Controllers:

Bang Bang:

- PushController: Checks which of the front sensors' proximity is greater (farther away from the puck) and steers accordingly.
  E.g. if the proximity of the right sensor is greater, the robot turns left.

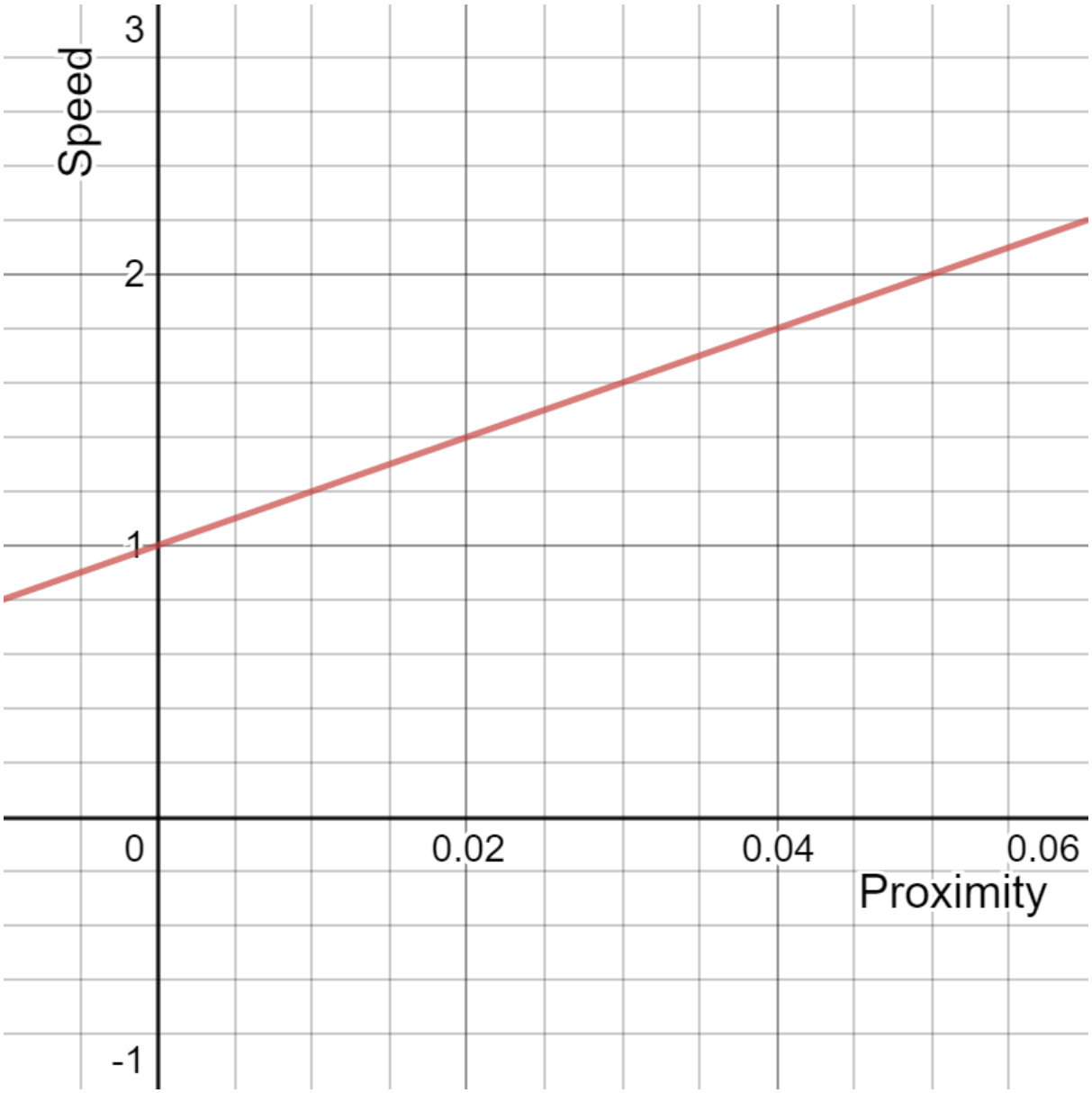
- DoorController: Calculates the center of the door relative to the center of the image and steers accordingly.
  E.g. if the center of the door is to the left of the center of the image, the robot turns left.

- DoorProxController: Same as the DoorController but stops if one of the two front sensors detect the door.

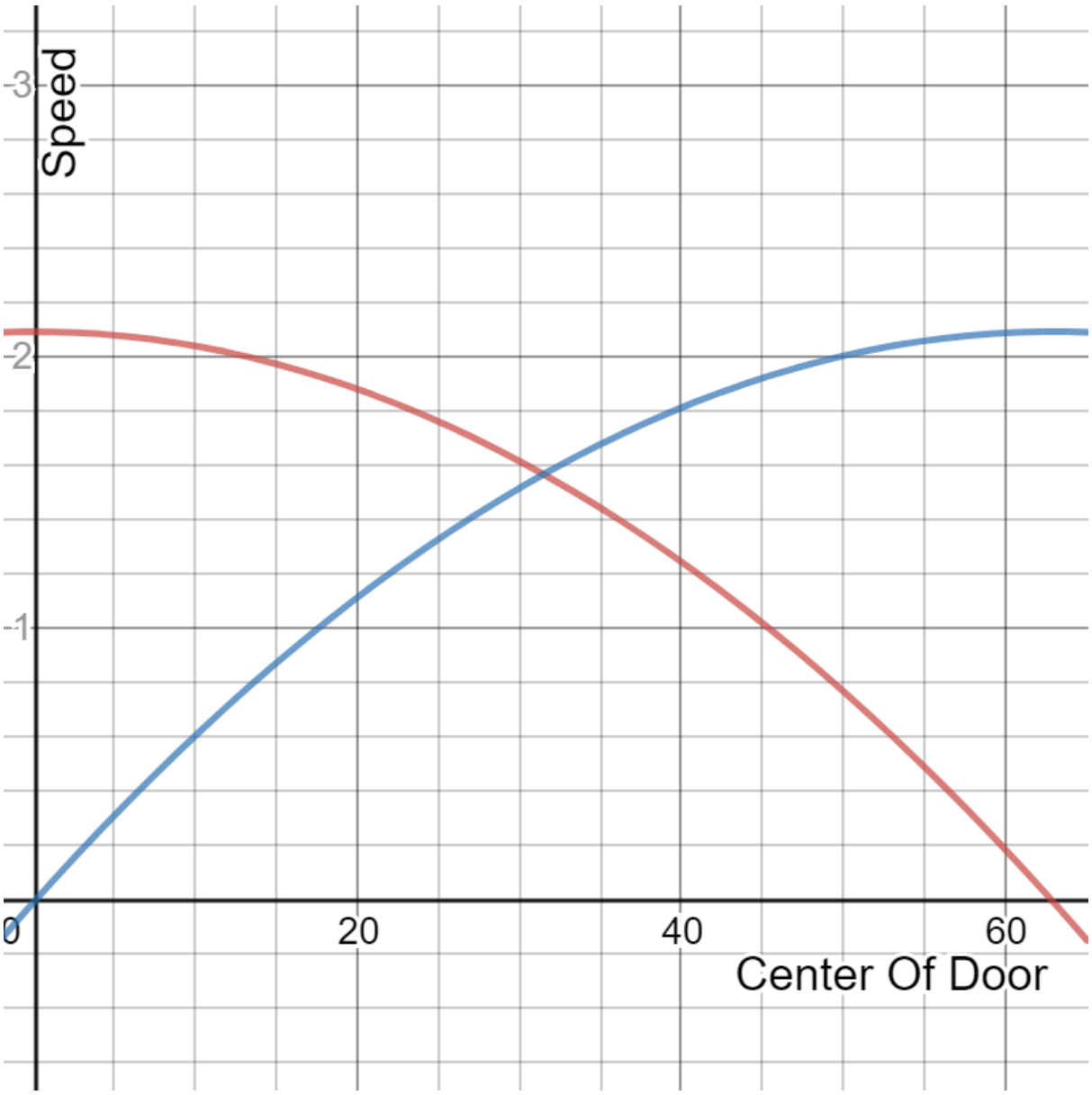
- WallFollowController: Drives straight ahead until a wall is found, then positions the robot with its left side facing towards the wall and turns either left or right if the robot gets too close or too far away from the wall. At corners, the robot stops and turns right until the correct position is reached again.

Proportional:

- PushController: Calculates the speeds of the wheels according to the front sensors' values.
  Formula: $speed = v \times 20 + 1$ (with v = value of right or left sensor)

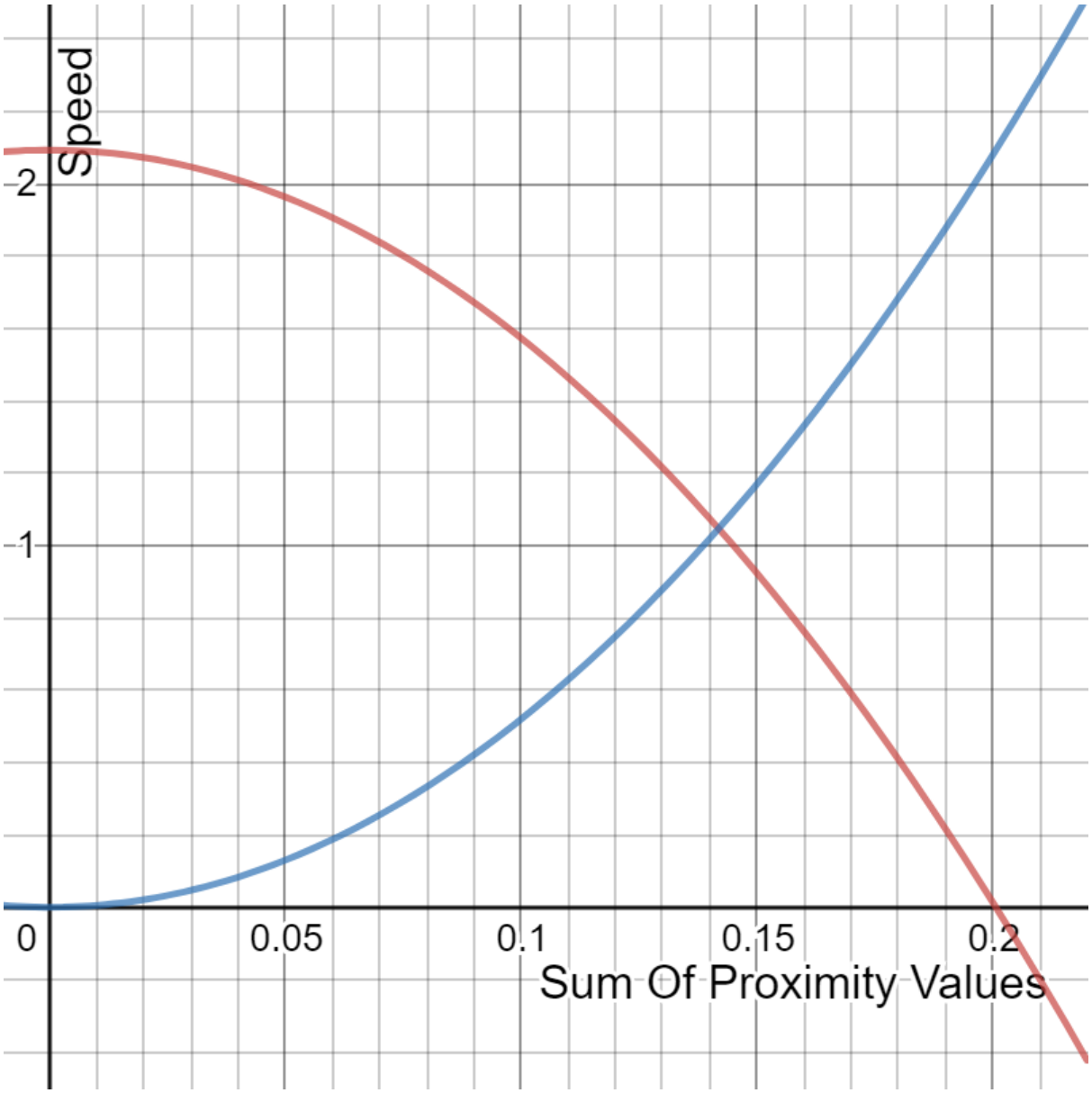


- DoorController: Calculates the center of the door relative to the image and sets the speeds of the wheels proportional to the center-position.
  Formula: $leftSpeed = -0.00053(center - 63)^2 + Max\_Speed$ (blue)
  $rightSpeed = -0.00053center^2 + Max\_Speed$ (red)



- DoorProxController: Same as the DoorController but uses a different formula as soon as the door is reached to stop at a proximity of 0.03.
  leftSpeed and rightSpeed as seen in the graph above.
  Formula: $leftSpeed = leftSpeed \times (vl - 0.03)$ (vl = value of left sensor)
  $rightSpeed = rightSpeed \times (vr - 0.03)$ (vr = value of right sensor)

- WallFollowController: This controller is only partially proportional and does not fully work.
  Same as the Bang Bang controller but uses a formula to keep the robot close to the wall. If the robot gets to far away it turns to the left; if it gets too close it steers to the right.
  Formula: $leftSpeed = -52 \times (\sum vl)^2 + Max\_Speed$ (vl = left sensor values; red)
  $rightSpeed = 52 \times (\sum vl)^2$ (vl = left sensor values; blue)



Daniel Vogel, Pascal Grones

We did not use a specific architecture, but simply used inheritance.