Two-Wheel Balancing Robot

This robot uses sensors to balance itself on two wheels by moving in the required direction. The balancing about two wheels provides a great deal of freedom and mobility, as it can easily avoid obstacles.

Introduction

- Inspired by a Segway, which can transport people while balancing on two robots, this robot is a miniature which can do the same.
- It works on the basis of Linear Inverted Pendulum, wherein a continuous output dynamically balances an unstable mass.
- It senses the acceleration continuously and uses PID control algorithm to create acceleration in the opposite direction.
- The opposite acceleration is created by moving its two wheels. The speed of the motors is controlled using Pulse-Width Modulation.

Stage 1: Design of Chassis

1. A basic Mechanix set is used to construct a chassis for the two wheeled robot. Two motors were joined on the left and right of the robot. The Mechanix base was chosen as it is easily removable and reconfigurable according to the needs.
2. Two support bases are constructed to mount the Microcontroller board and the IMU sensor.
3. An extra weight was added to the top, so as to create a larger inertia to rotation.

Stage 2: Setup and Test of sensors and motors

1. Mount the microcontroller board and the IMU on the chassis, complete its wiring.
2. The output of IMU is to be given to an input pin, and the registers should be declared accordingly.
3. The motors should be connected to the output of the motor driver instead of connecting directly to the microcontroller, as a microcontroller can neither provide the required voltage nor the current for correct operation of motors.
4. Recheck the IMU power supply wiring as it is a very delicate and sensitive device. Connect the power pins properly.
5. Test the IMU output and transmit it to Hyperterminal using UART communication.
6. Check the variation of the output along with the variation of angles. Calibrate the output if necessary.
7. Start the internal timer of microcontroller and test the speed variation of motors. Using PWM, the on time and off time of motor driver enable pin is varied, which corresponds to the speed variation of motors.

Stage 3: IMU Filtering and Fusion

1. Using the output of UART, check for any noisy output. A sudden drop to value of 0 or a sudden high value indicates noise of the accelerometer.
2. Plot the graph of the output if possible to see the noise and its effect visually.
3. Apply any filtering algorithms as necessary. For a lesser magnitude of noise, use a simple Low Pass filter algorithm.
4. If the readings are still noisy, check the connections for any fault. If noise still persists, use an advanced filter like Kalman Filter algorithm to reduce the noise.
5. After obtaining the readings from accelerometer and gyroscope, fuse the sensor readings to obtain a proper orientation of the robot along with magnitude of acceleration.
6. The Fusion factor used for combining the two readings is dependent on individual systems. It works by combining two outputs, usually one being a low frequency output and the other being a higher frequency one.
7. The output combines the best parts available from both these readings and avoids errors such as gyroscopic drift.
8. Use UART to verify readings. Processing can be done by saving the reading continuously in a text file and plotting it in another program.

Stage 4: PID control programming

1. The final fused sensor reading is then given as an input to the Proportional-Integral-Derivative control algorithm.
2. Firstly, the output is to be declared as an addition to the OCR of the PWM output of motors. This is a really important step as our output needs to be acceleration not velocity, as we need a pseudo-torque to maintain the balance.
3. At the vertical position of the robot, measure readings from the sensors, so as to obtain a definite angle of tilt. This angle/reading is to be used as a setpoint for the PID algorithm.
4. For the PID control, use only proportional term at start. The output of this algorithm is to be continuously added to the OCR value of the motor PWM. Make sure the output does not exceed 255 to avoid overflow errors.
5. For the opposite direction, assign a negative value to the input, so as to move the motors in opposite direction.
6. Use the sign of sensor readings, not the output of PID to assign the direction of motor movement, as the robot should be able to move forward and accelerate backward at the same time.
7. Adjust the ‘P’ term of the algorithm so that the motors almost start the ‘Bang-Bang’ like state, i.e. suddenly accelerate to almost full speed when crossing the setpoint/vertical position.
8. Slowly increment the ‘D’ term so that the robot balances on its two wheels. As the angle of tilt increases, the robot should accelerate more towards the direction of tilt so as to balance it.
9. Make sure that it is highly reactive. A slow response will usually make the robot fall flat on its face. A minimum update speed of 40Hz-50Hz is recommended.
10. If the robot still fails to balance, try varying the ‘I’ term of the PID algorithm. Make sure the output adds on to the existing OCR.

11. The PID tuning requires a lot of time and trials, but once tuned, the robot is really immune to any variations. Tip: Vary the parameters of the PID algorithm in the order of log scale. Eg: 0.1, 0.33, 1 and so on.

Pitfalls and Tips

1. The centre of gravity of the body can be imbalanced by adding weights on the top of the robot, thereby increasing the rotational inertia. This will help in balancing the robot in a better way.
2. Make sure you are adding on the output of the PID onto the existing OCR value of PWM so as to simulate acceleration, instead of velocity.

Warnings:

1. **Make sure the IMU is connected in the correct fashion as it is a very sensitive and delicate device.**
2. **Do not exceed the voltages of the IMU in any way. Even small variations will cause overheating and hence failure of the IMU.**
3. **Make sure that your motor driver IC (usually L293D)can handle the current that your motor requires. If the motor driver overheats a lot, consider switching to a better rated motor driver (L298N). Use heatsinks wherever overheating occurs.**

Improvements

1. A simple offset to the PWM output will cause a constant velocity to be added to the reading required for balancing the robot. This offset can be varied and controlled to make the robot move in any direction required, while balancing on its two wheels. This would be a significant improvement over balancing on the spot.
2. A difference between the offsets given to individual PWM of motors would make it possible to turn the robot by differential driving.
3. If the PID constants are tuned properly, the robot could even climb up slopes without toppling over.
4. A wireless control can be done by Bluetooth Communication or Wi-Fi Modules, to make the robot completely mobile.

Author: Shubham M Wagh *Date: 12th Septemberr 2013*

Electrical Engg. Dept., V.N.I.T. Nagpur