LIDAR Concept of Operation

ANATOMY
Lidar systems have a rotating sensor collecting multiple measurements to measure in a 2D plane. (Some have 3D, by other methods).

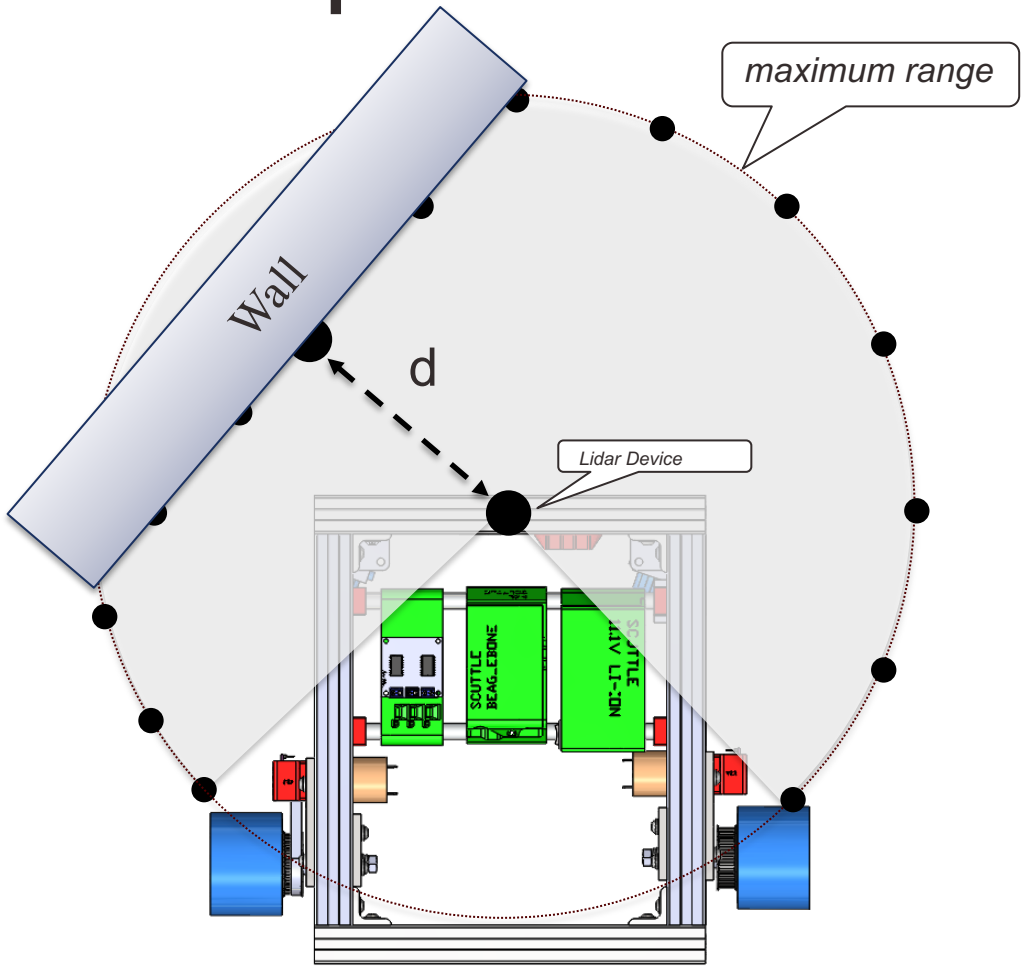
METHOD
Lidar emits a beam of light and receives the reflection. distance is based on Time of Flight concept.

POWER
TiM561 uses about 2.1 watts during operation, mainly due to driving the motor and driving a strong IR emitter diode.

FAILURE MODES
Just like light, a Lidar beam can be absorbed by very dark objects and can be mis-directed by highly reflective objects which are non-perpendicular to the beam.

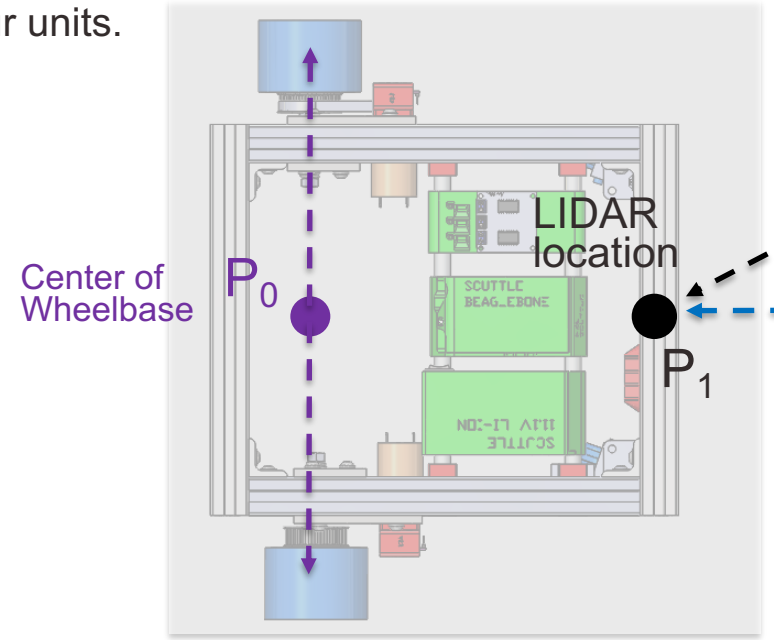
DATA QUALITY
The lidar has *variable resolution* in a sense! 0.33 degrees offers 5mm point spacing at a 1m distance, and at 10 meters, 50mm point spacing.

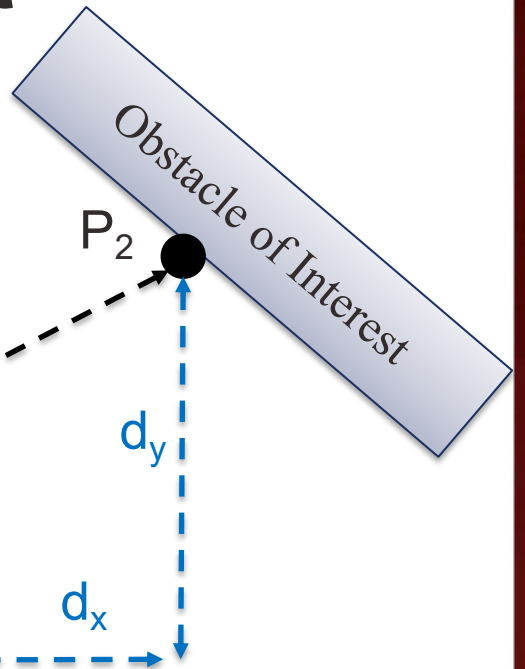
APPROPRIATE USE
To be successful in using the device, you need to see the datasheet to understand the parameters of your device.



LIDAR – measuring a point

- P_1 is the location of the lidar.
- The points will be initially measured from lidar and returned as pairs given by:
- **[d (mm), α (degrees)]**
- Python's numpy library performs math in radians. It is easy to convert back and forth but you must be aware of your units.





Software For LIDAR

Key Points:

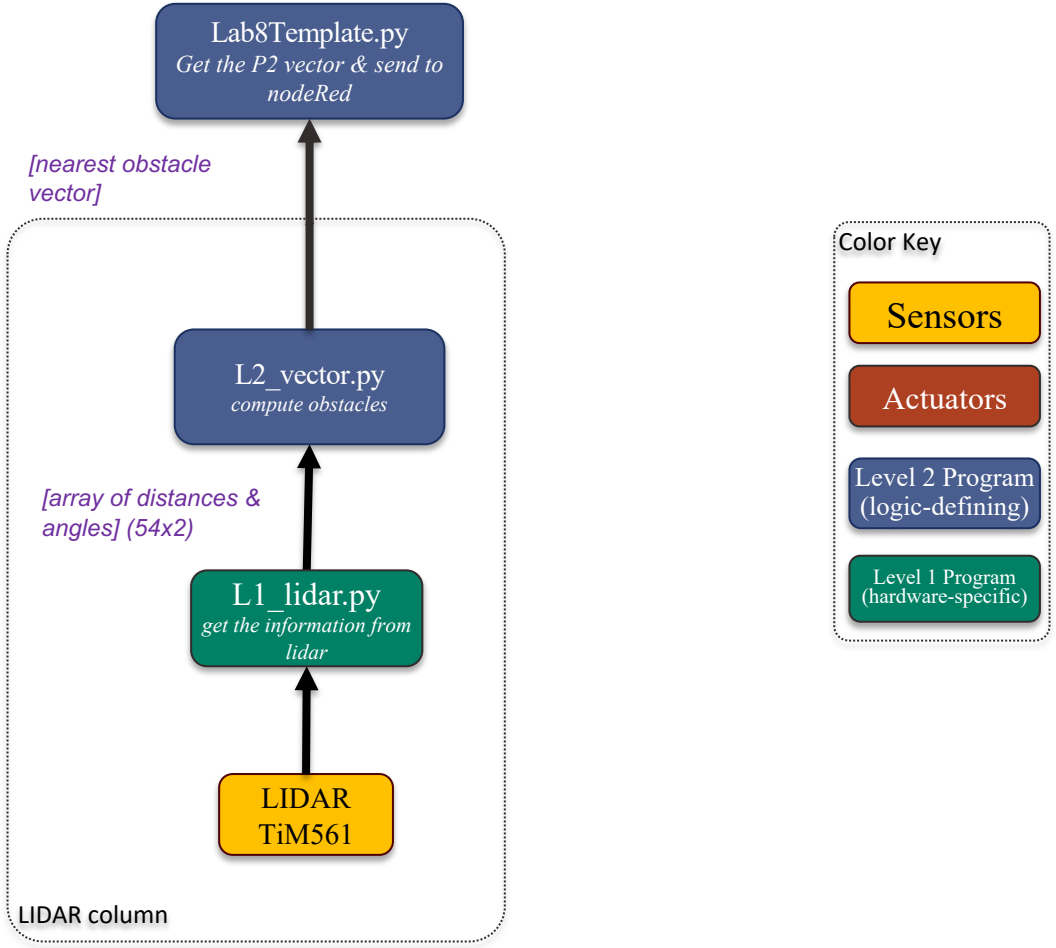
Software is using the numPy library to handle vectors and matrices. numPy computation is faster than raw python and requires proper syntax.

Lidar scan frequency: 15hz, so you cannot get new measurements faster than 66ms.

L1_lidar.py returns 54 measurements by default and can return over 800 single points if desired, for more resolution.

TiM561 LIDAR returns distances in meters. Distances under 16mm are returned as error codes in case of poor reflection or other problem for a given measurement.

L2_vector.py can manipulate measurements, with functions such as returning the nearest point, combining cartesian vectors, and converting vectors from polar to Cartesian coordinates.



Global Location of Obstacle

Determine the global location of an obstacle:

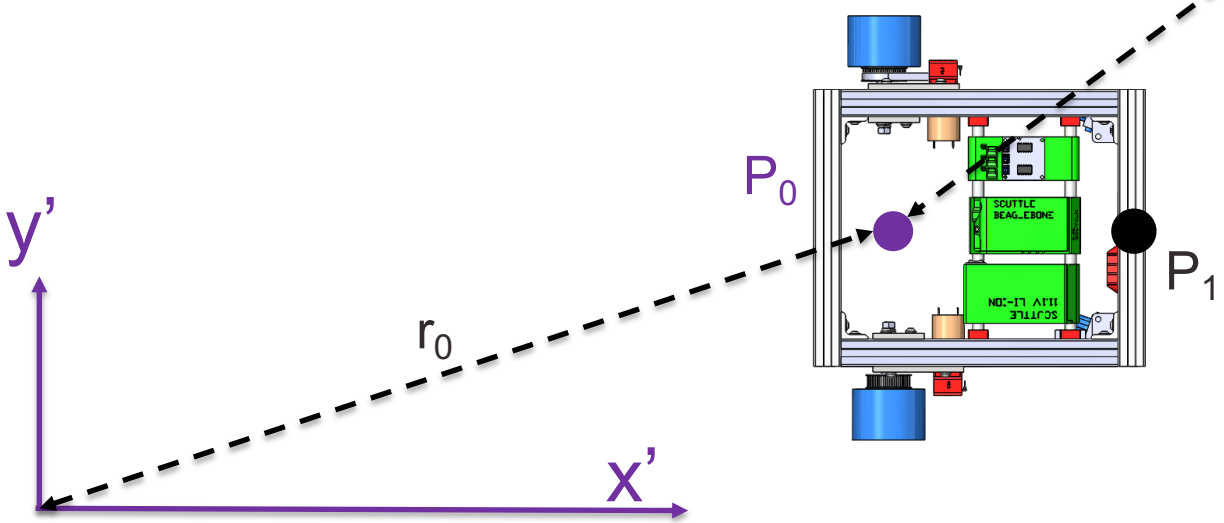
You must add vector r_0 and vector r_{02}

First, your vector r_{02} must be generated using knowledge of the location of the LIDAR on the robot.

Where is the obstacle located in the room?

r_{0-2}

vector r_{02} is not given automatically.



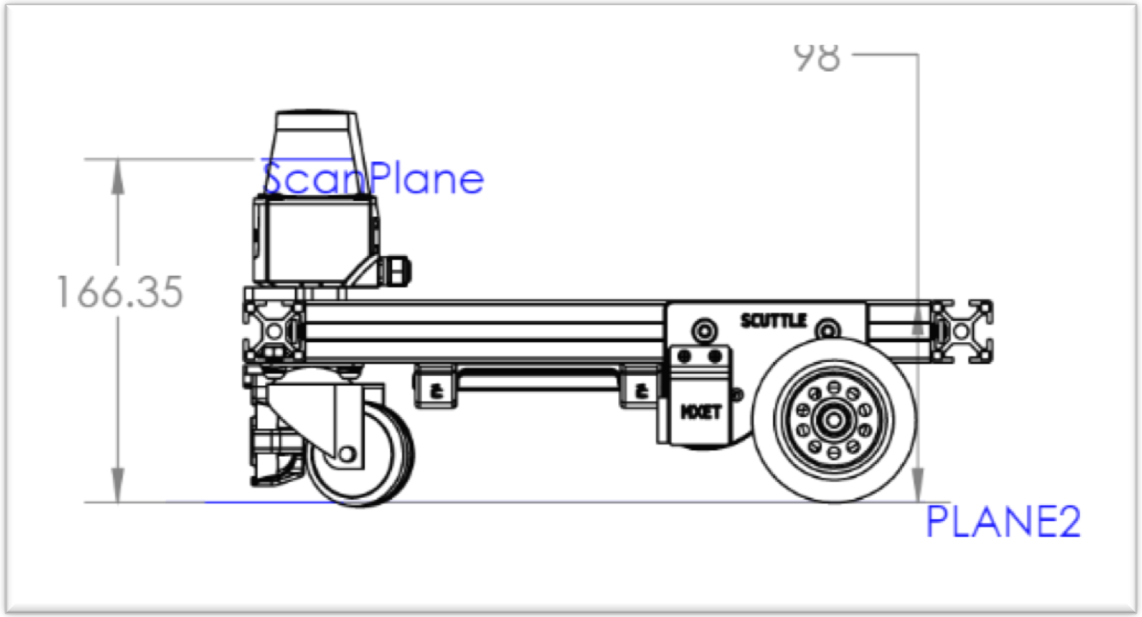
global coordinate frame

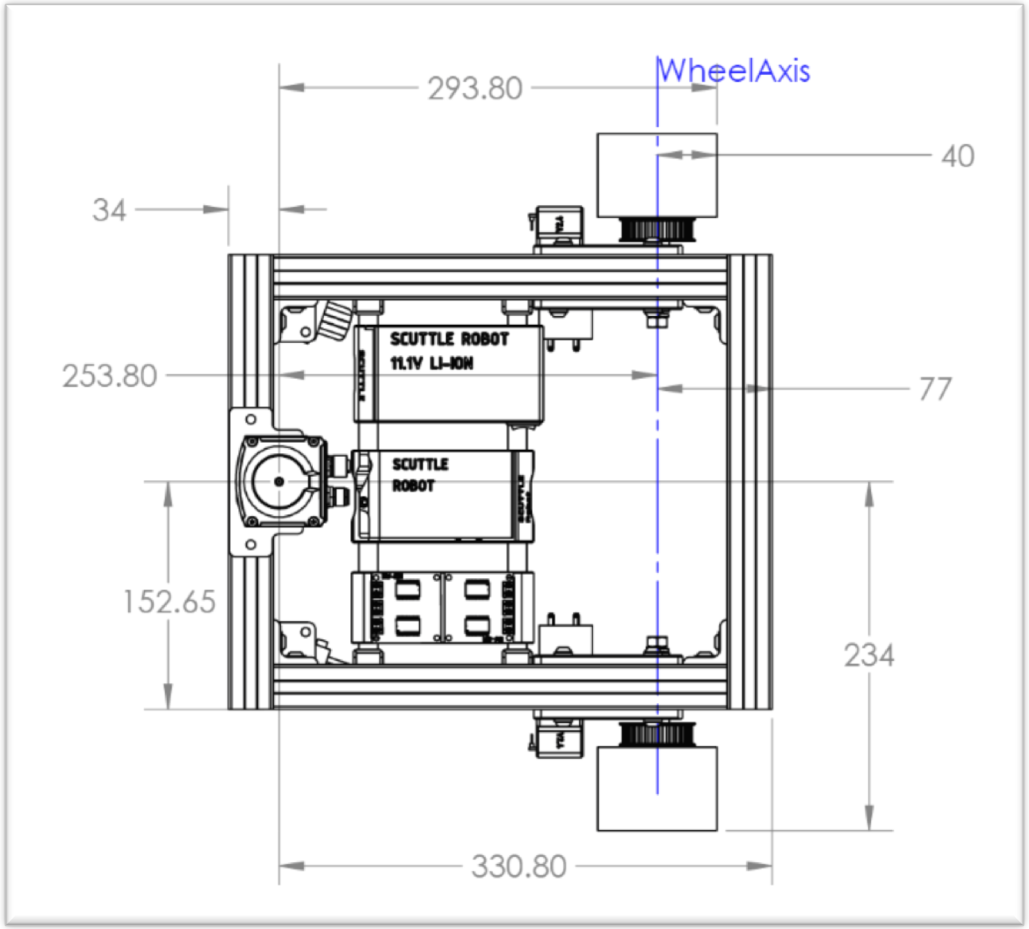
Global Location of Obstacle

Determine the global location of an obstacle:

Lidar is located at positive 254mm in the x-direction on the robot.

The lidar beam is 166 mm above the floor.



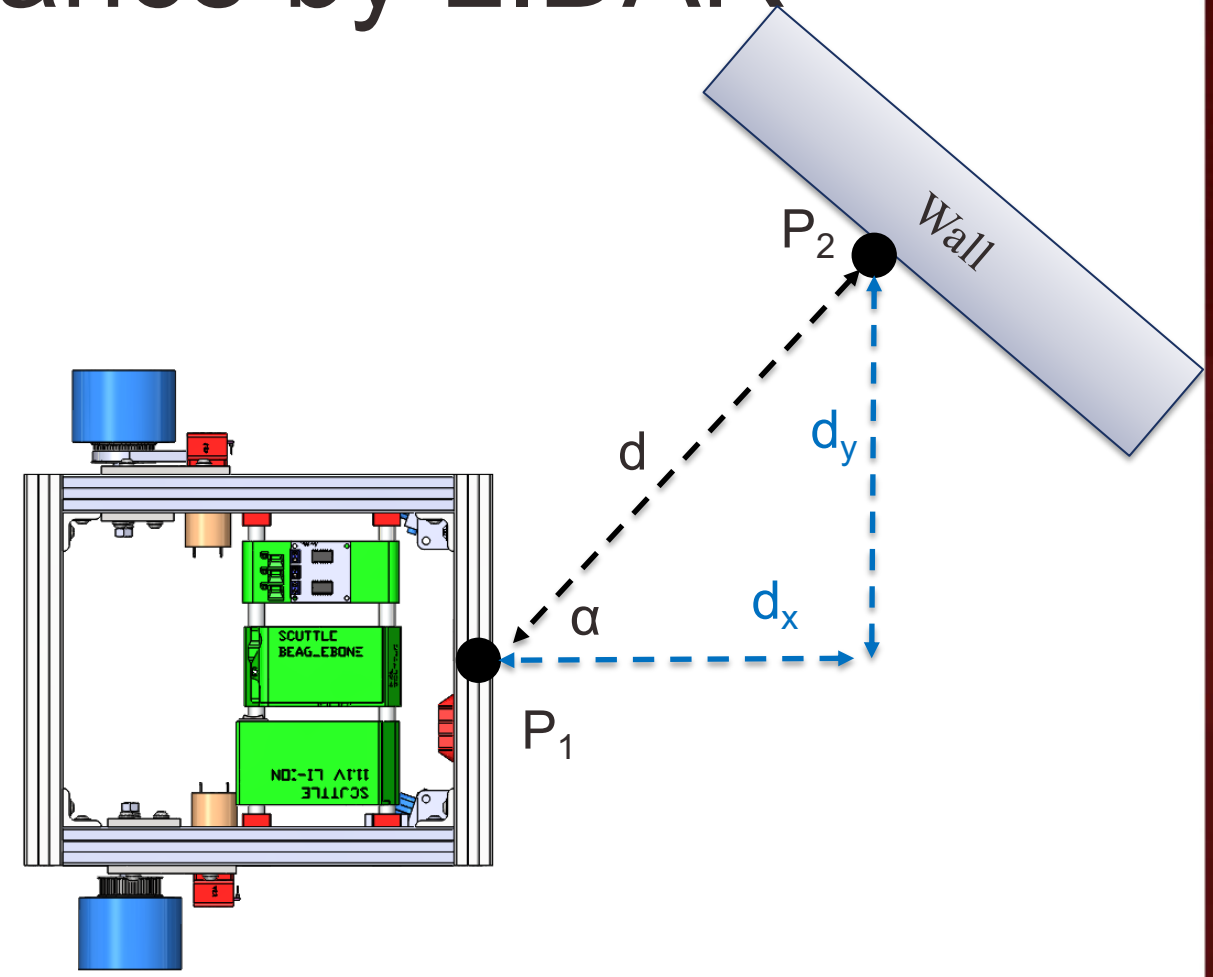


Obstacle Avoidance by LIDAR

One method to avoid obstacles is to generate an imaginary spring which pushes on your robot and depends on the nearest obstacle.

D_y is the y-component of distance d

D_x is the x-component of distance d



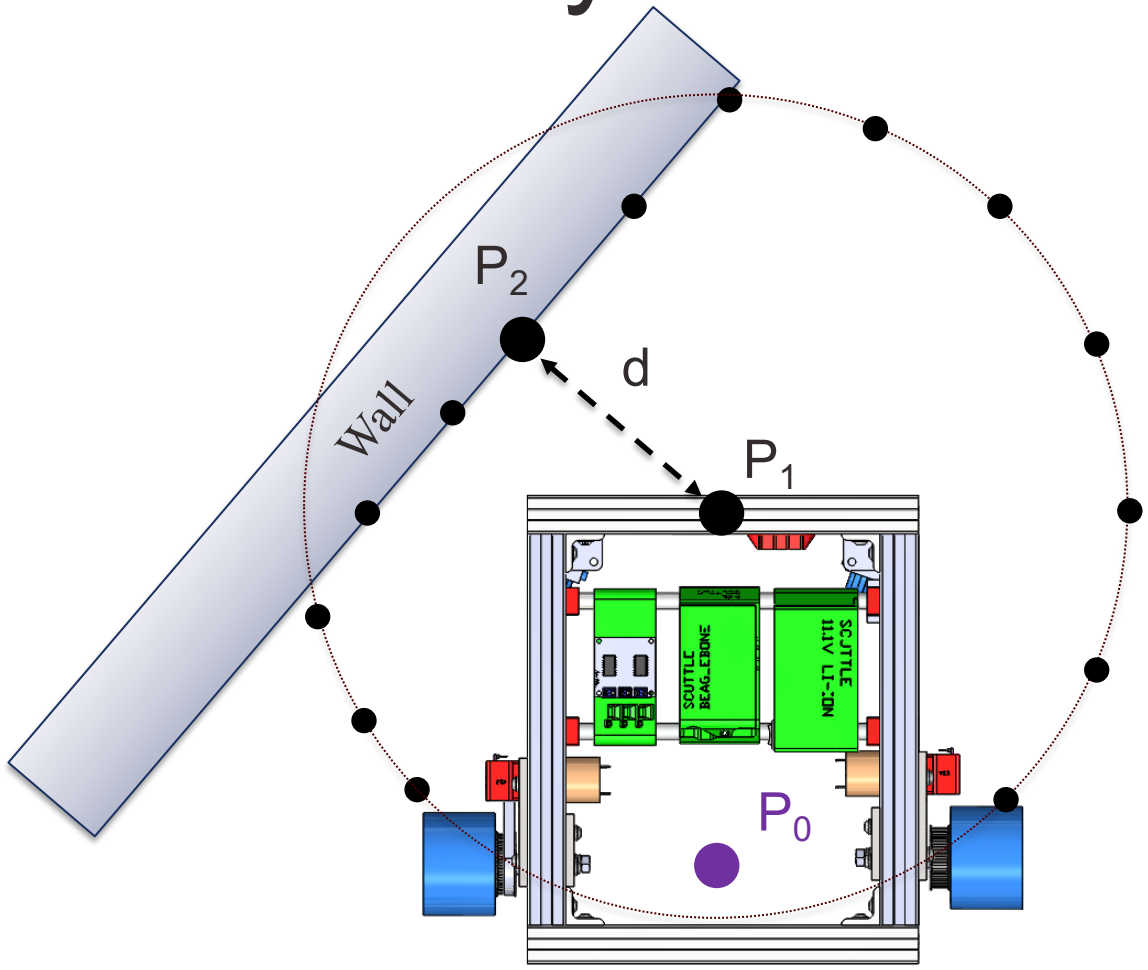
Obstacle Avoidance by LIDAR

Strategy:

The obstacle avoidance feature will try to detect the nearest objects to the robot and apply an “invisible force” to prevent the robot from crashing. The force is intended to act like a spring which is anchored to the nearest obstacle and pushes the robot at a point on the body, referred to as P1.

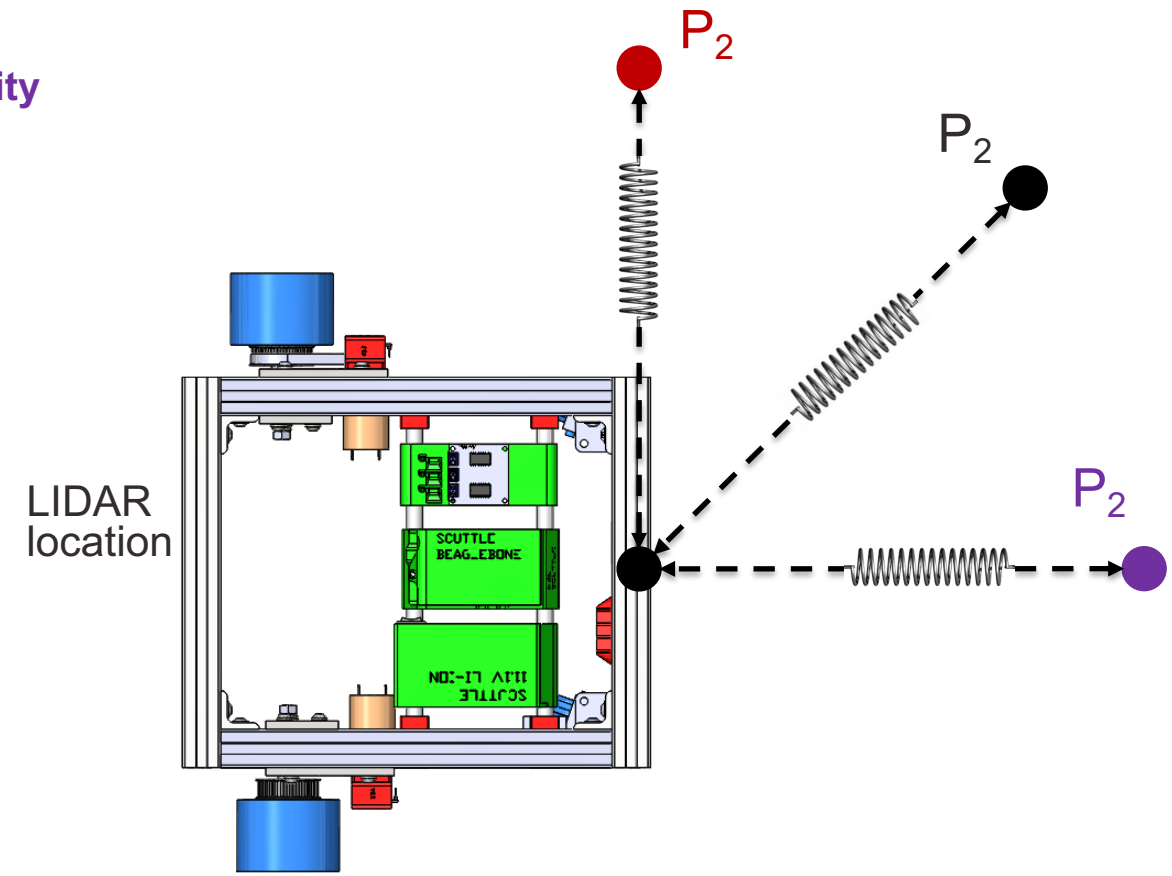
The obstacle avoidance only deals with the body-fixed frame

- Define P_1 as a point of interest on our robot.
- P_2 is assigned to the nearest point detected by the LIDAR scan.
- d is the distance between point 1 and point 2
- We would like to handle all of these variables in:
  - body-fixed frame
  - Cartesian coordinates



Obstacle Avoidance – influence on velocity (translational and angular)

- If P2 is detected straight ahead, the force influences **velocity**
- If P2 is detected at the side, the force influences **steering**
- If P2 is detected at an angle, the force will **influence both**



Obstacle Avoidance – Variable Force

- If d is large, the force is low
- If d is small, the force is high
- If d is larger than d_{max}, the force is absent