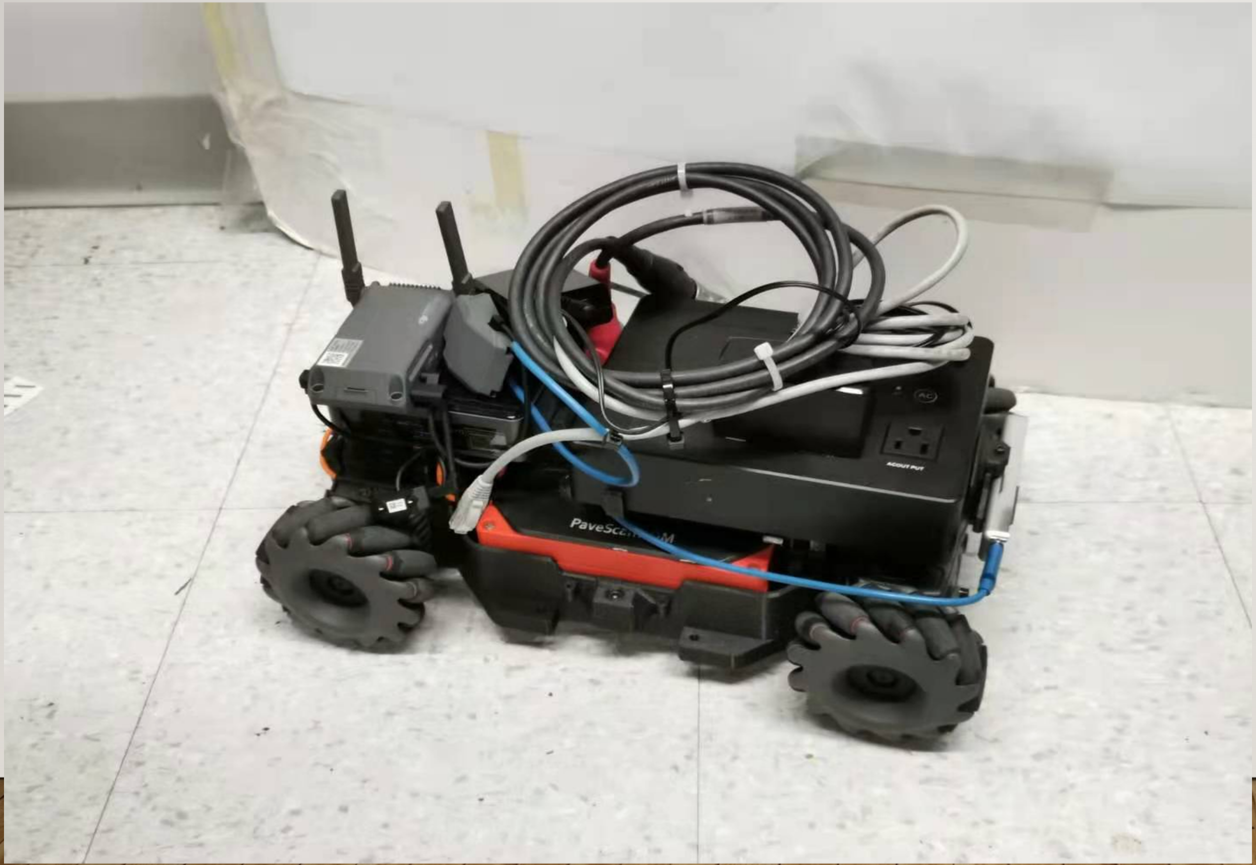
GPR-ROBOMASTER

MENTOR: JIZHONG XIAO

STUDENT: ZHICHUAN TAN

Task: program a robot equipped with GPR (ground penetrating radar) to move along a designed trajectory

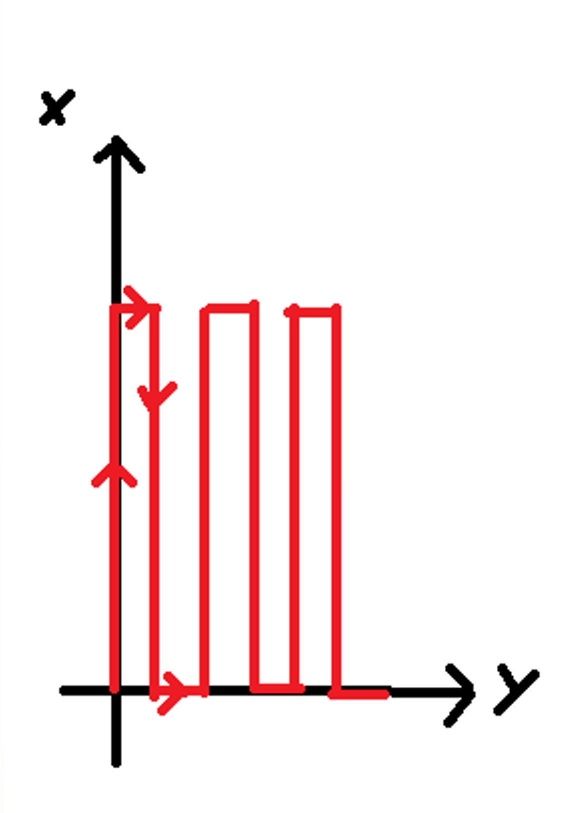


DJI Robomaster robot

TRAJECTORY

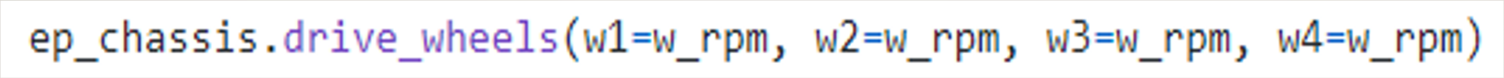
- Cover an area
- Square trajectory
- One cycle: forward, rightward, backward, rightward
- # Cycles depends on the area and GPR resolution
- e.g.: width = 1 m (y = 1), resolution = 0.05 m

  so, cycles = 1 / (2 * 0.05) = 10



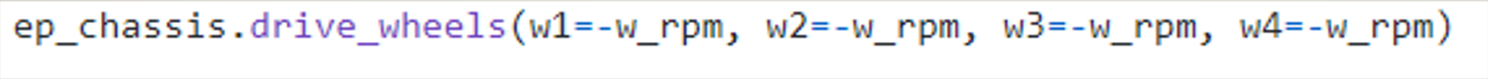
MOVE

- Mecanum wheels
- Control the speed of the wheels to move and rotate
- Programming language: Python



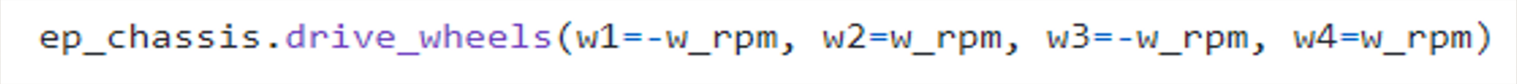
```
ep_chassis.drive_wheels(w1=w_rpm, w2=w_rpm, w3=w_rpm, w4=w_rpm)
```

Move forward



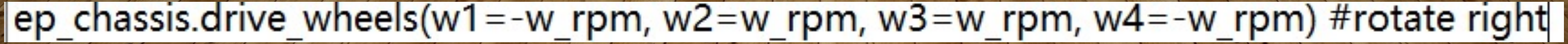
```
ep_chassis.drive_wheels(w1=-w_rpm, w2=-w_rpm, w3=-w_rpm, w4=-w_rpm)
```

Move backward

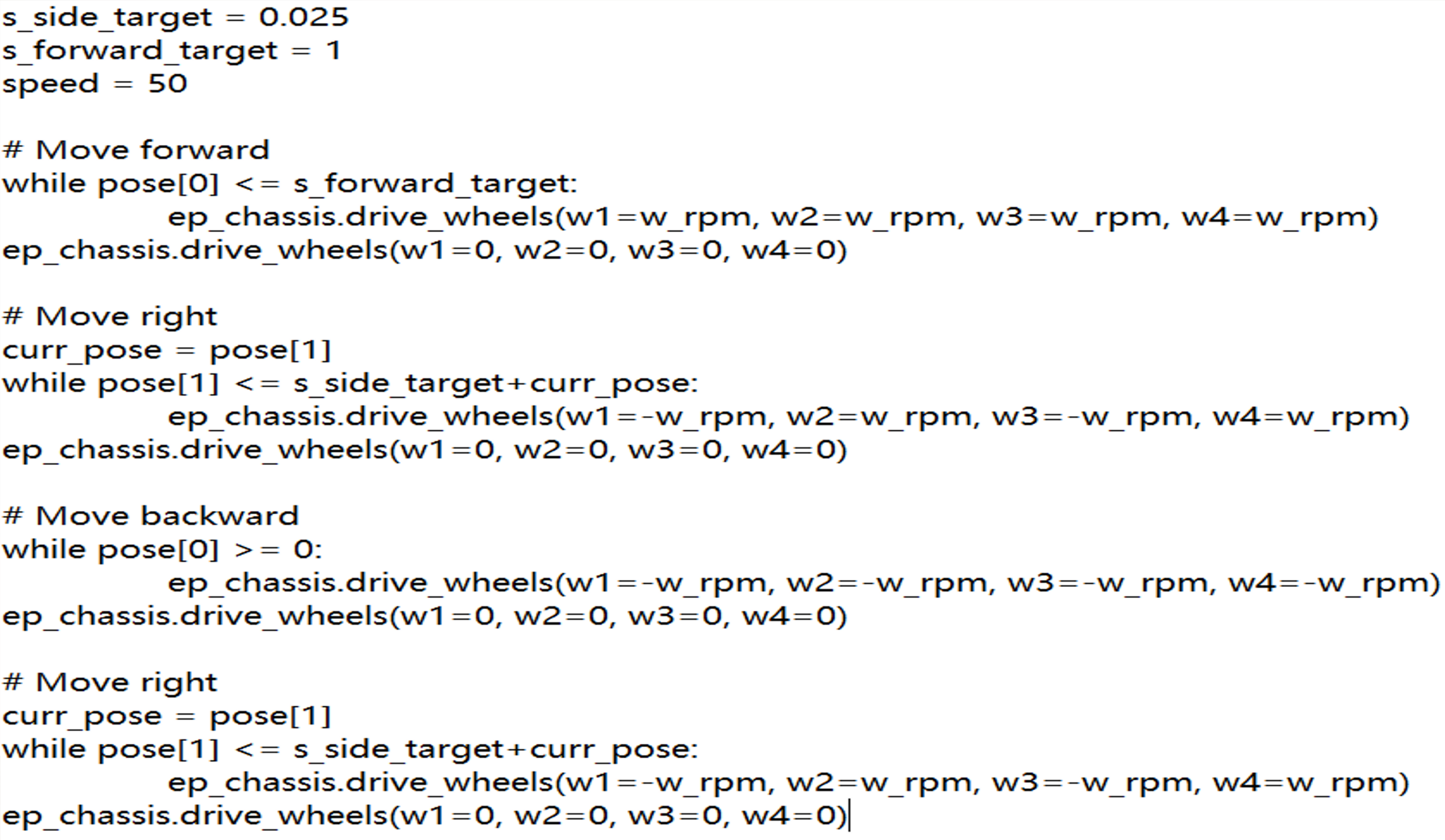


```
ep_chassis.drive_wheels(w1=-w_rpm, w2=w_rpm, w3=-w_rpm, w4=w_rpm)
```

Move rightward



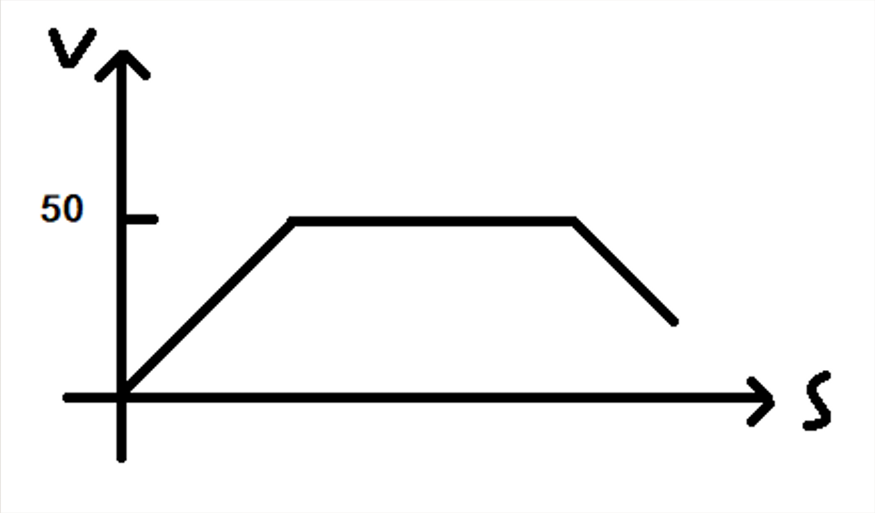
```
ep_chassis.drive_wheels(w1=-w_rpm, w2=w_rpm, w3=w_rpm, w4=-w_rpm) #rotate right
```

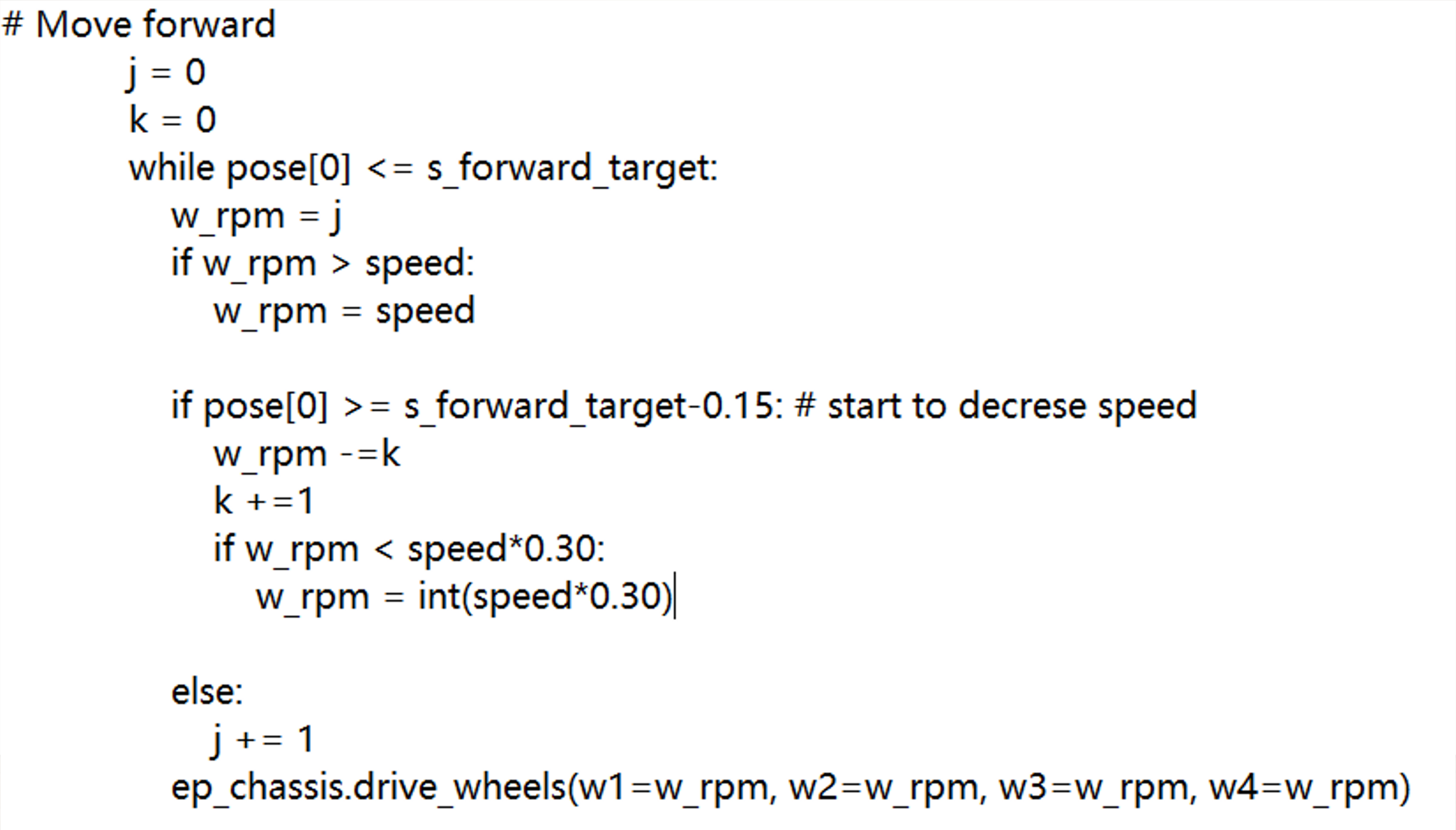


```
s_side_target = 0.025
s_forward_target = 1
speed = 50

# Move forward
while pose[0] <= s_forward_target:
        ep_chassis.drive_wheels(w1=w_rpm, w2=w_rpm, w3=w_rpm, w4=w_rpm)
ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)

# Move right
curr_pose = pose[1]
while pose[1] <= s_side_target+curr_pose:
        ep_chassis.drive_wheels(w1=-w_rpm, w2=w_rpm, w3=-w_rpm, w4=w_rpm)
ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)

# Move backward
while pose[0] >= 0:
        ep_chassis.drive_wheels(w1=-w_rpm, w2=-w_rpm, w3=-w_rpm, w4=-w_rpm)
ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)

# Move right
curr_pose = pose[1]
while pose[1] <= s_side_target+curr_pose:
        ep_chassis.drive_wheels(w1=-w_rpm, w2=w_rpm, w3=-w_rpm, w4=w_rpm)
ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
```

PROBLEM AND SOLUTION

- Movement is not smooth

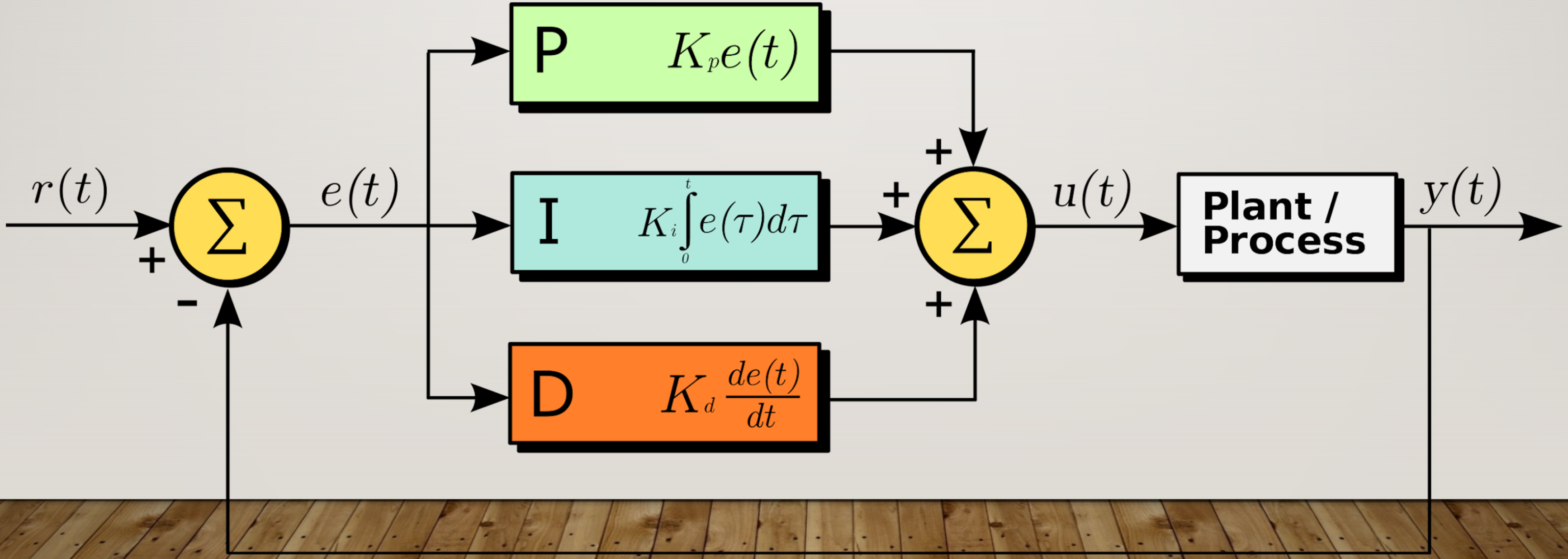




```
# Move forward
            j = 0
            k = 0
            while pose[0] <= s_forward_target:
                w_rpm = j
                if w_rpm > speed:
                    w_rpm = speed

                if pose[0] >= s_forward_target-0.15: # start to decrese speed
                    w_rpm -=k
                    k +=1
                    if w_rpm < speed*0.30:
                        w_rpm = int(speed*0.30)

                else:
                    j += 1
                ep_chassis.drive_wheels(w1=w_rpm, w2=w_rpm, w3=w_rpm, w4=w_rpm)
```

PROBLEM AND SOLUTION

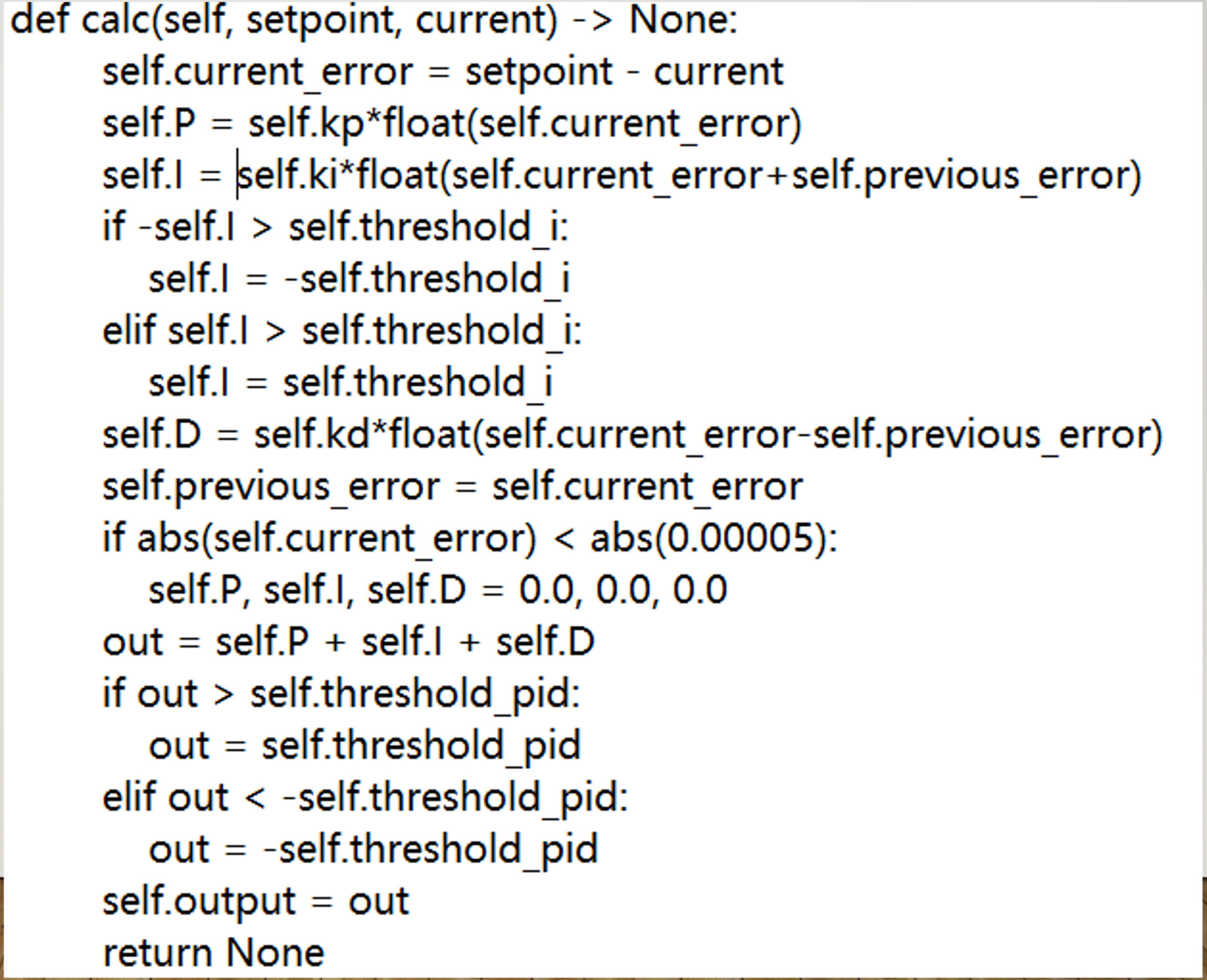
- Robot drifts and rotates when moving
- Solution: PID control



PID CONTROL CALCULATION

- Current error = expected value - current value
- P = Kp * current error
- I = Ki * (current error + previous error)
- D = Kd * (current error - previous error)
- PID output = P + I + D
- Increasing Kp → respond faster; oscillate if too large
- Increasing Ki → reduce error; large overshoot if too large
- Increasing Kd reduce overshoot; amplify noise if too large →

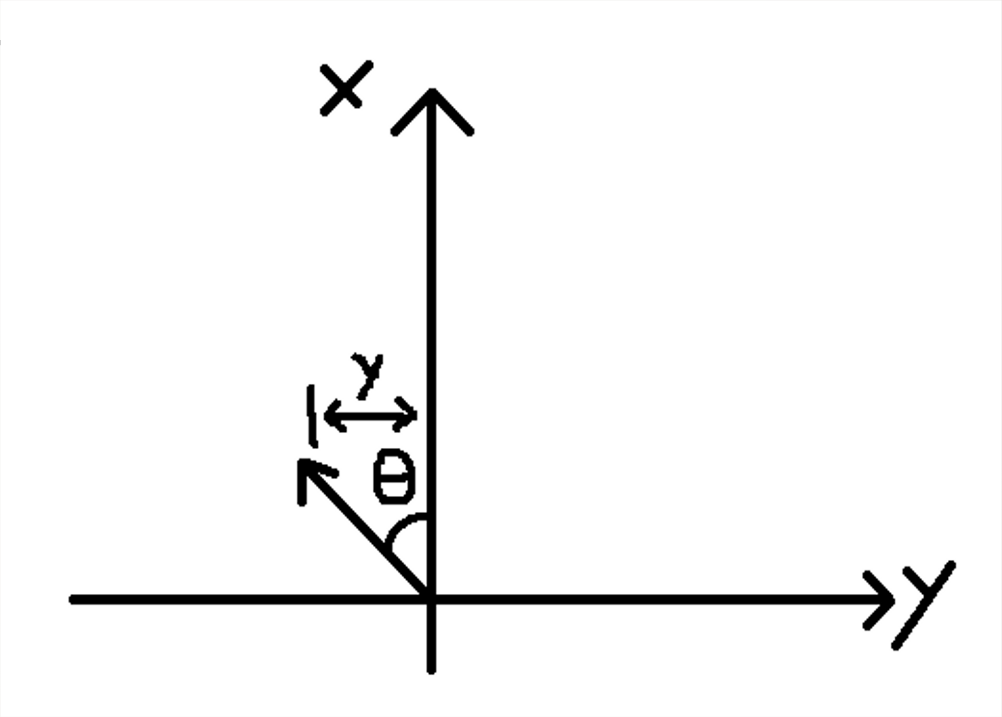
PYTHON CODE

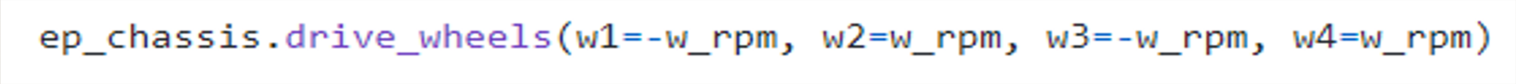


```
def calc(self, setpoint, current) -> None:
    self.current_error = setpoint - current
    self.P = self.kp*float(self.current_error)
    self.I = self.ki*float(self.current_error+self.previous_error)
    if -self.I > self.threshold_i:
        self.I = -self.threshold_i
    elif self.I > self.threshold_i:
        self.I = self.threshold_i
    self.D = self.kd*float(self.current_error-self.previous_error)
    self.previous_error = self.current_error
    if abs(self.current_error) < abs(0.00005):
        self.P, self.I, self.D = 0.0, 0.0, 0.0
    out = self.P + self.I + self.D
    if out > self.threshold_pid:
        out = self.threshold_pid
    elif out < -self.threshold_pid:
        out = -self.threshold_pid
    self.output = out
    return None
```

CONTROL YAW AND Y POSIT

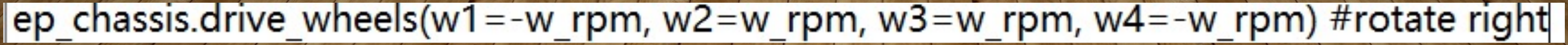
- Yaw error : 0 - theta
- Y error : y-setpoint - y
- Add PID output to the speed
- Rotate : control yaw
- Move : control y





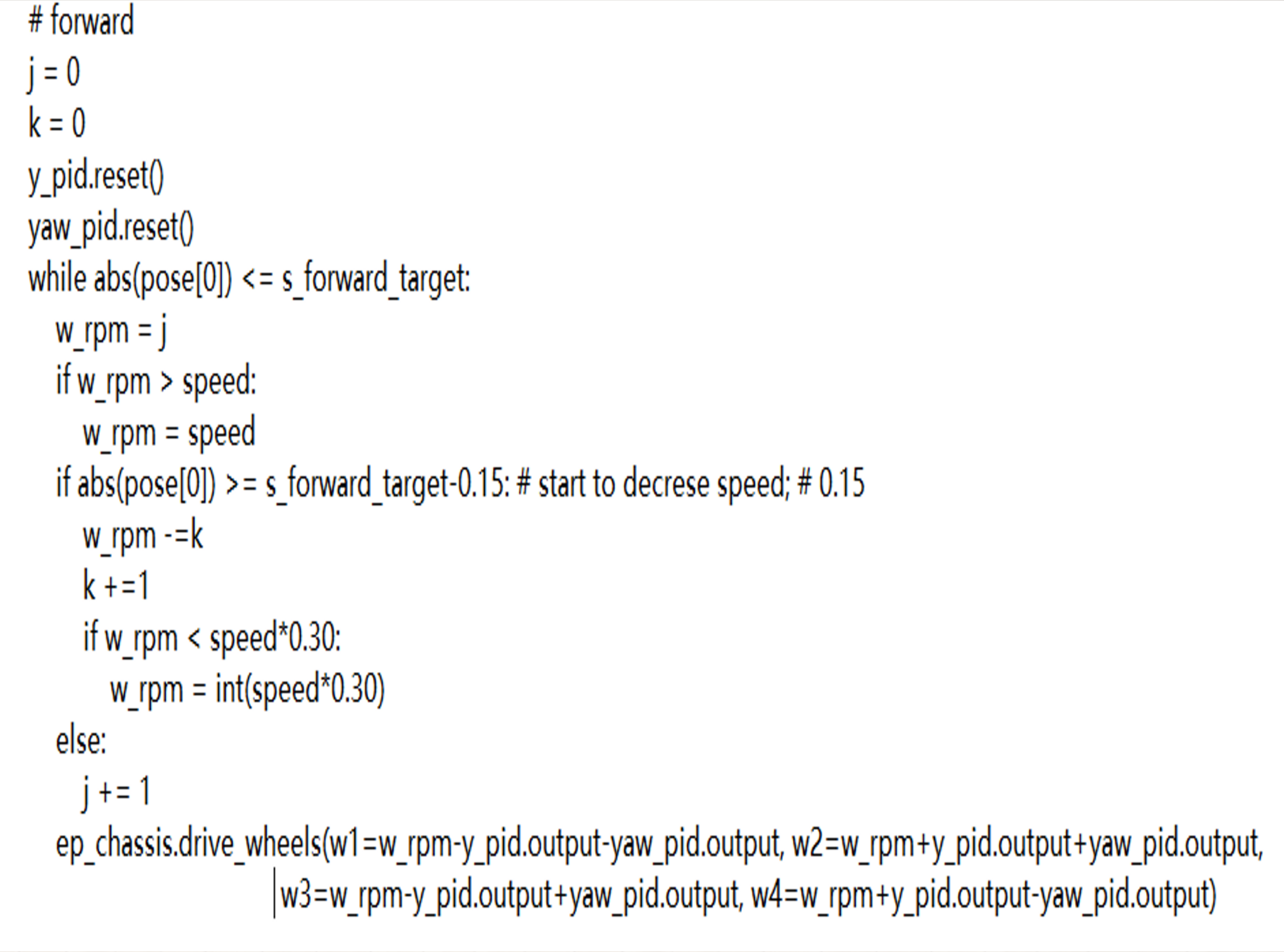
```
ep_chassis.drive_wheels(w1=-w_rpm, w2=w_rpm, w3=-w_rpm, w4=w_rpm)
```

Move rightward



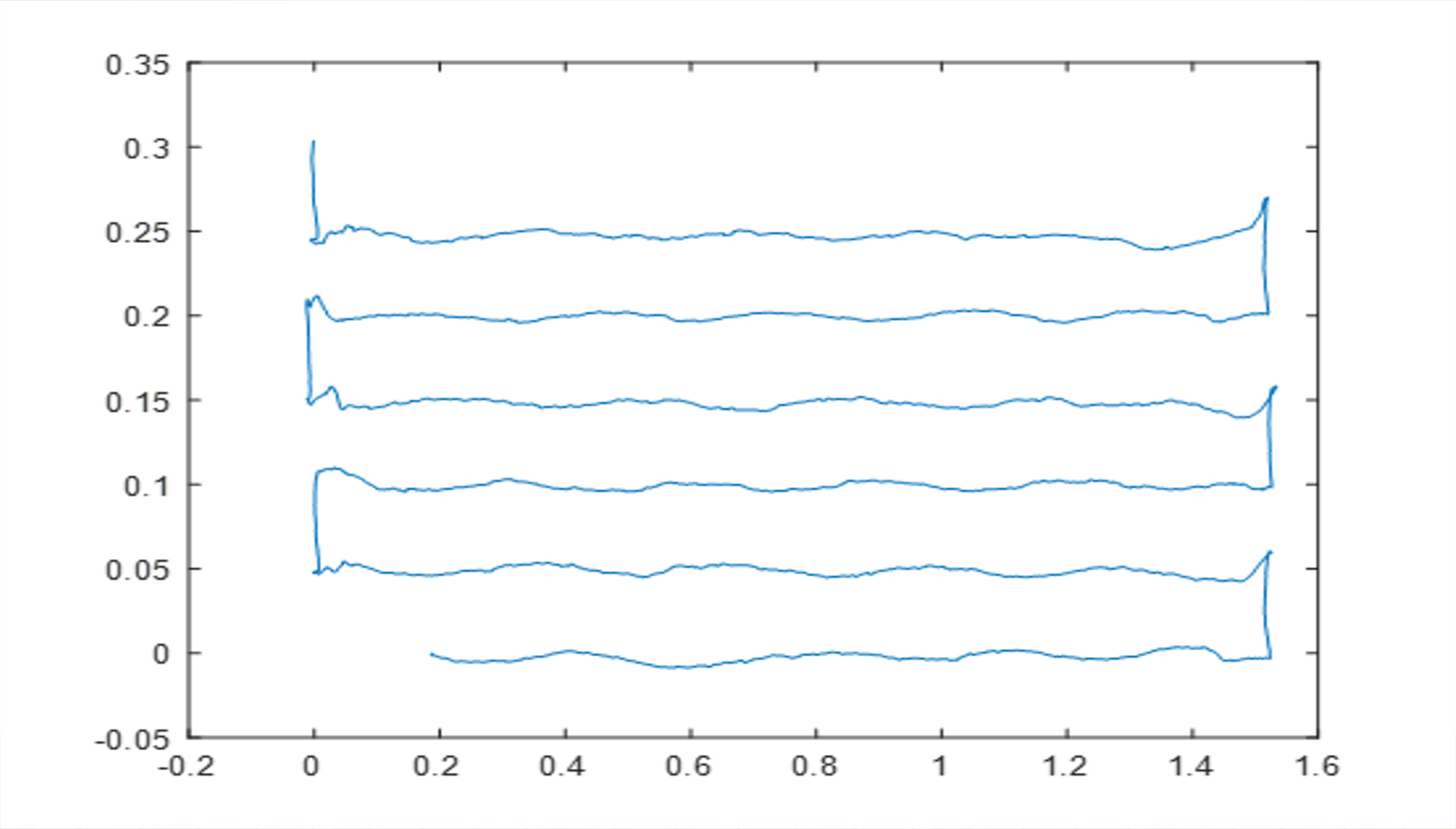
```
ep_chassis.drive_wheels(w1=-w_rpm, w2=w_rpm, w3=w_rpm, w4=-w_rpm) #rotate right
```

PYTHON CODI



```
# forward
j = 0
k = 0
y_pid.reset()
yaw_pid.reset()
while abs(pose[0]) <= s_forward_target:
    w_rpm = j
    if w_rpm > speed:
        w_rpm = speed
    if abs(pose[0]) >= s_forward_target-0.15: # start to decrese speed; # 0.15
        w_rpm -=k
        k +=1
        if w_rpm < speed*0.30:
            w_rpm = int(speed*0.30)
    else:
        j += 1
    ep_chassis.drive_wheels(w1=w_rpm-y_pid.output-yaw_pid.output, w2=w_rpm+y_pid.output+yaw_pid.output,
                            w3=w_rpm-y_pid.output+yaw_pid.output, w4=w_rpm+y_pid.output-yaw_pid.output)
```

RESULT



Thank You