Self-Balancing Robot

Ricky Huang, 008410701, John Strube, 010072595, CmpE 150 Spring 2017

***Abstract –*** The goal of this project was to create an autonomous self-balancing robot controlled by a microcontroller. By receiving data from an accelerometer, the robot can adjust the motor’s speed to compensate for its tilt.

1. Introduction

The main parts of the robot consist of an accelerometer, a motor driver, and two motors. As the accelerometer detects a tilt in a certain direction, the motors move in the same direction to cancel the tilt. The speed of the motors also depends on the degree of tilt, but it is further processed through a PID controller to make sure the robot does not overcompensate and overrun the balancing target.

1. Design Methodology

The robot was made primarily with a dsPIC33FJ128MC802 microcontroller, an accelerometer, an H-Bridge motor driver, and two 100:1 DC motors. The accelerometer communicated with the microcontroller via SPI. The H-bridge motor driver was controlled with PWM which in turn drove the two motors.

1. *Parts List*

* Microchip dsPIC33FJ128MC802 microcontroller
* Microchip DM330013-2 Microstick II
* PCB board
* 2 x Pololu 1101 100:1 6V DC motors
* L298 Dual H-bridge DC motor controller
* PMOD ACL2 3-axis MEMS accelerometer
* 3D printed chassis
* 2 x 3D printed wheels

1. *Hardware Design*

The robot’s hardware was built around the Microstick II for ease of use and convenience in adjusting internal parameters during testing. The two devices connected to the microcontroller are the dual H-bridge motor driver and the accelerometer. The accelerometer is connected through SPI. The two motors drive each wheel on either side and is connected to the dual H-bridge motor driver, which controls the speed of the motors with pulse-width modulation (PWM).

The chassis was 3D printed to fit the main controller assembly along with housing the two motors on each side. The wheels were also 3D printed, but the material used did not have enough friction, so grooves on the wheels were made and rubber bands were wrapped around the wheels to gain traction.

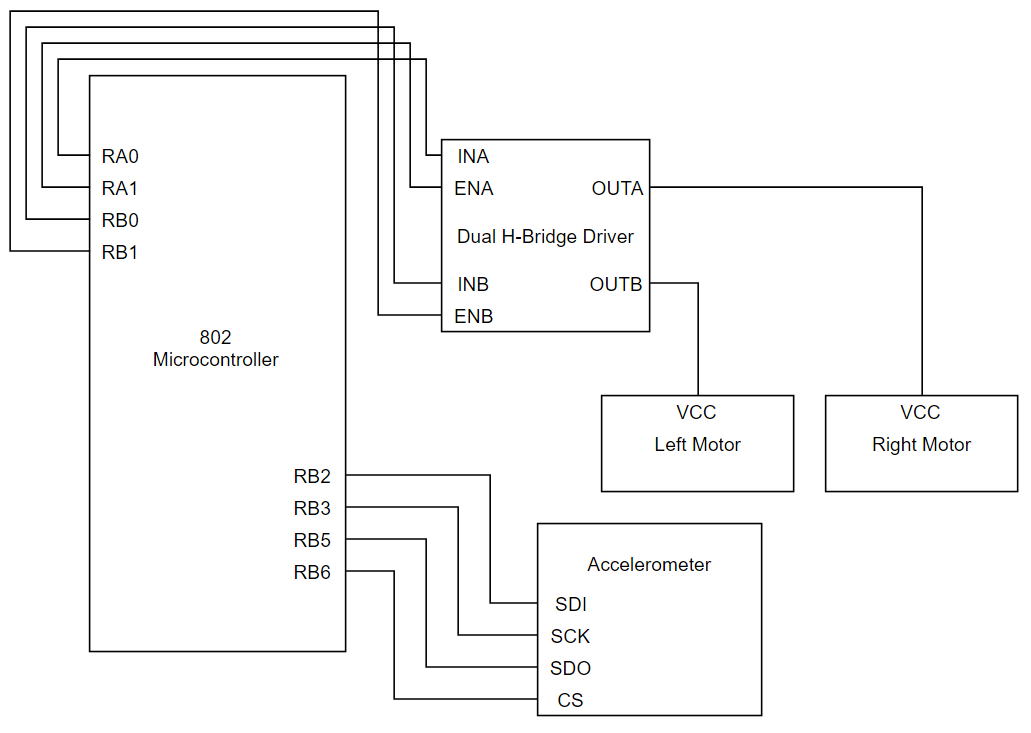


Figure 1: Hardware microarchitecture of the self-balancing robot.

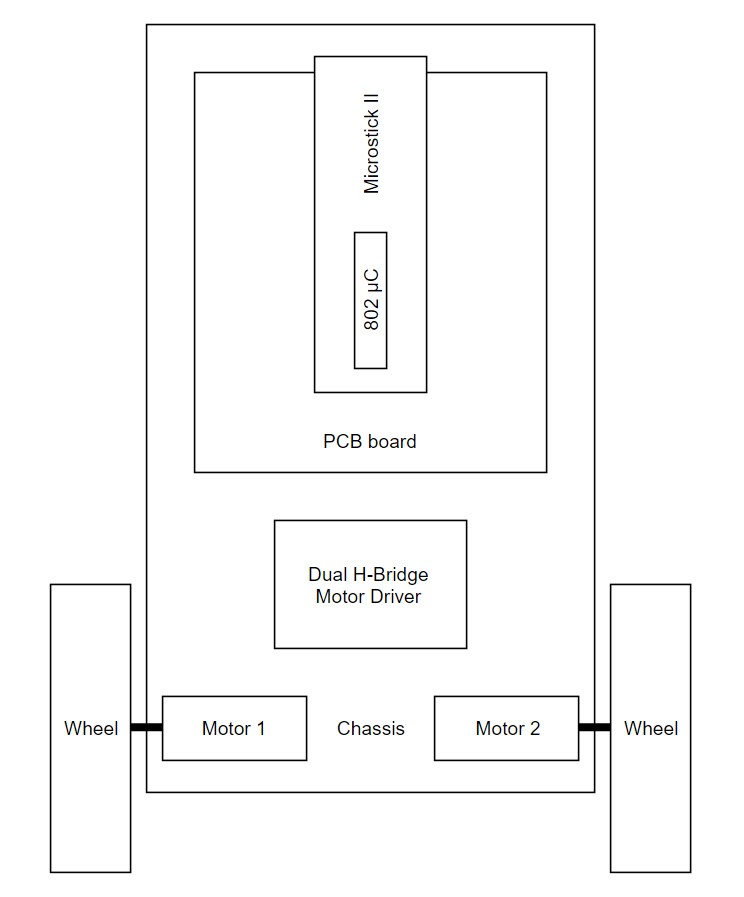


Figure 2: A diagram of the full assembly of the device.

1. *Software Design*

All self-balancing controls and computations were done using the microcontroller. For smoother motor control, the clock speed was overclocked to 8Mhz. Since the robot only tilts forwards and backwards, only the X-axis was used in our application. The speed of the two motors were controlled using PWM, and fine-tuned using proportional-integral-derivative (PID) controller. The PID controller was used to have a continuous feedback system that calculated the error value from the desired angle, and the speed of the motors were adjusted accordingly. The main algorithm of the program was a loop to retrieve the tilt data from the accelerometer, calculate how fast the motors spin through PID, and then perform the motor action.

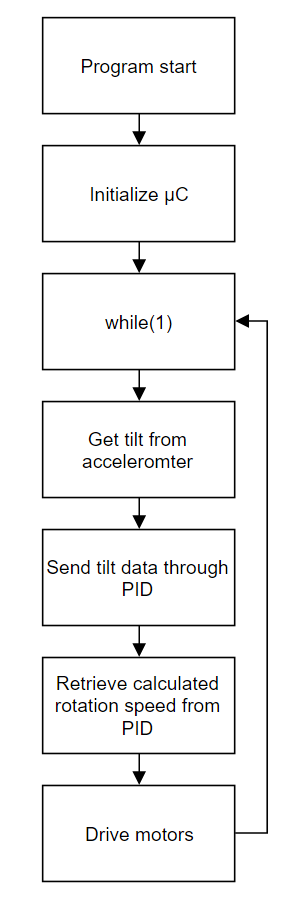


Figure 3: Flowchart of the self-balancing algorithm.

1. Testing Procedures and Results

The self-balancing robot was put through countless trials in order to fine tune the parameters for self-balancing action. During testing of the motors, pulsing the motor driver at high frequencies destroyed one of our motors due to the voltage ramping down too slowly, because the driver runs on a 12V power supply and our motors are only rated for 6V. In order to fix this issue, we programmed the PWM to not exceed a duty cycle of 50%.

Further testing revealed that the accelerometer was not calibrated correctly, and was sending the wrong data to the microcontroller. To fix this, we added an offset in each axis to correct the tilt data retrieved from the accelerometer.

Unfortunately, we were unable to create a working balancing robot. This was due to the fact that the motors used did not provide enough torque to self-correct itself, and performed jerky movements instead of smooth motor rotation. Furthermore, the parameters of the PID needed to be constantly tweaked, and the ideal values were not able to be calculated.

1. Conclusion

In conclusion, even though a functional device was not achieved by the deadline, further testing and adjustments of the robot may be able get the self-balancing action working correctly. Larger motors with more torque could’ve been used so low speeds would still provide enough torque to drive the wheels.