

Lesson 19 Self-balancing

19.1 Overview

The self-balancing is developed based on the MPU6050 sensor. Once the self balancing function is enabled, it strives to ensure that the perspective of the robot camera remains horizontal. When this function is running normally, the robot is restricted from performing other operations. To disable this feature, simply click on the relevant option again.

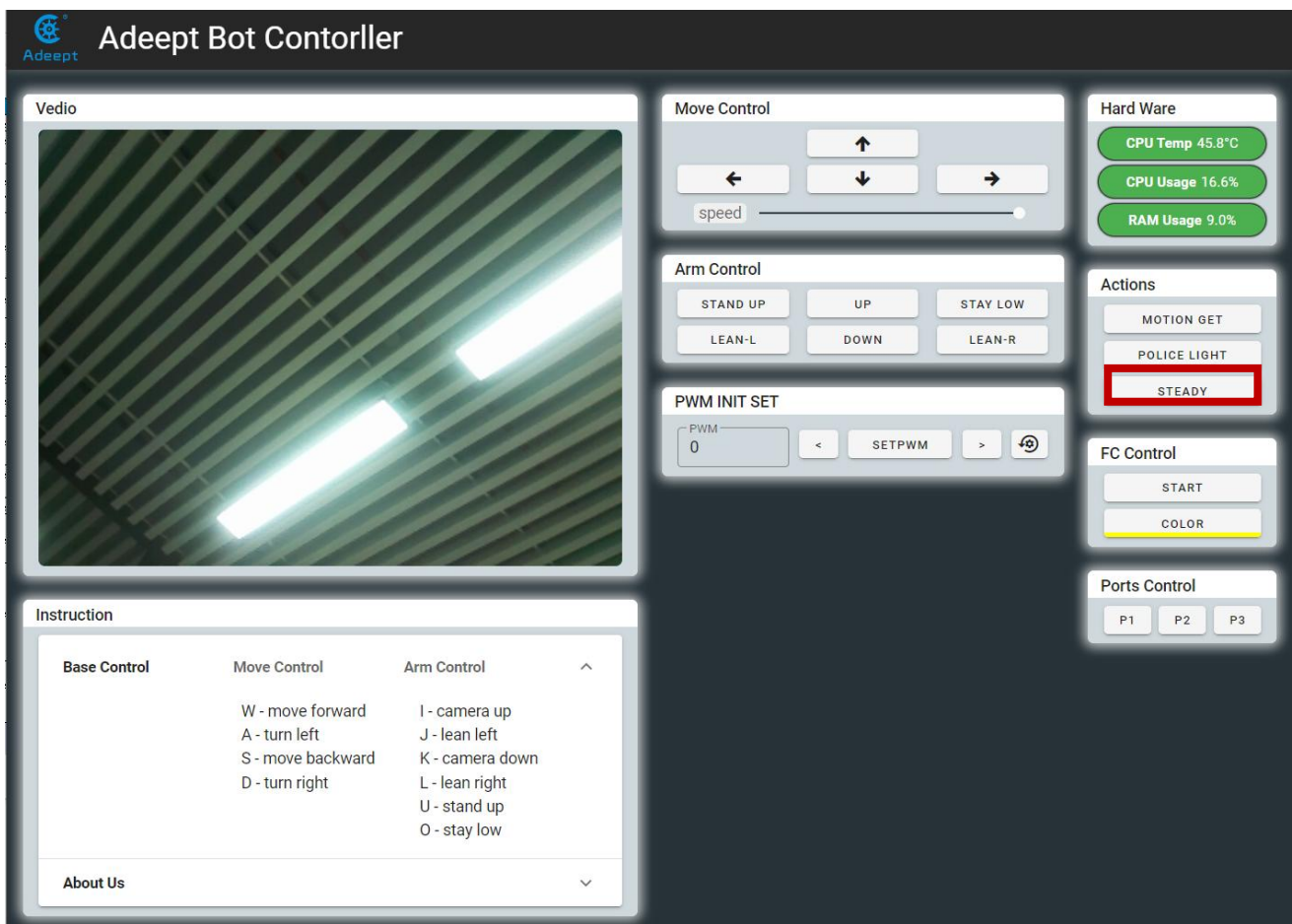
19.2 Introduction to Self-balancing

The self-balancing utilizes the data from the MPU6050 sensor to detect the orientation and movement of the robot. By continuously monitoring the sensor readings, it can adjust the camera's position to maintain a horizontal view. This is achieved through a series of algorithms that calculate the necessary adjustments based on the detected tilt angles. The MPU6050 provides accurate acceleration and gyroscope data, which serves as the foundation for the self-balancing and camera - stabilization mechanisms. This ensures that regardless of the robot's movement or orientation changes, the camera's line of sight remains parallel to the horizon, offering a consistent and stable visual output.

19.3 Running the Self-balancing Function

Running the Automatic Obstacle Avoidance program

1. Start the Adeept_DarkPaw Robot. It may take about 30-50s to boot.
2. After Adeept_DarkPaw is turned on, open the Chrome browser on your mobile or computer, enter the IP address of your Raspberry Pi and access port ":5000" into the IP address bar, like this: **192.168.3.31:5000**. The web controller will then be displayed on the browser.



3. After clicking "**STEADY**", The Adeept_DarkPaw camera will remain level.

4. Click "**STEADY**" again to disable the function.

19.4 Code

The main code is as follows. For the complete code, please check [SpiderG.py](#).

```

01 def steady():
02     global sensor
03     if steadyMode:
04         if MPU_connection:
05             try:
06                 accelerometer_data = sensor.get_accel_data()
07                 X = accelerometer_data['x']
08                 X = kalman_filter_X.kalman(X)
09                 Y = accelerometer_data['y']
10                 Y = kalman_filter_Y.kalman(Y)
11
12                 X_error = X-X_steady
13                 Y_error = Y-Y_steady
14
15                 if abs(X_error)>mpu_tor or abs(Y_error)>mpu_tor:
16                     status_GenOut(0, Y_error*P, X_error*P)

```

```
17         direct_M_move()
18     except:
19         time.sleep(0.1)
20         sensor = mpu6050(0x68)
21     pass
```