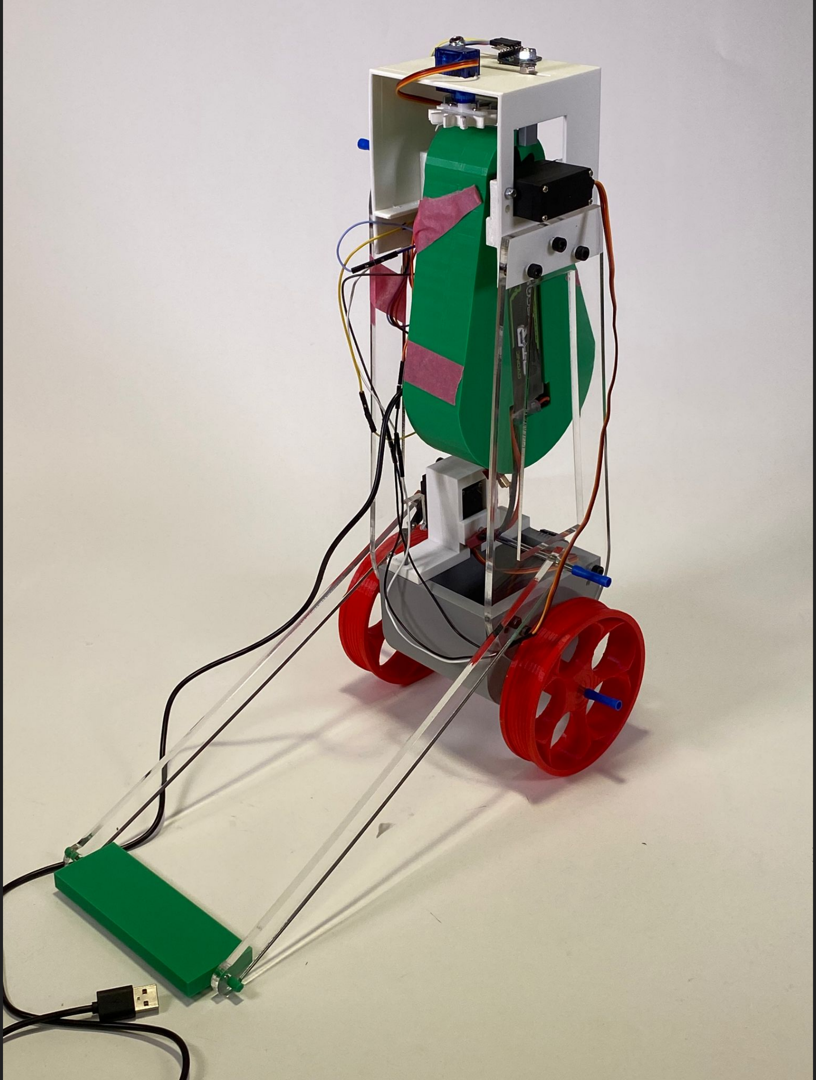
PendulumRoller

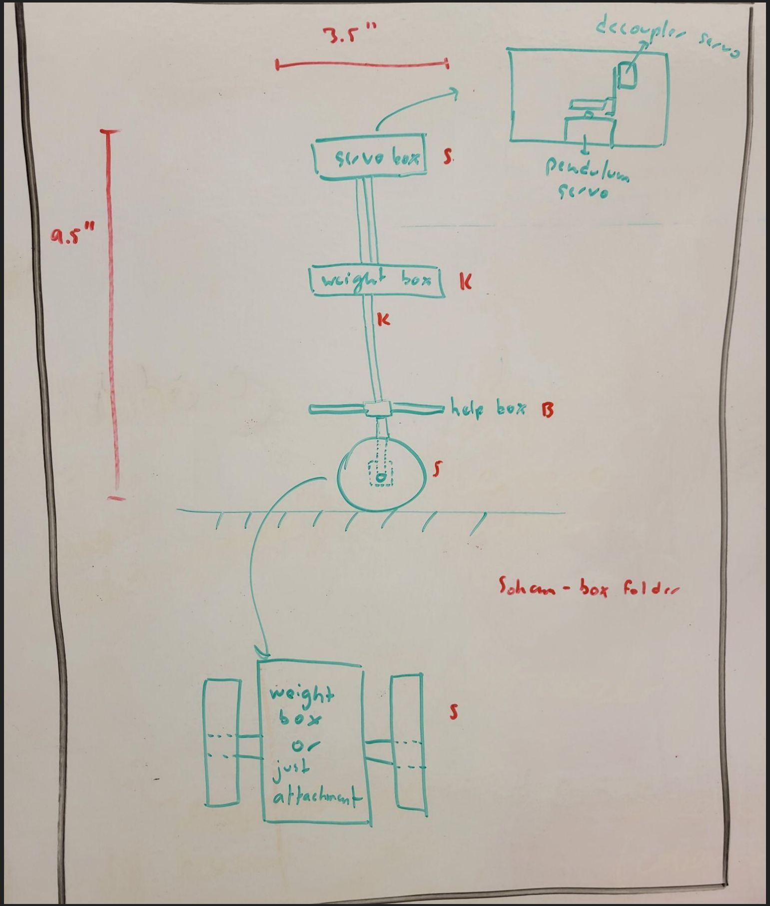


HW 4 Group 2

Kat Allen
Soham Gaggenapally
Benjamin McDermott

Initial Conception

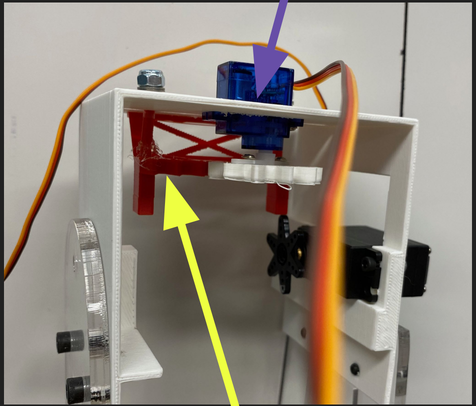
- Balancing Robot
- Raspberry Pi and battery included onboard near the servos for eas(ier) cable management
- Freewheel on the bottom
- Pendulum driven by servo motor controls balance and motion



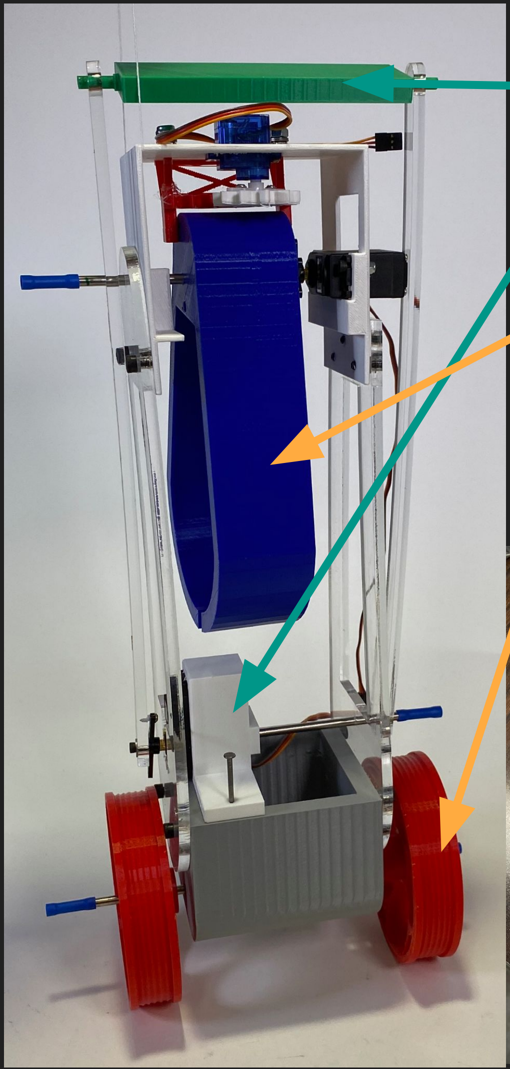
Ben

Creativity and Innovation

Vertical Pendulum Decoupler reduces width of servo box to allow 90° fall



Alignment Support simplifies rack and pinion alignment

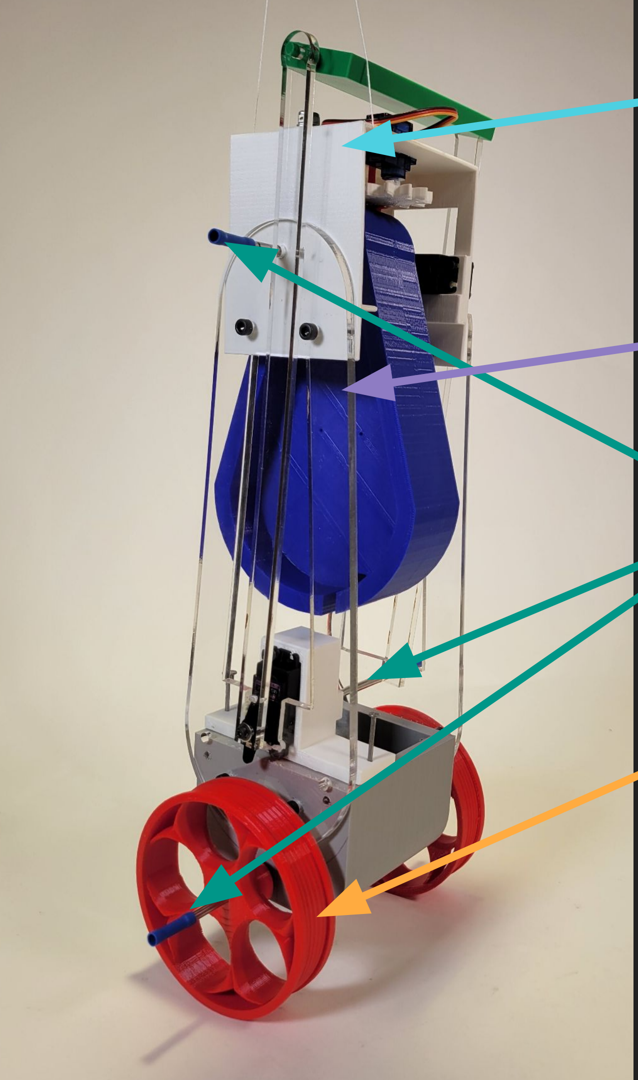


Lifting mechanism allows fall recovery

Pendulum control and free-spinning wheels eliminate the need for 360° servomotors

Onboard storage for Pi, ESP32 and battery helps manage cables and allows them to double as pendulum mass

Kat



Servo Box
Holds pendulum support, controller and decoupling mechanism

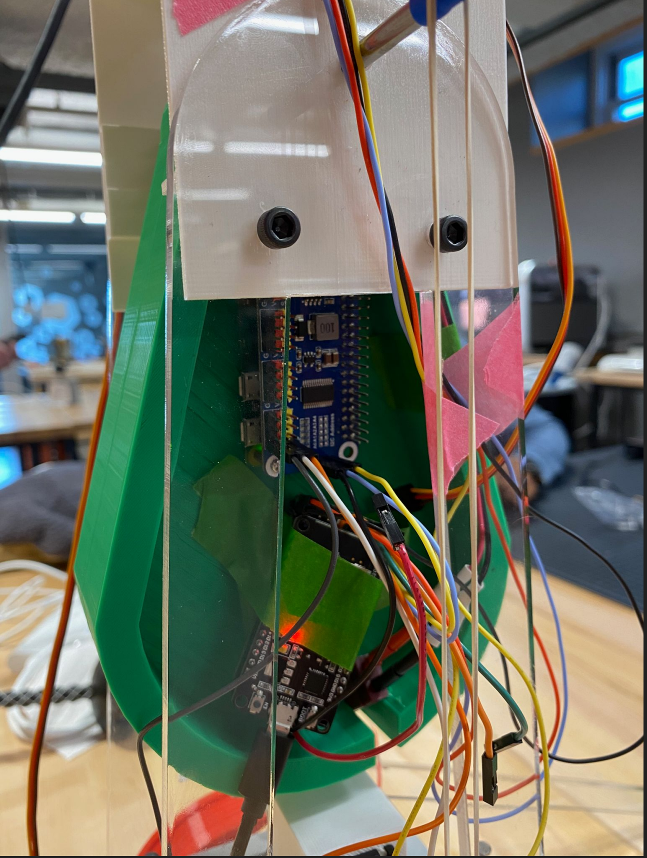
Pendulum
Provides control via rotation to balance, holds electronics and power

Metal Axels allow low-friction rotation for pendulum, wheels, and lifter assembly

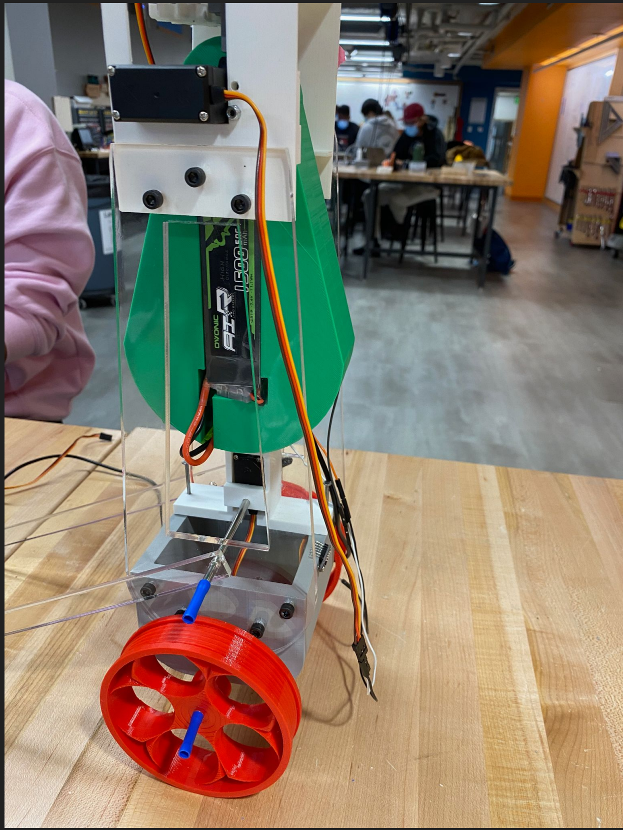
Wide Freewheel Base
Allows free rotation in x-direction, restricts motion in y-direction

Main Assembly

Kat

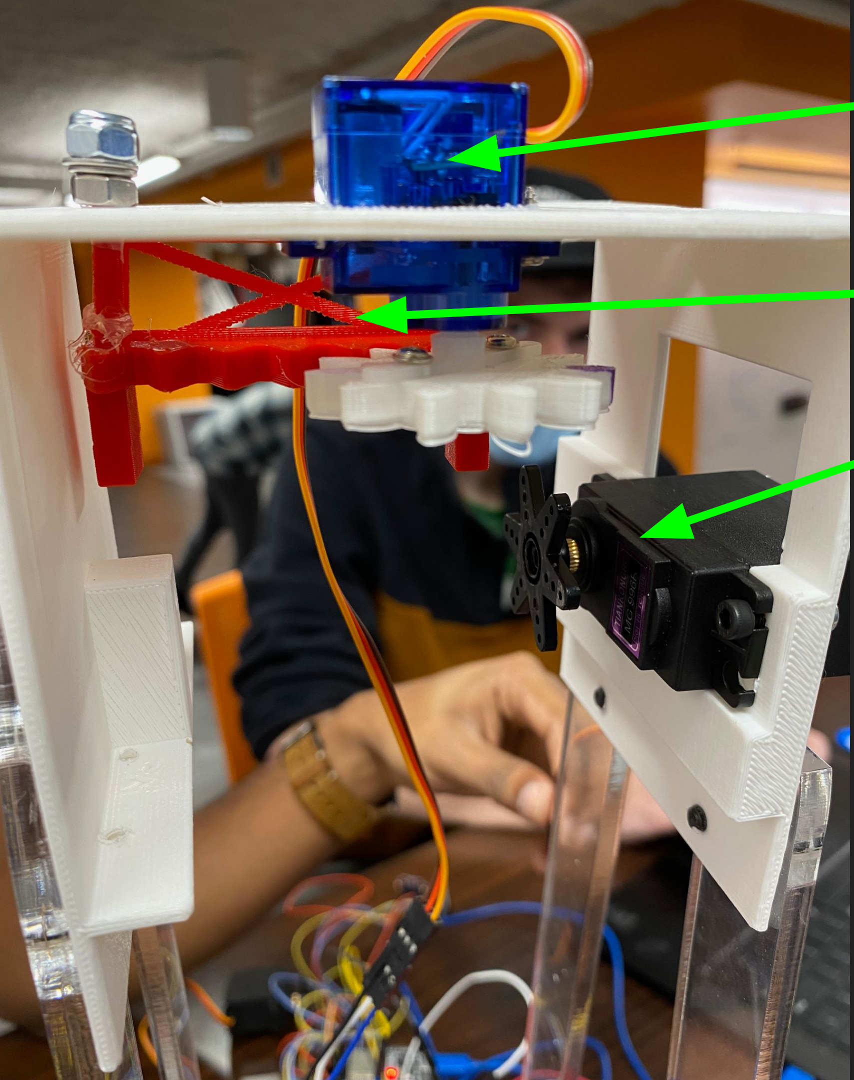


- Storage for Raspberry Pi and ESP32
- Storage for Battery
- Decoupleable connection to servo horn
- Routing for power through the part
- Weight distributed evenly around the centerline for easier mechanics
- Through hole allows axel to support weight during operation and when decoupled



Pendulum Design

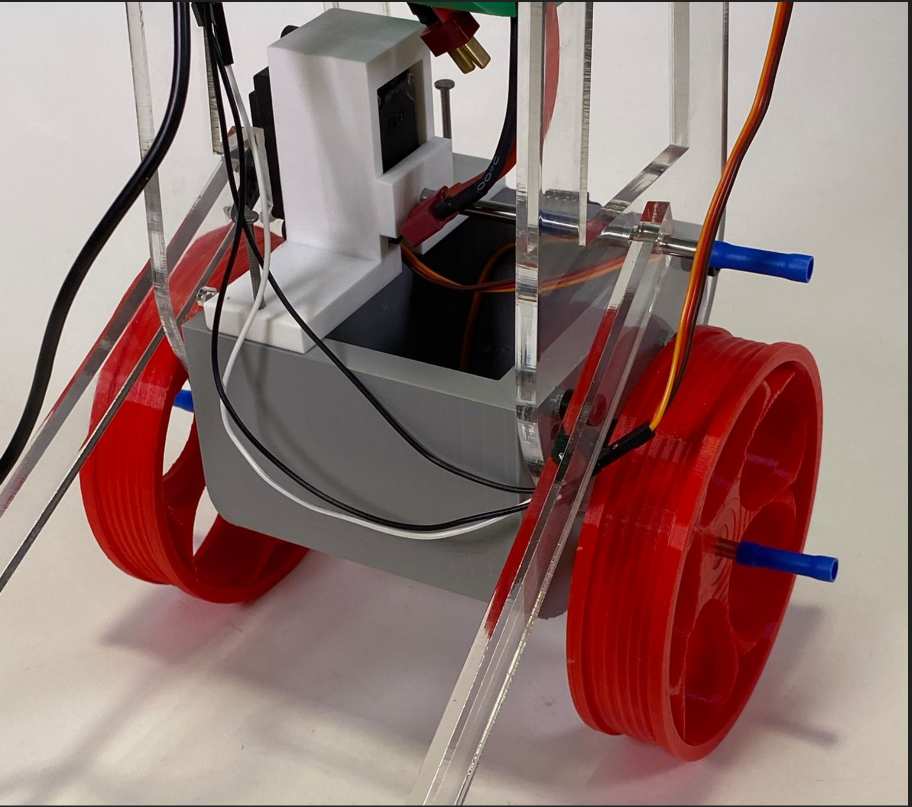
Kat



- Small servo drives the decoupling mechanism, prevents damage to servo when fully falling down
- Rack is designed to counteract moments with just one slot
- Large servo drives the pendulum and provides control input
- Vertical, narrow configuration allows robot to fall freely to ±90°

Servo Box and Decoupling Mechanism

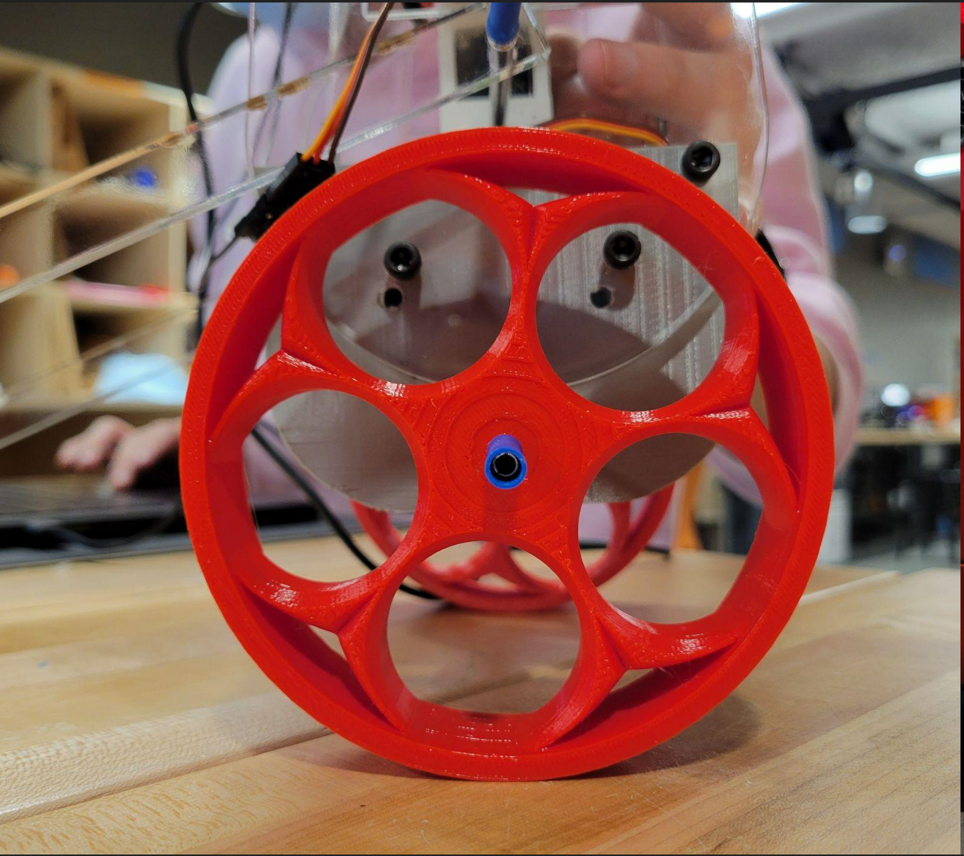
Soham



- Freewheel does not require any 360º servo motion or motors
- 3D printed body can be self-tapping, eliminating the need for loose hardware inside the box
- Wide wheel base helps prevent uncontrolled Y-axis motion
- Narrow, rounded box in the X-axis allows robot to fall to ±90º when unpowered

Wheel Box

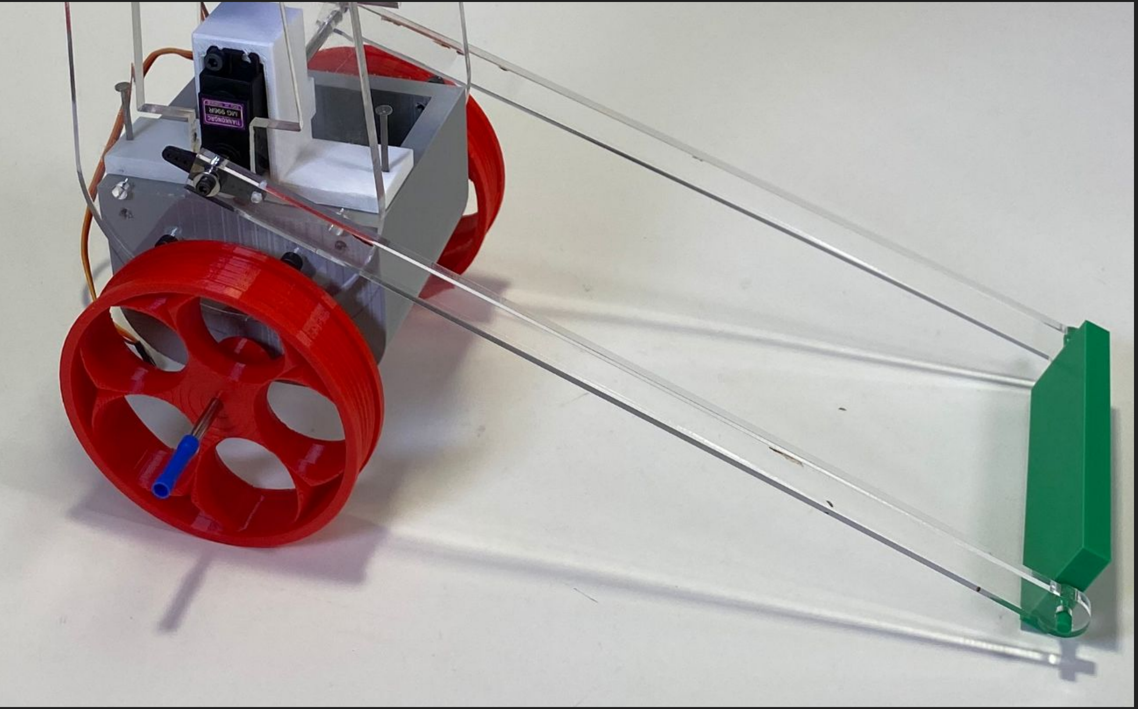
Soham




Wheels

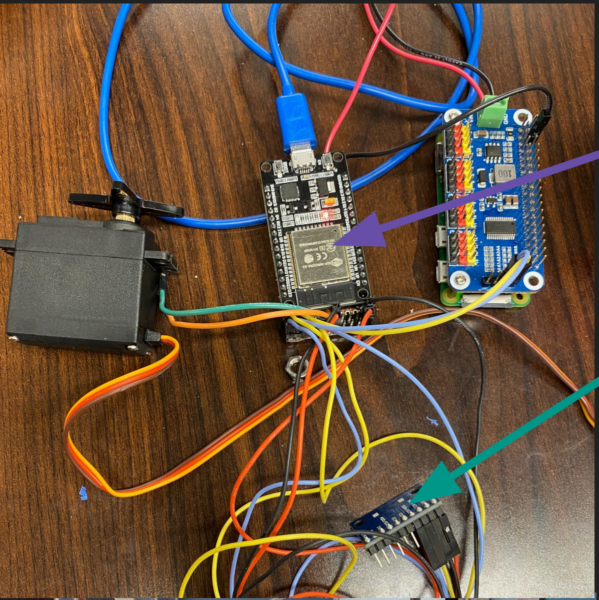
Soham

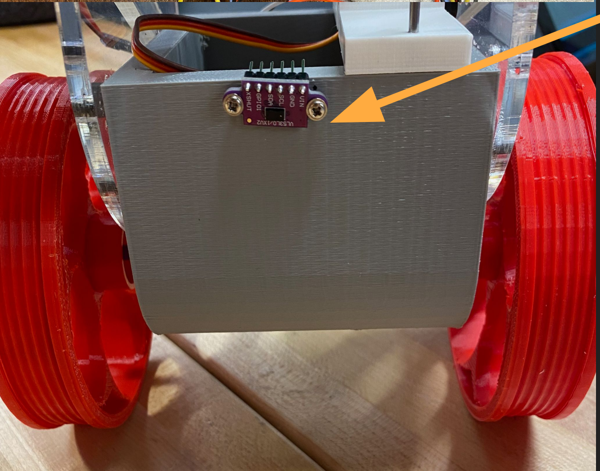
Lifter Arm



- Designed to lift the robot from a ±90° fall
- Works as a “kickstand” to allow the robot to balance when unpowered
- Second lifter servo would provide more power to lift the robot fully vertical

Soham

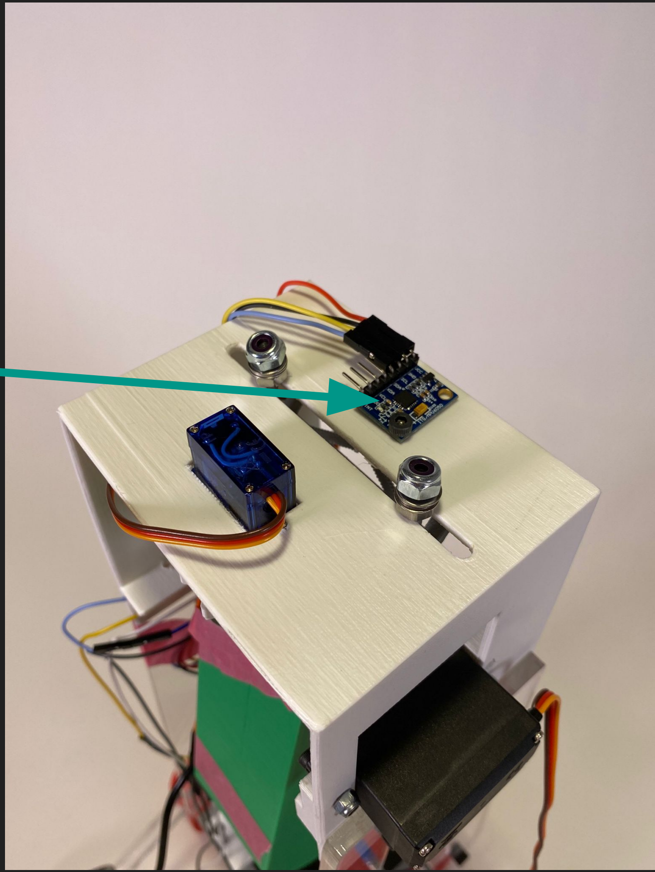




ESP32 provides rapid feedback control and sensing from IMU and LIDAR

IMU - MPU 6050
provides position & acceleration data for the robot body

LIDAR provides distance measurement and sensor fusion data during fall recovery



Control

Ben

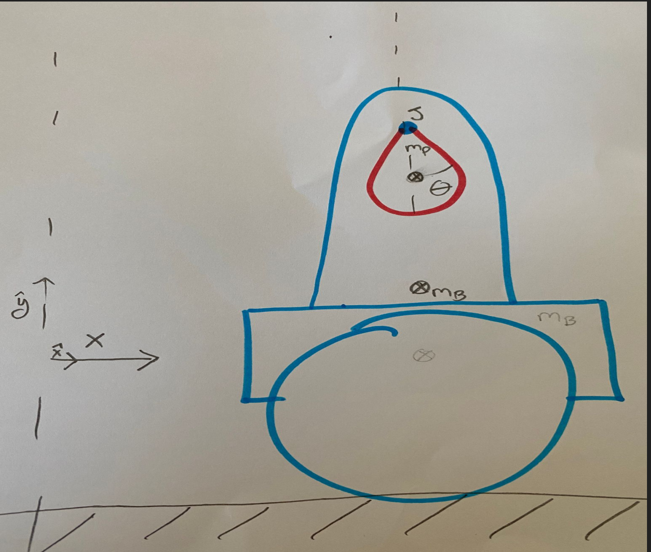
Kat

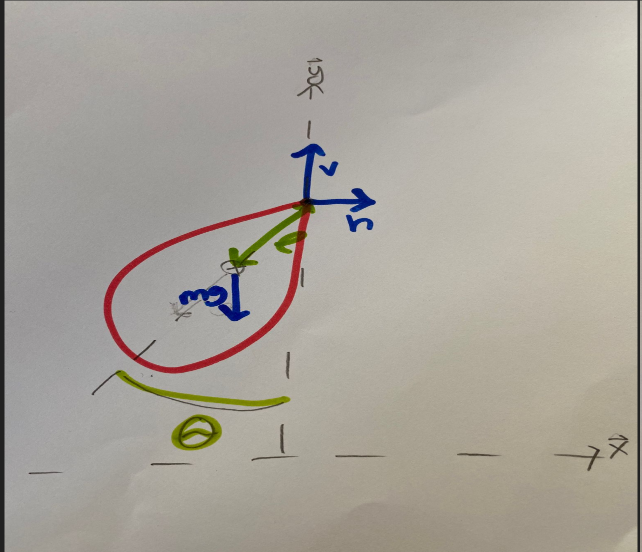
Theoretical Control:
Free Body Diagram and Equations of Motion

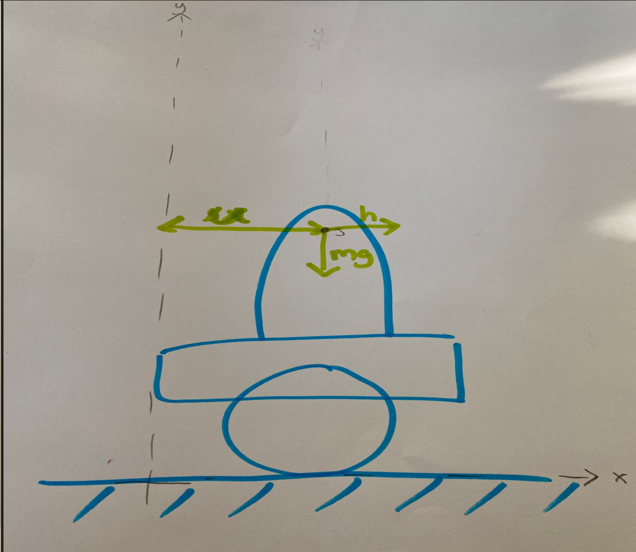
$(M+m)x'' + ml\Theta'' = u$
$Mlx'' + (I+ml^2)\Theta'' - mgl\Theta = 0$
$u = \cos(alpha) / height\_robot$

$\Theta\_p = (-m\_p * I\_p * \Theta'' * + (M\_total*I\_p - M\_total*I\_p) * \Theta'' + math.cos(alpha) * h\_robot) / (-g * M\_total)$

- Similar to cart and inverted pendulum
- Model pendulum as 30º circle segment for moment of inertia calculations
- Small angle assumption for pendulum control







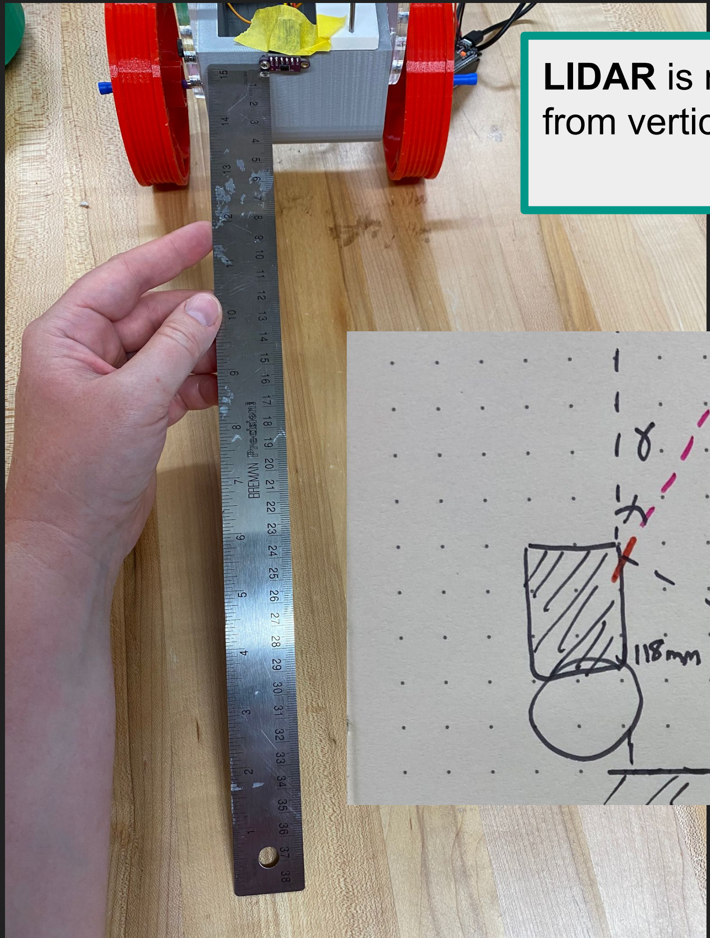
Implemented Control System / Code

PID controller with data from the IMU

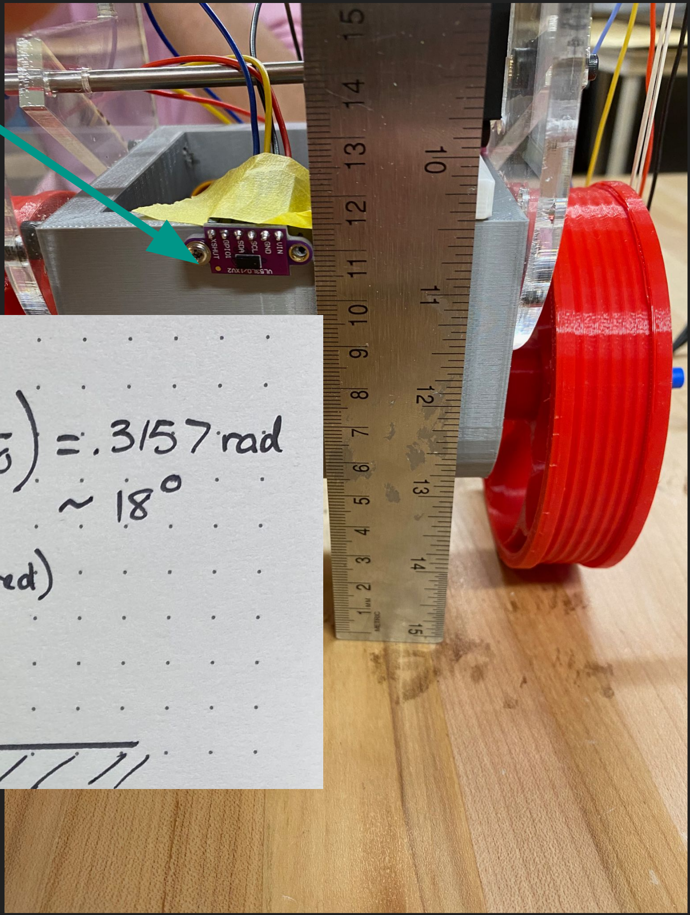
```
/////////////////////////PROPORTIONAL ERROR//////////////
float pid_p = kp*error;
/////////////////////////INTERGRAL ERROR/////////////////
float pid_i = pid_i+(ki*error);
/////////////////////////DIFFERENTIAL ERROR//////////////
float pid_d = kd*((error - previous_error)/elapsedTime);
/////////////////////////TOTAL PID VALUE/////////////////
PID = pid_p + pid_d + pid_i;
// Calculating the pulse we need to send to get the desired angle
float theta_p = (zero + (PID * ratio));
faboPWM.set_channel_value(15, theta_p);
```

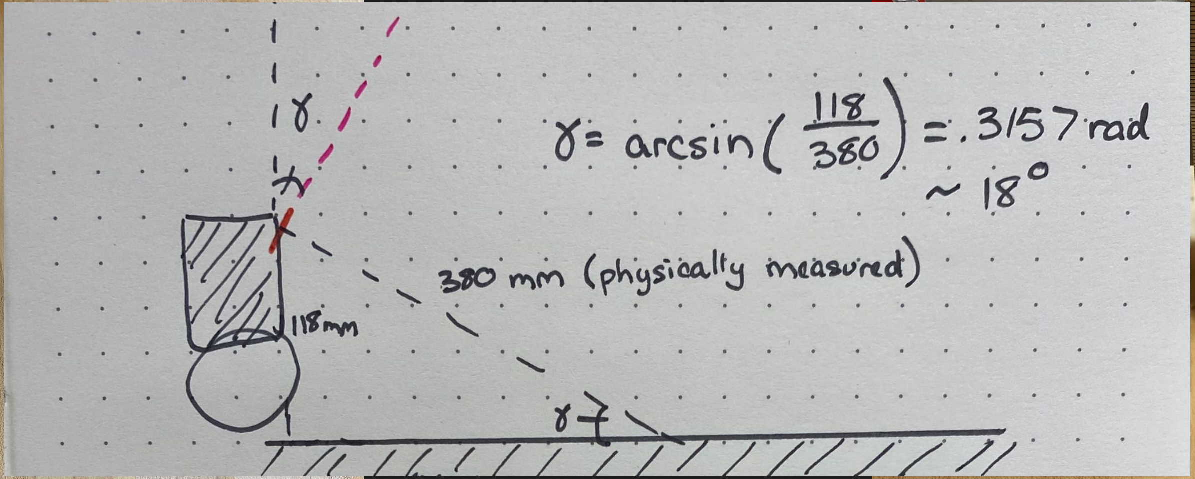
Ben

LIDAR vs IMU - measuring robot body angle

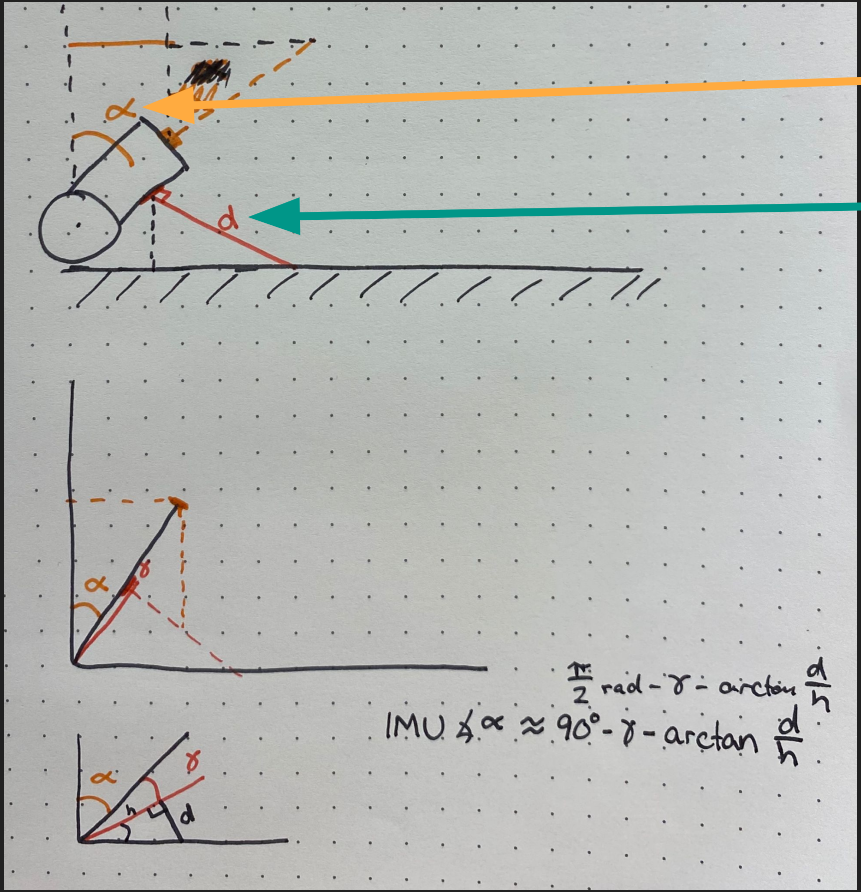


LIDAR is mounted approximately 18º from vertical on the wheel base front



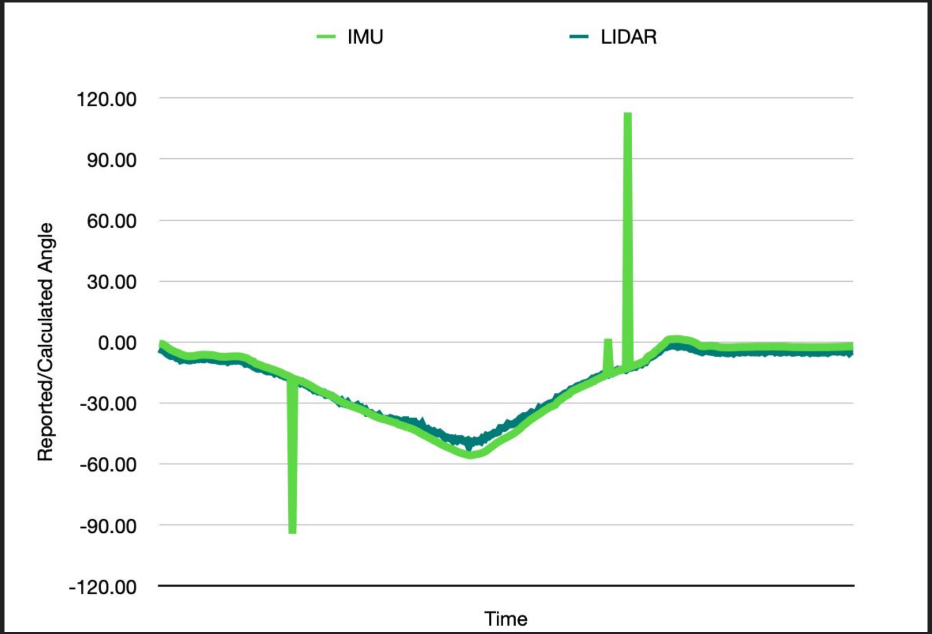


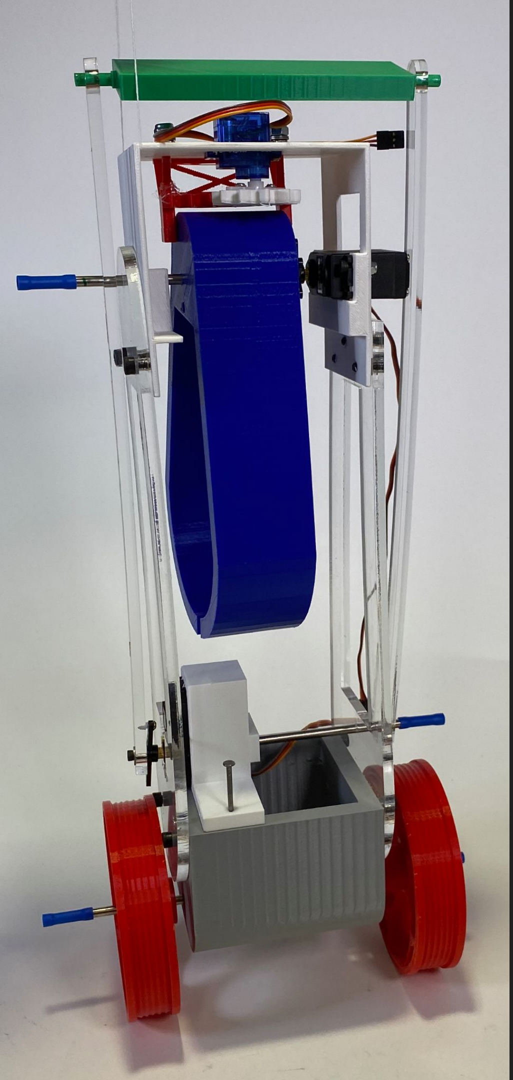
LIDAR vs IMU - measuring robot body angle



***α* = IMU angle measurement**

d = LIDAR measurement





Thank You