that can run at up to 300 Hz. This gives the ability to send new position commands at a faster rate. I find I need to update at about 30 Hz for the smoothest motion.

Status: I've got the GUI party done and a start on the other Python files.

On 11/19/2022 11:26 AM, Chris Albertson wrote:

If you look at the building code. There is a rather complex formula for stairs. By law, we are only allowed to build stairs that follow the formula. The formula was designed based on human leg lengths and how our knees and hips move so it should be no surprise that we can walk up and down stairs. They are designed for us.

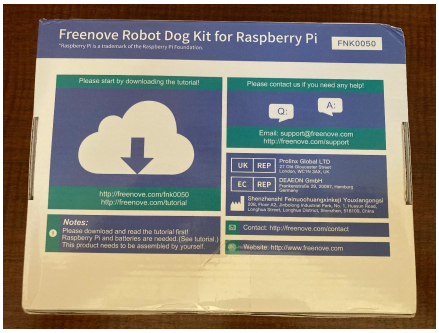
The small robot has very different leg geometry.

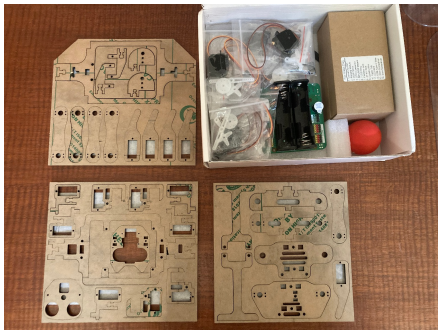
Boston Dynamics solved the stair problem by sizing the robot for stairs. They knew the building code and hence the range of allowed stair dimensions, and designed a robot for that. BD Spot does stairs very well. I'd say a LOT BETTER than the Unitree robot. BD Spot uses a depth sensor and a software optimizer for footfall placement. Notice that spot's feet hit the stair treads dead-on center each time. I'd say it has "perfect" execution.
This is the best video I could find https://youtu.be/ dnBuwZdmq8

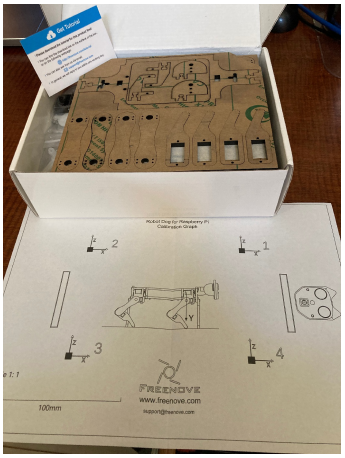
My smaller real-dog has learned a new way to climb stairs. He is very small. He stands on his rear legs and places his front paws on the step and wines and cries until someone picks him up and carries him up the stairs. This actually could work for a robot and would not be as hard to implement.

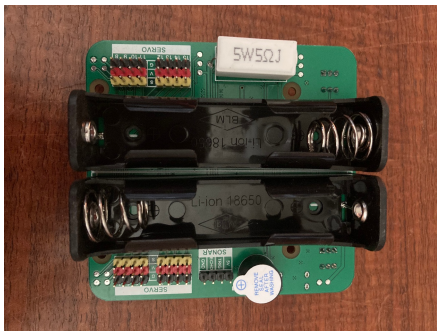
correspondence end
***

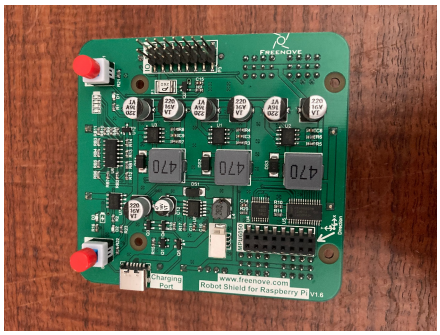
2022.09.13
Robot Dog Kit arrived today!











https://freenove.com/tutorial.html
allows you to choose download eg 0050

https://freenove.com/fnk0050/
automatically downloads .zip file

On unzipping the fnk0050.zip file, I get an error on two of the ---.dll files “file name too long”. Email sent to Freenove.

On 9/13/2022 9:40 PM, support@freenove.com wrote:
Dear customer,
Please try this one:

https://drive.google.com/file/d/1rRrmtcX6ePmtjQjCpr01ftllK7U5382x/view?usp=sharing

We will wait your feedback.

Denzel
Freenove Support Team
---
www.freenove.com

Opened the unzipped file Tutorial.pdf:

Tutorial.pdf

First, read the document About_Battery.pdf in the unzipped folder.

About_Battery.pdf
1, Robot Dog Kit for Raspberry Pi
4WD Smart Car Kit for Raspberry Pi
Both flat and button top 3.7V 18650 batteries with discharge current >10A.

https://www.amazon.com/EBL-Rechargeable-Lithium-Battery-Batteries/dp/B088ZQ8SDK/ref=sr_1_39?crid=155BTYV2H7MC3&keywords=EBL+18650+battery&qid=1663117890&sprefix=ebl+18650+battery%2Caps%2C91&sr=8-39

Scroll through the Tutorial.pdf, parts list, Raspberry Pi options GPIO interface, until you get to p20 “Install a System”.

Install a System

https://www.raspberrypi.com/software/operating-systems/

Recommended system for most users:
Raspberry Pi OS with desktop and recommended software
Release date: April 4th 2022
System: 32-bit
Kernel version: 5.15
Debian version: 11 (bullseye)
Size: 2,277MB
Show SHA256 file integrity hash

2022.09.14
Update Raspberry Pi Imager to v1.7.3
Custom
2022-09-06-raspios-bullseye-armhf-full.img.xz
Hostname: raspberrypi
Username: **pi**
Password: **FIDO**
Local settings
WiFi
SSH
telemetry (?)

BOOT Raspberry Pi

Windows PowerShell:
PS C:\Users\jhphe> **ping raspberrypi.local**

```
Pinging raspberrypi.local [2601:2c4:57f:df70::bd1a] with 32 bytes of data:
Request timed out.
Reply from 2601:2c4:57f:df70::bd1a: time=1ms
Reply from 2601:2c4:57f:df70::bd1a: time<1ms
Reply from 2601:2c4:57f:df70::bd1a: time=1ms

Ping statistics for 2601:2c4:57f:df70::bd1a:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

PS C:\Users\jhphe> **ssh pi@2601:2c4:57f:df70::bd1a**

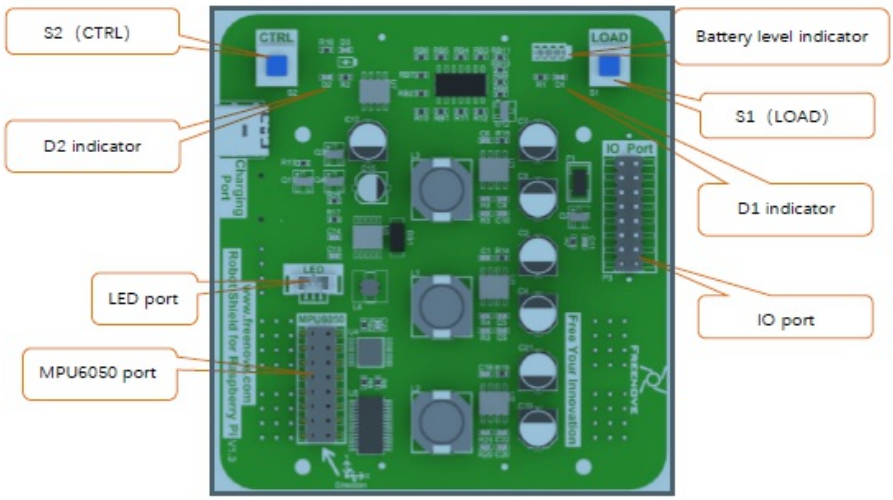
```
The authenticity of host '2601:2c4:57f:df70::bd1a (2601:2c4:57f:df70::bd1a)' can't be
established.
ECDSA key fingerprint is SHA256:rX0Bx2Yi8y8dWqqt31nW8jIgy0Ukz2JcAAgHqoGaa5o.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '2601:2c4:57f:df70::bd1a' (ECDSA) to the list of known hosts.
pi@2601:2c4:57f:df70::bd1a's password: FIDO
Linux raspberrypi 5.15.61-v7+ #1579 SMP Fri Aug 26 11:10:59 BST 2022 armv7l
```

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.79  netmask 255.255.255.0  broadcast 10.0.0.255
..
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
...
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.30  netmask 255.255.255.0  broadcast 10.0.0.255
...
```

```
pi@raspberrypi:~ $ sudo raspi-config
Interface/Enable: VNC
Performance/GPU Memory: 256 (my option)
Advanced/Expand filesystem (my option)
REBOOT
```

Install VNC viewer on laptop:
https://www.realvnc.com/en/connect/download/viewer/
"Use VNC Viewer without signing in"
Connect to **10.0.0.79**
Success

p36 Robot Shield for Raspberry Pi



p38 Chapter 1 Install Python Libraries (Required)

Step 1 Obtain the Code and Set python3 as default
pi@raspberrypi:~ $ **cd ~**
pi@raspberrypi:~ $ **git clone https://github.com/Freenove/Freenove_Robot_Dog_Kit_for_Raspberry_Pi**
Success

pi@raspberrypi:~ $ **python --version**
Python 3.9.2

Step 2 Configuration
Enable i2c
Raspberry Pi Configuration
Interfaces / enable I2C
[SSH & VNC already ON]

Set I2C Baud Rate
pi@raspberrypi:~ $ **sudo nano /boot/config.txt**
The default I2C Baud Rate is 100000. Now we change it to 400000, because this can speed up the response
speed of the servos to make robot dog walk faster. If the baud rate is 100,000, the robot walks slowly.
Scrolling the middle of the mouse to find dtparam=i2c_arm=on, and add "i2c_arm_baudrate=400000"

dtparam=i2c_arm=on**, i2c_arm_baudrate=400000**
^O, **^X**

Patch for OS 2021-10-30 and Later.
pi@raspberrypi:~ $ **cat /etc/rpi-issue**
Raspberry Pi reference 2022-09-06
If the result is later than 2021-10-30, execute following steps.
pi@raspberrypi:~ $ **cd ~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Patch**
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Patch $ **sudo sh ./patch_for_bullseye.sh**
cp: cannot stat '/opt/vc/lib/libmmal.so': No such file or directory
cp: cannot create regular file '/opt/vc/lib/libmmal.so': No such file or directory
patched complete!
If there is "patched complete", just ignore "opt/vc/lib/libmmal.so doesn' exist"and move on.

pi@raspberrypi:~ $ **sudo nano /boot/config.txt**
1 Add # before camera_auto_detect=1 „³ **#** camera_auto_detect=1
2 Add f in dtoverlay=vc4-kms-v3d „³ dtoverlay=vc4-**f**kms-v3d
3 Add following in the end.
start_x=1
gpu_mem=128
hdmi_force_hotplug=1
hdmi_ignore_edid=0xa5000080
hdmi_group=2
hdmi_mode=82
Finally press Ctrl+O, Enter, then Ctrl+X.

Additional supplement
Raspbery Pi, other than 4B and 400, needs to disable the audio module, otherwise the LED will not work
properly.
1. Create a new snd-blacklist.conf and open it for editing
pi@raspberrypi:~ $ **sudo nano /etc/modprobe.d/snd-blacklist.conf**

blacklist snd_bcm2835

2. We also need to edit config file.
pi@raspberrypi:~ $ **sudo nano /boot/config.txt**
dtparam=audio=on

Step 3 Run the Installation Program
pi@raspberrypi:~ $ **cd ~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code**
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code $ **sudo python setup.py**
...
Now the installation is successful.
Please restart raspberry pi

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code $ **sudo reboot**

Chapter 2 Assemble Robot p48
2022.09.15

Step 1 Install Disk Servo Arms
Take out 12 disk servo arms from the servo packages.
There's supposed to be 12. I have 13. Spare?
Only 3 screws provided per servo?
DON'T use the screws in the servo bag! Too small and will break!
*Use the ones in the parts bag "M1.2*7 self-tapping Screw x 60"*
Only 3/4 screws fit. One odd hole in each disk is offset.
Done.

Step 2 Install Body Bracket

Step 3 Install Shield p50

Step 4 Install MPU6050 p51

Step 5 Install LED module

Install the top bracket.

Step 6 Install Raspberry Pi p53

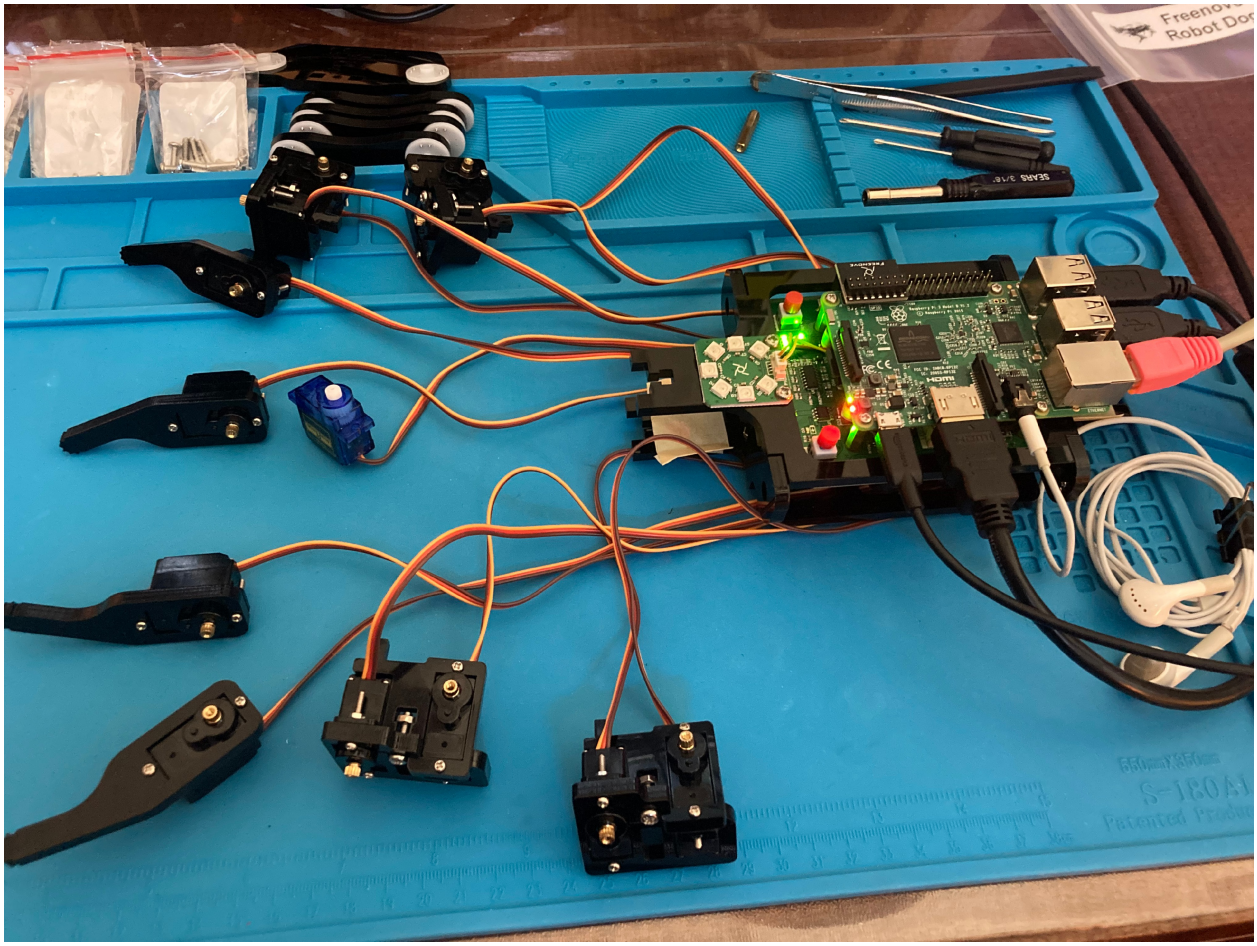
Step 7 Install Connector p54

Step 8 Install Servo to Acrylic Board p56

Step 9 Run Servo Program (Necessary) p59

pi@raspberrypi:~ $ **cd**
~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Ser
ver $ **sudo python Servo.py**
Now servos will rotate to 90°.
If they have already been at 90°, nothing will be observed.
Please keep the program running when installing the servos.
After that, you can press ctrl-C to end the program.



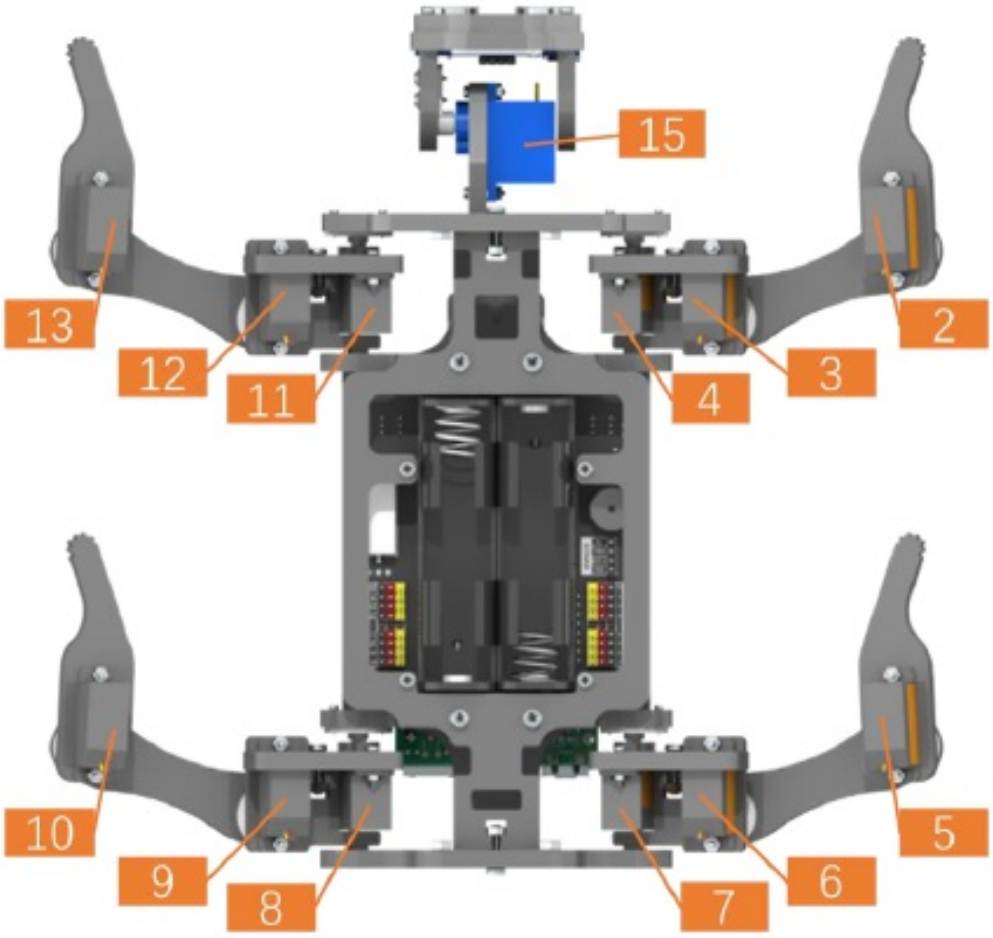
Step 10 Assemble Legs to Body p62.

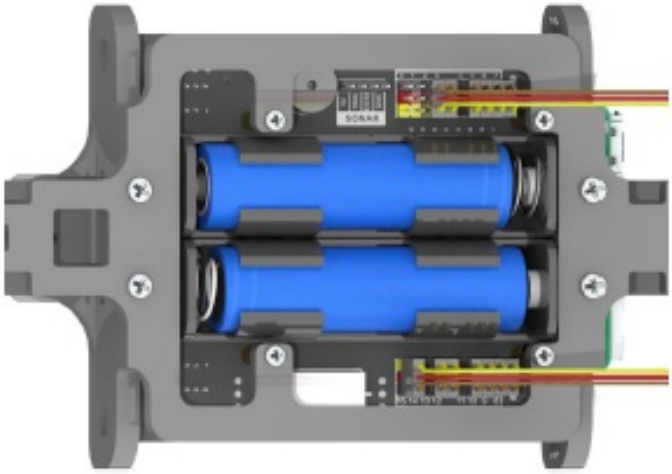
Step 11 Assemble Head p66.

Step 12 Assemble Head to Body p67.
We will complete wiring in next section. Shut down the Raspberry Pi first. Do NOT turn on Raspberry Pi
until chapter 3.

Step 13 Wiring p68.

Connect servos according to the numbers. Note servo ports 0, 1, 14 are not connected to servo. They are spare.





Wiring of Ultrasonic module (Note: Do NOT connect wrongly. If you connect 5V to GND, it may
damage the ultrasonic module)

Wiring of camera
Pay attention to the blue side of camera cable.
(Note: plugging and unplugging the cable requires the Raspberry Pi to be powered off, otherwise
the camera module may be burned.)

Step 14 Install Calibration Support p71.

Step 15 Verify Assembly p72.
Turn on two switches and run following two commands again.
pi@raspberrypi:~ $ **cd ~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server**
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python Servo.py**

Chapter 3 Module test (Required) p75.
pi@raspberrypi:~ $ **i2cdetect -y 1**
As shown in the figure below, the addresses 0x40, 0X48, and 0X68 corresponds to the PCA9685 chip, ADS7830 chip, and MPU6050 module, respectively.

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python test.py Servo**

should say:

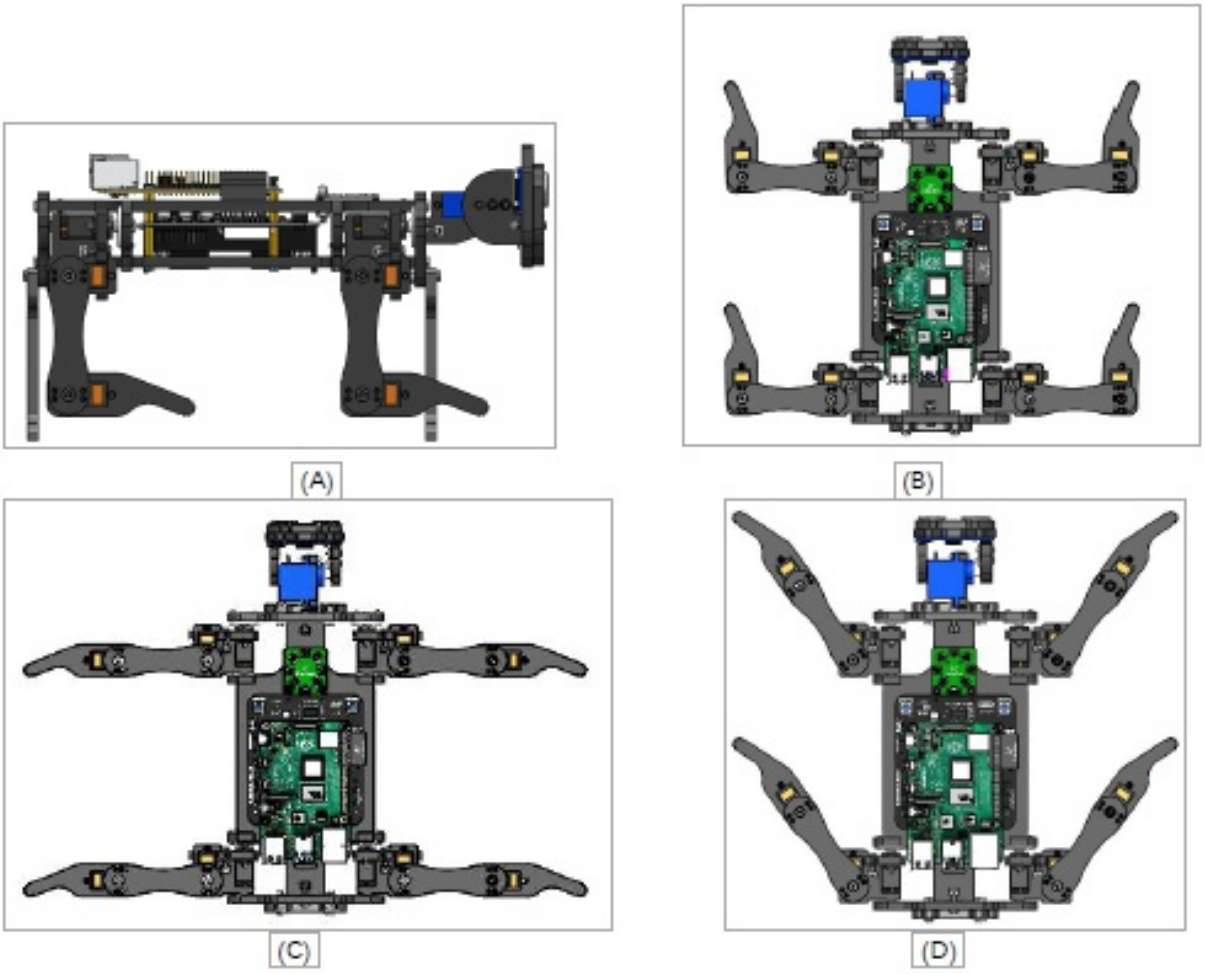
Program is starting...

Result:

After assembly in the previous chapter, the robot should look as shown in picture A.

After servo test program is executed, the robot's posture will change to A, B, C, D gradually, which indicates the servo channel works normally.

If the situation is not correct, check the servo wiring.



Unfortunately, FIDO doesn't follow the above routine.
Particularly the left hind leg hip #7 abducts very slowly.
Right front elbow #12 fails to extend.
Will need to rig up a servo tester to check each one out.

Tried getting all neat with wire managment, but then needed to undo in order to reconfirm the servo wire placement.

Will need to adapt Servo.py to rest each servo in succession. Meanwhile - check the other functions:

ADC Module

pi@raspberrypi:~ $ **cd Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server**

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python test.py ADC**

```
Program is starting ...
The battery voltage is 7.647058823529411V
...
^C
End of program
```

Ultrasonic module

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python test.py Ultrasonic**

readings correspond closely to distance between acrylic parts board and dog's face.

LED

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python test.py Led**

```
Program is starting ...
Red wipe
Green wipe
Blue wipe
White wipe
End of program
```

Colors change accordingly

Buzzer

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python test.py Buzzer**

```
Program is starting ...
1S
2S
3S
End of program
```

Camera

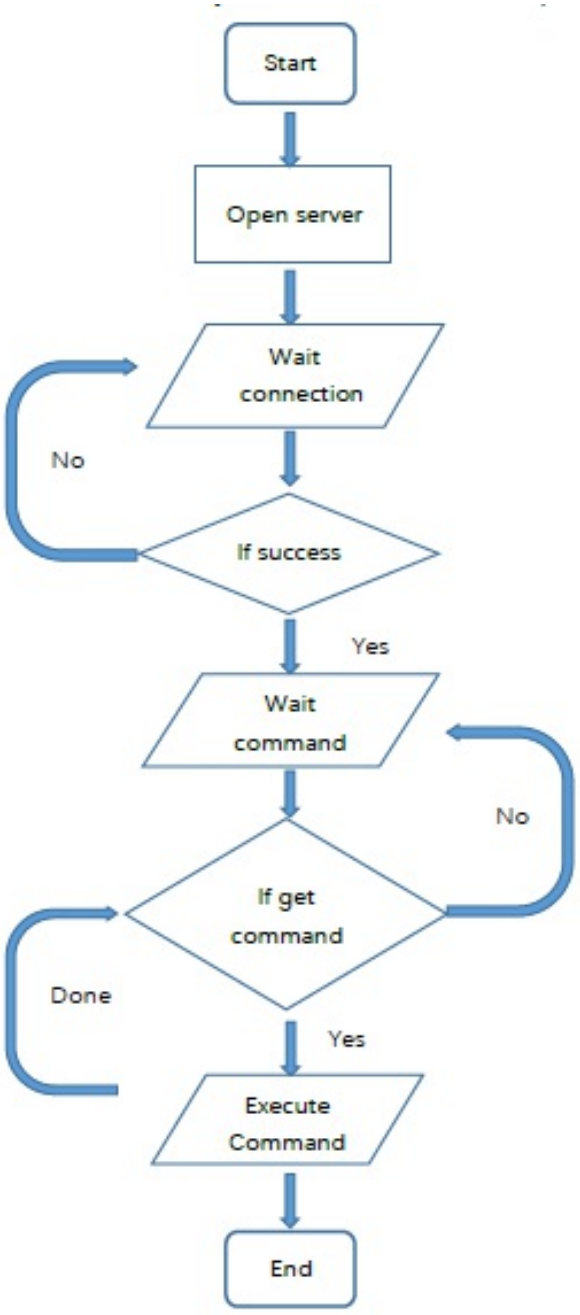
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **vcgencmd get_camera**

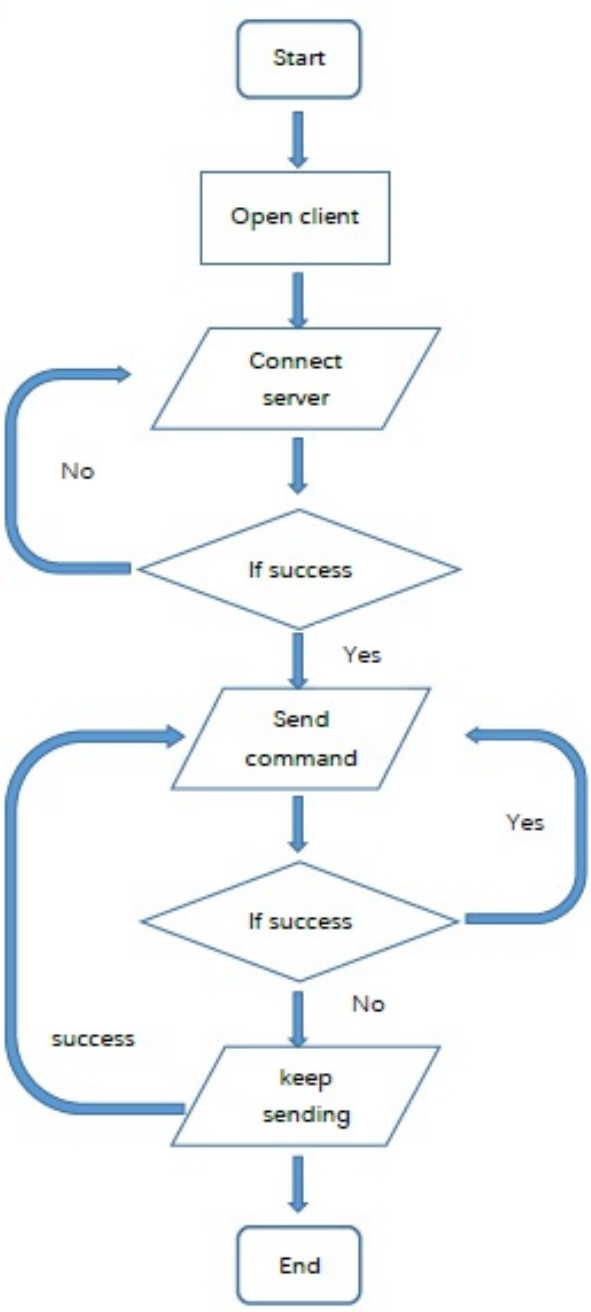
```
supported=1 detected=1, libcamera interfaces=0
```

pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **python camera.py**

Chapter 4 Robot Dog p86.

This robot has rich functions, such as real-time video, LED, ultrasonic ranging. The server and client are established, based on Python3 and PyQt5. They communicate via TCP/IP protocol. The robot can be controlled remotely within a local area network (LAN).





Open Server

Step 1 Login Raspberry Pi via VNC viewer

Because server and client use GUI. You need use VNC viewer as remote desktop way.

Step 2 Run commands

```
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ sudo python main.py
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
Server address: 10.0.0.30
libEGL warning: DRI2: failed to authenticate
```

If you don’ like the interface, you can also enter the commands to open the server. It is more convenient.

```
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ sudo python main.py -t -n
Open TCP
Server address: 10.0.0.30
```

Sever Auto Start

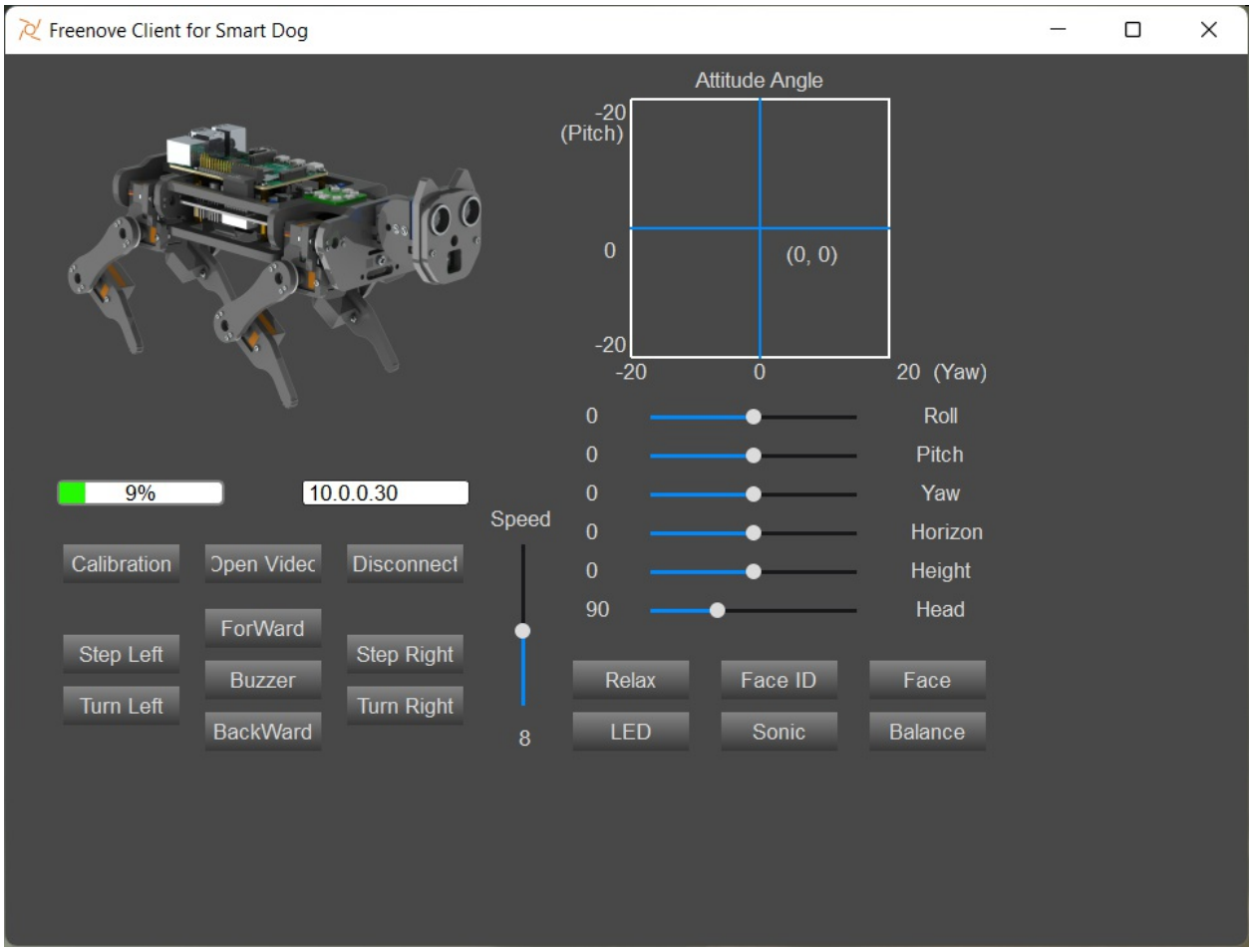
```
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ cd ~
pi@raspberrypi:~ $ sudo touch start.sh
pi@raspberrypi:~ $ sudo nano start.sh
      #!/bin/sh
      cd "/home/pi/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server"
      pwd
      sleep 10
      sudo cp point.txt /home/pi
      sudo python main.py
pi@raspberrypi:~ $ sudo chmod 777 start.sh
pi@raspberrypi:~ $ mkdir ~/.config/autostart/
pi@raspberrypi:~ $ sudo nano .config/autostart/start.desktop
      [Desktop Entry]
      Type=Application
      Name=start
      NoDisplay=true
      Exec=/home/pi/start.sh
pi@raspberrypi:~ $ sudo chmod +x .config/autostart/start.desktop
pi@raspberrypi:~ $ sudo reboot
```

Note: To cancel auto start, please delete the files “start.sh”and “start.desktop”created above.

Client p93.
The client can receive video data and commands from the server, and can send commands to the server. And it can run on different systems, such as windows, macOS and so on. However, you need to install related software and libraries when running it.

Run Client on Windows system

For best results extracted -master.zip to C:/Program Files (x86) then execute C:\Program Files (x86)\Freenove_Robot_Dog_Kit_for_Raspberry_Pi-master\Application\windows**Client.exe Run as administrator**.



1.You need to open the Raspberry Pi and Turn on the server,
pi@raspberrypi:~/Freenove_Robot_Dog_Kit_for_Raspberry_Pi/Code/Server $ **sudo python main.py**

2.Enter the Raspberry Pi's IP address in the white IP edit box,
10.0.0.30
3.Click "Connect" to connect client to the Raspberry Pi.

LOW VOLTAGE WARNING. CHECK POWER SUPPLY.
Fido battery 9%. REPLACE!

After the connection is successful, you need to calibrate the robot in Calibration section. After the calibration is completed, the robot dog can be controlled to move.
You can refer to this video: https://youtu.be/l2v9PdwQdvY

Taking each leg in turn, abduct or adduct (+Z-Z) the hip to align with the target spot. Move foot forward or back (+X-X) to align. Then lower the foot to not quite touch the calibration sheet. I adjust the vertica axis (-Y+Y) [note +Y pushes DOWN] to just allow a piece of paper to slide under the foot to be sure it's not tipping the balance.

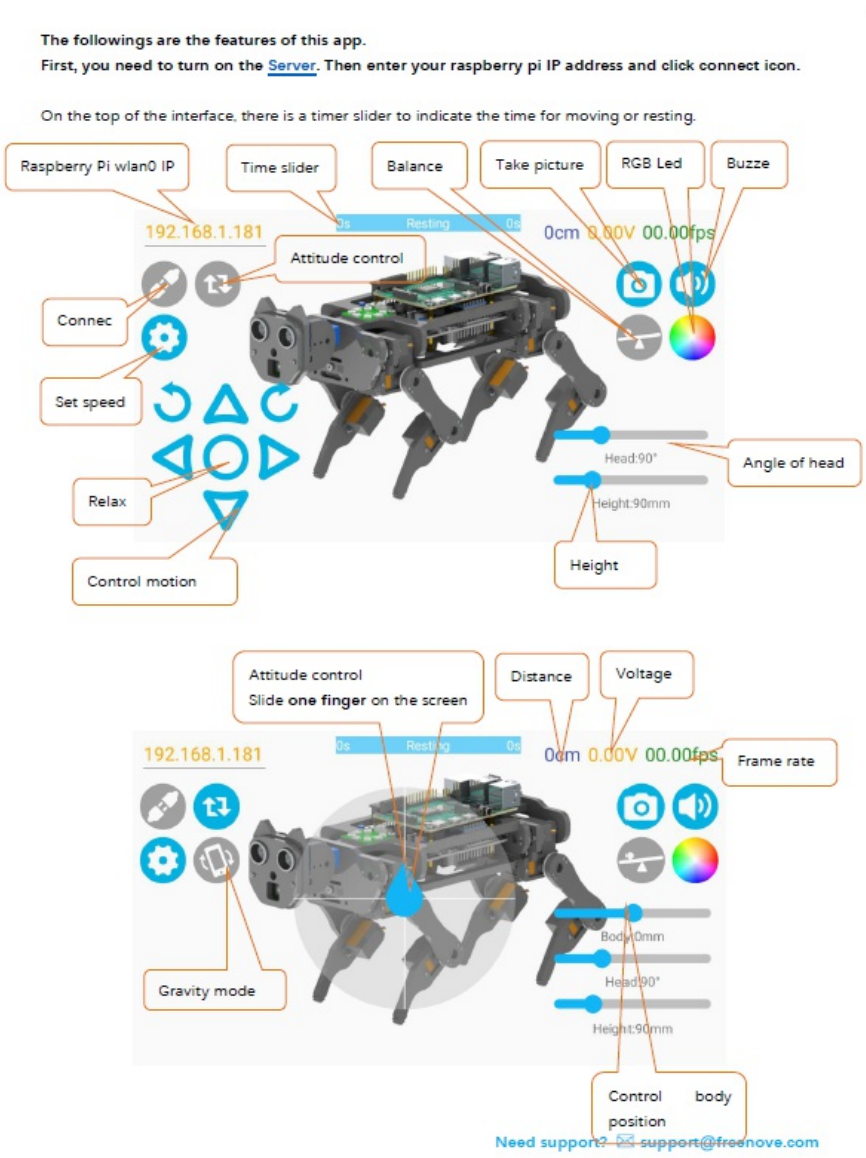
Right foreleg elbow servo #14 doesn't seem to respond.
But I can get the foot close enough to the calibration spot.

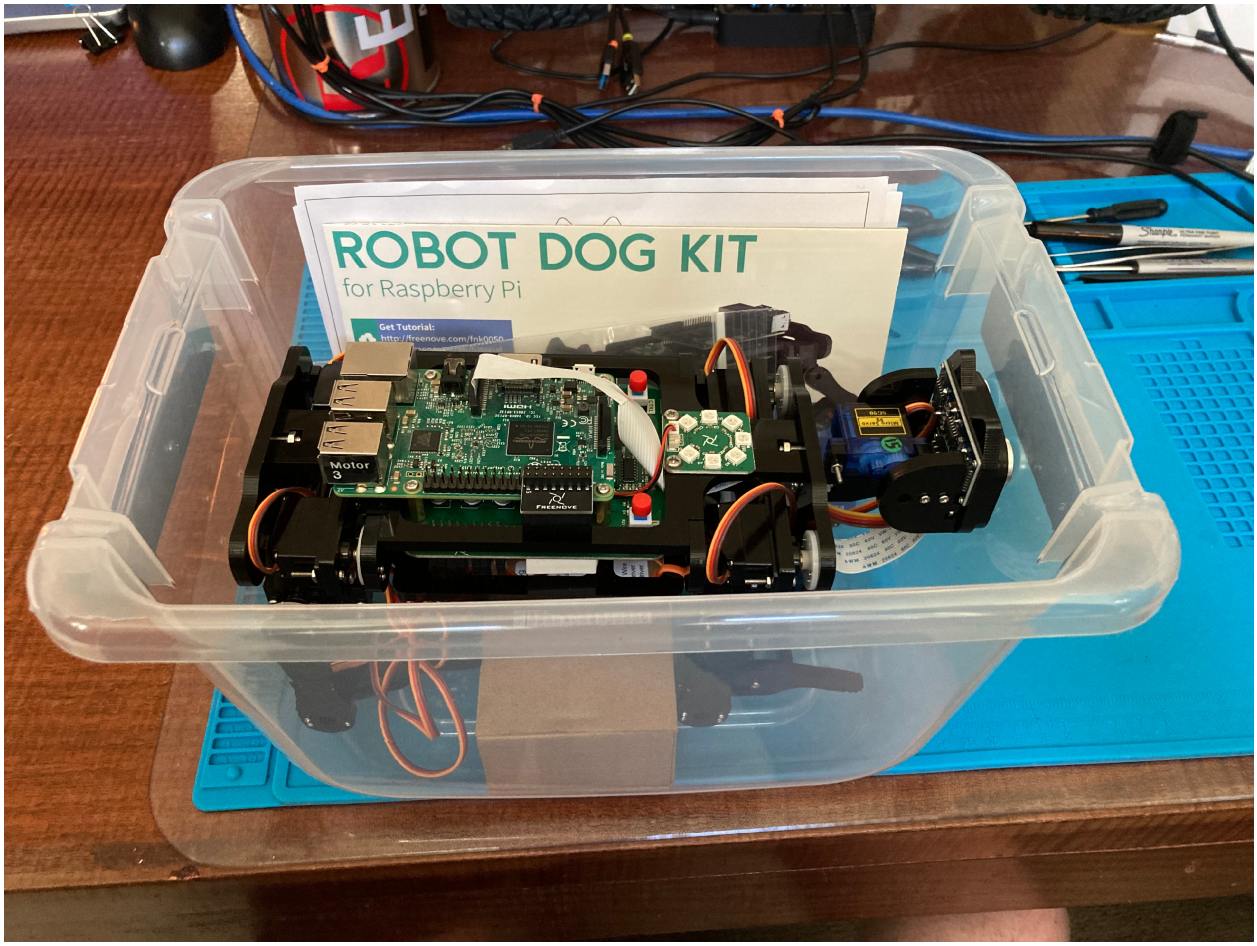
SAVE

Open Video:	shows dog-eye view.
Relax:	assumes standing position briefly, then 90o sit
LED:	opens LED window with 6 different modes and color adjustment
Sonic:	show distance from face to object
Ball/Face	tracks orange ball or recognizes trained faces. Don't know how to make forget mis-recognized face.
Step:	steps forward, backward, left, or right. Turns. This only works if the dog is balanced and all 4 limbs are working and calibrated correctly. Not!

You can download and install the Freenove iOS app by searching freenove in app store.

Relax mode.
a) When the robot dog moves for 3 minutes in total, it will feel tired (the servo will get hot). In order to protect the servo, the robot will get into relax mode for 1 minute. During this time, it won’ respond to motion command. You can still use the functions of LED, buzzer, real-time video and so on.
b) When the robot dog moves for <3 minutes and then the robot rest for 1 minute. The timer will start from 0. Then the robot can move for 3 minutes again.
c) If the robot isn’ tired and is standing, when the robot does not receive motion command for 10s, it will get into relax mode. In this situation, it will responds to any commands.





https://www.lowes.com/pd/Hefty-1-625-Gallon-6-5-Quart-Clear-Tote-with-Latching-Lid/1000505511

Fits like it was made for it!

2022.09.24
2d ago visited Roberto. He gave me a servo tester. [We also worked on the LIDAR adapter for the OSR.] Testing demonstrated at least 2 servos were dead. Roberto loaned me 4 same make/model servos as the original (with with the labels missing on my kit servos). Replaced the 2 bad servos. Today trying to do the calibration but both Windows.exe and Client.exe will briefly open a terminal window, but then disappear! :-(

Solved problem by deleting the unzipped files from my Robot Dog folder and unzipping the “-master.zip” file into C:/Program Files (x86) to avoid the long file name issue.

Then run client.exe As Administrator to bring up the GUI.

Was able to successfully calibrate the dog on the calibration chart, “save” it, unmount the dog and take it for a walk on my desktop mat. It’s not as agile as Spot, but for $130 vs $74,500 I’m satisfied! The other GUI functions work ok, too: buzzer, LEDs, camera, movements, sonar.