REFERENCE LIST

[1] B.Armstrong-Helouvry, P. Dupont, C. Canudas de Wit, “A survey of models, analysis tools and compensation methods for control of machines with friction,” Automatica, vol.30. no. 7, pp 1083-1138, 1994.

[2] C.Iurian, F.Ikhouane, J.Rodellar, R.Grino, (2005) “Identifaction of a system with dry friction”, University of Barcelona.

[3] Wahyudi, (2003) “Friction Identification and Compensation for High Precision Motion Control Sysyem – Part 1: Friction Identification”, *Processing of industrial Electronics Seminar (IES).*

[4] I. Virgala, P. Frankovsky, M. Kenderova, (2013) “Friction Effect Analysis of a DC Motor” Transaction on American Journal of Mechanical Engineering, Vol.1 No. 1, 1-5.

[5] Rajasekhar Gundala; Narain, A., "Friction compensation using neural networks applicable to high precision motion control systems in manufacturing," *Industrial Informatics, 2005. INDIN '05. 2005 3rd IEEE International Conference on* , vol., no., pp.808,812, 10-12 Aug. 2005.

[6] Wahyudi; Tijani, I.B., "Friction compensation for motion control system using multilayer feedforward network," Information Technology, 2008. ITSim 2008. International Symposium on , vol.4, no., pp.1,7, 26-28 Aug. 2008.

[7] B.Amstrong, (1993) “Stick slip and control in low-speed motion”, *IEEE transaction on automatic control*, vol. 38, No. 10.

[8] Lee, C.K.; Pang, W.H ; , "Adaptive control to parameter variations in a DC motor control system using fuzzy rules,” *Second International* Conference on Intelligent Systems Engineering, pp.247-254, Sept. 1994.

[9] Xin Qian,Youyi Wang:, “Motion Control of a Linear Brushless DCMotor Drive System with Nonlinear FrictionCompensation”2004 Intematlonal Conference on Power System Technology - POWERCON 2004Slngapore vol.2, pp. 1276 - 1281, 21-24 Novembar 2004.

[10] Ciliz, M.K.; Tomizuka, M., "Neural network based friction compensation in motion control," Electronics Letters , vol.40, no.12, pp.752,753, 10 June. 2004.

[11] Tzes, A.; Pei-Yuao Peng; Guthy, J., "Genetic-based fuzzy clustering for DC-motor friction identification and compensation," *Control Systems Technology, IEEE Transactions on* , vol.6, no.4, pp.462,472, Jul 1998.

[12] Babaie, M.; Khanzadi, M.R., "Precision motion control for an X-Y table using the LOLIMOT Neuro-Fuzzy Friction compensation," *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on* , vol., no., pp.2300,2304, 15-18 Dec. 2007

[13] Dinesh Chinthaka, M.K.C.; Punchihewa, R.U.G.; Illangarathne, N.C.; Harsha S.Abeykoon, A.M., "Practical Friction Compensation for DC Motors Using the Disturbance Observer" *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems (ICAEES),2014 International Conference on,* 22-23 Aprill 2014.

[14] Dayarathna, H.A.N.D.; Prabuddha, L.L.G.; Ariyawansha, K.L.D.N.J.; Chinthaka, M.K.C.D.; Abeykoon, A.M.H.S.; Branesh Pillai, M., "Sensorless contact position estimation of a mobile robot in pushing motion," *Circuits, Power and Computing Technologies (ICCPCT), 2013 International Conference on* , vol., no., pp.344,349, 20-21 March 2013

[15] Abeykoon A.M.Harsha S, Senevirathne Hasala R. (2012). Disturbance Observer Based Current Controller for a Brushed DC Motor. *Proceeding of the 6th IEEE Conference on Information and Automation,* September, 2012, Beijing, China,59-64.

[16] T.Murakami, K.Ohnishi, (1993) " Observer Based Motion Control-Application to Robust Control and Parameter Identification", IEEE transaction on Industrial Electronics.

[17] Ali, Y.S.E.; Noor, S.B.M.; Bashi, S.M.; Hassan, M.K.; ,"Microcontroller performance for DC motor speed controlsystem,"Power Engineering Conference, 2003. PECon 2003. *Proceedings. National* , vol., no., pp. 104- 109, 15-16 Dec. 2003.

[18] Guoshing. Huang, Shuocheng. Lee (2008). PC-based PID speed control in DC motor. *Proceedings of the International Conference on Audio, Language and Image Processing*, July, 2008, Shanghai, 400-407.

[19] Wahyudi,(2008) "Artificial Intelligent Based Friction Identification using Multilayer Feedforward Network", *Proceeding of the 4th International Colloquium on signal Processing and its Applications (CSPA).*

[20] Mizuochi, M.; Tsuji, T.; Ohnishi, K.; , "Improvement of disturbancesuppression based on disturbance observer," *Advanced Motion Control*, 2006. 9th IEEE International Workshop on ,vol.,no., pp.229-234, 0-0 0.

[21] Dinesh Chinthaka, M.K.C.; Punchihewa, R.U.G.; Harsha S.Abeykoon, A.M., "Disturbance Observer based Friction Compensator for a DC Motor" Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI), 2014 *International Conference on*, 14-17 May 2014.

[22] J Annstrong-Helouvry B., Dupont P. and De Wit C., 1994. "A survey of models, analysis tools and compensation method for the control of machines with friction", Automatica, Vol. 30, No.7 (1994) pp. 10831138.

[23] K. Ohnishi, M. Shibata, T. Murakami : "Motion control for advancedmechatronics," Mechatronics, IEEE/ASME Transactions on , vol.1, no.1pp.56-67, Mar 1996.

[24] Katsura, S.; Irie, K.; Ohishi, K.; , "Wideband Force Control by Position-Acceleration Integrated Disturbance Observer," *Industrial Electronics,IEEE Transactions on* , vol.55, no.4, pp.1699-1706, April 2008.

[25] T.Murakami, K.Ohnishi, (1993) "Disturbance Observer Based Motion Control- Application to Robust Control and Parameter Identification", IEEE transaction on Industrial Electronics.

[26] Katsura.;Y.Matsumoto,K.Ohnishi:"Modelingofforcesensingandvalidationof disturbanceobserverforforcecontrol,"*IndustrialElectronicsSociety,2003. IECON'03.The29thAnnualConferenceoftheIEEE*,vol.1,no.,pp.291-296vol.1,2-6Nov.2003.

[27] K.Ohishi, K.Ohnishi and K.Miyachi,"Torque-speed regulation of DC motor based on load torque estimation," *IEEJ IPEC-Tokyo,*, 1983,vol.2, pp. 1209-1216.

[28] Asif Sabanovic, Kouhei Ohnishi (2011) *Motion control systems*, IEEE press John Willey and sons (Asia) pte Ltd, (First Edition).

[29] A.M Harsha S.Abeykoon, Kouhei Ohnishi: "Improvement of Tactile Sensation of a Bilateral Forceps Robot by a Switching Virtual Model," *Transaction on Advanced Robotics*", Vol. 8, pp. 789-806 (18), 2008.

[30] A.M Harsha S.Abeykoon, Kouhei Ohnishi: "Virtual tool for bilaterally controlled forceps robot-for minimally invasive surgery," *Transaction on international Journal of Medical Robotics and Computer Assisted Surgery*", ISSN 1478-5951, VOL 3; No 3, pp. 271-280, 2007.

[31] www.mbed.org

[32] M. Branesh Pillai: "Parameters Estimation for Motion Control And Friction Compensation", University of Moratuwa, MSc thesis, July 2013.

[33] Hasala R. Senevirathne, A.M.Harsha S.Abeykoon, M. Branesh Pillai:"Disturbance Rejection Analysis of a Disturbance Observer basedVelocity Controller," ICIAfS'12, Proceeding of the 6th IEEE*Conference*, Sep 27-29, 2012.

APPENDIX A:POSITION BASED STATIC FRICTION ESTIMATION ALGORITHM

```
#include "mbed.h"
#include "rtos.h"
#include "qeihw.h"
#include "SDFileSystem.h"

QEIHW qei_s(QEI_DIRINV_NONE, QEI_SIGNALMODE_QUAD,
QEI_CAPMODE_2X, QEI_INVINX_NONE );

#define G1 1.0//100.0
#define G2 1.0 //500.0

// Safety for mbed unused pins
DigitalIn safety_19(p19);
DigitalIn safety_25(p25);
DigitalIn safety_26(p26);
DigitalIn safety_21(p21);
DigitalIn safety_22(p22);
DigitalIn safety_27(p27);
DigitalIn safety_28(p28);

PwmOutpwm_s_clk(p23);  // clockwise rotation pwm pin
PwmOutpwm_s_anticlk(p24);    //anti-clockwise rotation pwm pin

DigitalOutReset_AB_S(p29);
DigitalOutReset_CD_S(p30);

DigitalInM_Dir(p9);
DigitalInS_Dir(p10);

DigitalInIndex(p12);

AnalogIncurrent_sensor_s_p(p17); // current sensor input for SLAVE positive
AnalogIncurrent_sensor_s_n(p18); // current sensor input for SLAVE negative

DigitalOutled1(LED1);
DigitalOutled3(LED3);

DigitalInsignal(p20);  //external signal to close the file

Timer timer;                  // For the controller
FILE *fp;
Ticker ticker;
```

```
intdt_us= 100;                  // define main loop time in us
floatdt = dt_us/1000000.0;      //loop time in seconds for calculations

intcounter_time;
int counter =0;
intcounter_old =0;

//Current Sensor Directions
intSlave_Direction = 0;

// Encoder Constants
floatconstencoder_pulses_s = 10000.0;

#define PI 3.141592653
#define Gd 0.5 //1200.0 //cutoff frequency of the DOB
//#define Gv 0.0


// PID parameters for Current - Loop
floatconstIkp_s = 4.5, Iki_s = 10.0, Ikd_s = 0.1;

// PID parameters for velocity - Loop
floatI_ref_s = 0.0, I_err_s = 0.0, I_res_s = 0.0, I_tmp_s = 0.0, tem_I_s = 0.0, d_I_s =
0.0,I1_act_s=0.0;
floatv_ref_s = 60.0, v_err_s = 0.0, v_res_s = 20.0,v_res_s_rpm = 0.0, v_tmp_s = 0.0,
tem_v_s = 0.0, d_v_s = 0.0;
floatduty_s = 0.0;

// Low pass filter gain for Current - loop
floatconstG_filcon_I_s = 1.0;
// Low pass filter gain for Current - loop
floatconst G_filcon_I1_s = 100.0;


// Storing actual current flow
floatI_act_s = 0.0;

// Parameters to calculate current rotational speed
floatencoder_s_prv = 0.0;
floatencoder_s_now = 0.0;

// Motor Constant and Inertia
floatconstJ_const_s = 0.0000800; //0.0000268;
floatconstKt_const_s = 0.134;
floatconstKt_constinv_s = 1.0/0.134;
```

```
floattmp_s = 0.0,ob_sum_s = 0.0,ob_sum_s1=0.0;

floati_com_s = 0.0;
floatfric_m = 0.0,fric_s = 0.0,i_rto_m = 0.0,i_rto_s = 0.0;


floatx_res_s = 0.0;
floatve_sum_s = 0.0;
floatpwm_I_S= 0.0;
floatpid_V_I_S= 0.0;

float point =0.0;
floatde_s =0.0, temp_s1=0.0, v_res_s1 = 0.0, temp_s= 0.0, ddx_s = 0.0;

floatpos_ref=0.0, pos_err=0.0, p_pos_err=0.0, i_pos_err=0.0;
int32_t position_old=0.0,tem_d_pos_err=0.0, tmp_d_pos_err=0.0, d_pos_err=0.0;
float count=0.0,current=0.0,duty=0.0,pos_ref_old=0.0, PWM_pos=0.0,current_res;
floatI_res = 0,I1_act = 0,Duty = 0,I_act = 0;
int i=0,cycles=0;
int32_t position = 0;
int negative=0,positive=0;
int n =20000,r=1;
intcountup=0,t=1;

voidpwm_init(void) {

pwm_s_anticlk.period_us(10);
pwm_s_clk.period_us(10);

    // Set the ouput duty-cycle, specified as a percentage (float)

pwm_s_anticlk.write(0.0f);
pwm_s_clk.write(0.0f);

   //ENABLE RUNNING MODE  (H BRIDGE ENABLE)

Reset_AB_S = 1;
Reset_CD_S = 1;
}

//$$$$$$$$ CURRENT CONTROLLER $$$$$$$$$$$$$$$

floatcurrent_pid_s(){
if (pos_ref_old == pos_ref){
```

```
Slave_Direction = S_Dir.read();
if(Slave_Direction == 0){                              // clockwise
I_res_s=  current_sensor_s_p.read();
        I1_act_s = -1.0*((I_res_s*3.3) / 0.717075441532258);
}else if(Slave_Direction == 1){
I_res_s=  current_sensor_s_n.read();                   // anticlockwise
        I1_act_s = ((I_res_s*3.3) / 0.724138445564516);
    }
I_act_s += G_filcon_I1_s*(I1_act_s-I_act_s)*dt;

I_err_s = I_ref_s - I_act_s;
I_tmp_s += Iki_s * dt * I_err_s;
tem_I_s += dt * d_I_s;
d_I_s = (I_act_s - tem_I_s) * G_filcon_I_s;

pwm_I_S=((I_err_s * Ikp_s) + I_tmp_s + (d_I_s * Ikd_s));

   }
return duty;
}

floatkp_pos= 0.0005;//0.0005;//0.00006;
floatki_pos = 0.03;//0.03;//0.004;//0.0008;
floatkd_pos =0.000001;//0.000002;//0.000001; //0.00649999999;
floatpos_filcon = 10000;//2000;//100;//10000;


//$$$$$$$ POSITION CONTROLLER $$$$$$$$$$$

voidposition_pid_s() {
if (pos_ref_old != pos_ref){
qei_s.SetDigiFilter(480UL);
qei_s.SetMaxPosition(0xFFFFFFFF);
position = qei_s.GetPosition();
pos_err = pos_ref - position;
p_pos_err = kp_pos*pos_err;
i_pos_err += ki_pos*pos_err*dt;
tem_d_pos_err += pos_filcon*dt*tmp_d_pos_err;
tmp_d_pos_err = pos_err - tem_d_pos_err;
d_pos_err = kd_pos*tmp_d_pos_err;

PWM_pos = p_pos_err + i_pos_err + d_pos_err;
duty = PWM_pos;
countup++;
}
}
```

```
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$$$ POSITION VERIFICATION $$$$$$$$$$$$

voidpositon_check (){
if((I_ref_s == 0.0)){
if ((pos_ref!=position )){
negative++;
}
if ((pos_ref==position )){
positive++;
}

if ((pos_ref==position )&&(pos_ref_old != pos_ref)&&(positive>(negative+100))){
pos_ref_old = pos_ref;
position_old = position;
negative=0;
positive=0;
countup=0;
   t=1;
duty = 0.0;
   }
}
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$ POSITION SHIFT $$$$$$$$$$$$$$$$$$$

voidposition_shift(){
if (countup>=200000){
pos_ref = position_old + t;
t++;
countup=0;
negative=0;
positive=0;
   }
  }

//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$$$ PWM GENERATOR $$$$$$$$$$$$$$$$$$$$

voidPWM_Generator_s(){

duty_s = duty;
```

```
if (duty_s> 0.0) {
if (duty_s> 0.98) {
duty_s = 0.3;
        }
pwm_s_clk = 0.0;
pwm_s_anticlk = duty_s;
    }

if (duty_s< 0.0) {
if (duty_s< -0.98) {
duty_s = -0.3;
        }
pwm_s_anticlk = 0.0;
pwm_s_clk = -1.0*duty_s;
    }
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$$$$$ POSITION CONTROLLER $$$$$$$$$$$

voidposition_control(){
if (pos_ref_old == pos_ref){
position = qei_s.GetPosition();
if (position_old == position){
I_ref_s += 0.001;
count = I_ref_s;
duty = I_ref_s;
    }

if (position_old != position) {
current = I_ref_s;
I_res = I_res_s;
    I1_act = I1_act_s;
I_act = I_act_s;
    Duty = duty;
pos_ref = position_old + 1;
I_ref_s = 0.0;
duty = 0.0;
    }

}
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

intcleanup_module(void){
```

```
pwm_s_anticlk = 0.0;
pwm_s_clk = 0.0;

    //RESET H BRIDGE
Reset_AB_S = 0;
Reset_CD_S = 0;

   led1=0;

   led3=0;


return 0;
}

//RTOS
// Control Part - main code  $$$$$$$$
voidControl_body() {
current_pid_s ();
position_control();
position_pid_s();
PWM_Generator_s();
positon_check ();
position_shift();
PWM_Generator_s();
        led1=!led1;
counter++;
   }
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$ DATA STORING THREAD $$$$$$$$$$$$
void thread_2(void const *argument){

SDFileSystemsd(p5, p6, p7, p8, "sd");
        FILE *fp = fopen("/sd/v100.dat", "w");

if(fp == NULL) {
for(int i=0;i<5;i++){
        led3=!led3;
wait(1.0);
           }
        }

while((position_old)<=40000){

if((current != 0)){
```

```
fprintf(fp,"%d %f %f %f %f \n",position_old,current,I_res,I1_act,I_act);

counter_old=counter;
        led3=!led3;
current = 0;
           }
        }
fclose(fp);
timer.stop();
cleanup_module();
ticker.detach ();
    }

int main() {
dt =dt_us/1000000.0;
pwm_init();
timer.start();
    Thread thread2(thread_2, NULL, osPriorityNormal, (DEFAULT_STACK_SIZE *
4.0));
ticker.attach_us(&Control_body, dt_us);
   }
 //$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

 //_____ END_______________
```

APPENDIX B:POSITION BASED TOTAL FRICTION ESTIMATION ALGORITHM

```
#include "mbed.h"
#include "rtos.h"
#include "qeihw.h"
#include "SDFileSystem.h"
#include <math.h>
#define QEI_RESET_POSOnIDX QEI_CON_RESPI

QEIHW qei_s(QEI_DIRINV_NONE, QEI_SIGNALMODE_QUAD,
QEI_CAPMODE_2X, QEI_INVINX_NONE);

// Filter Coefficient
#define G1 1.0
#define G2 1.0
#define PI 3.141592653

// Safety for mbed unused pin
DigitalIn safety_11(p11);
DigitalIn safety_13(p13);
DigitalIn safety_14(p14);
DigitalIn safety_19(p19);
DigitalIn safety_20(p20);
DigitalIn safety_25(p25);
DigitalIn safety_26(p26);
DigitalIn safety_21(p21);
DigitalIn safety_22(p22);
DigitalIn safety_27(p27);
DigitalIn safety_28(p28);
DigitalIn safety_9(p9);
DigitalIn safety_15(p15);
DigitalIn safety_16(p16);
DigitalInIndex(p12);

PwmOutpwm_s_clk(p23);          // clockwise rotation pwm pin for SLAVE
PwmOutpwm_s_anticlk(p24);      //anti-clockwise rotation pwm pin for SLAVE

DigitalOutReset_AB_S(p29);
DigitalOutReset_CD_S(p30);

DigitalInS_Dir(p10);
AnalogIncurrent_sensor_s_p(p17); // current sensor input for SLAVE positive
```

```
AnalogIncurrent_sensor_s_n(p18); // current sensor input for SLAVE negative

DigitalOutled1(LED1);
DigitalOutled3(LED3);

Timer timer;                    // For the controller
FILE *fp;
Ticker ticker;

intdt_us= 100;                  // define main loop time in us
floatdt;      //loop time in seconds for calculations
floatramp_time = 3.0;
floatdelta_v = 0.0;

intcounter_time;
int counter=0;
intcounter_old=0;

//Current Sensor Directions
intMaster_Direction = 0;

// Encoder Constants
floatconstencoder_pulses_s = 10000.0;
floatconst G_filcon_I1_s = 1.0; // 350.0

// PID parameters for velocity - Loop
float  vkp_s = 0.070, vki_s = 0.0000001, vkd_s =0.09;//p=0.02, i=0.000001
floatv_ref_s = 0.0,v_err_s = 0.0, v_res_s = 0.0;
float  d_v_err_s = 0.0, prev_v_err_s=0.0,I_v_err_s=0.0,de_s =0.0;
floatduty_s = 0.0;
floattorque_dob = 0.0,torque_motor=0.0;

// parameters for Current - Loop
floatI_ref_s = 0.0, I_err_s = 0.0, I_res_s = 0.0, I_tmp_s = 0.0, tem_I_s = 0.0, d_I_s =
0.0,ddx_s = 0.0;
floatI_act_s = 0.0,I1_act_s=0.0,I_ref_s1 = 0.0,II_act_s=0.0;

//float constG_filcon_v_s = 300.0; //1.0;//100.0;//300.0; //100.0;//1.0
floatencoder_s_prv = 0.0;
floatencoder_s_now = 0.0;

// Motor Constant and Inertia
floatconstKt_const_s = 0.134;
floatconstJ_const_s = 0.0000268;
floatconstKt_constinv_s = 1.0/0.134;
floattmp_s = 0.0,ob_sum_s = 0.0,ob_sum_s1 = 0.0;
```

```
floati_com_s = 0.0;
floatx_res_s = 0.0;
floatve_sum_s = 0.0;
floatpwm_I_S= 0.0;
floatpid_V_I_S= 0.0;

float temp_s1=0.0, v_res_s1 = 0.0, temp_s= 0.0;
floatdde_s = 0.0, v_res_s_prv=0.0,accelaration_s=0.00,temp_s2=0.0,temp_s3
=0.0,acce_res_s =0.0;


intcounter_b=0,count=0;
int32_t position = 0;
int32_tpos=20000,p=0,r=0;
float current=0.0,duty=0.0;
int a=0,b=200,c=400,d=600,e=800,f=1000,g=1200,h=1400;
int i=1600,j=1800,k=2000,l=2200,m=2400,n=2600,aa=2800,ab=3000;
int ac=3200,ad=3400,ae=3600,af=3800,ag=4000,ah=4200,ai=4400,aj=4600;
intba=4800,bb=5000,bc=5200,bd=5400,be=5600,bf=5800,bg=6000,bh=6200;
int bi=6400,bj=6600,ca=6800,cb=7000,cc=7200,cd=7400,ce=7600,cf=7800;
int cg=8000,ch=8200,ci=8400,cj=8600,da=8800,db=9000,dc=9200,dd=9400;
int de=9600,df=9800;
int ea=100,eb=300,ec=500,ed=700,ee=900,ef=1100,eg=1300,eh=1500,ei=1700,
ej=1900;
int fa=2100,fb=2300,fc=2500,fd=2700,fe=2900,ff=3100,fg=3300,fh=3500,fi=3700,
fj=3900;
intga=4100,gb=4300,gc=4500,gd=4700,ge=4900,gf=5100,gg=5300,gh=5500,
gi=5700,gj=5900;
int ha=6100,hb=6300,hc=6500,hd=6700,he=6900,hf=7100,hg=7300,hh=7500,
hi=7700,hj=7900;
int ia=8100,ib=8300,ic=8500,id=8700,ie=8900,ifa=9100,ig=9300,ih=9500,ii=9700,
ij=9900;

voidpwm_init(void) {
pwm_s_anticlk.period_us(10);
pwm_s_clk.period_us(10);

 // Set the ouput duty-cycle, specified as a percentage (float)
pwm_s_anticlk.write(0.0f);
pwm_s_clk.write(0.0f);

   //ENABLE RUNNING MODE  (H BRIDGE ENABLE)
Reset_AB_S = 1;
Reset_CD_S = 1;
}
```

```
//$$$$$ CURRENT ESTIMATION $$$$$$$$$$$$$$$$$$
voidcurrent_pid_s(){
Master_Direction = S_Dir.read();

if(Master_Direction == 0){                              //master clockwise
I_res_s = current_sensor_s_p.read();
        I1_act_s = -1.0*((I_res_s*3.3) / 0.74787687701613);

}else if(Master_Direction == 1) { //master anticlockwise
I_res_s=  current_sensor_s_n.read();
        I1_act_s = 1.0*((I_res_s*3.3) / 0.713239227822580);

    }
II_act_s += G_filcon_I1_s*(I1_act_s-I_act_s)*dt;
I_act_s = II_act_s;

 }
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$$$ VELOCITY COMMAND $$$$$$$$$$$$$$$$$$$$$$$$$$$$

float rpm = 500;
voidvelocity_command(){

if(abs(v_ref_s) < abs(rpm)){
v_ref_s += delta_v;}
else {
v_ref_s=rpm;
 }
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$$$$ VELOCITY CONTROLLER $$$$$$$$$$$$$$$

voidvelocity_pid_s() {

qei_s.SetDigiFilter(480UL);
qei_s.SetMaxPosition(0xFFFFFFFF);
position = qei_s.GetPosition();
    //buffer[counter]=position;
encoder_s_now = position * 2.0 * PI / encoder_pulses_s;
de_s = encoder_s_now - encoder_s_prv;
encoder_s_prv = encoder_s_now;
    //v_res_s = ((de_s/dt_us)*(1000000.0 * 60.0))/(2.0*PI);
ddx_s=((de_s/dt_us)*(1000000.0 * 60.0))/(2.0*PI);
temp_s += G1*ddx_s*dt;
```

```
   temp_s1 += G2*v_res_s*dt;
v_res_s = temp_s - temp_s1;
v_err_s = v_ref_s - v_res_s;
   //d_v_err_s = (v_err_s - prev_v_err_s) * G_filcon_v_s;
d_v_err_s = (v_err_s - prev_v_err_s) *dt ;
prev_v_err_s = v_err_s;
I_v_err_s += v_err_s * dt_us;
pid_V_I_S= vkp_s * v_err_s + vkd_s * d_v_err_s + vki_s * I_v_err_s;
   p=position/10000;
if ((position>=10000*p)){
position = (position-10000*p);
   }
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$$$ PWM GENERATOR $$$$$$$$$$$$$$$$$$$$$

voidPWM_Generator_s() {
duty_s = pid_V_I_S;
if (duty_s> 0.0) {
if (duty_s> 0.95) {
duty_s = 0.95;
      }
pwm_s_clk = 0.0;
pwm_s_anticlk = duty_s;
   }

if (duty_s< 0.0) {
if (duty_s< -0.95) {
duty_s = -0.95;
      }
pwm_s_anticlk = 0.0;
pwm_s_clk = -1.0 * duty_s;
   }
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

intcleanup_module(void){
pwm_s_anticlk = 0.0;
pwm_s_clk = 0.0;
Reset_AB_S = 0;
Reset_CD_S = 0;
   led1=0;
   led3=0;
return 0;
}
```

```
//$$$$$$$$$$ POSITION CHECK $$$$$$$$$$$$$$$$$$

voidposition_check(){
if
(((position==a)&&(a<100))||((position==b)&&(b<300))||((position==c)&&(c<500))||
((position==d)&&(d<700))||
((position==e)&&(e<900))||((position==f)&&(f<1100))||((position==g)&&(g<1300))
||((position==h)&&(h<1500))||((position==i)&&(i<1700))||((position==j)&&(j<1900
))
||((position==k)&&(k<2100))||((position==l)&&(l<2300))||((position==m)&&(m<25
00))||((position==n)&&(n<2700))||((position==aa)&&(aa<2900))||((position==ab)&
&(ab<3100))
||((position==ac)&&(ac<3300))||((position==ad)&&(ad<3500))||((position==ae)&&(
ae<3700))||((position==af)&&(af<3900))||((position==ag)&&(ag<4100))||((position=
=ah)&&(ah<4300))
||((position==ai)&&(ai<4500))||((position==aj)&&(aj<4700))||((position==ba)&&(ba
<4900))||((position==bb)&&(bb<5100))||((position==bc)&&(bc<5300))||((position=
=bd)&&(bd<5500))
||((position==be)&&(be<5700))||((position==bf)&&(bf<5900))||((position==bg)&&(
bg<6100))||((position==bh)&&(bh<6300))||((position==bi)&&(bi<6500))||((position
==bj)&&(bj<6700))
||((position==ca)&&(ca<6900))||((position==cb)&&(cb<7100))||((position==cc)&&(
cc<7300))||((position==cd)&&(cd<7500))||((position==ce)&&(ce<7700))||((position
==cf)&&(cf<7900))
||((position==cg)&&(cg<8100))||((position==ch)&&(ch<8300))||((position==ci)&&(
ci<8500))||((position==cj)&&(cj<8700))||((position==da)&&(da<8900))||((position=
=db)&&(db<9100))
||((position==dc)&&(dc<9300))||((position==dd)&&(dd<9500))||((position==de)&&(
de<9700))||((position==df)&&(df<9900))||((position==ea)&&(ea<200))||((position=
=eb)&&(eb<400))
||((position==ec)&&(ec<600))||((position==ed)&&(ed<800))||((position==ee)&&(ee
<1000))||((position==ef)&&(ef<1200))||((position==eg)&&(eg<1400))||((position==
eh)&&(eh<1600))
||((position==ei)&&(ei<1800))||((position==ej)&&(ej<2000))||((position==fa)&&(fa
<2200))||((position==fb)&&(fb<2400))||((position==fc)&&(fc<2600))||((position==f
d)&&(fd<2800))
||((position==fe)&&(fe<3000))||((position==ff)&&(ff<3200))||((position==fg)&&(fg
<3400))||((position==fh)&&(fh<3600))||((position==fi)&&(fi<3800))||((position==fj)
&&(fj<4000))
||((position==ga)&&(ga<4200))||((position==gb)&&(gb<4400))||((position==gc)&&(
gc<4600))||((position==gd)&&(gd<4800))||((position==ge)&&(ge<5000))||((position
==gf)&&(gf<5200))
||((position==gg)&&(gg<5400))||((position==gh)&&(gh<5600))||((position==gi)&&(
gi<5800))||((position==gj)&&(gj<6000))||((position==ha)&&(ha<6200))||((position=
=hb)&&(hb<6400))
```

```
||((position==hc)&&(hc<6600))||((position==hd)&&(hd<6800))||((position==he)&&(
he<7000))||((position==hf)&&(hf<7200))||((position==hg)&&(hg<7400))||((position
==hh)&&(hh<7600))
||((position==hi)&&(hi<7800))||((position==hj)&&(hj<8000))||((position==ia)&&(ia
<8200))||((position==ib)&&(ib<8400))||((position==ic)&&(ic<8600))||((position==id
)&&(id<8800))
||((position==ie)&&(ie<9000))||((position==ifa)&&(ifa<9200))||((position==ig)&&(i
g<9400))||((position==ih)&&(ih<9600))||((position==ii)&&(ii<9800))||((position==ij
)&&(ij<10000))){
pos = position;
current = I_act_s;
duty = duty_s;
counter_b++;
}
}
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

//$$$$$$ DATA WRITING THREAD $$$$$$$$$$$$$$$$

void thread_2(void const *argument){

SDFileSystemsd(p5, p6, p7, p8, "sd");
        FILE *fp = fopen("/sd/v-10.dat", "w");   //.........................

if(fp == NULL) {
for(int i=0;i<5;i++){
        //led1=!led1;
wait(1.0);
            }
          }
while(counter_b<10000){
  //if(counter>=(counter_old+1)){

if (((a<100)&&(pos==a))){
a++;
count++;
fprintf(fp, "%d %f %f\n",pos,duty,current );
        led3=!led3;
          }
```

Shouldbe repeated until 10000

```
if ((pos==ij)&&(ij<10000)){
ij++;
count++;
fprintf(fp, "%d %f %f \n",pos,current,duty );
```

```
        led3=!led3;
        }
    }
fclose(fp);
timer.stop();
cleanup_module();
ticker.detach ();
wait(1.0);
  }

//$$$$$$$$$$$ Control Part - main code $$$$$$$

voidControl_body() {

velocity_command();
velocity_pid_s ();
current_pid_s ();
if (counter>50000) {

position_check();
        }
PWM_Generator_s();
counter++;
        led1=!led1;
   }
//$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

int main() {
dt =dt_us/1000000.0;
delta_v = (dt * rpm)/ramp_time;
pwm_init();
timer.start();
     Thread thread2(thread_2, NULL, osPriorityNormal, (DEFAULT_STACK_SIZE
* 2.25));
ticker.attach_us(&Control_body, dt_us);
   }

//$$$$$$$$$_____________ END____________________
```