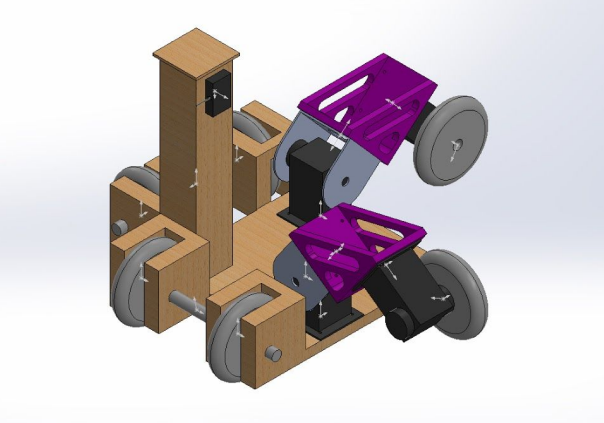
Winter 2020 P1 Purple Team

Ragav Subramanian, Madi Devore, Christian Soto, Matt Maurer, David Marsh

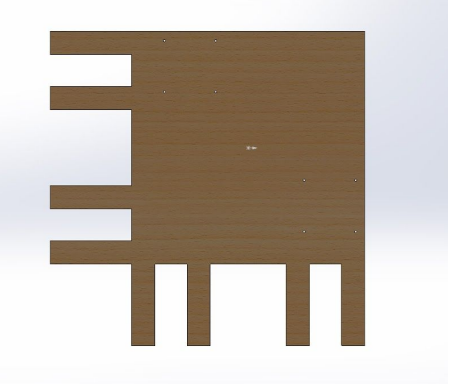


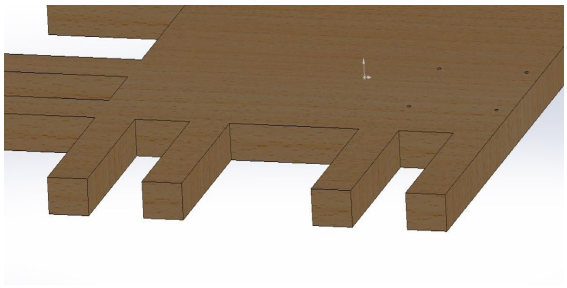
Robot Assembly:

Base

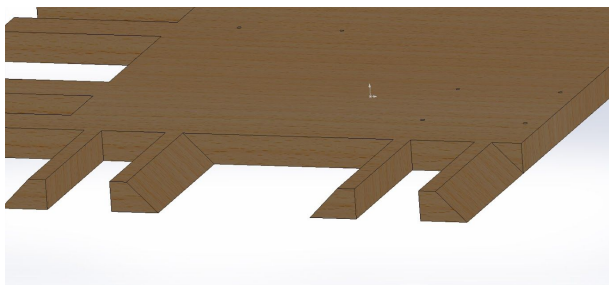
Laser cut the base from Base1.DWG out of ½” foam core. Line up two tailplates and two plastic spacers over the pilot holes in the base. Fasten with two drywall screws and a wood spacer for each tailplate.

By hand, cut a 45 degree bevel from each side of the 4 forks protruding from the base. The cut should go from the inside top edge of the fork to the outside bottom edge.





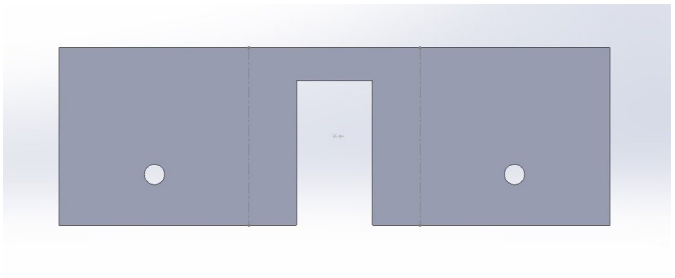
Before Cut



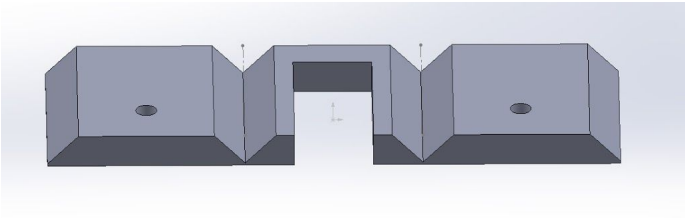
After 45° Cut

Passive Wheels

Wheel Wells (4):



The wheel wells are laser cut with the base in Base1.DWG. By hand, cut out a channel on either side of the partial cuts so that the piece can fold 90 degrees. The cut should be on the inside of the fold.
Hand cut a 45 degree bevel into the bottom of each side of the wheel well so that it will sit flush with the hand cut bevel on the base.
Using hot glue in the two seams, fold the two corners so that a 90 degree angle is formed.
Add tape around corners to strengthen the seam.

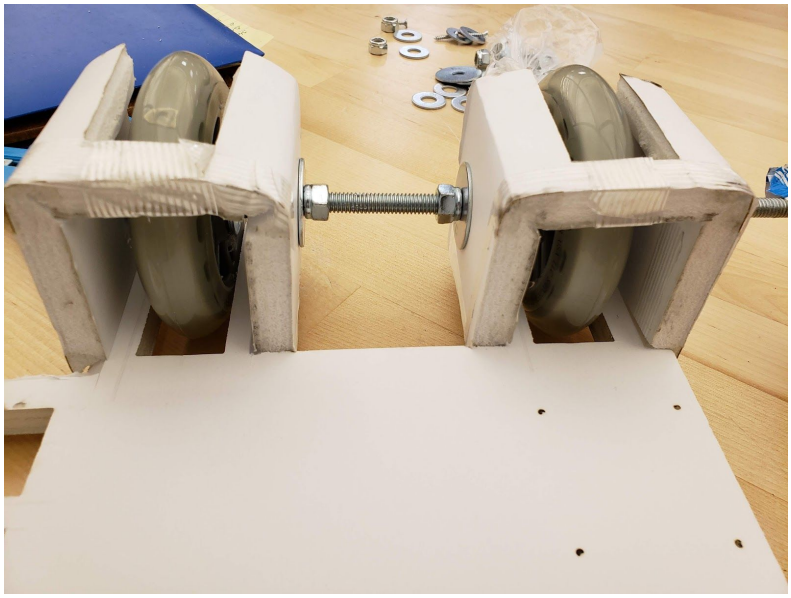


Repeat for all 4 wheel wells.

Axle Assembly (2):

Take a 5/16” threaded rod and add components in this order from left to right:

1. Wheel 1
    a. 5/16” lock nut
    b. 5/16” fender washer
    c. left side of wheel well 1
    d. 5/16” lock nut
    e. 5/16” washer
    f. razor scooter wheel
    g. 5/16” Washer
    h. 5/16” lock nut
    i. right side of wheel well 1
    j. 5/16” fender washer
    k. 5/16” lock nut



2. Wheel 2
    a. 5/16” lock nut
    b. 5/16” fender washer
    c. left side of wheel well 2
    d. 5/16” lock nut
    e. 5/16” washer
    f. razor scooter wheel
    g. 5/16” Washer
    h. 5/16” lock nut
    i. right side of wheel well 2
    j. 5/16” fender washer
    k. 5/16” lock nut

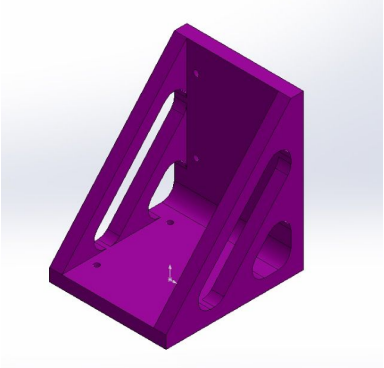
This is a lengthy process because all lock nuts need to be threaded down the rod. The fender washers should gently press on the outside of the foamcore. The lock nuts inside the wheel wells are for aligning the wheels.

Continue threading the lock nuts until the wheel wells line up with the forks in the base and the wheels are centered in the wheel wells.

Attach the wheel wells to the base with tape and hot glue.

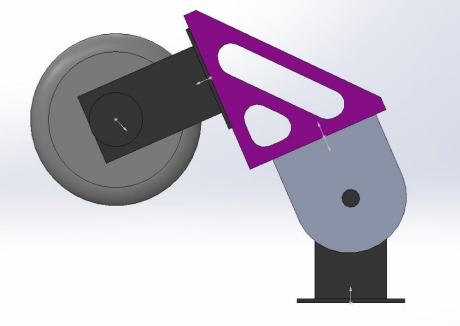
Repeat for the second axle assembly.

Drive wheels (2):



Attach the rotating tailplate of a cr dynamixel module to a large wooden wheel using a modlock. <ADD MORE DETAIL>

3D print RightAngle3.stl Pla with around a 20% infil. This part is used to attach the servo motor and the rotational motor together, but it allows the drive arm to lift the edge of the robot so only the wheels facing the intended direction of motion touch the surface.

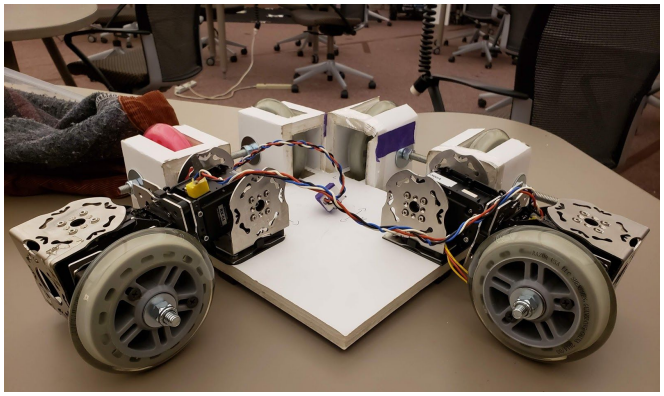


Attach the bottom tailplate of the cr dynamixel module to a tailplate on the 3d printed joint using a modlock.

Attach the bottom tailplate of a u-bar dynamixel module to the other tailplate on the 3d printed joint using a modlock.

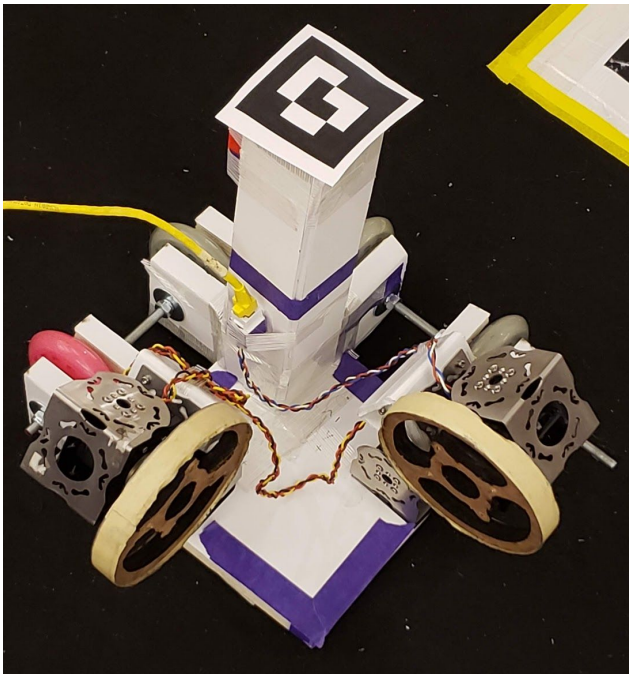
Attach the top of the u bracket to a tailplate on the base using a modlock. The 3d printed joint is mounted correctly if the wheel touches the ground before the motor.

Repeat for the second drive wheel. Link all the motors together with cable jumpers.



Tower:

Cut a 2x10cm strip of foamcore and tape it in the corner created by the two middle wheel wells. This is a spacer to prevent the wheels from touching the tower.
Build a 5x5x30cm square prism out of ⅜” foamcore. Tape the prism to the spacer and to the base.
Lower the wheel that moves the robot in the direction of the whiteboard to its drive position. Use a level to mount the laser so that the beam is level when the wheel in the direction of the whiteboard is engaged.
Connect the laser to a motor with the cable jumpers.
Tape the robot april tag to the top of the tower. Tape the cable with the rj45 connector to the middle of the tower.



Software

The software is broken up into four parts: scorpionMasterControl.py, scorpionPlans.py, scorpionReIX.py, and scorpionSimIX.py. The script scorpionMasterControl.py is the main which calls upon functions from the files. The scorpionReIX.py and scorpionSimIX.py scripts are identical at a higher level but implementation-wise, one affects the real-world robot and the other the simulated robot respectively. The scorpionPlans.py script has the Move and Autonomous plans that work for both scorpionReIX.py scorpionSimIX.py. In essence, we call these high level functions that allow us to abstract away the autonomous and manual specific implementation.

The simulated robot can move in both directions in pure vertical and pure horizontal directions. This is to reflect the physical design of our robot which moved in this way except for slight turns caused by noise.

Our autonomous algorithm moves in a spiral until the robot hits the next waypoint. Once it hits the waypoint it takes the x and y distance from the first and next waypoints and applies an equivalent amount of movement to the robot in the direction from the first waypoint to the next. If it hits the waypoint then the current waypoint is updated and it spirals again repeating the process but trying to find the current next waypoint. If it misses the waypoint the robot will move in a spiral until it hits it.