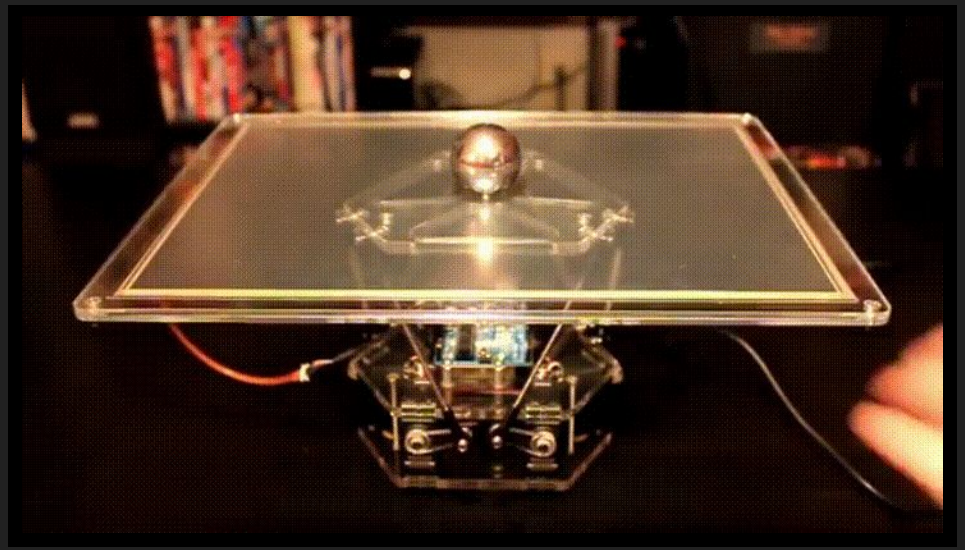
FPGA
Ball and Plate

Adam Rodriguez
Kaname Favier
Cameron Ordone

Overview

Keep a ball at some point on the plate.



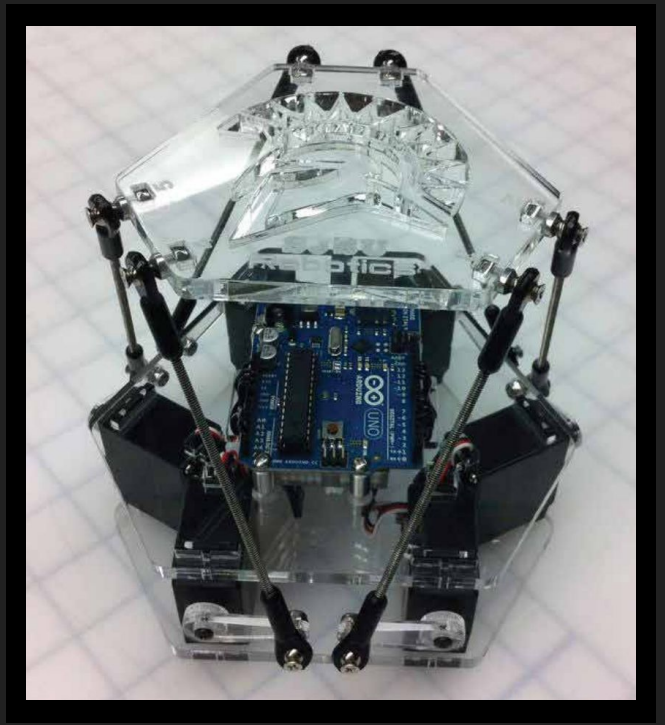
Hardware

Plate is a resistive touch screen; fast and cheap way to get ball position.

- Interfaced through separate IC that talks over SPI.

Platform is a servo-actuated Stewart platform (hexapod).

- Capable of 6-DOF but we’re limiting to two: roll and pitch.



Goals

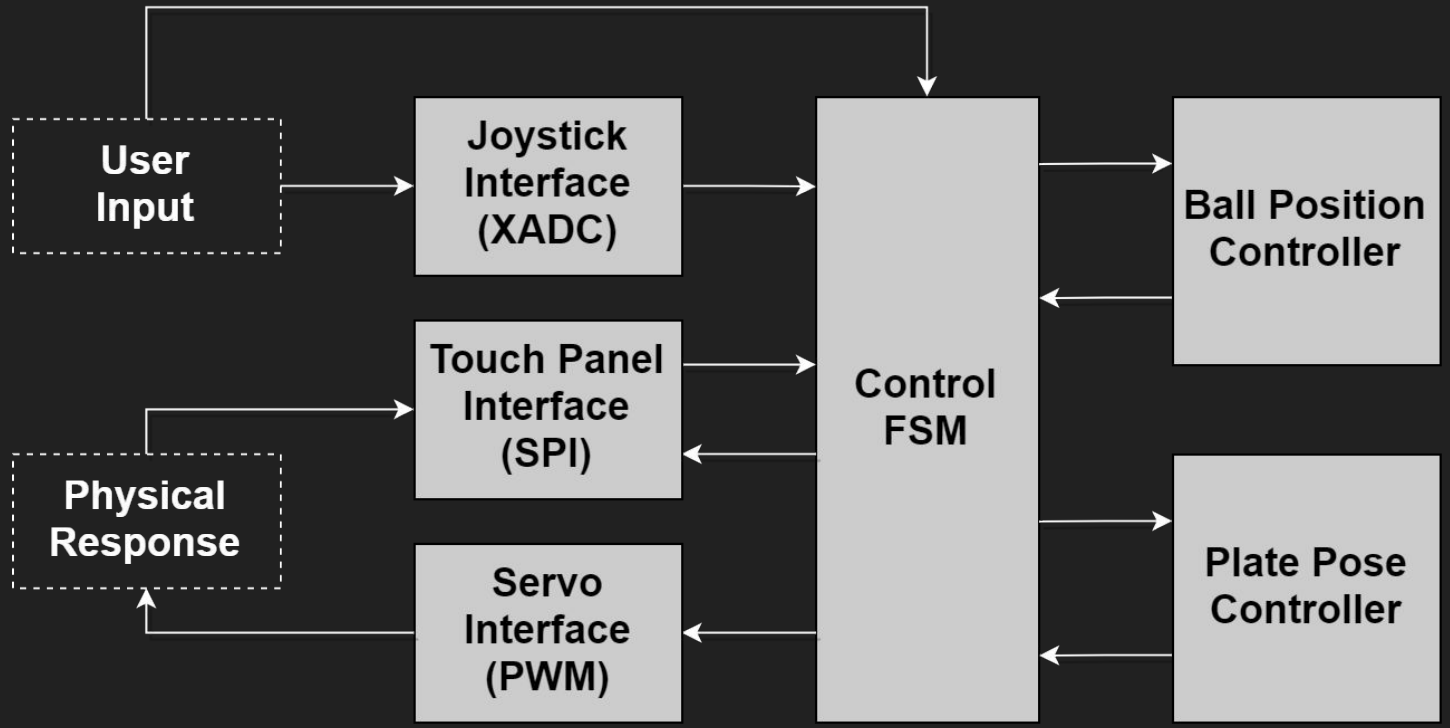
Baseline

- *Manual mode*: Joystick controls plate tilt directly.
- *Feedback mode*: Plate tilts automatically. Joystick adds offset to setpoint.

Stretch

- Extend *feedback mode* to include paths (circle, ellipse, figure-8, etc.)
- Extend *manual mode* to alternatively accept input from an IMU.

Block Diagram

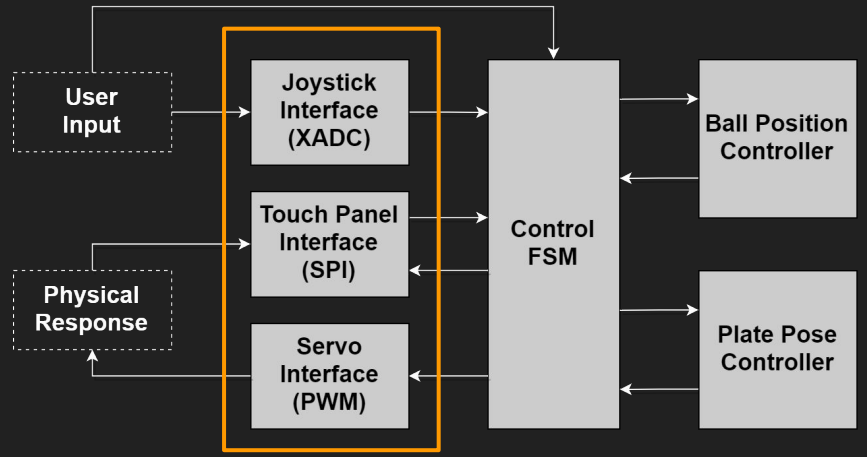


Interfaces

Sample joystick potentiometer voltages to 12-bits with Nexys 4 ADCs.

Receive 12-bit touch panel X and Y samples from Adafruit IC over SPI.

Convert calculated servo angle into the appropriate PWM signal.

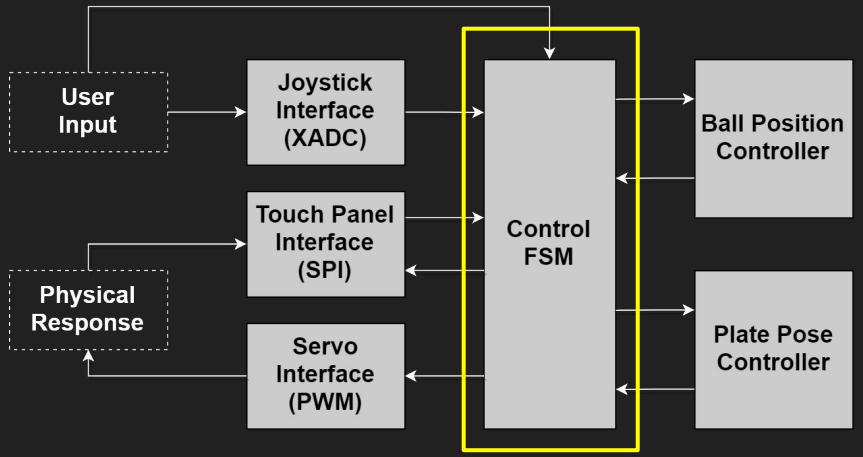


Control FSM

Responsible for managing data flow between modules.

Handles transitions between manual and feedback mode.

Stores the current setpoint for feedback mode.



Ball Position Controller

Error between the real and desired ball states sets new roll and pitch values.

Axes controlled separately (X-axis error → roll, Y-axis error → pitch).

Plan to do LQR.

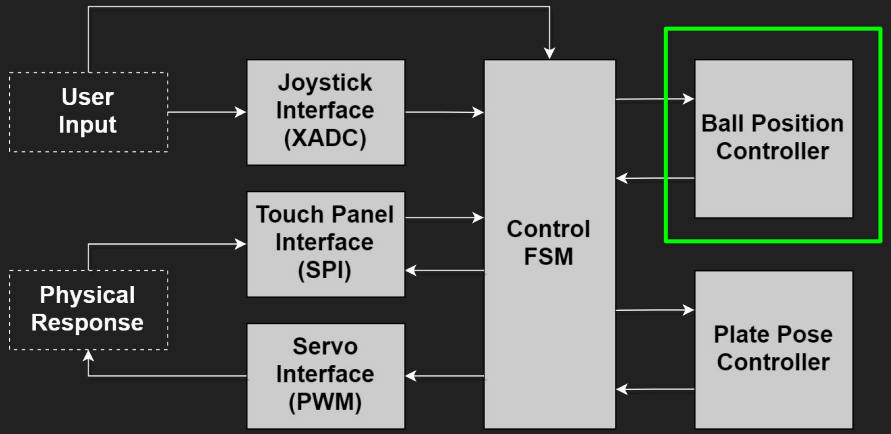
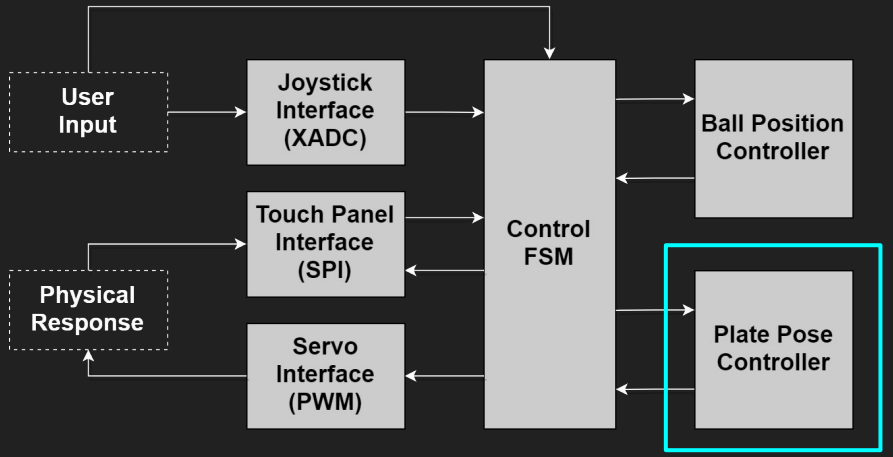


Plate Pose Controller

Inverse kinematics of our Stewart platform.

- Given a pose (translation and rotation), calculates the required leg lengths.
- Then, converts these leg lengths into servo angles (**the hard part**).

Likely most time-intensive module.



Pose Controller Insights

The rotation matrix contains:
- 3 s-c pairs
- min. 12 products

Luckily, we can avoid products by changing them to right-shifted sums.
We can also set yaw (ψ) = 0.

Updated matrix contains:
- 4 s-c pairs (11 if $\psi \neq 0$)
- 0 products

$$
{}^{P}\mathbf{R}_{B} = \mathbf{R}_{z}(\psi)\mathbf{R}_{y}(\theta)\mathbf{R}_{x}(\phi)
$$

$$
= \begin{pmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix}
$$

$$
\text{If } \mathbf{R}_{z} = \mathbf{I},
$$

$$
{}^{P}\mathbf{R}_{B} = \begin{pmatrix} \cos\theta & \frac{1}{2}[\cos(\theta-\phi) - \cos(\theta+\phi)] & \frac{1}{2}[\sin(\theta+\phi) + \sin(\theta-\phi)] \\ 0 & \cos\phi & -\sin\phi \\ -\sin\theta & \frac{1}{2}[\sin(\theta+\phi) - \sin(\theta-\phi)] & \frac{1}{2}[\cos(\theta+\phi) + \cos(\theta-\phi)] \end{pmatrix}
$$

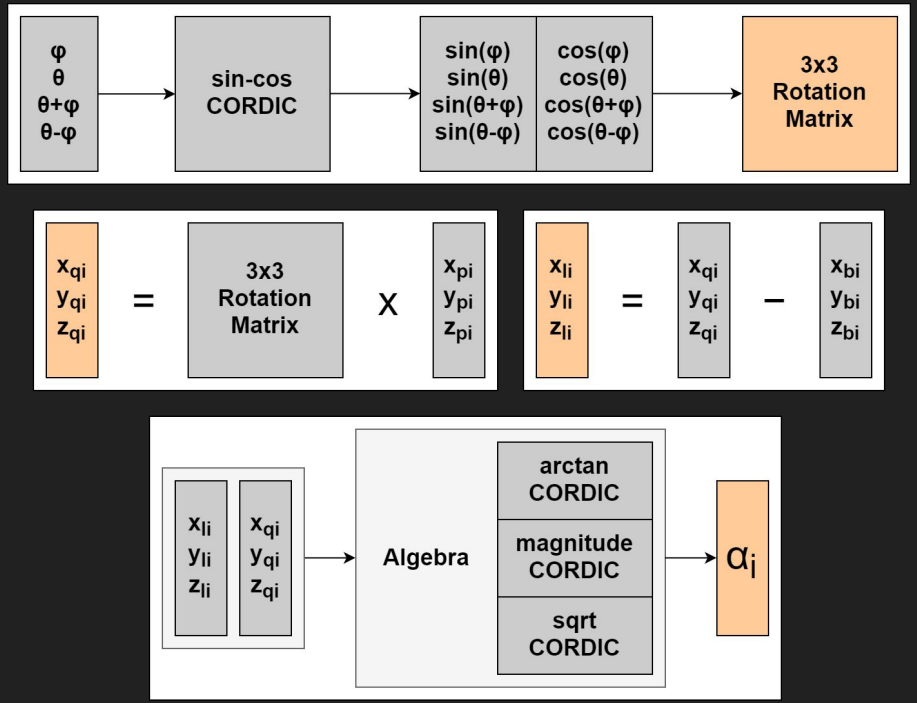
Pose Block Diagram

Three-step process.

1. Populate rotation matrix.

2. For each leg, find the new location of its plate joint ($q_i = \boldsymbol{R}p_i$). Then, subtract the position of its servo's output shaft to get its "length" ($l_i = q_i - b_i$).

3. Solve for servo angles (α_i).



Timeline

	Adam	Kaname	Cameron
11/4 - 11/10	Write and test PWM module. Begin writing pose controller.	Begin writing control FSM.	Write and test joystick interface. Fabricate touch panel mount.
11/11-11/17	Work on pose controller.	Begin writing manual mode for control FSM. Input modules working with FSM.	Write and test touch panel interface. Assemble platform and test with an Arduino.
11/18-11/24	Test pose controller on platform and debug.	Test and debug manual mode. Begin writing ball position controller.	Help with pose controller.
11/25-12/2	Help or work on stretch goal.	Test and debug ball position controller.	Add feedback mode to control FSM and debug.
12/3-12/9	Final testing and debugging + stretch goals (if time)		