SELF BALANCING BOT

COURSE PROJECT DOCUMENTATION

CS684 PROJECT

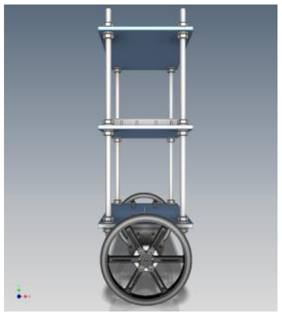
BY ABHISHEK GHOGARE (153059006)

TABLE OF CONTENT

INTRODUCTION

ABSTRACT

Two wheeled balancing robots are based on inverted pendulum configuration which rely upon dynamic balancing systems for balancing and maneuvering. These robot bases provide exceptional robustness and capability due to their smaller size and power requirements. Outcome of research in this field had led to the birth of robots such as Segway, Murata boy etc. Such robots find their applications in surveillance & transportation purpose. This project is based on development of a self-balanced two-wheeled robot which has a configuration similar to a bicycle. In particular, the focus is on the electro-mechanical mechanisms & control algorithms required to enable the robot to perceive and act in real time for a dynamically changing world. While these techniques are applicable to many robot applications, the construction of sensors, filters and actuator system is a learning experience.



Two wheeled balancing robot is a classic engineering problem based on inverted pendulum and is much like trying to balance a broom on the tip of your finger.

The word balance means the inverted pendulum is in equilibrium state, which its position is like standing upright 90 degrees. However, the system itself is not balance, which means it keeps falling off, away from the vertical axis. Therefore, MPU is needed to provide the angle position of the inverted pendulum or robot base and input into the microcontroller, which the program in itself is a balancing algorithm. The microcontroller will then provide a type of feedback signal through PWM control to turn the motor clockwise or anticlockwise, thus balancing the robot. The code is written in C software and compiled for the Tiva Launchpad tm4c123g microcontroller, which is interfaced with the sensors and motors. The main goal of the microcontroller is to fuse the wheel encoder, gyroscope and accelerometer sensors to estimate the attitude of the platform and then to use this information to drive the reaction wheel in the direction to maintain an upright and balanced position. The basic idea for a two-wheeled dynamically balancing robot is pretty simple: move the actuator in a direction to counter the direction of fall. In practice this requires two feedback sensors: a tilt or angle sensor to measure the tilt of the robot with respect to gravity, an accelerometer to calibrate the gyro thus minimizing the drift.

PROBLEM STATEMENT

Aim is to make a two-wheeled robot stand upright and balance in addition to maintaining the position and stability.

REQUIREMENTS

HARDWARE REQUIREMENTS

1. DC motors with high resolution encoder
2. Motor driver L298N
3. MPU9250
4. 12v Battery
5. Tiva Launchpad TM4C123G microcontroller

SOFTWARE REQUIREMENTS

1. Code Composer Studio
2. Energia for real time graph plotting

MODULES

I^2C PROTOCOL MODULE

A I^2C communication module of Tiva board is used to communicate with MPU9250.

I^2C (Inter-Integrated Circuit), pronounced I-squared-C, is a multi-master, multi-slave, single ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors). It is typically used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication.

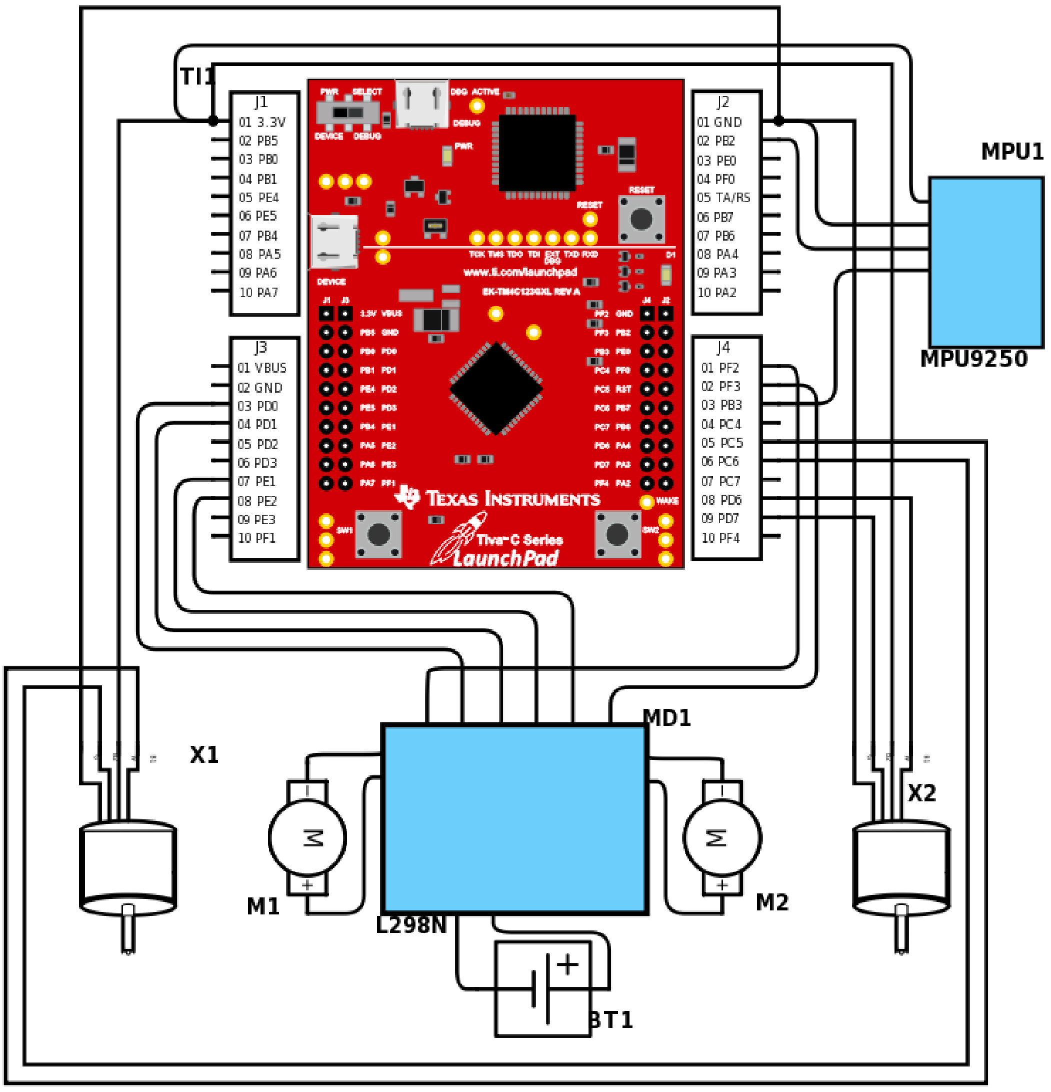
QUADRATURE ENCODER MODULE

A high-resolution quadrature encoder requires special handling. Tiva board comes with two quadrature encoder modules, can be used one for each wheel.

PID CONTROLLER

A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism (controller) commonly used in industrial control systems. A PID controller continuously calculates an error value e(t) as the difference between a desired setpoint and a measured process variable and applies a correction based on proportional, integral, and derivative terms, (sometimes denoted P, I, and D respectively) which give their name to the controller type.

CIRCUIT DIAGRAM



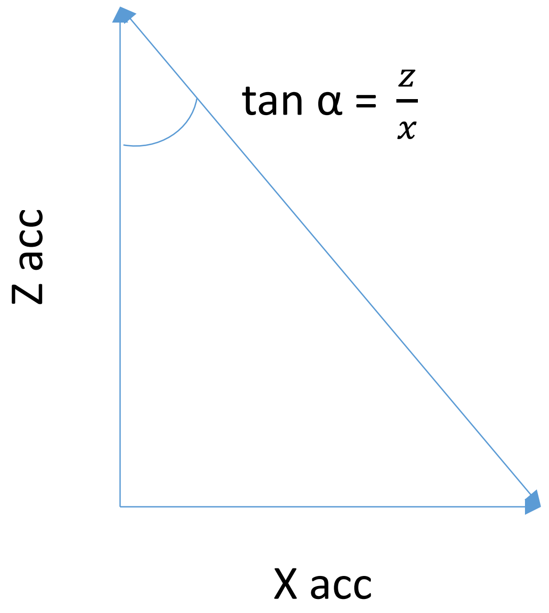
PROCEDURE

ANGLE CALCULATION

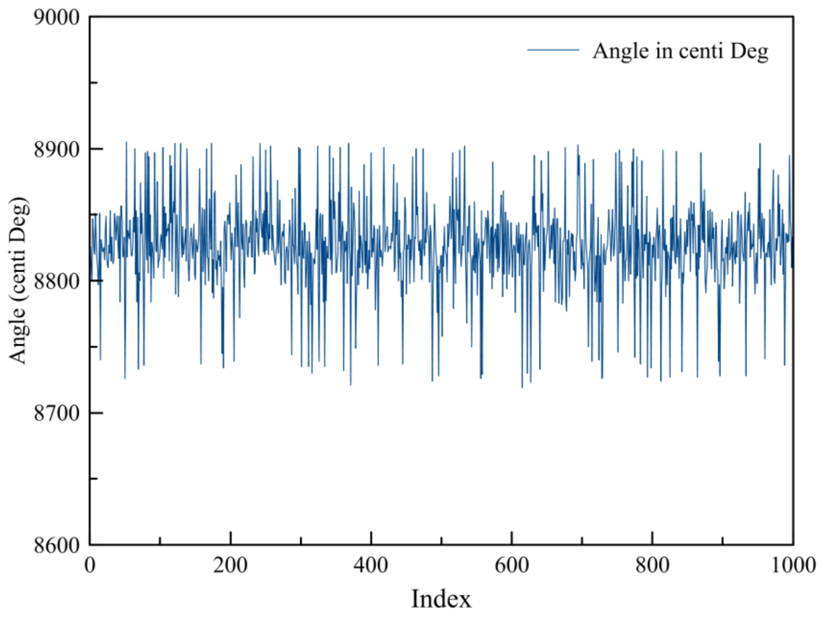
Based on readings from MPU, we can calculate angle by two methods.

1. Using accelerometer
2. Using gyroscope

ACCELEROMETER

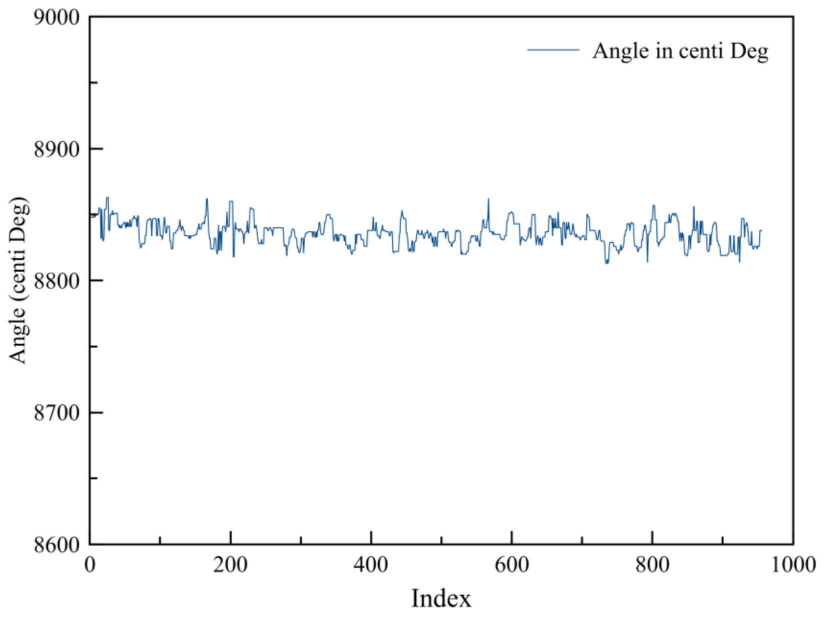


We are calculating tile angle using x and z components of accelerometer. It gives two advantages over single axis calculation. First, it excludes calibration of maximum value of accelerometer along any axis. Second, it minimizes angle calculation error, since two axes are used.



As shown in above graph, the calculated angle is very noisy even in steady state. A filtering mechanism should be applied.

We are using median filter, in which a median value from a predefined sized sliding window over samples is used.

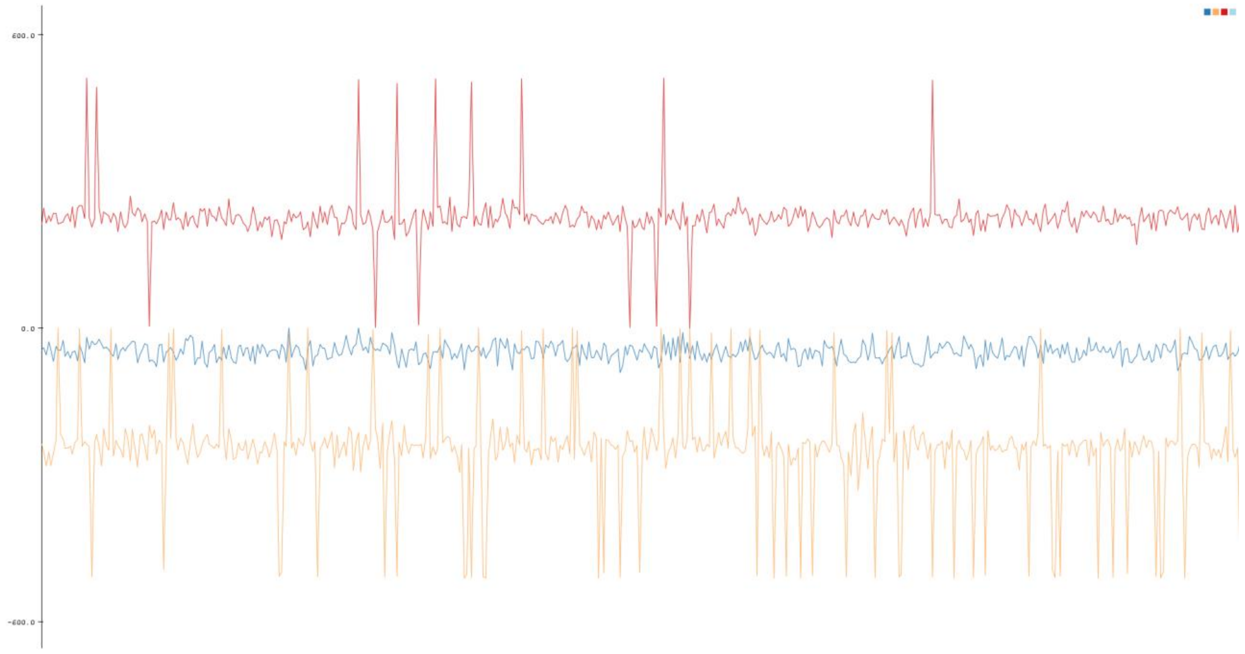


Above graph shows the filtered angle calculation.

GYROSCOPE

A gyroscope gives the angular velocity. We need to integrate it over time to calculate the actual angle.

In steady state, gyroscope does not give zero output, but some error offset.



Above graph shows the steady state values of the three axes of gyroscope. Each axis has different offset value. This value should be carefully calibrated.

We need to subtract this offset from the gyroscope output before integrating it to calculate the angle.

COMPLEMENTARY FILTER

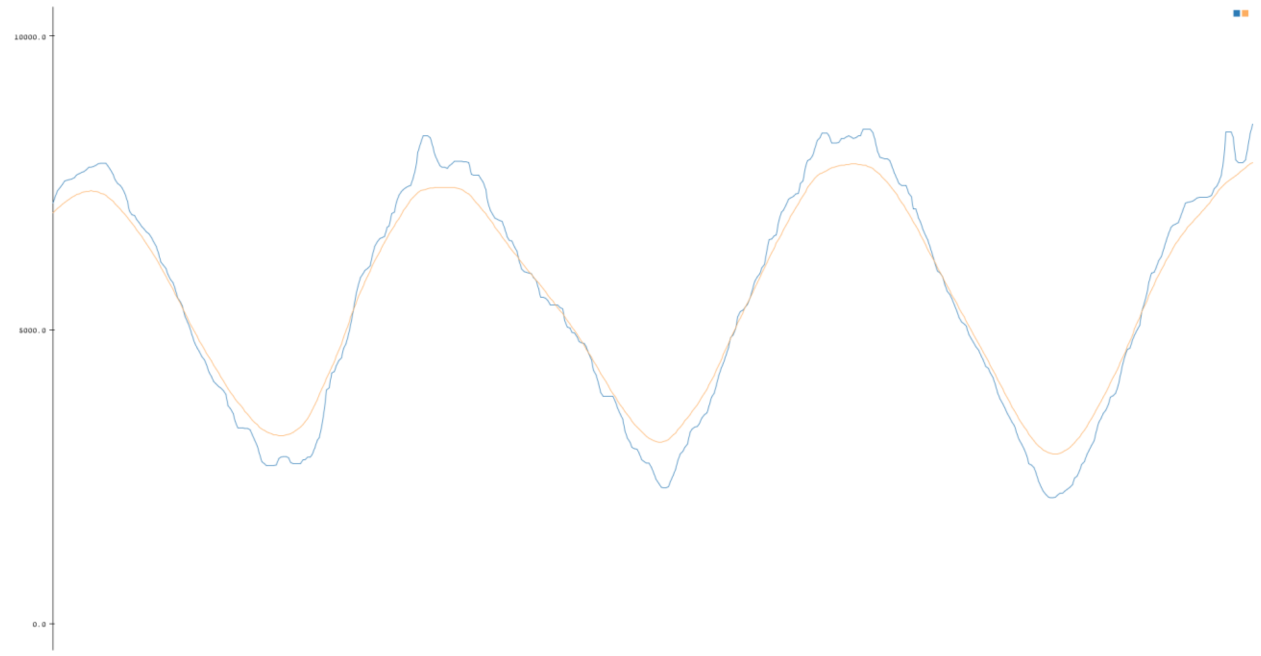
The gyroscope give very accurate and smooth results but the angle drifts from actual angle over time due to integration of error while calculating the angle. Whereas, accelerometer gives very noisy results but it does not drift over time.

Best of both results can be used to get smooth and accurate angle. We are using a complementary filter to fuse both angles calculated from accelerometer and gyroscope.

We are using following formula:

- $(a_a - a_g)\tau^2 dt^2 + 2(a_a - a_g)\tau dt \quad + \mathrm{v}dt$
  - a_a : Angle calculated using accelerometer

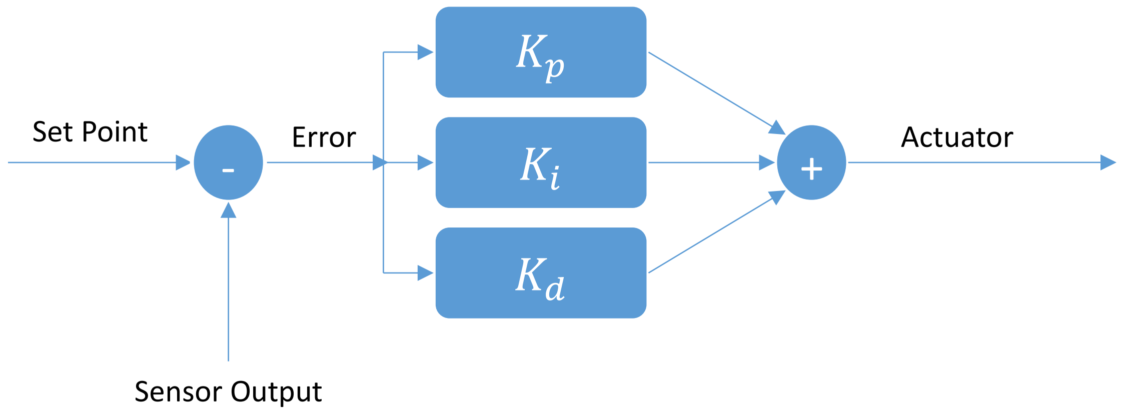
- a_g : Angle calculated using gyroscope
- τ : Convergence factor, how quickly gyroscope angle should converge with accelerometer angle
- v : Angular velocity
- dt : Change in time



Results are quite accurate and smooth. Above graph shows angle calculated using accelerometer (blue) and filtered final angle (orange), when MPU was waved around Y axis.

PID CONTROLLER

We are using PID controller library pointed in reference section. A PID controller controls the PWM input to motors based on the PID constants and the input/setpoint parameters of the PID.



- K_p : Proportional term
- K_i : Integral term
- K_d : Derivative term

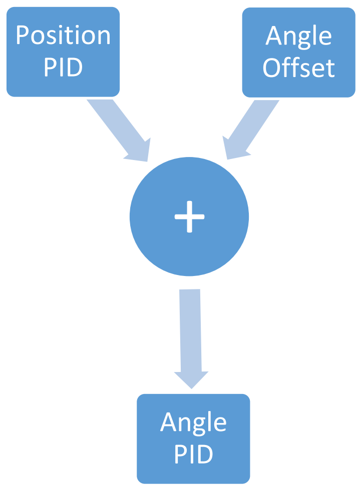
The calculated angle is supplied to the angle PID, and the output of the PID is set to motor PWM load.

At this point bot manage to balance the upright angle, but fails to maintain a fixed position on the surface.

QUADRATURE ENCODER INTEGRATION

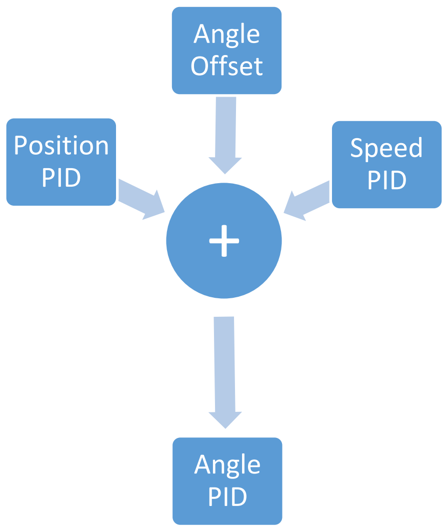
We can get the robot position from the wheel encoders connected to the DC motor. An error angle can be added to the setpoint of the angle PID in the direction in which the robot should move, so that robot would move in that direction to maintain that angle.

A new position PID can be used to generate this offset angle to maintain the fixed robot position.



After testing, it was observed that robot was able to balance and at the same time trying to maintain the starting position, but was not able to maintain its speed, resulting unnecessary movement.

Solution to this problem was to add another offset to the setpoint angle of the angle PID in opposite direction of the velocity of the robot.



RESULTS

After cascading the position PID, speed PID to angle PID, the robot manages to maintain the fixed starting point and also a stable state.

FUTURE WORK

We can directly use a special PID “target speed PID” to achieve desired motor speed. So that, there won’t be any need to add PWM offset to motors.

Use onboard DMP to offload MPU value processing

Both encoder integration

Remote control

Height independent PIDs

REFERENCES

1. PID controller library
    - http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
2. Quadrature encoder
    - http://www.dynapar.com/Technology/Encoder_Basics/Quadrature_Encoder/
    - http://www.ni.com/tutorial/7109/en/