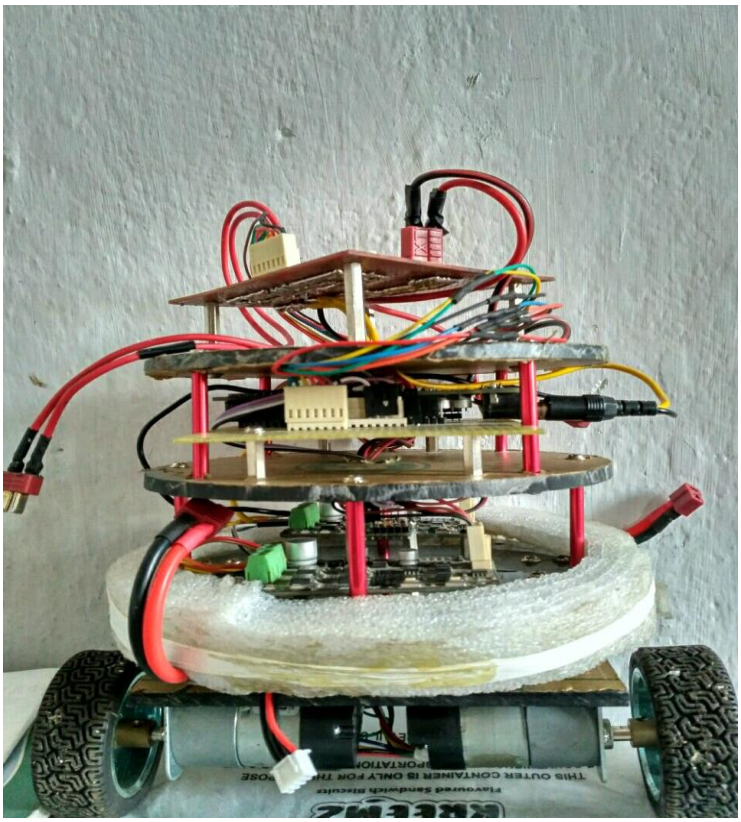
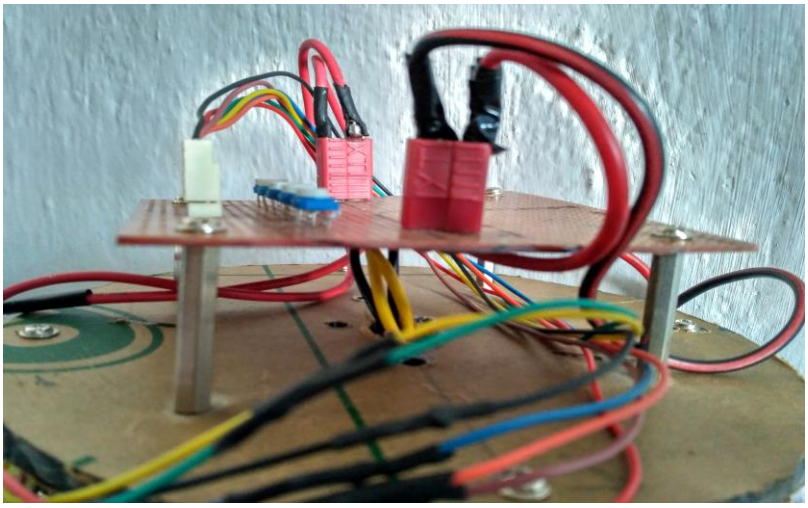
Description of the hardware design of the balance bot:



The above image shows a snap of the balance bot constructed by our team. It consists of four different sections separated by circular plates which are described as :

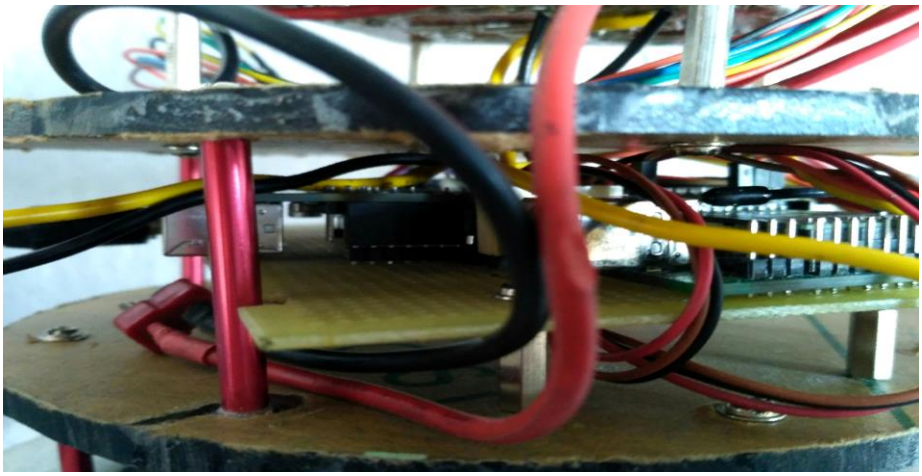
1. Power distribution and 'PID constants control board:



This board of the robot consists of 12 volts lipo jacks for providing power to the Arduino microcontroller and also to the motor drivers for running the two motors.

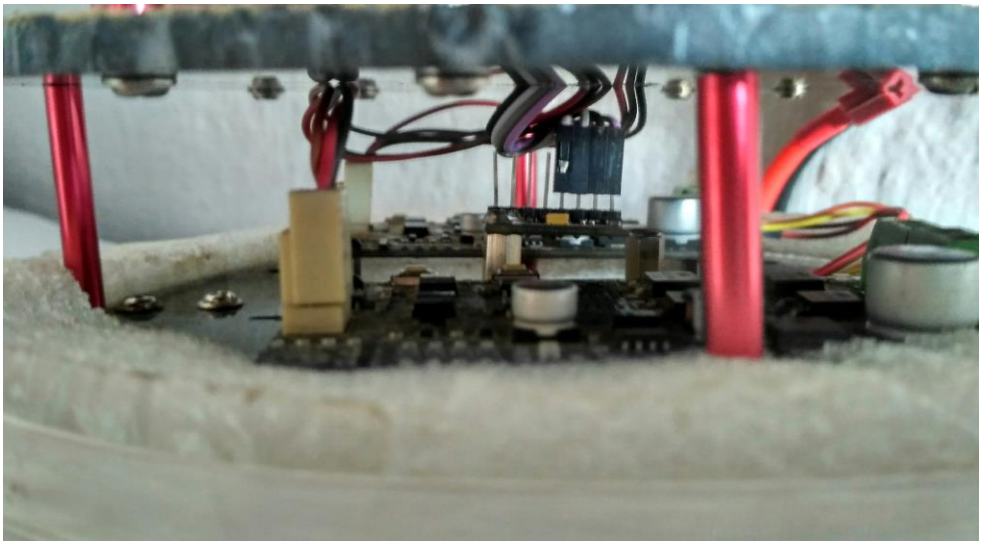
There are potentiometers also to vary the values of PID constants for achieving a fine tuning of the robot for it's desired balancing.

2. The Controlling part:



This part consists of the main controlling unit of the brain. It mainly consists of the microcontroller(Arduino) attached to the perf board and also there is a Xbee inserted into the xbee module which is connected to the Arduino with the help of Rx and Tx pins in addition to vcc and ground. Basically it is the programmer of the Robot.

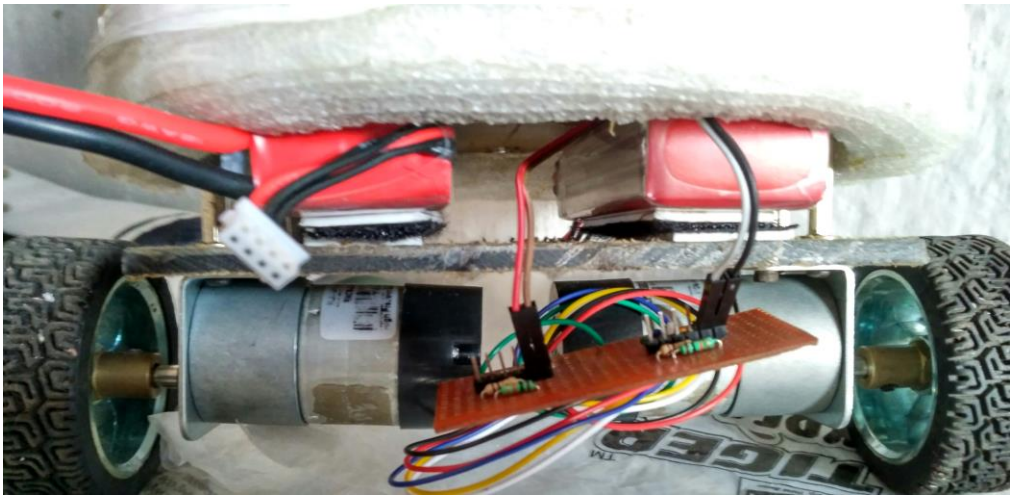
3.The Sensing and driving part:



This part consists of the Inertial Measurement Unit(GY-80) sensor for measuring the tilt angle of the bot with the help of accelerometer and gyroscope sensors inbuilt in it. It communicates with the arduino serially with the help of I2C buses(SCL, SDA).

Besides, this part contains motor drivers for driving the two motors which are used to drive the wheels and the robot as a large.

4. The part consisting of power source and motors:



This part consists of two quadrature encoder motors and board for the connection of six pins of each motor(m1a, m1b, encodera, encoderb, vcc, ground) to the Arduino. Also there are wheels and batteries for providing 12 volts to motor drivers and the arduino through it's 12 volt jack.

Reasons for selecting the design:

1. The top three plates are circular because mass is uniformly distributed in a circle. And also the circular plates are symmetric about any line through the center(diameter) so that the robot can perfectly balance itself in any of it's orientation with respect to the Y-axis. Also the circular bot is symmetric with respect to all the three axes(x, y and z). So according the sensor values, taking desired actions will be easier.
2. In our robot design apart from the top three circular plates there is the lowest rectangular plate which holds the batteries and the board for motor connection. It is basically for safety so as to increase the height of the robot as a whole in a case when the bot tends to fall down. Also the bot in that case would get a little more time to balance itself.
3. The metal studs that are used to connect the circular plates are attached at such positions so that their weight is distributed uniformly.

Challenges faced during designing:

1. The most challenging part was the PID tuning of the robot. Many trials were undertaken by varying the constants kp, ki and kd. The bot had been very shaky. Then after tuning the kd value a little, the bot was able to balance.
2. Another challenge encountered was in cutting the fiber plates into a circular shape for the robot and fixing the microcontroller, sensors, circuit boards on those plates so that the entire mass of the bot is uniformly distributed in each section and the center of gravity of the bot is near the sensor(GY-80).

3. Now the batteries were connected and placed at the top circular plate at first but then it became too difficult to balance the bot. Shifting the batteries towards the lowest plate made our job easy.

Explanation of the PID control Algorithm:

Effect of proportional :

proportional tries to reduce the error i.e it counters the error. it is the current error in the variable taken into account.

proportional is calculated as current error i.e, current value - setpoint . here the set point is the reference point where we want to control the variable.

In balance bot variable is the tilt angle of the bot here 'P' is calculated from equation < P=current_tilt - balancepoint >. Here the balance point is the value of the tilt angle at which bot balances properly and the current tilt is the value of current angle of the bot calculated through the help of sensor. With the help of 'P' bot tries to balance at the balance point angle

Effect of integral:

Integral is the sum of error in the variable taken in the account.

In the balance bot the integral is calculated from equation < I = I + P > Here p is proportional value and I is integral.

The main purpose of the 'integral' term in the PID controller is when the error increases to a greater extent in a particular direction, it is used to reduce it by making the bot to move in the other direction.

Effect of differential:

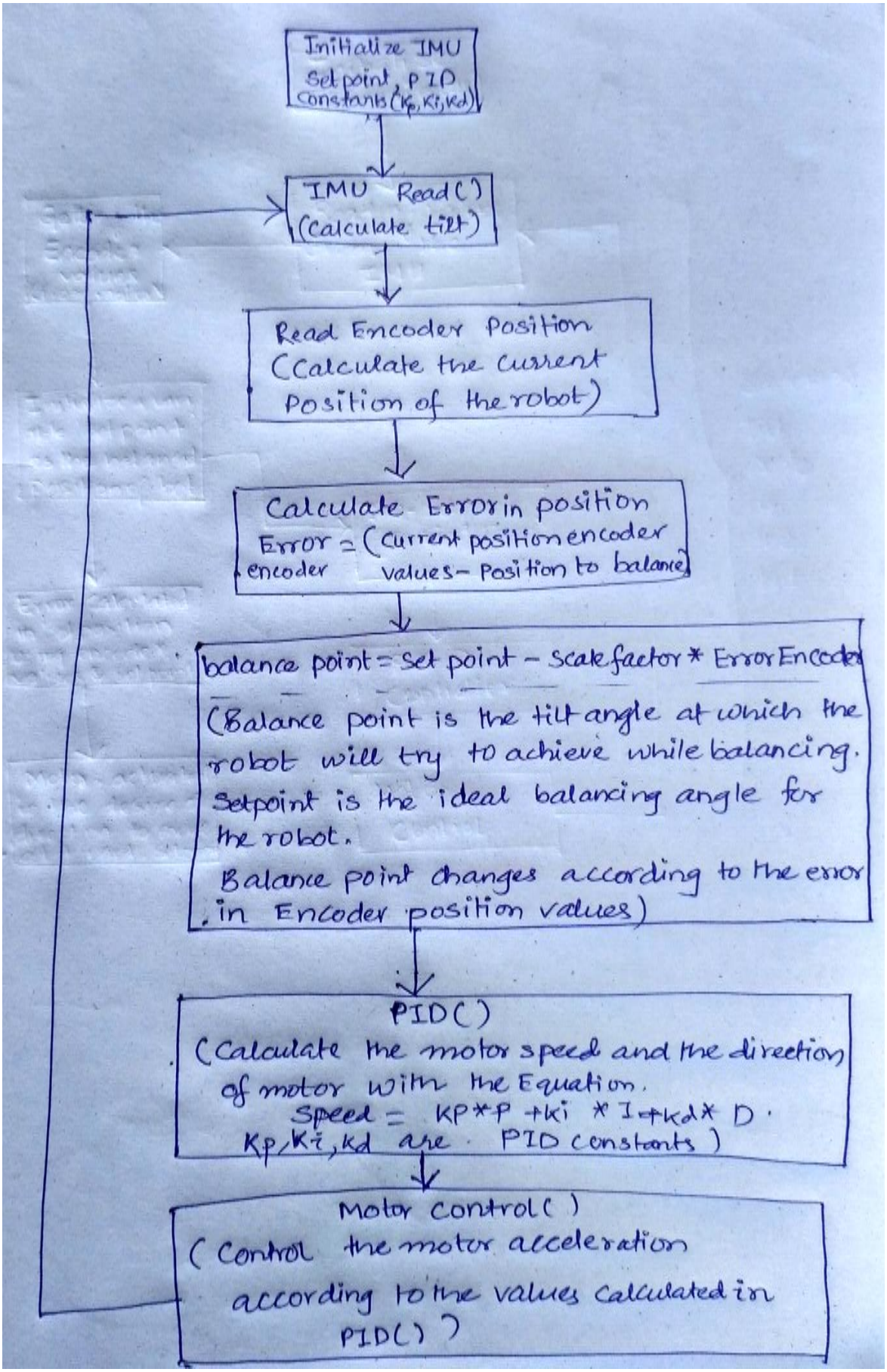
Differential is the change in error i.e the difference between the and current error and the previous error in the variable taken into account.

In the balance bot the diffenential is calculate from the equation < D=P-previous_p > . Here D is the differential to be calculated , P is the proportional and pre_p is the previous proportional .

Diffenential contols the sudden change error i.e in balance bot of bot bends suddenly in any direction it will balance itself by providing a sudden jerk at that instant.

Controlling the Tilt angle and Position of the balance bot

(Along with functional block diagram)


Initialize IMU
Set point, PID constants (Kp, Ki, Kd)
IMU Read()
(Calculate tilt)
Read Encoder Position
(Calculate the current position of the robot)
Calculate Error in position
Error encoder = (Current position encoder values - Position to balance)
balance point = set point - scale factor * Error Encoder
(Balance point is the tilt angle at which the robot will try to achieve while balancing.
Setpoint is the ideal balancing angle for the robot.
Balance point changes according to the error in Encoder position values)
PID()
(Calculate the motor speed and the direction of motor with the Equation.
Speed = KP*P + ki * I + kd * D.
Kp, Ki, Kd are PID constants)
Motor control()
(Control the motor acceleration according to the values calculated in PID())

We control the tilt angle and position of the balance bot as per the functional block diagram in the above image which is described as under:

1. At first the I2Cdevices such as GY-80(IMU) here are initialised for beginning the serial communication over I2C buses(SCL and SDA) between the microcontroller and the IMU. The set point of the bot that is it's balance position is also initialised.

2. The Gy-80 is used to read the values through the help of it's accelerometer and gyroscope sensor and subsequently the combined tilt angle is calculated.

3. Position encoder values were read due to the rotation of the quadrature encoder motor. Then error in position encoder value is calculated as:

Error encoder= Current position encoder value - position encoder value at which the bot is to balance.

4.Then a value called balance point is calculated using the following formula:

balance point = setpoint - scale factor*error encoder

Here, balance point is the tilt angle which the robot will try to achieve while balancing. **Setpoint** is the ideal balancing angle of the robot

Balance point changes according to the error in encoder position values.

This is how the position of the bot is controlled.

5. Then, simultaneously the values of IMU(GY-80) are fed into the PID controller. Subsequently the motor speed and it's direction is calculated from the equation:

speed = kp*p + ki*i + kd*d

where kp,ki,kd are PID constants.

Then commands were given to the motor according to the values computed by the PID controller.Again IMU values were read and the cycle continues.

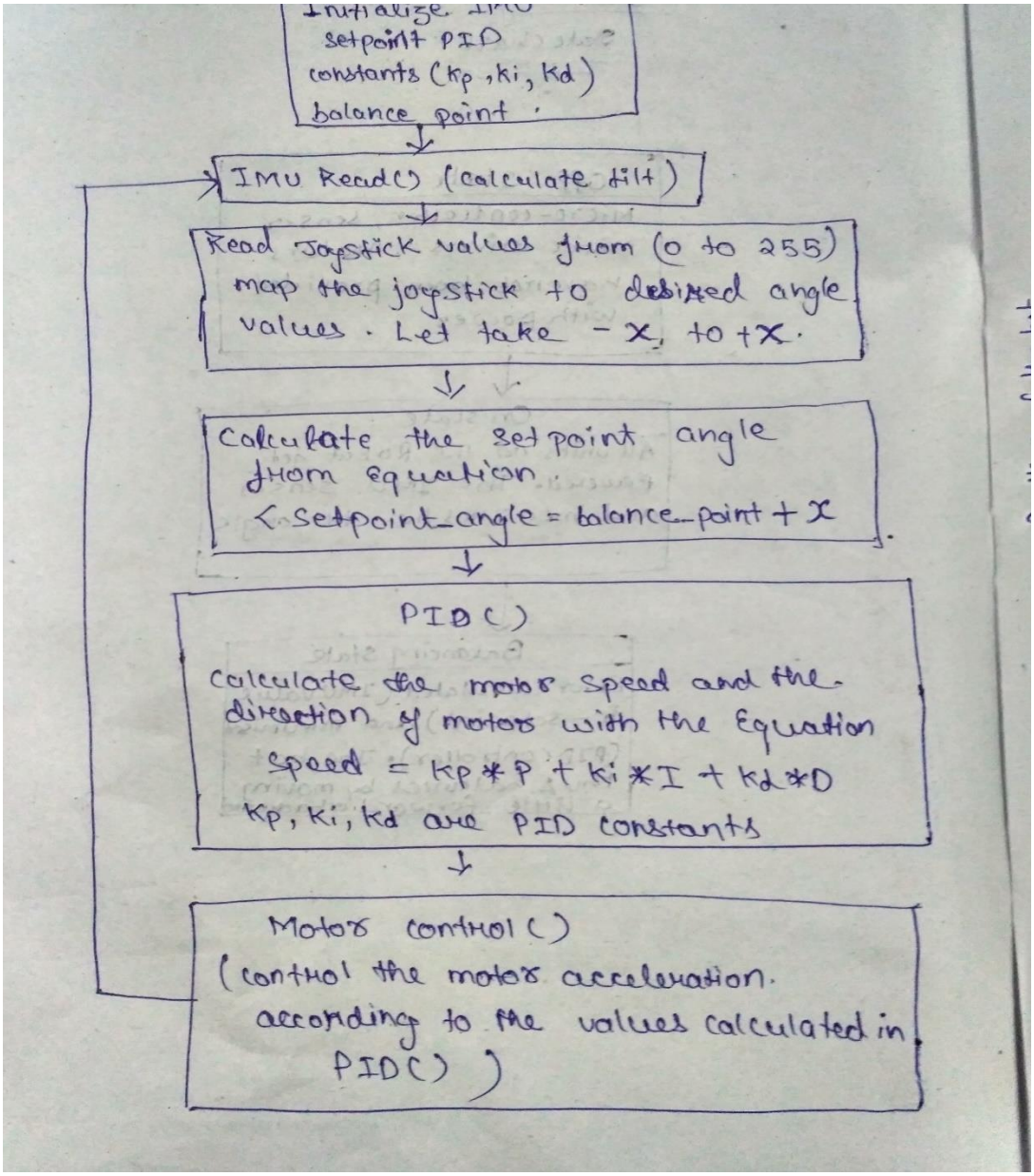
Providing motion commands through the joystick for different actions:

To move the bot in linear motion:

The joystick gives analog values in two dimensional plane considering x and y axes. Along the two axes the analog values range from 0 to 255. For linear motion of the balance bot i.e., forward and backward we need analog values of the joystick only along the y-axis (or the other one as our choice). Here the analog value 128 is the midpoint and analog values from 0 to 255 are mapped to values say from -1 to 1. This mapped value is added to the set point of the balance bot and if the final result is greater than the initial set point, the bot moves forward. Otherwise if the final result is less than the initial set point, the bot moves backward. Accordingly both the motors are given the same speed. The target values for mapping are kept as low as possible to enable the bot to balance itself in the forward as well as backward direction.

In other words if the joystick is moved along the positive y axis, the bot moves forward and if it is moved along the negative y-axis the bot moves backward.

The Functional block diagram for the bot to move in linear motion:

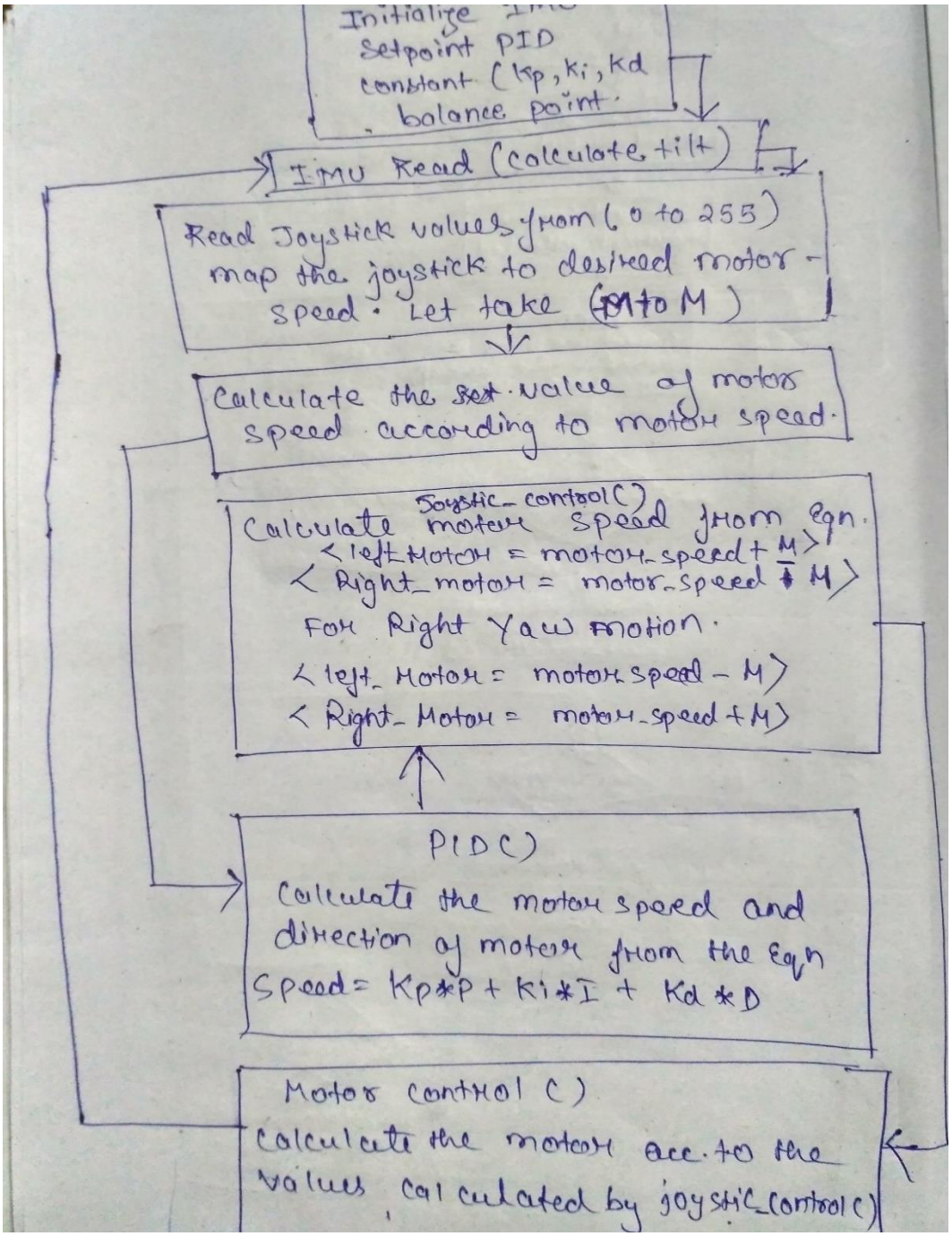


To move the bot in circular motion:

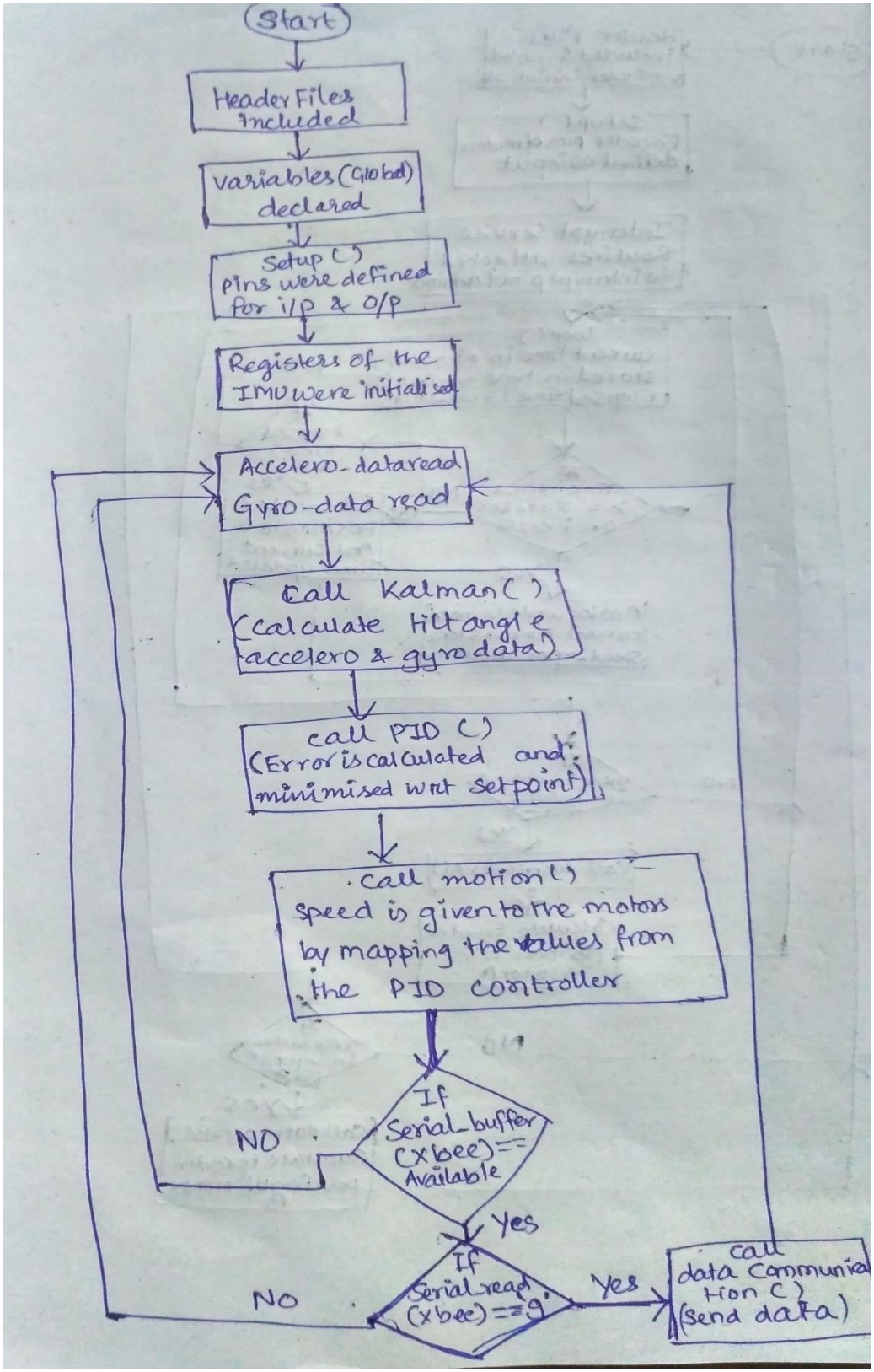
In this case we have to take the help of the other perpendicular axis also. Now speed of one of the wheel is kept constant and for the other wheel we have to map the analog values of the joystick along say x-axis i.e., 0 to 255 to say –M to +M (M is chosen according to our convenience). Thus this mapped value is added to speed of the other wheel so as to move in a circle.

Then while moving the joystick left would move the bot in a left circle (speed of the right wheel is mapped) and moving the joystick right would move the bot in a right circle (speed of the left wheel is mapped).

The Functional block diagram for the bot to move in circular motion:



Flow chart for Robot construction and Balancing:



State chart of the Robot: