side project of the term (S24)

Robot dog

08/20/2024

Rubie Luo

Abstract

A quadruped refers to an animal with 4 legs. Quadruped robots have become increasingly popular in recent years, which is easily attributed to their flexibility in locomotion. In comparison to humanoid robots, it is significantly easier to balance on 4 legs; Compared to wheeled-robots, it can jump, climb, and handle stairs. Quadruped robots are favored for rough terrain and unknown or sporadic use cases. These robots have extensive applications in various fields such as space exploration, military application, industrial use, and many more. The development and innovation of such robots are expensive and extensive. For example, *Spot*, by Boston Dynamics is of the most well-known quadruped robots, with a market price of around $75, 000 USD. The goal of this project is to make an affordable quadruped robot, exploring the development and engineering required to make them successful in industry. The result is a small quadruped composed of 3D printed components, with a price point of approximately $100. The robot has the ability to walk in all directions and rotate 360°.

This project is open source! Check out the github here, and read **Assembly Instructions** to make it or improve it.

Acknowledgements

Thank you Shaqeeb Momen and Frank Rao. You are both the goat.

Contents

1 Introduction

1.1 Background

Robotic quadrupeds have been explored and engineered dating back to as early as the 1800s. L.A. Rygg sought to create the first mechanical horse in 1893. The idea is seen in fig 1. There is no evidence that it was ever made.

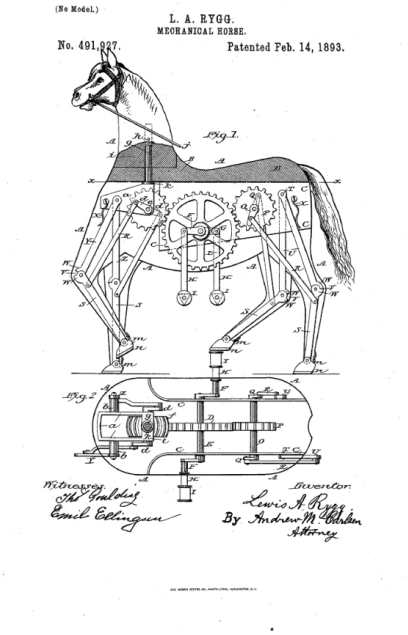

Figure 1: L.A. Rygg's mechanical horse patent.

Innovation has come a long way since then. The MiniCheetah (see fig 2), developed at MIT, is the first robot quadruped to do a back-flip. It uses brushless motors and a planetary gear with a 6:1 reduction, making it incredibly agile and robust [1].

Figure 2: MIT MiniCheetah

Another university-developed quadruped is the ANYmal (see fig 3) built at ETH Zurich [2]. The robot also became a product on the market through ANYbotics, a robotic spinoff of ETH. The ANYmal helps with industry grade inspections, and is built robustly in order to handle 10kg loads.

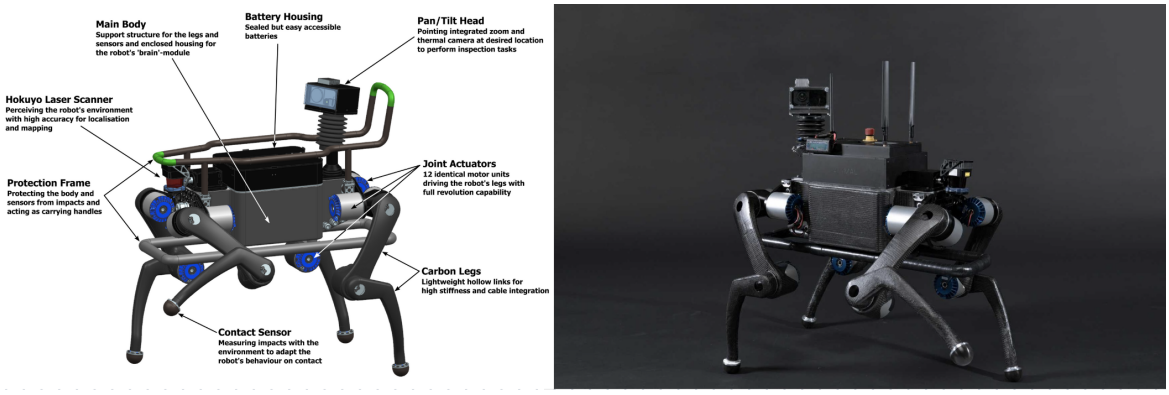

Figure 3: ETH Zurich ANYmal

Another key player in the quadrupedal robot market is Ghost Robotics, who's main use-case is military. See more.

Along with ANYbotics, the other two dog robots that tend to dominate the market are Unitree's GO2 and Boston Dynamic's Spot. There are many arguments and debates as to which is better, but this project focuses on Spot because the person who made it (me) really likes Spot. Both can be seen in fig 4, and for more info on either, visit Spot or GO2.

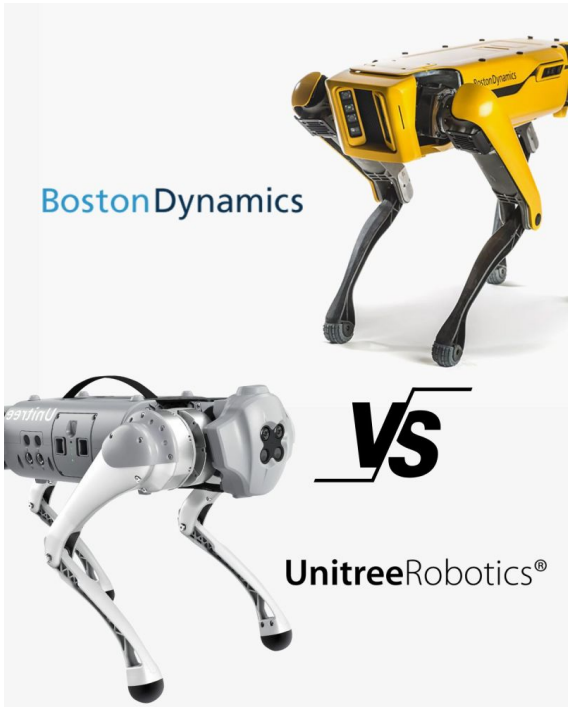

Figure 4: Boston Dynamics' Spot and Unitree's GO quadrupeds.

Spot was first unveiled in 2016, and made available to the public in 2019. Since then, it's been used for industrial purposes, construction, university/research, and the energy sector. The robot costs a steep price of $75,000 USD. The goal of this project is to create a smaller version of Spot that is more affordable. My original goal was to complete a project using more difficult inverse kinematics all on my own, and a robot dog definitely did the trick.

2 Scope

The project requirements were to walk in all directions and turn. It was also a requirement to use microservos since they would result in the cheapest and smallest robot.

2.1 Figuring things out

One of the first sources of inspiration was James Bruton's OpenDog project. The playlist of videos detailing his exact process is found here.

Another helpful model was Aaed Musa's TOPS. This was an incredibly well-researched project and I would highly recommend his video on his own custom QDD. While this quadruped had precise and impressive control, it was way out of my budget. His process is thoroughly detailed here.

I also found other hobbyist quadruped robots with a quick google search for "robot dog diy". All of the projects combined contributed a lot to my understanding of robotic dogs and heavily influenced ideas.

2.2 Robot Specifications

The final quadruped uses 12 microservos, 3 per leg. The MG90S servo was selected due to its small size and low cost. Each leg has a shoulder pitch, shoulder roll, and elbow pitch, illustrated in fig 5. The original goal was to be remote controlled, although because it was not a requirement, it was scrapped due to time constraints. The design of the dog gives it the ability to be innovated on at a later time, for example, if time permits, a camera could be integrated into the front which would open many new use case opportunities.

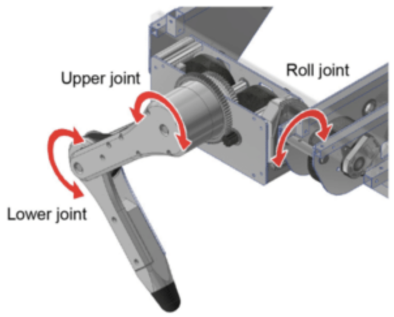

Figure 5: The 3 degrees of freedom in each leg of a quadruped. [3]

The robot uses an ArduinoMEGA as the microcontroller. This component was selected due

to the high number of PWM ports (14), which would support all 12 servos on the bot. No other microcontroller was as cheap and easy to use, while supporting 12 servos without an IO expander. A 7.4V LiPo battery is used for power, and two buck converters are used to regulate voltage. The ArduinoMEGA runs off 6-20V, while the servos accept 4.8-6V. These were originally both run off the same buck converter at 6V but this was scrapped. When both systems ran off of one buck converter, the current draw was likely too high. The behaviour seen was the setup() function in the arduino repeating every time it tried to move. The overall electrical system is seen in fig 6.

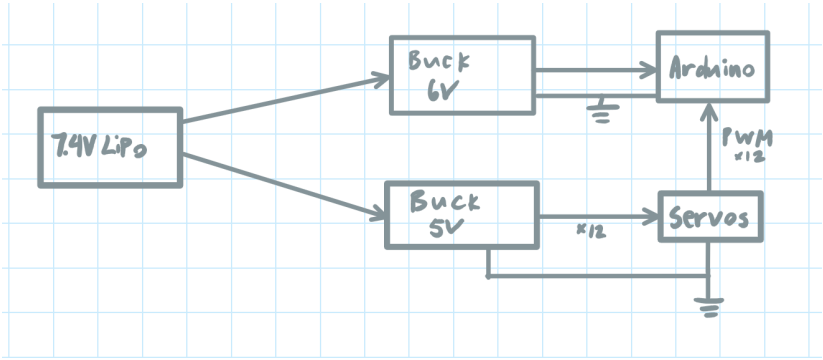

Figure 6: Electrical system

3 Design Considerations

One of the first observations when looking at quadrupeds was the knee direction. In particular, ANYmal is unique in having knees on both side pointing into the body. Even so, all other quadrupeds walk forward in the direction where the knees bend in. The purpose of this is very specifically so that they can walk up stairs or steeper terrain. In the case of walking up stairs backwards, the knee would bump into the next stair before it could go up.

Notice that earlier I specified 3 degrees of freedom and not 3 motors. There are a variety of ways to actually actuate these movements, it is just key that they are there. Some industrial quadrupeds use hydraulic or pneumatic actuators, however that is outside the scope of this project.

There are also different ways to transfer the energy. Boston Dynamics keeps their engineering fairly confidential, but there is speculation that the knee motor is connected to a worm gear that actuates the elbow pitch joint (Read more). In other prototypes, all three motors are near the shoulder area, with a joint down to move the elbow. This can be seen in many hobbyist dogs, as seen in fig 7.

Figure 7: Leg linkage quadrupeds, Dingo by Nathan Ferguson (left) and Kangal by Baris Alp (right)

The legs are basically the entire robot, and took the most prototyping and iteration. The biggest mechanical design decision I had to make was whether to use a linkage design, or simply a motor at each joint. The primary reason to opt for a linkage is to keep the heavy motors near the center of the robot, reducing inertia in each leg, and allowing for smoother movement. However, because of the fairly light motors I was designing with, I made the preliminary decision that it should not affect the robot too much. Thus, the latter option was selected for the final due to ease of design and easier inverse kinematics.

3.1 Overall Design

The final design in SolidWorks is shown in fig 8.

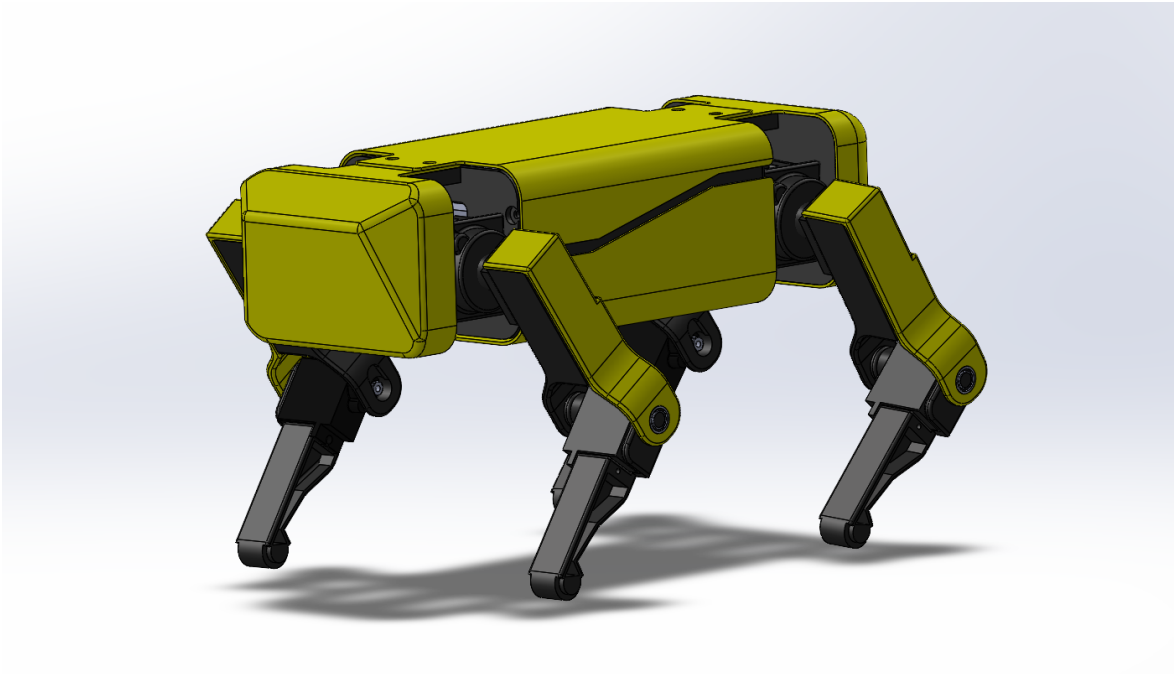

Figure 8: Final design

The dog was partially master-modeled, which is to say that each part is referenced to a common sketch. Namely, if a leg had to change dimensions, it should only be changed in the one sketch and the parts of the leg will update automatically. Similarly, if the shape of the body were to be changed, it also only needs to be changed in one sketch, and each part is subsequently updated.

The whole bot measures 225.26mm long, 84mm wide, and 174.97mm tall at full extension. Each section of the leg is 60mm long from centre to centre of the joints.

3.2 Challenges

This is the second iteration of the design. The first iteration was slightly larger, with the lower leg being 70mm and the upper leg at 65mm. **This original version could not walk as it would stall under its own weight**.

Since the battery was the heaviest component on the robot, it made sense to experiment with lower capacity batteries that were smaller and lighter. In the end, the original battery was kept because no significant difference was seen in the 30g weight decrease.

The importance of shortening the legs was immediate. Due to the longer legs, it would apply more force to the motor seen clearly by the torque formula. With the shorter legs seen on the final design, the robot was able to stand on its own (even unpowered) and walk.

This version has the body as small as possible while being able to house the ArduinoMEGA. In order to make it even lighter, all parts were printed at 5% infill.

4 Software Design and Implementation

In order to design the software, the requirements were first listed out, followed by what was needed to meet the requirements, and correspondingly breaking into sections and drawing a block diagram with the needs. The final block diagram is shown in fig 9.

The overall software design becomes very simple with the block diagram, the blocks represent classes and the arrows show the inputs/outputs to control. Each requirement of the block is a function of the class. The final code has 4 classes; Quadruped, Leg, Gait, and Point. Full code.

A large portion of the code is made to accommodate the absolutely-positioned servos. This means that during assembly, the servos must be clocked a certain way, and the code has to conform to it as well. 2 legs will always be inverted in the orientation now which had to be acknowledged and dealt with in code. Due to the clocking of servos not always being perfect, each servo has offsets to set as well. This all goes into tuning the dog, which can be tedious.

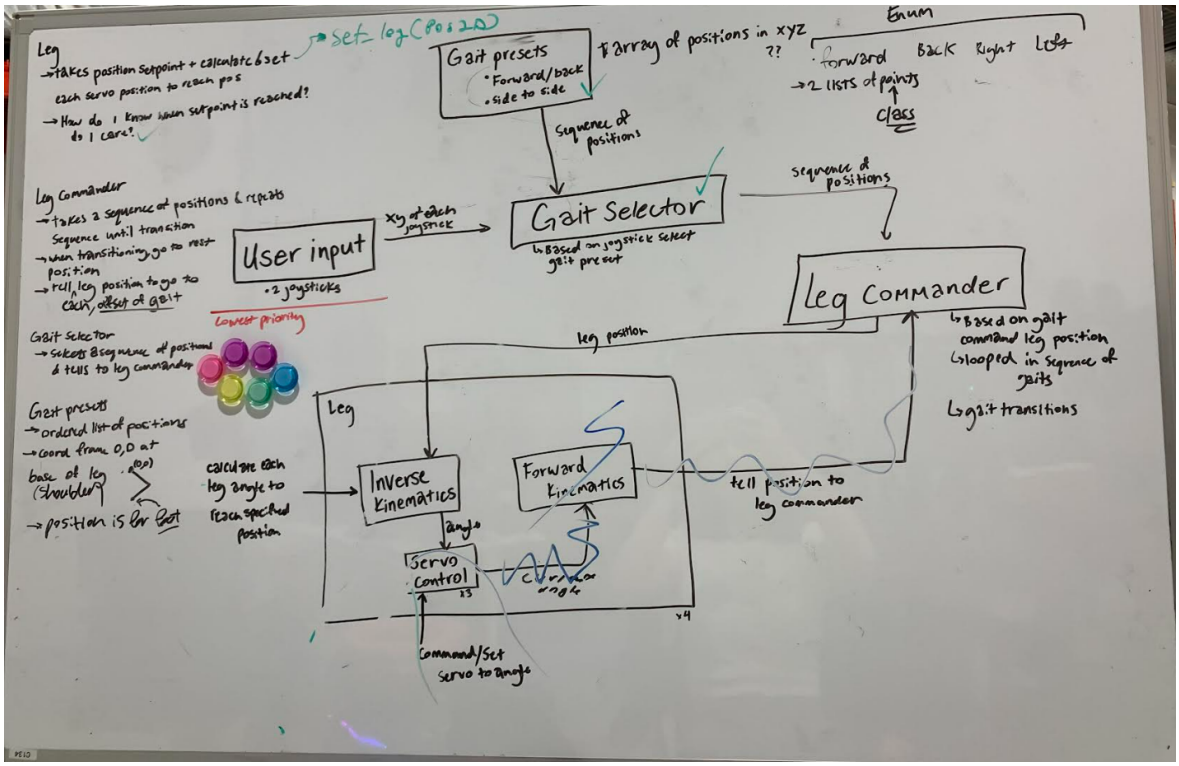

Figure 9: Software block diagram

One thing to note is that the user input got scrapped due to time, although if it were to ever be implemented it would be a matter of connecting to the UART pins on the Arduino and programming it with 2 joysticks, one for strafing and one for turning.

In the planning process, forward kinematics was scrapped as there was no need for it, but it was left visible to be aware of the process it took to get there.

4.1 Inverse Kinematics

The heart of inverse kinematics, at least in this context, is just very simple trig. The most difficult part of implementing the inverse kinematics was simply the inverted motors and separating positives from negatives. While designing the inverse kinematics, the calculations I derived were not quite working, so I did the calculations by hand for the first gait. This then led me to the calculation used is shown in fig 10.

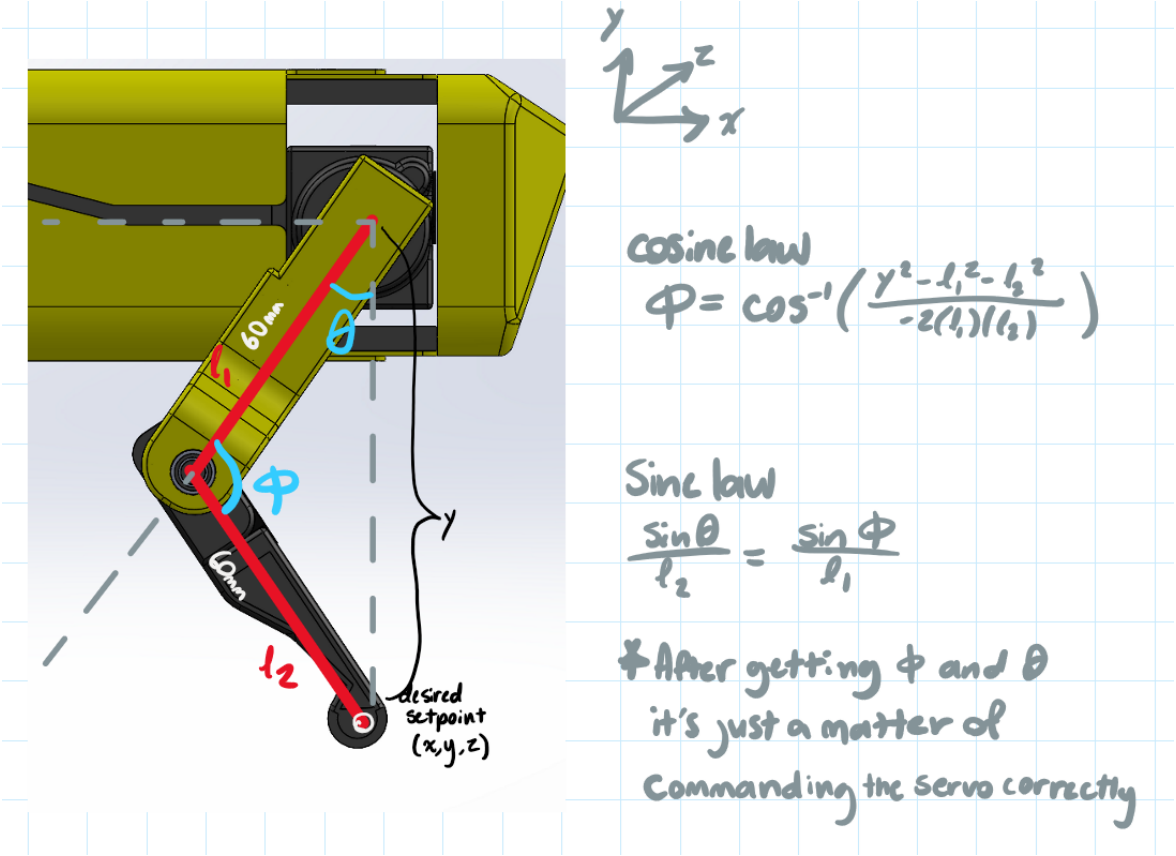

Figure 10: Inverse Kinematics

5 Assembly Instructions

The rest of this document's purpose is to provide the reader with enough information to make the dog if desired. I will try to go through all details of the process, but I understand some parts are very difficult to grasp. if things are unclear and you are truly trying to create the dog, please reach out to me directly and I will do my best to help out rubie.luo@uwaterloo.ca.

5.1 Mechanical

For a list of all parts needed for the project, scroll down to BOM/COTS List.

The first step of your process will be to 3D print all the parts. All components were printed with PLA. In the github repo, I've separated the parts to print in yellow and the parts to print in black for anyone who wants to replicate the Boston Dynamics look. Feel free to print in whatever colour you want. I used the Bambu Labs P1P, and I highly recommend this printer to anyone getting started with 3D printing.

The print settings I used for everything is 5% infill and 3 wall layers for all sides (bottom and top layer as well). The 15% infill is the only setting that will be helpful in the long run, wall loops and basically anything else can be customized.

Assembling the dog should be fairly intuitive with the CAD on hand. The dog uses M2 bolts for everything with the exception of M4 to secure the top leg pieces together, and to attach the yellow casings to the structure. The only unintuitive part is the servo horn on the elbow joint. I decided to trade-off the manufacturing process for the advantage of a smaller joint, meaning the servo horn has to be cut in half to fit in its given area. I started by cutting it in half, and then sanded it down until it fit.

Figure 11: Location of servo horn that requires modifications.

Also, clocking the servos while putting it together is necessary to make sure the dog can reach all ranges of movements it requires. I clock the elbow servo to be colinear with the upper leg at its farthest range of motion so that it can only bend to the front of the robot. The shoulder pitch is clocked similarly, but parallel with the body. The shoulder roll is clocked to have 90 degrees as the normal standing position, which would just be perpendicular to the body. By farthest range of motion, I mean either 180 or 0 degrees, based on whether that leg is inverted. The easiest way to do this is with the testing servos code in the github. Allow the servo to move from 180 to 0 and unplug it when it is at the end of its motion and about to go in the direction required. Then, just screw it in the position mentioned above, with the exception of the shoulder roll which can just be commanded to 90 degrees and attached in position.

I chose not to clock all of them to 90 (which would be easier since inversion doesn't matter, 90 is in the middle) because I did not want to guess what the middle of the required movement would need to be. Also, almost all servos do not actually reach a full 180 degrees of motion, which made it important to only clock the extreme of the servo to a point farther than it would ever have to go.

5.2 Electrical

Starting this project, electrical was the most foreign concept out of everything I knew. Controlling 12 servos is also not easy, and a lot of research went into doing this.

For the electrical system, I soldered a universal PCB as shown in fig 12. This was the easiest way, at the time, to connect all the servos to one power and ground; although a simple PCB would make this much simpler. I used male pin headers because the servo uses female pin plugs, so rather than needing another wire to connect to the board, the servo connects directly to the board.

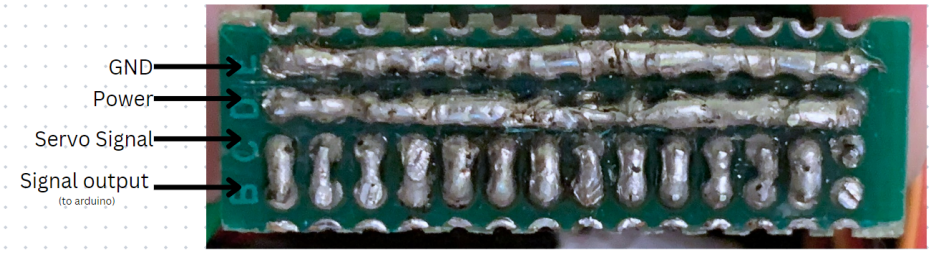

Figure 12: Universal PCB

The battery I got has two output connectors, which was incredibly convenient for my two buck converters. I used a connector for each buck which eliminated the need for another PCB to separate the output into 2. In case your battery does only have one output connector, attach the output to another universal PCB which would lead to two outputs in parallel. (if you know PCB design in KiCAD or Altium and wish to order a pcb, your life becomes much easier, unfortunately due to time I couldn't finish it but I would've liked to combine the buck converters into a custom pcb).

The buck converters I got required soldering for the input and output lines. Some buck converters you can purchase will have screw terminals instead which would make it much easier for modularity. On mine, I bought the two female connector ends and soldered that to the buck input. I soldered female pins to the output, although this depends on where you choose to route the output.

Since the buck converters are adjustable step-down, make sure to adjust them. Connect the battery to the buck and use a multimeter to probe the other side. Turn the small screw counterclockwise to lower the output voltage or clockwise to increase it. You may have to turn it many times before the number begins to change, this just depends on the orientation it is in when you buy it, which is unknown.

I stepped the buck for the Arduino to 6V, and the buck for the servos to 5V. Although the Arduino prefers to be slightly higher, be mindful that the buck is converting down from 7.4V, and due to a minimum dropout voltage, it will not convert to voltages too close to 7.4V.

Connect all the servos to the universal PCB. Make sure you know which order they are in, it will help with programming it. Connect all the signal pins to the Arduino's PWM pins. Attach the buck output to the Arduino's VIN and GND pins. The other buck can have the ground connected to all the servo grounds. I used a rocker switch to turn off/on the servos and connected it after this buck, meaning that the positive buck output goes to one side of the switch, and the other side of the switch connects to the power of the servos.

5.3 Programming

To tune the leg, command each leg to 0 or 180 (based on inversion) and figure out how far it is from where the desired clocking position was. Keep changing the angle until it is exactly in place. The difference in the numbers is the offset. Call set_offsets() in Quadruped.cpp and set each offset accordingly.

Since most of the programming is done already, this will mainly be how to change parts of the code to play around with gaits and speeds. Feel free to use a different MCU and start the programming from scratch.

To get started, in the repo go to Software and copy the whole dog folder. Paste this into the Arduino folder, which is by default in Documents. When you open arduino now, you should see the dog project in sketchbooks.

Connect the arduino to the laptop and upload code. By default, this code just makes the dog walk forward continuously.

To change or add gaits, add an array of points for the bottom of the foot to be to Gaits.h. Name the gait and add it to the enum at the end of the file. Add this as an option in Gait.cpp. To see the gait, call set_gait() in dog.ino before the walk() command.

To change the speed between each gait, change the delay in Quadruped.cpp. The delay is in milliseconds.

6 COTS List

The exact parts I used are linked. Two improvements I would've loved to do but ran out of time is to replace the universal PCB by designing everything into one board. Additionally, any MCU can be used to replace the Arduino MEGA as long as you can program it. Many other options may also require an IO expander board since 12 pwm ports are needed.

Part	Quantity	Notes
Arduino Mega	1	Any MCU can be used if you know how to program it
MG90S Servo	12	Should come with servo horns and screws
Buck Converter	2	Adjustable step down
7.4V LiPo	1	
JST connector	1	Depends on the battery you buy**
JST-PH connector	1	Depends on the battery you buy**
Rocker Switch	1	
Universal PCB + Pin headers	N/A	
Pin wires	N/A	
Metric bolts	See notes	M2-12 (x12) M2-16 (x8) M4-8 (x10)
Metric nuts	See notes	M2 (x20) M4 (x10)
Bearings 5mm ID 9mm OD	4	

Everything else that is needed for assembly is a 3D printer, soldering iron + solder, M2 and M4 allen keys, and a screwdriver for servos.

References

[1] W. Bosworth, S. Kim and N. Hogan, "The MIT super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion," 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)

[2] "Generation D ANYmal Technical Specifications → Leading Autonomy & Mobility → Intelligent Inspection → Industrial Grade." Available: https://www.anybotics.com/anymal-technical-specifications.pdf

[3] A Reciprocal Excitatory Reflex Between Extensors Reproduces the Prolongation of Stance Phase in Walking Cats: Analysis on a Robotic Platform - Scientific Figure on ResearchGate. Available: https://www.researchgate.net/figure/Leg-design-of-the-quadruped-robot-A-Degree-of-freedom-for-the-legs-B-Detailed_fig2_350734605