**CENG4480 Embedded System Development and Applications**

**Computer Science and Engineering Department**

**The Chinese University of Hong Kong**

**Laboratory 9: Self-balancing Robot (2) (software)**

November, 2017

**Introduction**

In this lab you will complete your self-balancing robot by coding the program and tuning the PID constants such that make your robot to standing up.



**Figure 1. CENG4480 self-balancing robot program flow chart**

**Objectives**

* To learn how to develop software to control the system
* To familiar with the practical work in engineering

Procedures

1. **Calculate the angle from accelerometers values**

* On the provided skeleton program Lab9.ino add the angle calculation codes as following:

Ayz=atan2(RwAcc[1],RwAcc[2])\*180/PI; //angle measured by accelerometer

Ayz-=offset; //adjust to correct balance point

1. **Add the complement and Kalman filters**

* On the provided skeleton program Lab9.ino add the complement and Kalman filters codes as following:

Angy = 0.998\*(Angy+GyroIN[0]\*interval/1000)+0.002\*Ayz; //complement filter

kang = kalmanCalculate(Angy, GyroIN[0],interval); //kalman filter

Serial.println(kang);

1. **Add the PID calculation and update the speed of motors**

* On the provided skeleton program Lab9.ino add the PID calculation and update the speed of motors as following:

if ((abs(kang)>=minangle)&&(abs(kang)<maxangle)){

delta=kang;

diff = delta - last;

diff2 = delta - last2;

diff = constrain(diff,-maxdiff,maxdiff);

diff2 = constrain(diff2,-maxdiff,maxdiff);

last2 = last;

last = delta;

LRspeed = P\*delta + I\*accu\*interval\*0.001 + D\*(diff\*100+diff2\*100)/interval;

accu+=delta;

accu = constrain(accu,-maxaccu,maxaccu);

}

else {

LRspeed = 0;

accu = 0;

last=0;

diff=0;

}

1. **Calibrate the offset**

* After adding all codes in Lab9.ino then upload it to the Arduino board
* Hold the robot vertically
* Open the COM window and find out the offset value
* Change the offset value in Lab9.ino accordingly
* Upload the Lab9.ino to Arduino board again

1. **Tuning the PID constants**

* Increase the P value in the step of 50 upload to the Arduino each time until the robot start to oscillate (move back and forth)
* Increase I in the step of 50 so that the robot accelerates faster when off balance
* Increase D in the step of 10 so that the robot would move about its balanced position more gentle, and there shouldn’t be any significant overshoots
* If first attempt doesn’t give the satisfying results, reset PID values and start over again with different value of P
* Repeat the steps until you find a certain PID value which gives the satisfactory results
* A fine tuning can be done to further increase the performance of PID system
* In fine tuning, PID values are restricted to neighboring values and effects are observed in practical situations

1. **Demo to your TAs**

* The longer your robot standing up time the higher you will get the bonus marks

**END**