Course: CDA 4621 **Professor:** Alfredo Weitzenfeld PhD
Name : Chance Hamilton **Support:** Wall Following PID

1. Calculating Saturation Speed:

This function calculates the saturation speed of the robot's motors. It takes a velocity value as input and ensures that it doesn't exceed a maximum limit or fall below its negative counterpart. The function returns the saturated velocity.

Algorithm 1 Calculating Saturation Speed

function FORWARD_SATURATION(v)
  $max\_v \leftarrow$ *robot's motor max velocity*
  if $v \geq max\_v$ **then**
    $v \leftarrow max\_v$
  else if $v \leq -max\_v$ **then**
    $v \leftarrow -max\_v$
  end if
  return v
end function

2. Calculating Forward Speed Using PID:

This function computes the forward speed of the robot using a PID (Proportional-Integral-Derivative) control approach. It measures the distance to an object in front of the robot using lidar sensors, calculates an error relative to a desired distance to maintain, and adjusts the forward speed proportionally to this error. The result is the velocity to apply to the robot's motors.

Algorithm 2 Calculating Forward Speed Using PID

function FORWARD_PID($D\_maintain = 0.5, K\_p = 5$)
  $fd \leftarrow$ *Minimum lidar measurement in range of front measurements*
  $error \leftarrow fd - D\_maintain$
  return FORWARD_SATURATION($K\_p \cdot error$)
end function

3. Calculating Motor Speeds Using PID for Wall Following:

This function implements a PID-based wall-following algorithm. It calculates the left and right motor velocities required for the robot to follow a wall, either on its right or left side. It considers factors like the minimum desired distance from the wall, the distance readings from lidar sensors, and adjusts the motor speeds accordingly to maintain the desired distance from the wall. The function provides the left and right motor velocities as output, allowing the robot to execute wall-following behavior effectively.

Algorithm 3 Calculating Motor Speeds Using PID for Wall Following

```
function WALL_FOLLOW_PID(D_min = 0.45, K_p = 2, wall = 'R')
    V_f ← FORWARD_PID(robot, K_p = 5)
    rd ← Minimum lidar measurement in range of right measurements
    ld ← Minimum lidar measurement in range of left measurements
    if wall = 'R' then
        error ← D_min − rd
        if rd < D_min then
            V_r ← FORWARD_SATURATION(V_f)
            V_l ← FORWARD_SATURATION(V_f − abs(K_p · error))
        else if rd > D_min then
            V_r ← FORWARD_SATURATION(V_f − abs(K_p · error))
            V_l ← FORWARD_SATURATION(V_f)
        else if ld < .75 then
            error ← 0.75 − ld
            V_r ← FORWARD_SATURATION(V_f − abs(K_p · error))
            V_l ← FORWARD_SATURATION(0)
        else
            V_l ← V_f
            V_r ← V_f
        end if
    else
        error ← D_min − ld
        if ld < D_min then
            V_r ← FORWARD_SATURATION(V_f − abs(K_p · error))
            V_l ← FORWARD_SATURATION(V_f)
        else if ld > D_min then
            V_r ← FORWARD_SATURATION(V_f)
            V_l ← FORWARD_SATURATION(V_f − abs(K_p · error))
        else if rd < 0.75 then
            error ← 0.75 − rd
            V_r ← FORWARD_SATURATION(0)
            V_l ← FORWARD_SATURATION(V_f − abs(K_p · error))
        else
            V_l ← V_f
            V_r ← V_f
        end if
    end if
    return V_l, V_r
end function
```

4. Main Loop:

This is the central control loop of the robot's behavior. It iteratively performs simulation steps as long as the Webots simulator doesn't stop the controller. Inside the loop, it implements wall-following behavior with PID control, adjusting the left and right motor velocities based on sensor readings. Additionally, it checks if the robot is too close to a wall in front and performs rotations to avoid collisions.

Algorithm 4 Main Loop

```
while robot.experiment_supervisor.step(robot.timestep) ≠ −1 do
    V_l, V_r ← wall_follow_PID(robot, wall = wall_to_follow)
    robot.leftMotor.setVelocity(V_l)
    robot.rightMotor.setVelocity(V_r)
    fd ← Minimum lidar range measurement in range of front measurements
    if fd < 0.25 and wall_to_follow =' R' then
        robot.rotate(−90)
    end if
    if fd < 0.25 and wall_to_follow =' L' then
        robot.rotate(90)
    end if
end while
```