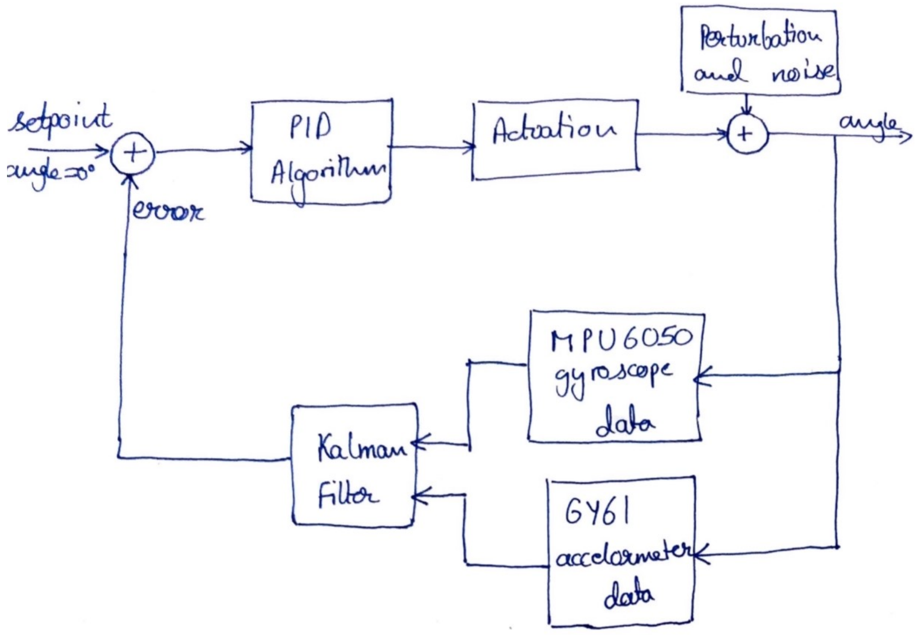
Self Balancing Bike

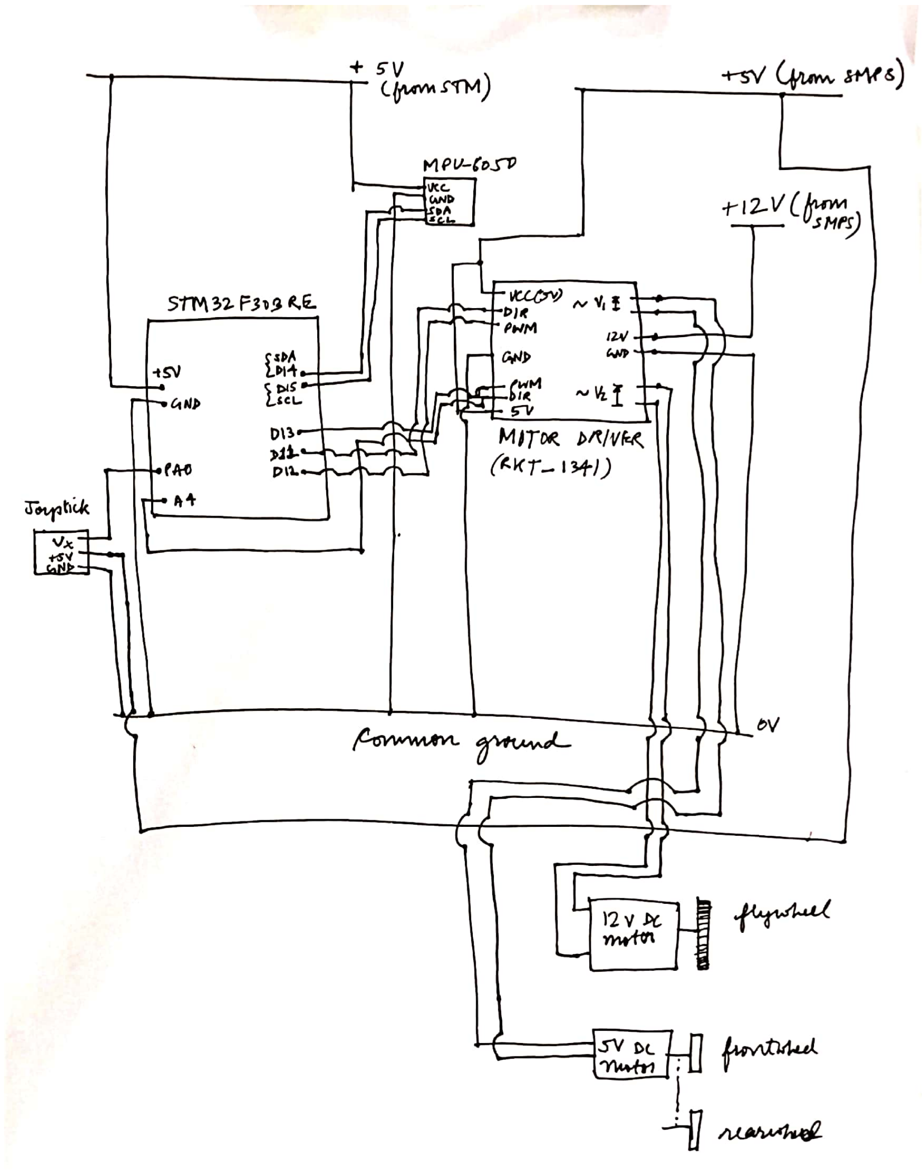
Agastya Seth — Suchit Jain — Saurabh Gangadharan

May 2019

1 Block Diagram



2 Circuit Diagram



3 Objective

The objective of our project was inspired from the latest development in motorbike technology - self balancing. The idea was simple - make a prototype of a motorbike that would stand on its axis, and would counteract any swivel from that position.

The working principle involves various sensors to measure the deviation from its mean position, and a flywheel mounted on the bike, perpendicular to the axis of the wheels of the bike. We would measure the deviation from the defined axis, and make the flywheel rotate in the direction of the movement, which would give the bike angular momentum in the direction opposite to the deviation to counteract the effect of the swivel. This idea is simple yet different. What initially came to our mind is that balancing can be achieved through weight shifting that can be done by fixing rails, perpendicular to the line of motion but in its plane, on which the weight can be made to move through an electric motor. But this is not a very efficient approach because once we approach nearer to the balancing point we'll have to shift weights more rapidly , i.e. the response time should be good which cannot be done through weight-rail mechanism . Also in the weight-rail mechanism there is significant amount of jerks when the weights suddenly change their direction of motion.

Tilt sensing is the crux of this project and the most difficult part as well. We started off thinking that an accelerometer would be sufficient to measure angle as we can measure the direction of gravity w.r.t the accelerometer and get the tilt angle w.r.t the vertical. This approach is correct, but only for slow angular velocities, at high angular velocities the accelerometer tilt angle begins to lag giving wrong tilt data. Then we changed our line of thought and decided to include a Gyro sensor that would measure angular velocity and angle can be found by integrating but this approach fails at slow angular velocities due to gyroscopic drift(small errors in slow velocities integrate and accumulate into a big error).

We would then use Kalman Filtering to combine the readings of both the sensors (accelerometer at low angular velocities and gyro sensor for high angular velocities), thereby giving a more precise tilt angle.

There are a plethora of accelerometer and gyro sensors on the market, which are prototyping-friendly. We decided to use the MPU6050 (acclerometer & gyro sensor) and the GY-61 (only accelerometer). Further, we wanted to make it remote controlled for moving the bike front-and-back. To implement this, we decided to use a pair of XBee node pairs (Coordinator and Endpoint) to act as the receiver for our microcontroller (STM32) and the remote respectively. The remote would take input from a joystick and would send a pair of coordinates to the coordinator.

4 Achievement

We successfully set up communication with the Gryo/Accelerometer Sensor - MPU6050. The z-axis readings (the ones with least variance) were used to calculate the angle of the title of the bike. The drive to the flywheel motor was controlled based on the tilt angle using a PID algorithm. The observed effect of the flywheel rotation on the chassis was as expected. (This involved using the I2C communication and PWM features of the STM contoller.)

The front-wheel drive was controlled by another DC motor. The rotation was controlled using a joystick. This involved successful use of PWM generation and ADC features of the STM32F303RE microcontroller.

Once we started getting values from the sensor, the next line of business was to make the motor of the flywheel follow the deviation from the mean position, using PID control algorithm.

Last but not the least, the integration of the various components was done on the chassis to the extent possible. The motors of the flywheel and front-wheel were mounted on the chassis, along with the wheels and gyro

5 Demonstration

For our demo, we were unable to get the front-wheel motor to work, primarily because it was interfering with our sensor readings. We are still unable to explain why this problem arose, but for demonstration, we thought we'd rather prioritize more on the tilt sensing/resolving side of things as that is the heart of the project.

We mounted the MPU6050 on the bike chassis along with the high-torque DC motor (for the flywheel). The flywheel itself were nothing but two chassis wheels stuck together on the DC motor shaft. The rest of the components would lie on the table, and the bike would be tethered to them using wires. This was done primarily to reduce the weight and dimensions of the bike.

We were successfully able to showcase the tilt resolving function of our bike. The bike would mitigate the tilt by making the flywheel rotate corresponding to the extent and sharpness of the tilt using PID control algorithm.

We were unable to get the XBee to wirelessly transmit joystick coordinates to the STM board. We had tested this functionality in the past, but due to some error in the configuration side of things in the XCTU software, the XBee pair was working anomalously. We, therefore pivoted to connect the joystick directly as an ADC input to the STM board. This didn't really compromise on the functionality of our bike as it was anyway being controlled tethered to the STM board on the table.

6 References

http://students.iitk.ac.in/projects/roboticsclub_selfbalance
http://iedprojects2015.blogspot.com/2015/03/self-balancing.html