Connections:

MPU-6050	
MPU-6050 Pin	**Arduino Pin**
GND	GND
Vcc	5V
SCL	A5
SDA	A4
INT	D2
L298N Motor Driver	
L298N Motor Driver Pin	**Arduino Pin**
12V	Battery +ve
GND	GND + Battery -ve
5V	Vin
ENA	D3
IN1	D4
IN2	D5
IN3	D7
IN4	D8
ENB	D9

Code:

Press here to visit GitHub repository

I- Overview:

- Before diving into the code explanation, it's good to mention that it's divided into several layers and not a one chunk piece of code as a (.ino) file. Each layer is divided into its own cpp (.cpp) and header (.h) files. The header file contains all constants' definitions in the form of (#define) statements as well as the functions' prototypes, while the cpp file contains functions' declarations.
- The layers in our robot are:
    - Application: App.cpp – App.h
    - Motors: motors.cpp – motors.h
    - MPU-6050: mpu6050.cpp – mpu6050.h
    - PID: pid.cpp – pid.h
    - Test: test.cpp – test.h
- All besides the (.ino) file which calls the App_start() and App_init() functions present in the application layer.
- First the application layer (upper most layer) and it has the following functions:
    - App_init() function that calls the initialization functions for the motors, PID controller and MPU-6050.
    - App_start() function where the application logic for the whole system runs.
- Second the motors layer and it has the following functions:
    - motors_setup() to set pins direction(input or output)
    - move(int speed) and it takes speed as an input parameter holding both the magnitude and sign to move the robot either forward or backward with this speed.
    - moveForward() to move the wheels in one direction with full speed.
    - moveBackward() to move the wheels in the other direction with full speed.
    - stop() to stop both wheels motion.
- Third is the MPU-6050 layers and it has the following functions:

    - mpu_setup() to calibrate it and initialize its internal processor(DMP).
    - dmpDataReady() which is an ISR that's triggered when new data is available to be fetched out.
    - mpu_update() to constantly update the sensor's readings.
    - return_roll(), return_pitch() and return_yaw() to return inclination angles around the x-axis, y-axis and z-axis respectively.
- Fourth is the PID layer and it has the following functions:
    - PID_setup() to apply initial configurations as sampling time and mode either automatic or manual.
    - PID_run() to run the PID algorithm inside it and let it do its calculations.
    - get_pid_output() to return the output calculated by PID which in our case is motor speed ranging from (-255, 255).
- Last but not least the Test layer to test each of the above layers separately and check if they function correctly. It contains the following functions:
    - test_motors() where we call the move(int speed) function of motors and send different speeds using Serial connection. Speeds are either positive or negative to move in the corresponding direction.
    - test_mpu6050() and it basically returns the pitch angle to be printed to the serial monitor to check that readings are correct.
    - test_pid(int input) that takes inclination angle as input and returns the output speed that should be sent to motors. This is to check that the algorithm is running correctly and it's output is reasonable relative to the input angle fed to it.
- That was all for the different layers in the code, what functions are there in each and a brief summary for each function and its usage.

II- Explanation: