AUTODESK Instructables

Stewart Platform

By thiagohersan in CircuitsRobots

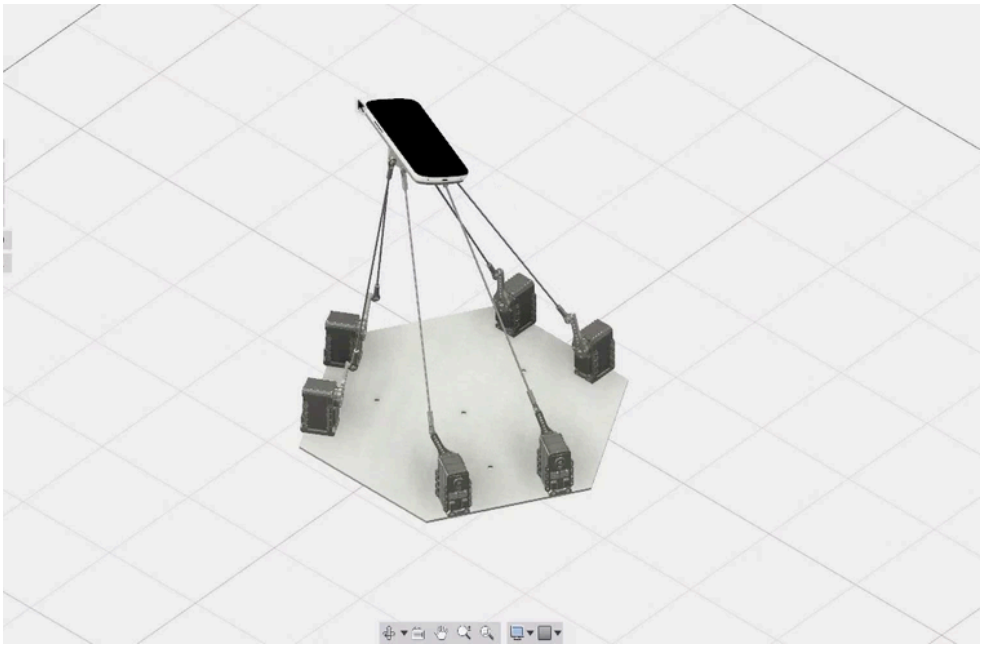
Introduction: Stewart Platform

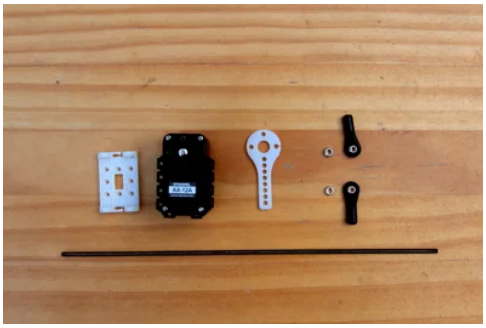
We're working on a project where we explore robotic body language as a way to augment/replace purely digital communication protocols. For this, we need to move a cellphone with as many degrees of freedom as possible. A robot arm would have been a good option, but for aesthetic reasons, we don't want the mechanism to draw too much attention to itself. After a bit of research, we've settled on a structured called a Stewart Platform.

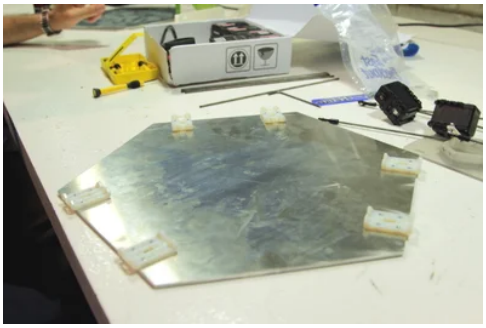
The Stewart Platform is a really amazing and versatile building block of robotics. It exists in many sizes and is used for many purposes. Lots has been written about it. But sometimes it's hard to find a nice intro to the basic operation of its mechanism.

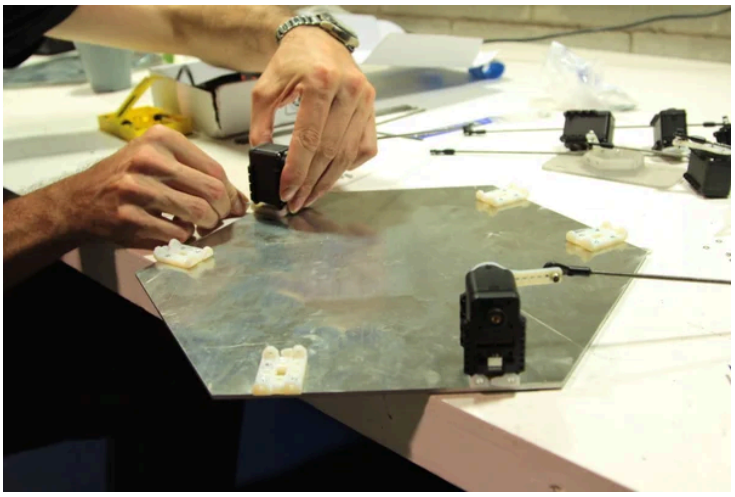
After reading a couple of papers and trying out a couple of different implementations, this is the one that worked for us.

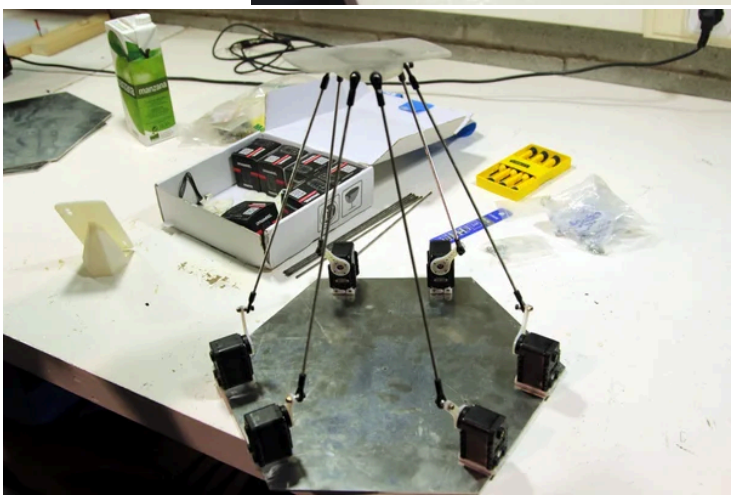
Step 1: Hardware: Modeling, Cutting, Printing and Assembling











We designed the platform using servo motors because they are easier to find and cheaper than linear actuators. The range of motion is a bit more limited, but we were designing for expressiveness and not so much for range.

Before cutting metal and programming motors, Radamés designed a version of our platform using Fusion 360 in order to get some intuition about the range of movements and motor positioning.

The materials we used were:

- 1x RaspberryPi (with 5V/2A power supply)
- 6x Dynamixel AX-12A motors
- 1x UART circuit (with 12V/6A power supply)
- 1x aluminum base
- 1x plastic platform
- 12x M3 ball joints
- 6x threaded rods
- 6x servo brackets
- 6x servo horns
- lots of tiny screws

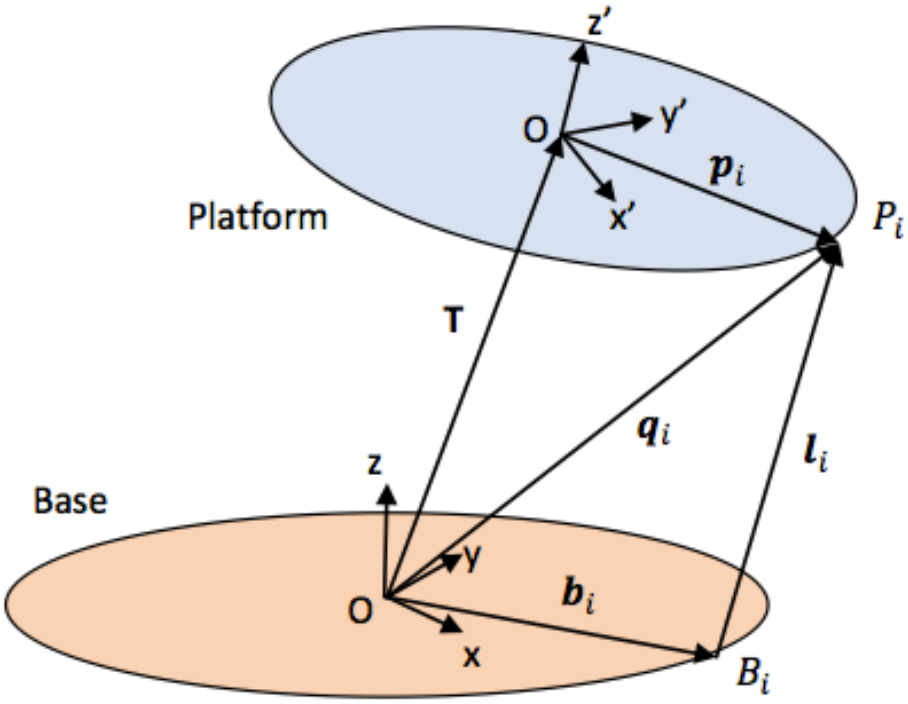
The servo brackets and horns came with the motors we bought, but we also modeled them in case we needed replacement parts. Simpler acrylic pieces can be used to hold whatever object you need on top of the platform, but since we wanted to hold cellphones in an upright position, we ended up designing a custom platform. STLs are attached here. The servo horns can also be cut from 1/8” acrylic or wood using a laser cutter.

Step 2: Testing

We tested our AX-12A library and motor driver circuit using one of the motors attached to our platform. We quickly realized that the servos on the platform can’t be driven independently of each other; in most cases, all six motors have to move simultaneously in order to achieve a desired position or pose.

This is a video showing how all the motors are affected when we drive only one motor up and down.

Step 3: Math: Papers



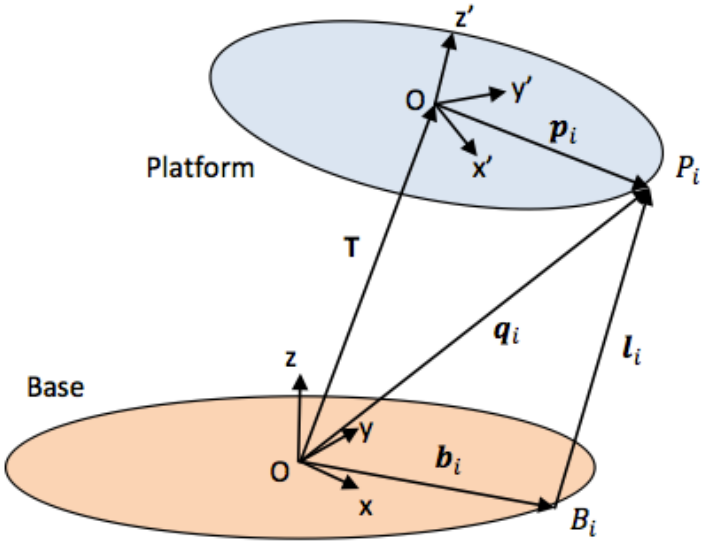
Before we started sending signals to all the servos, we figured it would be worthwhile to get familiar with the mathematics of the Stewart Platform.

Unlike articulated robotic arms, the Stewart Platform's inverse kinematics are simpler than its forward kinematics. What this means is that it's easier to calculate the leg lengths and motor parameters given a desired position for the platform, than to calculate where the platform is located for a given set of motor parameters. This is fine; we really want the inverse kinematics anyway, and that way we avoid solving a system of 18 non-linear equations with 40 possible solutions.

Some of the papers that we found were not specific to servo-based Stewart Platforms; they simply described the math based on desired leg lengths, and sometimes assumed that linear actuators would be used. This is the case for this MICS journal paper, which describes a very specific Stewart Platform and focuses on its forward kinematics.

We also found two papers that were more specific for servos. This paper by Filip Szufnarowski describes the inverse kinematic problem very nicely, but it was this document by an unknown author from the Wokingham U3A Math Group that had the most detail and cleanest notation. For example, this image that labels all the relevant points of a Stewart Platform with their corresponding coordinate system.

Step 4: Math: Joint Equations



$$+ {}^{P}\mathbf{R}_{B} \cdot$$

$$R_x(\varphi)$$

$$\begin{matrix} -\sin\psi\cos\varphi + \cos\psi\sin\theta\sin\varphi & \sin \\ \cos\psi\cos\varphi + \sin\psi\sin\theta\sin\varphi & -\cos \\ \cos\theta\sin\varphi & \end{matrix}$$

The inverse kinematics problem of a Stewart Platform can be broken up into two stages:

(1) Given a desired position and orientation for the platform, how far is each joint on the platform from its corresponding base joint,

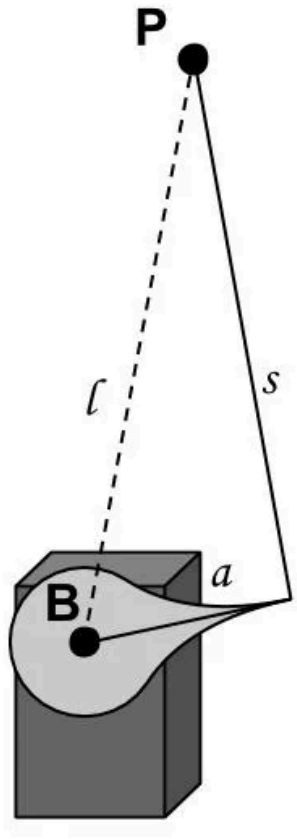
and

(2) What servo angles, if any, put each platform joint in the positions calculated in the previous step.

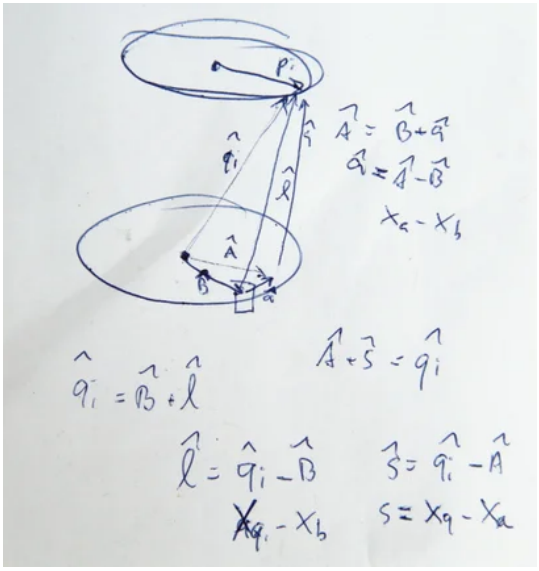
The first problem is easy to solve; once you have the appropriate points and coordinate systems defined like in the above image, the distances between base joints and platforms joints can be calculated with simple matrix operations for rotation and translation.

Namely, the length of each leg is calculated using the li equation in the image above. T is the translation vector between base coordinate system and platform coordinate system (where you want the platform to move), bi and pi are the locations of the joints in base and platform coordinate systems, respectively, and PRB is a rotation matrix describing how you want the platform to rotate. PRB is also described in detail in one of the images above.

Step 5: Math: Length Equations



$$l = \frac{L}{\sqrt{M^2+N^2}} - s$$



The second part of the inverse kinematics problem is a little bit trickier. For each servo, given a platform joint position P, relative to the base joint position B, and fixed lengths for the servo horn a and support leg s, what is the servo arm angle that satisfies the distance l calculated in the previous step.

Since l increases as you vary the servo arm angle from -90° to +90° (relative to base plane), one way to solve for this angle is to do a binary search over the angle values, and find the one that more closely satisfies all the distance constraints. This is done on the code for this Stewart Platform.

But, the Wokingham U3A Math Group document actually steps through the derivation of a closed-form expression for this angle, using some pretty sweet geometry, algebra and trigonometry tricks.

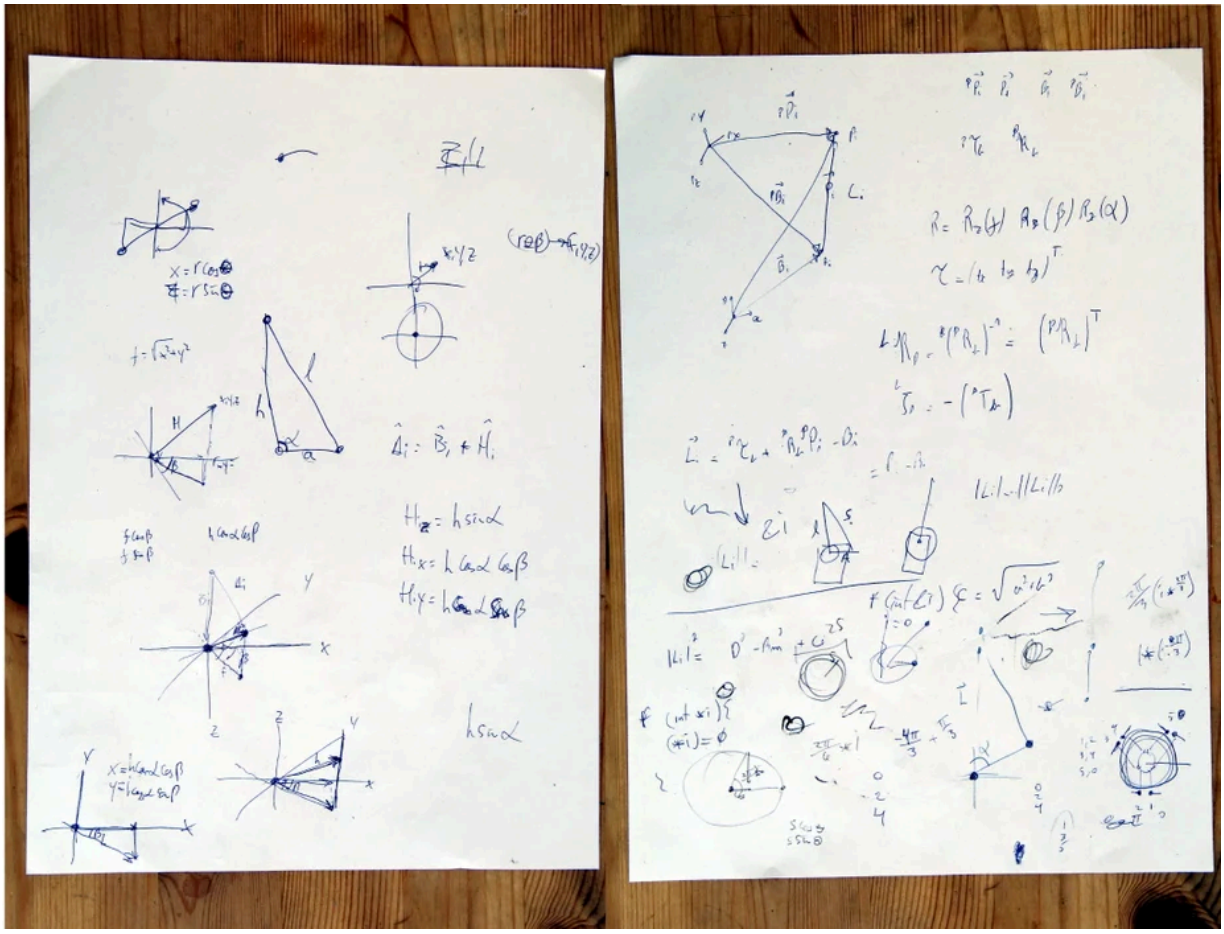
The form of the equation for the angle is shown in one of the images above (read the paper for the full derivation and definition of the variables).

We actually found a small bug in this part of the U3A document: instead of using the p values for the platform joint positions in the platform coordinate system, you have to use the q values, which are relative to the base coordinate system.

Step 6: Math: Testing

We wrote a simple Stewart Platform simulator for our platform, to see the range of movements that it will be able to achieve, and to double-check the math. The code is in Processing, and is on github.

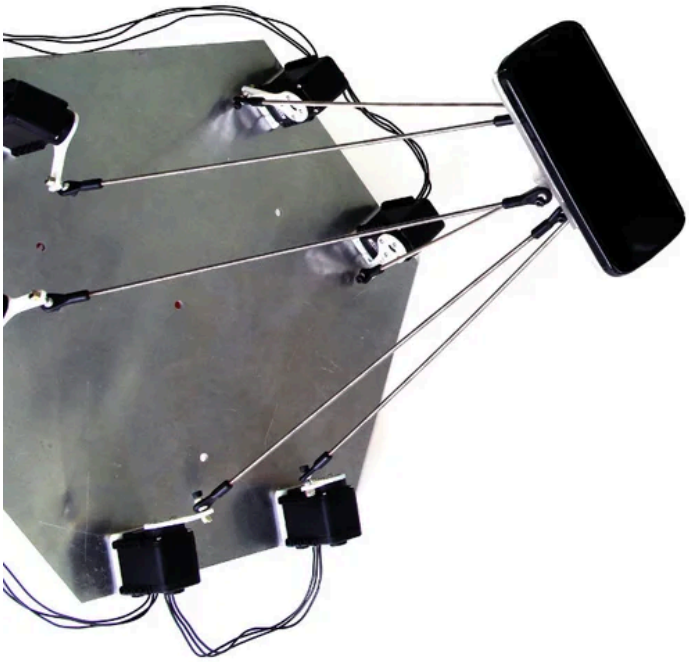
Step 7: Controlling the Stewart Platform



Once the math was verified, it wasn't hard to extend the code to get real-time communication going between the simulator and the platform. We used OSC to send motor angle packets to the RaspberryPi controlling the platform, and a bit of code to transform these angles into AX12-A motor commands. The code for controlling the motors using the simulator is on github.

Some of this initial controlled movement can be seen on the videos above.

Step 8: Links



- Information about the overall project.
- Somerelevantblogposts.
- List of references.
- OtherPlatforms on Instructables.