REFERENCES

1. F. Grasser, A. D'arrigo, S. Colombi, and A. C. Rufer, "JOE: A Mobile, Inverted Pendulum," *IEEE Trans. on Industrial Electronics*, 49, 1, (2002): 107–114.

2. D. Kamen, "Segway HT," technical report, URL: `http://www.segway.com/`, 2001.

3. M. A. Clark, J. B. Field, S. G. McMahon, and P. S. Philps, "EDGAR, A Self-Balancing Scooter," technical report, The University of Adelaide, Australia, October 2005.

4. D. P. Anderson, "Nbot, a Two Wheel Balancing Robot [Online]," technical report, Southern Methodist University, URL: `http://www.geology.smu.edu/~dpa-www/robo/nbot/`, 2003.

5. H. Steven, "Legway, a Two Wheel Balancing Robot," technical report, URL: `http://www.teamhassenplug.org/robots/legway/`, 2002.

6. N. Shiroma, O. Matsumoto, S. Kajita, and K. Tani, "Cooperative Behaviour of a Wheeled Inverted Pendulum for Object Transportation," in *Proc. IEEE/RSJ International Conf. on Interlligent Robots and Systems*, (1996): 396–401.

7. K. Pathak, J. Franch, and S. Agrawal, "Velocity Control of a Wheeled Inverted Pendulum by Partial Feedback Linearization," *Proc. IEEE Conf. on Decision and Control*, 4, (2004): 3962–3967.

8. K. Pathak, J. Franch, and S. Agrawal, "Velocity and Position Control of a Wheeled Inverted Pendulum by Partial Feedback Linearization," *IEEE Trans. Robotics*, 21, 3, (2005): 505–513.

9. J. Searock and B. Browning, "Learning to Prevent Failure States for a Dynamically Balancing Robot," in *Proc. AAAI*, (2005): 1312–1317.

10. Y. Takahashi, T. Takagaki, J. Kishi, and Y. Ishii, "Back and Forward Moving Scheme of Front Wheel Raising for Inverse Pendulum Control Wheel Chair Robot," in *Proc. IEEE International Conf. on Robotics and Automation*, (2001): 3189–3194.

11. Y. Takahashi, N. Ishikawa, and T. Hagiwara, "Inverse Pendulum Controlled Two Wheel Drive System," in *Proc. SICE Annual Conf.*, (2001): 25–27.

12. H. G. Nguyen, G. Kogut, R. Barua, and A. Burmeister, "A Segway RMP-Based Robotic Transport System," in *Proc. SPIE Mobile Robots XVII*, (2004): 1–12.

13. T. Sugihara, Y. Nakamura, and H. Inoue, "Real-Time Humanoid Motion Generation Through ZMP Manipulation Based on Inverted Pendulum Control," in *Proc. IEEE International Conference on Robotics and Automation*, (2002): 1404–1409.

14. J. Angeles, "An Innovative Drive for Wheeled Mobile Robots," in *IEEE/ASME Trans. on Mechatronics*, (2005): 43–49.

15. R. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Trans. /ASME-Journal of Basic Engineering*, 82, Series D, (1960): 35–45.

16. A. J. Baerveldt and R. Klang, "A Low-Cost and Low-weight Attitude Estimation System for an Autonomous Helicopter," in *Proc. IEEE International Conference on Intelligent Engineering Systems*, (1997): 391–395.

17. B. Barshan and H. F. Durrant-Whyte, "An Inertial Navigation System for a Mobile Robot," in *Proc. IEEE International Conference on Intelligent Engineering Systems*, (1993): 2243–2248.

18. B. Barshan and H. F. Durrant-Whyte, "Inertial Navigation System for a Mobile Robot," in *Proc. IEEE International Conference on Intelligent Engineering Systems*, (1995): 328–342.

19. J. Borenstein and L. Feng, "Gyrodometry: a New Method for Combining Data from Gyros and Odometry in Mobile Robots," in *Proc. IEEE International Conference on Robotics and Automation*, (1996): 423–428.

20. K. Komoriya and E. Oyama, "Position Estimation of a Mobile Robot using Optical Fiber Gyroscope," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, (1994): 143–149.

21. T. Lahdhiri, C. L. Carnal, and A. T. Alouani, "Cart-Pendulum Balancing Problem Using Fuzzy Logic Control," in *Proc. IEEE Southeastcon on 'Creative Technology Transfer - A Global Affair'*, (1994): 393–397.

22. V. Williams and K. Matsuoka, "Learning to Balance the Inverted Pendulum Using Neural Networks," in *IEEE International Joint Conf. on Neural Networks*, (1991): 214–219.

23. H. Kajiwara, P. Apkarian, and P. Gahinet, "LPV Techniques for Control of an Inverted Pendulum," in *IEEE Control Syst. Mag.*, (1999): 44–54.

24. K. Xu and X. Duan, "Comparative Study of Control Methods of Single-Rotational Inverted Pendulum," in *Proc. International Conf. on Machine Learning and Cybernetics*, (2002): 776–778.

25. D. D. Gobbo, M. Napolitano, P. Famouri, and M. Innocenti, "Experimental Application of Extended Kalman Filtering for Sensor Validation," *IEEE Trans. Control Sys. Tech.*, 9, 2, (2001): 376–380.

26. H. Liu and G. Pang, "Accelerometer for Mobile Robot Positioning," *IEEE Trans. Industry Appl.*, 37, 3, (2001): 812–819.

27. G. Pang and H. Liu, "Evaluation of a Low-cost MEMS Accelerometer for Distance Measurement," *Jounal of Intelligent and Robotic Systems*, 30, (2001): 249–265.

28. G. Welch and B. Gary, "An Introduction to Kalman Filtering," technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC. USA, 2002.

29. K. J. Åström and B. Wittenmark, *Computer-Controlled Systems.* United States of America: Prentice-Hall International, Inc., 1997.

30. J. L. Meriam and L. G. Kraige, *Engineering Mechanics.* England: JOHN WILEY & SONS, 2004.

31. T. Braunl, *Embedded Robotics.* Germany: Springer-Verlag, 2003.

32. P. Apkarian and P. Gahinet, "A Convex Characterization of Gain-Scheduled $\mathcal{H}_\infty$ Controllers," *IEEE Trans. Aut. Control*, 40, 5, (1995): 853–864.

33. C. W. Scherer, "Robust Mixed Control and Linear Parameter-Varying Control with Full Block Scaling," in *Advanced in Linear Matrix Inequality Methods in Control*, (1999): 187-207.

34. S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control.* England: JOHN WILEY & SONS, 1996.

35. K. Pitoonmanit, "An $\mathcal{H}_\infty$ Controller Design for Rotary Inverted Pendulum Using Linear Parameter-Varying Control Technique with Full Block Multiplier," Master's thesis, Department of Electrical Engineering, Faculty of Engineering, Chulalongkorn University, 2005.

APPENDICES

Appendix A

Matlab Source Code for Simulations of Controllers Design

A.1 Program for Simulation Open Loop Impulse Response

```
%==========================================================================
%                    Open_loop_impulse_response.m
%   Simulation of open loop impulse response of Balancing robot
%          This model includes the motor dynamics and gear
%                         Author: Lychek Keo
%                         22 February 2006
%==========================================================================
%==========================================================================
%                          Variable initialization
%==========================================================================
close all;
clear all;
g = 9.81; %Gravity(m/s^2)
r = 0.093; %Radius of wheel(m)
Mw = 1.282; %Mass of wheel(kg)
Mp = 9.6; %Mass of body(kg)
Jw = 0.0111 %Inertia of the wheel(kg*m^2)
Jp = 0.36; %Inertia of the body respect to ax x(kg*m^2)
Jpd = 0.563;%Inertia of the body respect to ax z(kg*m^2)
L = 0.152; %Length to the body's center of mass(m)
Dis = 0.35;
%Motor's variables
Km = 0.0333; %Motor torque constant (Nm/A)
Ke = 0.0333; %Back EMF constant (Vs/rad)
Ra = 0.62; %Nominal Terminal Resistance (Ohm)
nm = 0.85; %Motor efficiency
Jm = 0.000635; %Moment Inertia of the rotor
% Va voltage applied to motors for controlling the pendulum
%Gear's variables
ng = 0.8; %Gear efficiency
Kg = 23; %Gear ratio
%==========================================================================
%                  Parameters initialization calculation
%==========================================================================
Jeq = Jw+ng*Kg^2*Jm;
beta = (2*Mw+(2*Jeq/r^2)+Mp);
alpha = (Jp*beta+2*Mp*L^2*(Mw+Jeq/r^2));
%==========================================================================
%                              System matrices
%                    Denominator for the A and B matrices
%==========================================================================
A = [0 1 0 0;
0 (2*ng*nm*Km*Kg^2*Ke*(Mp*L*r-Jp-Mp*L^2))/(Ra*r^2*alpha) ((Mp*L)^2*g)/alpha 0;
0 0 0 1;
```

```
0 (2*ng*nm*Km*Kg^2*Ke*(r*beta - Mp*L))/(Ra*r^2*alpha) (Mp*g*L*beta)/alpha 0]
B = [ 0;
(2*ng*nm*Km*Kg*(Jp + Mp*L^2 - Mp*L*r))/(Ra*r*alpha);
0;
(2*ng*nm*Km*Kg*(Mp*L-r*beta)/(Ra*r*alpha))]
C = [1 0 0 0;
0 0 1 0]
D = [0;
0]
%=========================================================================
%                    Transfer function form of the state space model
%=========================================================================
[num,den] = ss2tf(A,B,C,D)
%=========================================================================
%                    Obtaining the impulse response of the system
%=========================================================================
T = 0:0.02:10;
U = zeros(size(T));
U(1) = 1; %input voltage
[Y,X] = lsim(A,B,C,D,U,T);
%=========================================================================
%          Plot open loop impulse response and pole-zero map of the system
%=========================================================================
plot(T,-Y);
title('Open_loop_impulse_response_of_the_system','fontsize',15);
ylabel('Position[m],Angle[rad]','fontsize',15);
xlabel('Time[s]','fontsize',15);
legend('Robot_Position','Tilt_angle');
axis([0 2.2 0 1]);
grid;
figure(2);
H = tf(1,den);
pzmap(H);
title('Pole-Zero_Map','fontsize',16);
ylabel('Imaginary_Axis','fontsize',16);
xlabel('Real_Axis','fontsize',16);
%=========================================================================
%                          End simulation open loop impulse response
%=========================================================================
```

A.2 Program for Design LQR Controller

```
%=========================================================================
%                                LQR_control.m
%                  Simulation of Balancing MIP with LQR control
%                  This model includes the motor dynamics
%                              Author: Lychek Keo
%                              22 February 2006
%=========================================================================
%                          Variable initialization
%=========================================================================
close all;
clear all;
g = 9.81; %Gravity(m/s^2)
r = 0.093; %Radius of wheel(m)
Mw = 1.282; %Mass of wheel(kg)
Mp = 9.6; %Mass of body(kg)
Jw = 0.0111; %Inertia of the wheel(kg*m^2)
```

```
Jp = 0.36; %Inertia of the body(kg*m^2)
L = 0.152; %Length to the body's center of mass(m)
%Motor's variables
Km = 0.0333; %Motor torque constant (Nm/A)
Ke = 0.0333; %Back EMF constant (Vs/rad)
Ra = 0.62; %Nominal Terminal Resistance (Ohm)
nm = 0.85; %Motor efficiency
Jm = 0.000635; %Moment Inertia of the rotor
% Va voltage applied to motors for controlling the pendulum
%Gear's variables
ng = 0.8; %Gear efficiency
Kg = 23; %Gear ratio
%Initial angle of pendulum
dTheta = 20*pi/180;
%=========================================================================
%                        Parameters initialization calculation
%=========================================================================
Jeq = Jw+ng*(Kg^2)*Jm;
beta = (2*Mw+(2*Jeq/(r^2))+Mp);
alpha = (Jp*beta + 2*Mp*(L^2)*(Mw + Jeq/(r^2)));
%=========================================================================
%                                System matrices
%                        Denominator for the A and B matrices
%=========================================================================
A = [0 1 0 0;
0 (2*ng*nm*Km*(Kg^2)*Ke*(Mp*L*r-Jp-Mp*(L^2)))/(Ra*(r^2)*alpha) (((Mp*L)^2)*g)/alpha 0;
0 0 0 1;
0 (2*ng*nm*Km*(Kg^2)*Ke*(r*beta - Mp*L))/(Ra*(r^2)*alpha) (Mp*g*L*beta)/alpha 0]
B = [ 0;
(2*ng*nm*Km*Kg*(Jp + Mp*(L^2) - Mp*L*r))/(Ra*r*alpha);
0;
(2*ng*nm*Km*Kg*(Mp*L-r*beta)/(Ra*r*alpha))]
C = [1 0 0 0;
0 0 1 0]
D = [0;
0]
%=========================================================================
%                        Obtaining the eigenvalues of the system matrix
%=========================================================================
%The eigenvalues of the system matrix A
%A positive value will indicate an unstable system
p = eig(A)
%=========================================================================
%                                LQR control design
%=========================================================================
%Q = C'*C is a 4 x 4 weighting matrix for the outputs
%Q could well be Identity matrix with size same with system matrix A
%R is a 1 x 1 weighting matrix for the input
%x is the weighting for the robot position
%y is the weighting for the pendulum position
x = 2000; y = 1000;
Q = [x 0 0 0;0 0 0 0;0 0 y 0;0 0 0 0];
R =1;
BRinverse = B*inv(R)*B';
P = are(A,BRinverse,Q);
%Feedback Gains for the system
K = inv(R)*B'*P
T = 0:0.02:5;
```

```
U = zeros(size(T));
U(1)= 24;
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
[Y,X] = lsim(Ac,Bc,Cc,Dc,U,T);
[n m] = size (X);
for i = 1:n
UU(i) = -K*X(i,:)';
end
new_poles = eig(Ac)
figure(1)
plot(T,[X(:,1) X(:,2)]);
xlabel('Time_[s]','fontsize',16);
title('_Impulse_response_of_the_plant_with_LQR_control','fontsize',16);
ylabel('Position[m],Velocity[m/s]','fontsize',16);
legend('X','XDot');
grid;
figure(2);
plot(T,[X(:,3) X(:,4)]);
xlabel('Time_[s]','fontsize',16);
title('_Impulse_response_of_the_plant_with_LQR_control','fontsize',16);
ylabel('Angle[rad],Angular_velocity[rad/s]','fontsize',16);
legend('Phi','PhiDot');
grid;
figure(3);
plot(T,UU);
xlabel('Time_[s]','fontsize',16);
ylabel('Optimal_control_u(t)','fontsize',16);
title('_Impulse_response_of_the_plant_with_LQR_control','fontsize',16);
grid;
%=========================================================================
%                          End LQR control design
%=========================================================================
```

A.3 Program for Design Pole-Placement Controller

```
%=========================================================================
%                          Pole-Placement.m
%          Simulation of Balancing Robot with Pole-Placement control
%              This model includes the motor dynamics and gear
%                          Author: Lychek Keo
%                          22 February 2006
%=========================================================================
%=========================================================================
%                          Variable initialization
%=========================================================================
close all;
clear all;
g = 9.81; %Gravity(m/s^2)
r = 0.093; %Radius of wheel(m)
Mw = 1.282; %Mass of wheel(kg)
Mp = 9.6; %Mass of body(kg)
Jw = 0.0111; %Inertia of the wheel(kg*m^2)
Jp = 0.36; %Inertia of the body(kg*m^2)
L = 0.152; %Length to the body's center of mass(m)
%Motor's variables
```

```
Km = 0.0333; %Motor torque constant (Nm/A)
Ke = 0.0333; %Back EMF constant (Vs/rad)
Ra = 0.62; %Nominal Terminal Resistance (Ohm)
nm = 0.85; %Motor efficiency
Jm = 0.000635; %Moment Inertia of the rotor
% Va voltage applied to motors for controlling the pendulum
%Gear's variables
ng = 0.8; %Gear efficiency
Kg = 23; %Gear ratio
%Initial angle of pendulum
dTheta = 20*pi/180;
%=========================================================================================
%                          Parameters initialization calculation
%=========================================================================================
Jeq = Jw+ng*(Kg^2)*Jm;
beta = (2*Mw+(2*Jeq/(r^2))+Mp);
alpha = (Jp*beta + 2*Mp*(L^2)*(Mw + Jeq/(r^2)));
%=========================================================================================
%                                    System matrices
%                              Denominator for the A and B matrices
%=========================================================================================
A = [0 1 0 0;
0 (2*ng*nm*Km*(Kg^2)*Ke*(Mp*L*r-Jp-Mp*(L^2)))/(Ra*(r^2)*alpha) (((Mp*L)^2)*g)/alpha 0;
0 0 0 1;
0 (2*ng*nm*Km*(Kg^2)*Ke*(r*beta - Mp*L))/(Ra*(r^2)*alpha) (Mp*g*L*beta)/alpha 0]
B = [ 0;
(2*ng*nm*Km*Kg*(Jp + Mp*(L^2) - Mp*L*r))/(Ra*r*alpha);
0;
(2*ng*nm*Km*Kg*(Mp*L-r*beta)/(Ra*r*alpha))]
C = [1 0 0 0;
0 0 1 0]
D = [0;
0]
%=========================================================================================
%                          Obtaining the eigenvalues of the system matrix
%=========================================================================================
%The eigenvalues of the system matrix A
%A positive value will indicate an unstable system
p = eig(A)
%=========================================================================================
%                                  Pole Placement control design
%=========================================================================================
%The system matrix have to be full rank for Pole Placement control
rankMIP = rank(ctrb(A,B))
P = [-1.2+j*2 -1.2-j*2 -6+j*5 -6-j*5]
K = place(A,B,P)
T = 0:0.02:5;
U = zeros(size(T));
U(1) = 24;
Ac = [(A-B*K)];
Bc = [B];
Cc = [C];
Dc = [D];
[Y,X] = lsim(Ac,Bc,Cc,Dc,U,T);
[n m] = size (X);
for i = 1:n
UU(i) = -K*X(i,:)';
end
```

```
figure(1)
plot(T,[X(:,1) X(:,2)]);
xlabel('Time [s]','fontsize',16);
title('Impulse response of the plant with complex conjugate poles','fontsize',16);
ylabel('Position [m], Velocity [m/s]','fontsize',16);
legend('X','XDot');
grid;
figure(2);
plot(T,[X(:,3) X(:,4)]);
xlabel('Time [s]','fontsize',16);
title('Impulse response of the plant with complex conjugate poles','fontsize',16);
ylabel('Angle [rad], Angular velocity [rad/s]','fontsize',16);
legend('Phi','PhiDot');
grid;
figure(3);
plot(T,UU);
xlabel('Time [s]','fontsize',16);
ylabel('Control input u(t)','fontsize',16);
title('Impulse response of the plant with complex conjugate poles','fontsize',16);
grid;
%==========================================================================
%                          End Pole-Placement control design
%==========================================================================
```

A.4 Program for LPV Controller Synthesis

```
%==========================================================================
%                                LPV_control.m
%                  Simulation of Balancing Robot with LPV control
%                    This model includes the motor dynamics
%                              Author: Lychek Keo
%                                10 June 2006
%==========================================================================
%                            Variable initialization
%==========================================================================
close all;
clear all;
g = 9.81; %Gravity(m/s^2)
r = 0.093; %Radius of wheel(m)
Mw = 1.282; %Mass of wheel(kg)
Mp = 9.6; %Mass of body(kg)
Jw = 0.0111; %Inertia of the wheel(kg*m^2)
Jp = 0.36; %Inertia of the body(kg*m^2)
L = 0.152; %Length to the body's center of mass(m)
%Motor's variables
Km = 0.0333; %Motor torque constant (Nm/A)
Ke = 0.0333; %Back EMF constant (Vs/rad)
Ra = 0.62; %Nominal Terminal Resistance (Ohm)
nm = 0.85; %Motor efficiency
Jm = 0.000635; %Moment Inertia of the rotor
% Va voltage applied to motors for controlling the pendulum
%Gear's variables
ng = 0.8; %Gear efficiency
Kg = 23; %Gear ratio
dTheta = 20*pi/180;
wp1 = 0.2;
wp2 = 0.15;
wu = 0.11;
```

```
n_d = [2];
n_wp = 2;
n_zp = 3;
n_u = 1;
n_y = 4;
%=============================================================================
%                        Parameters initialization calculation
%=============================================================================
Jeq = Jw+ng*(Kg^2)*Jm;
beta = (2*Mw+(2*Jeq/(r^2))+Mp);
alpha = (Jp*beta + 2*Mp*(L^2)*(Mw + Jeq/(r^2)));
%=============================================================================
%                                Calculate LPV Model
%=============================================================================
lfrs dl
A22 = -(2*ng*nm*Km*(Kg^2)*Ke*(-Mp*L*r+Jp+Mp*(L^2)))/(Ra*(r^2)*alpha);
A23 = (Mp*L)^2*g/alpha;
A24 = -(Jp+Mp*L^2)*dl/alpha;
A42 = (2*ng*nm*Km*(Kg^2)*Ke*(-Mp*L+r*beta))/(Ra*(r^2)*alpha);
A43 = Mp*g*L*beta/alpha;
A44 = -Mp*L*dl/alpha;
B21 = (2*ng*nm*Km*Kg*(-Mp*L*r+Jp+Mp*(L^2)))/(Ra*r*alpha);
B41 = -(2*ng*nm*Km*Kg*(-Mp*L+r*beta))/(Ra*r*alpha);
Alpv = [0 1 0 0;0 A22 A23 A24;0 0 0 1;0 A42 A43 A44];
Blpv = [0;B21;0;B41];
Clpv = eye(4);
Dlpv = zeros(4,1);
%=============================================================================
%                                Transform to LFT form
%=============================================================================
sys = abcd2lfr([Alpv Blpv;Clpv Dlpv],4);
[abcd,nS] = lfr2abcd(sys);
M11 = abcd.a;  M12 = abcd.b;
M21 = abcd.c;  M22 = abcd.d;
Anom1 = M22(1:4,1:4);
Bp1 = M22(1:4,5);
Cp1 = M22(5:8,1:4);
Dpp1 = M22(5:8,5);
Ddd1 = M11;
Cd1 = M12(:,1:4);
Ddp1 = M12(:,5);
Bd1 = M21(1:4,:);
Dpd1 = M21(5:8,:);
M1P_lpv1 = ss(Anom1,[Bd1 Bp1],[Cd1;Cp1],[Ddd1 Ddp1;Dpd1 Dpp1]);
%=============================================================================
%            Normalizes the variations of real uncertain parameters
%=============================================================================
Thetadmax = pi;
Thetamax = pi/3;
dmax = 1.5*Mp*L*Thetadmax*sin(Thetamax);
dmin = -dmax;
sys_norm = normlfr(sys,dmin,dmax);
%=============================================================================
%                        Pack state-space data into a SYSTEM matrix
%=============================================================================
[abcd2,nS] = lfr2abcd(sys_norm);
M11_2 = abcd2.a;  M12_2 = abcd2.b;
M21_2 = abcd2.c;  M22_2 = abcd2.d;
```

```
Anom2 = M22_2(1:4,1:4);
Bp2 = M22_2(1:4,5);
Cp2 = M22_2(5:8,1:4);
Dpp2 = M22_2(5:8,5);
Ddd2 = M11_2;
Cd2 = M12_2(:,1:4);
Ddp2 = M12_2(:,5);
Bd2 = M21_2(1:4,:);
Dpd2 = M21_2(5:8,:);
MIP_lpv_norm_t = pck(Anom2,[Bd2 Bp2],[Cd2;Cp2],[Ddd2 Ddp2;Dpd2 Dpp2]);
%=============================================================================
%                    Form interconnections of SYSTEM matrices
%=============================================================================
systemnames = 'MIP_lpv_norm_t_wp1_wp2_wu';
inputvar  = '[inc{2};r{2};u]';
outputvar = '[MIP_lpv_norm_t(1:2);wp1;wp2;wu;r(1)-MIP_lpv_norm_t(3);-MIP_lpv_norm_t(4);
            r(2)-MIP_lpv_norm_t(5);-MIP_lpv_norm_t(6)]';
input_to_MIP_lpv_norm_t = '[inc(1:2);u]';
input_to_wp1 = '[r(1)-MIP_lpv_norm_t(3)]';
input_to_wp2 = '[r(2)-MIP_lpv_norm_t(5)]';
input_to_wu  = '[u]';
sysoutname = 'MIP_aug_t';
cleanupsysic = 'yes';
sysic
%=============================================================================
%                  LPV controller synthesis with full block multiplier
%=============================================================================
[gamma,x,y,P,Ptilde] = opt_lpv(MIP_aug_t,n_d,n_wp,n_zp,n_u,n_y);
[Qe,Se,Re,Us,Vs,Ws,N_m,N_p] = multiplier_construct(P,Ptilde,x,y);
%=============================================================================
%                                Construct Gain K
%=============================================================================
[Ac,Bc,Cc,Dc]=construct_K(MIP_aug_t,n_d,n_wp,n_zp,n_u,n_y,Qe,Se,Re,gamma,N_m,N_p);
%=============================================================================
%                              Construct Gain Scheduling
%=============================================================================
[U11,U12,W11,W12,W21,W22,V11,V21] = scheduling(Us,Vs,Ws,n_d);
%=============================================================================
%                                  End Program
%=============================================================================
```

Appendix B

C Source Code for Balancing of MIP

B.1 Program for Running MIP Using DSPIC 30F4011

```
//*****************************************************************************************
//                          Main Program for Balancing MIP
//                Using Processor: DSPIC 30F4011 with Microchip C30 Compiler
//                                Author: Lychek Keo
//                                  17 July 2006
//*****************************************************************************************
//=========================================================================================
//                                Include Header File
//=========================================================================================
#include "p30f4011.h"
#include "serial.h"
#include "analog.h"
#include "kalman.h"
#include "timer.h"
#include "motor.h"
#include "qei.h"
//=========================================================================================
//                              Define Constant Parameter
//=========================================================================================
#define Vbc 0.004887585532746823069403714565004 9
//=========================================================================================
//                                Define Parameters
//=========================================================================================
float rate;
float angle;
float q_bias;
static const float X_K=-110;
static const float X_Kdot=-100;
static const float Phi_K=-1000;
static const float Phi_Kdot=-25;
static const float Delta_K=0;
static const float Delta_Kdot=-100;
static float X=0;
static float X_dot=0;
static float Phi=0;
static float Phi_dot=0;
static float Delta=0;
static float Delta_dot=0;
static float rat=0;
static float ax=0;
static float v_left=0;
static float p_left=0;
static float p_right=0;
static float v_right=0;
static int mvelocity_Left=0;
```

```
static int DnCount=0;
static int UpCount=0;
static long position_Right=0;
static long mposition_Left=0;
static signed long mposition_Right=0;
static signed long pos_valuc=0;
static signed long AngPos[2] = {0,0};
static signed int Speed=0;
static signed int old_speed=0;
static float update_L=0;
static float update_R=0;
static float Vtheta=0;
static float Vdelta=0;
static float c_left=0;
static float c_right=0;
static float c_for=0;
static float c_back=0;
static int c_Dncount=0;
static int c_Upcount=0;
static int st_Dncount=0;
static int st_Upcount=0;
static long mposition_turn=0;
static int mvelocity_turn=0;
static long mposition_st=0;
static int mvelocity_st=0;
static float C_Po_st=0;
static float C_Ve_st=0;
static float C_Po_turn=0;
static float C_Vc_turn=0;
static unsigned int duty_L=0;
static unsigned int duty_R=0;
static unsigned char updatedisable=0;
static unsigned int i=0;
//=============================================================================================
//                                        Interrupt for QEI
//=============================================================================================
void _ISR _QEIInterrupt(void)
{
  POSCNT=0;
  IFS2bits.QEIIF=0;
}
//=============================================================================================
//                                        Setup Timer
//=============================================================================================
void Setup_Timer(void)
{
  CloseTimer1();
  CloseTimer2();
  CloseTimer3();
  CloseTimer4();
  ConfigIntTimer1(T1_INT_PRIOR_1 & T1_INT_OFF);
  ConfigIntTimer2(T2_INT_PRIOR_1 & T2_INT_OFF);
  ConfigIntTimer3(T3_INT_PRIOR_2 & T3_INT_ON);
  ConfigIntTimer4(T4_INT_PRIOR_2 & T4_INT_ON);
  OpenTimer1(T1_ON & T1_GATE_ON & T1_IDLE_CON & T1_PS_1_1 & T1_SYNC_EXT_ON &
  T1_SOURCE_EXT,0xFFFF);
  OpenTimer2(T2_ON & T2_GATE_ON & T2_IDLE_CON & T2_PS_1_1 & T2_SOURCE_EXT,0xFFFF);
  OpenTimer3(T3_ON & T3_GATE_OFF & T3_IDLE_STOP & T3_PS_1_8 & T3_SOURCE_INT,0x9000);
```

```
  OpenTimer4(T4_ON & T4_GATE_OFF & T4_IDLE_STOP & T4_PS_1_1 & T4_SOURCE_INT,0x399A);
}
//*****************************************************************************************
//=========================================================================================
//                                    Initial QEI Port
//=========================================================================================
void qei_init(void)
{
  unsigned int config1;
  ADPCFG |= 0x0038;
  ConfigIntQEI(QEI_INT_PRI_0 & QEI_INT_ENABLE);
  POSCNT=0;
  MAXCNT=0xB7FF;
  config1=(QEI_DIR_SEL_QEB & QEI_INT_CLK & QEI_INDEX_RESET_DISABLE & QEI_CLK_PRESCALE_1 &
  QEI_GATED_ACC_DISABLE & QEI_NORMAL_IO & QEI_INPUTS_SWAP &  QEI_MODE_x2_MATCH &
  QEI_UP_COUNT & QEI_IDLE_CON);
  OpenQEI(config1 ,0);
}
//*****************************************************************************************
//=========================================================================================
//                        Calculate Position and Velocity of Motor Left
//=========================================================================================
void update_position_LEFT(void)
{
  mvelocity_Left = DnCount;
  mvelocity_Left -= UpCount;
  UpCount = ReadTimer1();
  DnCount = ReadTimer2();
  mvelocity_Left += UpCount;
  mvelocity_Left -= DnCount;
  mposition_Left += (long)mvelocity_Left << 2;
}
//=========================================================================================
//                      Calculate Position and Velocity of Command Turn
//=========================================================================================
void update_command_turn(void)
{
  mvelocity_turn = c_Dncount;
  mvelocity_turn -= c_Upcount;
  c_Upcount = c_right;
  c_Dncount = c_left;
  mvelocity_turn += c_Upcount;
  mvelocity_turn -= c_Dncount;
  mposition_turn += (long)mvelocity_turn;
}
//=========================================================================================
//            Calculate Position and Velocity of Command Forward or Backward
//=========================================================================================
void update_command_st(void)
{
  mvelocity_st = st_Dncount;
  mvelocity_st -= st_Upcount;
  st_Upcount = c_for;
  st_Dncount = c_back;
  mvelocity_st += st_Upcount;
  mvelocity_st -= st_Dncount;
  mposition_st += (long)mvelocity_st;
}
```

```
//*****************************************************************************************
//=========================================================================================
//                    Calculate Position and Velocity of Motor Right
//=========================================================================================
void update_position_RIGTH(void)
{
  pos_value = ReadQEI();
  AngPos[1] = AngPos[0];
  AngPos[0] = (long)pos_value;
}
//*****************************************************************************************
//                                    Initial Program
//*****************************************************************************************
void setup(void)
{
  TRISF=0;
  TRISE=0;
  TRISD=0xffff;
  TRISEbits.TRISE8 = 1;
  LATFbits.LATF6 = 0;
  Setup_Timer();
  uart2_init();
  init_ADC();
  qei_init();
  init_PWM();
  angle=-((Vbe*(float)read_adc(7))-2.37)*0.7614;
  q_bias=-((Vbe*(float)read_adc(8))-2.492)*5.1476;
}
//*****************************************************************************************
//                                    MAIN PROGRAM
//*****************************************************************************************
int main(void)
{
  setup();
  while(1)
  {
  }
}
//*****************************************************************************************
//=========================================================================================
//                                    Interrupt Timer 3
//=========================================================================================
void __attribute__((__interrupt__)) _T3Interrupt (void)
{
  IFS0bits.T3IF = 0;
//=========================================================================================
//                          Update Position and Velocity of the Motors
//=========================================================================================
  update_position_LEFT();
  update_position_RIGTH();
  v_left=(float)mvelocity_Left*0.8922/360;
  p_left=(float)mposition_Left*0.003814*0.5843/360;
  old_speed=Speed;
  Speed = (AngPos[0] - AngPos[1])*100/202;
  if (Speed>900)
  Speed=old_speed;
  if (Speed<-900)
  Speed=old_speed;
```

```
  position_Right += (long)Speed << 2;
  p_right=(float)position_Right*0.0039*0.5843/360;
  v_right=(float)Speed*0.9104/360;
//=============================================================================
//            Update Position and Velocity of the Command from Remote Control
//=============================================================================
  update_command_turn();
  update_command_st();
  C_Po_st=(float)mposition_st*0.07/100;
  C_Ve_st=(float)mvelocity_st*0.07;
  C_Po_turn=(float)mposition_turn*0.06/100;
  C_Ve_turn=(float)mvelocity_turn*0.06;
//=============================================================================
//                                  Calculate State of MIP
//=============================================================================
  Phi=angle*0.8;
  Phi_dot=rate;
  X=(p_left+p_right+angle*0.8*0.186)/2;
  X_dot=(v_left+v_right+rate*0.186)/2;
  Delta=p_left-p_right;
  Delta_dot=v_left-v_right;
//=============================================================================
//                       Calculate Control Input Using LQR Controller
//=============================================================================
  Vtheta=(X-C_Po_st)*X_K + (X_dot-C_Ve_st)*X_Kdot + Phi*Phi_K + Phi_dot*Phi_Kdot;
  Vdelta=(Delta-C_Po_turn)*Delta_K+(Delta_dot-C_Ve_turn)*Delta_Kdot;
  update_L=Vtheta/2+Vdelta/2;
  update_R=Vtheta/2-Vdelta/2;
//=============================================================================
//                            Update PWM for Control the Motor
//=============================================================================
  if (update_L<0)
    LATEbits.LATE0 = 0;
  else
    LATEbits.LATE0 = 1;
  duty_L=abs(update_L)*50;
  if (duty_L>1200)
    duty_L=1200;
  if (update_R<0)
    LATEbits.LATE2 = 0;
  else
    LATEbits.LATE2 = 1;
  duty_R=abs(update_R)*50;
  if (duty_R>1200)
    duty_R=1200;
  SetDCMCPWM(1,duty_L,updatedisable);
  SetDCMCPWM(2,duty_R,updatedisable);
//=============================================================================
//                          Sent State Variable to Computer Via RS232
//=============================================================================
  sprintf(sbuf,"%.3f_%.3f_%.3f_%.3f",X,X_dot,Phi,Phi_dot);
  putsUART2((signed int *)sbuf);
  while (BusyUART2());
  putsUART2((unsigned int *)"\r\n");
  while (BusyUART2());
  return;
}
//*****************************************************************************
```

```
//=============================================================================
//                          Interrupt Timer 4
//=============================================================================
void __attribute__((__interrupt__)) _T4Interrupt (void)
{
  IFS1bits.T4IF = 0; // Clear timer 4interrupt flag
//=============================================================================
//                      Wait for Calibration Kalman Filter
//=============================================================================
  if (i<=3000)
  {
    i=i+1;
    LATFbits.LATF6 = 0;
  }
  else
    LATFbits.LATF6 = 1;
//=============================================================================
//                          Calculate Kalman Filter
//=============================================================================
  rat=-((Vbe*(float)read_adc(8))-2.492)*5.1476;
  stateUpdate(rat);
  ax=-((Vbe*(float)read_adc(7))-2.37)*0.7614;
  kalmanUpdate(ax);
//=============================================================================
//                          Capture Remote Control
//=============================================================================
  if (PORTEbits.RE8==1&PORTDbits.RD0==1&PORTDbits.RD1==1&PORTDbits.RD2==0&PORTDbits.RD3==1)
    c_right=c_right+1;
  if (PORTEbits.RE8==1&PORTDbits.RD0==1&PORTDbits.RD1==0&PORTDbits.RD2==1&PORTDbits.RD3==1)
    c_left=c_left+1;
  if (PORTEbits.RE8==1&PORTDbits.RD0==1&PORTDbits.RD1==1&PORTDbits.RD2==1&PORTDbits.RD3==0)
    c_for=c_for+1;
  if (PORTEbits.RE8==1&PORTDbits.RD0==0&PORTDbits.RD1==1&PORTDbits.RD2==1&PORTDbits.RD3==1)
    c_back=c_back+1;
  if (PORTEbits.RE8==1&PORTDbits.RD0==1&PORTDbits.RD1==1&PORTDbits.RD2==0&PORTDbits.RD3==0)
  {
    c_for=c_for+1;
    c_right=c_right+1;
  }
  if (PORTEbits.RE8==1&PORTDbits.RD0==1&PORTDbits.RD1==0&PORTDbits.RD2==1&PORTDbits.RD3==0)
  {
    c_for=c_for+1;
    c_left=c_left+1;
  }
  if (PORTEbits.RE8==1&PORTDbits.RD0==0&PORTDbits.RD1==1&PORTDbits.RD2==0&PORTDbits.RD3==1)
  {
    c_back=c_back+1;
    c_right=c_right+1;
  }
  if (PORTEbits.RE8 ==1&PORTDbits.RD0==0&PORTDbits.RD1==0&PORTDbits.RD2==1&PORTDbits.RD3==1)
  {
    c_back=c_back+1;
    c_left=c_left+1;
  }
  return;
}
//*****************************************************************************
//                          End Main Program
```

```
//*************************************************************************************************
//*************************************************************************************************
//*************************************************************************************************
//                              Sub Program for calculate Kalman Filter
//                                          kalman.h
//                                       Author: Lychek Keo
//                                        17 July 2006
//*************************************************************************************************
//=================================================================================================
//                                        Include Header File
//=================================================================================================
#include "math.h"
//=================================================================================================
//                                        Define Parameters
//=================================================================================================
static const float dt = 0.0005;
static float P[2][2] = {{ 1, 0 },{ 0, 1 }};
float rate;
float angle;
float q_bias;
static const float R_angle = 0.01 ;
static const float Q_angle = 0.00005;
static const float Q_gyro  = 0.00001;
//=================================================================================================
//                                      Calculate State Update
//=================================================================================================
void stateUpdate(const float q_m)
{
  float q;
  float Pdot[4];
  q = q_m - q_bias;
  Pdot[0] = Q_angle - P[0][1] - P[1][0];
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
  rate = q;
  angle += q * dt;
  P[0][0] += Pdot[0] * dt;
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;
}
//=================================================================================================
//                                     Calculate Kalman Update
//=================================================================================================
void kalmanUpdate(const float incAngle)
{
  float angle_m = incAngle;
  float angle_err = angle_m - angle;
  float h_0 = 1;
  const float PHt_0 = h_0*P[0][0];
  const float PHt_1 = h_0*P[1][0];
  float E = R_angle +(h_0 * PHt_0);
  float K_0 = PHt_0 / E;
  float K_1 = PHt_1 / E;
  float Y_0 = PHt_0;
```

```
  float Y_1 = h_0 * P[0][1];
  P[0][0] -= K_0 * Y_0;
  P[0][1] -= K_0 * Y_1;
  P[1][0] -= K_1 * Y_0;
  P[1][1] -= K_1 * Y_1;
  angle += K_0 * angle_err;
  q_bias += K_1 * angle_err;
}
//*************************************************************************************************
//                                        End Sub Program
//*************************************************************************************************

////////////////////////////////////////////////////////////////////////////////////////////////////

//*************************************************************************************************
//                              Sub Program for Configuration of ADC Port
//                                           analog.h
//                                       Author: Lychek Keo
//                                         17 July 2006
//*************************************************************************************************
//=================================================================================================
//                                        Include Header File
//=================================================================================================
#include "adc10.h"
//=================================================================================================
//                                        Initial ADC Port
//=================================================================================================
void init_ADC(void)
{
  unsigned int Channel, PinConfig, Scanselect, Adcon3_reg, Adcon2_reg,Adcon1_reg;
  ADCON1bits.ADON = 0; /* turn off ADC */
  Channel = ADC_CH0_POS_SAMPLEA_AN6 & ADC_CH0_POS_SAMPLEA_AN7 & ADC_CH0_POS_SAMPLEA_AN8 &
  ADC_CH0_NEG_SAMPLEA_NVREF;
  SetChanADC10(Channel);
  ConfigIntADC10(ADC_INT_DISABLE);
  PinConfig = ENABLE_AN6_ANA & ENABLE_AN7_ANA & ENABLE_AN8_ANA;
  Scanselect = SKIP_SCAN_AN0 & SKIP_SCAN_AN1 & SKIP_SCAN_AN2 & SKIP_SCAN_AN3 &
  SKIP_SCAN_AN4 & SKIP_SCAN_AN5;
  Adcon3_reg = ADC_SAMPLE_TIME_10 & ADC_CONV_CLK_INTERNAL_RC & ADC_CONV_CLK_13Tcy;
  Adcon2_reg = ADC_VREF_AVDD_AVSS & ADC_SCAN_ON & ADC_ALT_BUF_OFF & ADC_ALT_INPUT_OFF &
  ADC_CONVERT_CH0 & ADC_SAMPLES_PER_INT_16;
  Adcon1_reg = ADC_MODULE_ON & ADC_IDLE_CONTINUE & ADC_FORMAT_INTG & ADC_CLK_MANUAL &
  ADC_SAMPLE_SIMULTANEOUS & ADC_AUTO_SAMPLING_ON;
  OpenADC10(Adcon1_reg, Adcon2_reg, Adcon3_reg,PinConfig, Scanselect);
}
//=================================================================================================
//                                      Create Read ADC Function
//=================================================================================================
int read_adc(char channel)
{
  ADCON1bits.SAMP = 1;
  while(!ADCON1bits.SAMP);
  ConvertADC10();
  while(ADCON1bits.SAMP);
  while(!BusyADC10());
  return(ReadADC10(channel));
}
//*************************************************************************************************
```

```
//                                    End Sub Program
//*****************************************************************************************

//////////////////////////////////////////////////////////////////////////////////////////

//*****************************************************************************************
//                          Sub Program for Configuration of PWM Port
//                                        motor.h
//                                    Author: Lychek Keo
//                                      17 July 2006
//*****************************************************************************************
//=========================================================================================
//                                    Include Header File
//=========================================================================================
#include "pwm.h"
//=========================================================================================
//                                    Clear Interrupt PWM
//=========================================================================================
void __attribute__((__interrupt__)) _PWMInterrupt(void)
{
  IFS2bits.PWMIF = 0;
}
//=========================================================================================
//                                    Clear Interrupt FLTA
//=========================================================================================
void __attribute__((__interrupt__)) _FLTAInterrupt(void)
{
  IFS2bits.FLTAIF = 0;
}
//=========================================================================================
//                                    Initial PWM Port
//=========================================================================================
void init_PWM(void)
{
  unsigned int config;
  unsigned int period;
  unsigned int sptime;
  unsigned int config1;
  unsigned int config2;
  unsigned int config3;
  unsigned int dutycyclereg;
  unsigned int dutycycle;
  unsigned char updatedisable;
  config = (PWM_INT_EN & PWM_INT_PR3 & PWM_FLTA_INT_PR5 );
  ConfigIntMCPWM( config );
  dutycyclereg = 1;
  dutycycle = 0x3FFF;
  updatedisable = 0;
  SetDCMCPWM(dutycyclereg,dutycycle,updatedisable);
  period = 0x0500;
  sptime = 0x0;
  config1 = (PWM_EN & PWM_OP_SCALE1 & PWM_IPCLK_SCALE1 & PWM_MOD_FREE);
  config2 = (PWM_MOD1_IND & PWM_PDIS3H & PWM_PEN2H & PWM_PEN1H & PWM_PDIS3L & PWM_PDIS2L &
  PWM_PDIS1L);
  config3 = (PWM_SEVOPS1 & PWM_OSYNC_PWM & PWM_UEN);
  OpenMCPWM(period,sptime,config1,config2,config3);
}
//*****************************************************************************************
```

```
//                                          End Sub Program
//*****************************************************************************************

//////////////////////////////////////////////////////////////////////////////////////////

//*****************************************************************************************
//                          Sub Program for Configuration of Serial Port
//                                          serial.h
//                                       Author: Lychek Keo
//                                         17 July 2006
//*****************************************************************************************
//=========================================================================================
//                                      Include Header File
//=========================================================================================
#include "uart.h"
#include "stdio.h"
//=========================================================================================
//                                      Define Parameter
//=========================================================================================
char sbuf[40];
//=========================================================================================
//                                      Clear Interrupt UART
//=========================================================================================
void _ISR _U2TXInterrupt(void)
{
  IFS1bits.U2TXIF=0;
}
void _ISR _U2RXInterrupt(void)
{
  IFS1bits.U2RXIF=0;
}
//=========================================================================================
//                                      Initial UART Port
//=========================================================================================
void uart2_init()
{
  unsigned int baudvalue;
  unsigned int U2MODEvalue;
  unsigned int U2STAvalue;
  CloseUART2();
  ConfigIntUART2(UART_RX_INT_EN & UART_RX_INT_PR6 & UART_TX_INT_EN & UART_TX_INT_PR4);
  baudvalue=31;
  U2MODEvalue=UART_EN & UART_IDLE_CON & UART_RX_TX & UART_DIS_WAKE & UART_DIS_LOOPBACK &
  UART_DIS_ABAUD & UART_NO_PAR_8BIT & UART_1STOPBIT;
  U2STAvalue=UART_INT_TX_BUF_EMPTY & UART_TX_PIN_NORMAL & UART_TX_ENABLE &
  UART_INT_RX_3_4_FUL & UART_ADR_DETECT_DIS & UART_RX_OVERRUN_CLEAR;
  OpenUART2(U2MODEvalue, U2STAvalue, baudvalue);
}
//*****************************************************************************************
//                                          End Sub Program
//*****************************************************************************************
```

B.2 Program for Monitoring Battery Using PIC 16F877

```
//*************************************************************************************************
//                              Main Program for Balancing MIP
//                        Using Processor: DSPIC 30F4011 with CCS Compiler
//                                      Author: Lychek Keo
//                                        17 July 2006
//*************************************************************************************************
//=================================================================================================
//                                      Define Processor
//=================================================================================================
#define _PIC16F877A_
#include <16F877A.h>
//=================================================================================================
//                                      Assign Pin Name
//=================================================================================================
#define DET_C PIN_B3
#define OUT1 PIN_C2
#define LD1 PIN_D0
#define LD2 PIN_D1
#define LD3 PIN_D2
#define LD4 PIN_D3
#define LD5 PIN_D4
#define LD6 PIN_D5
#define LD7 PIN_D6
#define LD8 PIN_D7
#define LD9 PIN_C0
#define LD10 PIN_C1
#define I2C_DECT PIN_C5
//=================================================================================================
//                                      Define Clock Input
//=================================================================================================
#define CLOCK_SP 20000000
//=================================================================================================
//                                      Define ADC Constant
//=================================================================================================
#define Vbe 0.004887585532746823069 4037145650049
//=================================================================================================
//        Configuration Oscillator, Code Protection, ADC Resolution, USART and LCD
//=================================================================================================
#fuses HS
#fuses NOLVP, NOWDT
#fuses NOPROTECT
#device ADC=10
#use delay (clock=CLOCK_SP)
#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)
#use i2c(Master, SDA=PIN_C4, SCL=PIN_C3)
#use fast_io(A)
#define use_portb_lcd
//=================================================================================================
//                                      Include Header File
//=================================================================================================
#include <math.h>
#include <stdlib.h>
#include <ds1307.c>
#include <LCD.c>
//=================================================================================================
//                                      Define Subroutine
```

```
//=============================================================================================
void batt_c(void);
void batt_level(void);
void display_date_hour(void);
void ds1307_set_date_time(void);
void ds1307_read_date_time(void);
void ds1307_write_byte(char addr, char value);
char ds1307_read_byte(char addr);
char bin2bcd(char binary_value);
char bcd2bin(char bcd_value);
//=============================================================================================
//                                        Define Variable
//=============================================================================================
int wait=0;
char sec;
char min;
char hrs;
char day;
char date;
char month;
char yr;
//=============================================================================================
//                                Set Indicator for Battery Level 0
//=============================================================================================
void batt_level_0(void)
{
  output_bit(LD1,0);
  output_bit(LD2,0);
  output_bit(LD3,0);
  output_bit(LD4,0);
  output_bit(LD5,0);
  output_bit(LD6,0);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================================
//                                Set Indicator for Battery Level 1
//=============================================================================================
void batt_level_1(void)
{
  output_bit(LD1,1);
  output_bit(LD2,0);
  output_bit(LD3,0);
  output_bit(LD4,0);
  output_bit(LD5,0);
  output_bit(LD6,0);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================================
//                                Set Indicator for Battery Level 2
//=============================================================================================
void batt_level_2(void)
{
```

```
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,0);
  output_bit(LD4,0);
  output_bit(LD5,0);
  output_bit(LD6,0);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================
//                          Set Indicator for Battery Level 3
//=============================================================================
void batt_level_3(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,0);
  output_bit(LD5,0);
  output_bit(LD6,0);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================
//                          Set Indicator for Battery Level 4
//=============================================================================
void batt_level_4(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,1);
  output_bit(LD5,0);
  output_bit(LD6,0);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================
//                          Set Indicator for Battery Level 5
//=============================================================================
void batt_level_5(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,1);
  output_bit(LD5,1);
  output_bit(LD6,0);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
```

```
}
//=============================================================================
//                          Set Indicator for Battery Level 6
//=============================================================================
void batt_level_6(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,1);
  output_bit(LD5,1);
  output_bit(LD6,1);
  output_bit(LD7,0);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================
//                          Set Indicator for Battery Level 7
//=============================================================================
void batt_level_7(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,1);
  output_bit(LD5,1);
  output_bit(LD6,1);
  output_bit(LD7,1);
  output_bit(LD8,0);
  output_bit(LD9,0);
  output_bit(LD10,0);
}

//=============================================================================
//                          Set Indicator for Battery Level 8
//=============================================================================
void batt_level_8(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,1);
  output_bit(LD5,1);
  output_bit(LD6,1);
  output_bit(LD7,1);
  output_bit(LD8,1);
  output_bit(LD9,0);
  output_bit(LD10,0);
}
//=============================================================================
//                          Set Indicator for Battery Level 9
//=============================================================================
void batt_level_9(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
```

```
  output_bit(LD4,1);
  output_bit(LD5,1);
  output_bit(LD6,1);
  output_bit(LD7,1);
  output_bit(LD8,1);
  output_bit(LD9,1);
  output_bit(LD10,0);
}
//=============================================================================
//                          Set Indicator for Battery Level 10
//=============================================================================
void batt_level_10(void)
{
  output_bit(LD1,1);
  output_bit(LD2,1);
  output_bit(LD3,1);
  output_bit(LD4,1);
  output_bit(LD5,1);
  output_bit(LD6,1);
  output_bit(LD7,1);
  output_bit(LD8,1);
  output_bit(LD9,1);
  output_bit(LD10,1);
}
//=============================================================================
//                      Subroutine for Delay 0.5s and Send Data to LCD
//=============================================================================
void xdelay(void)
{
  int16 value;
  float volt;
  if (input(DET_C)){
  delay_ms(500);
  value=read_ADC();
  printf("\n\rAnalog_adc_value :_%ld",value);
  volt=8.0653463016*Vbe*(float)value;
  printf("\n\rBattery_Charging :_%2.2f_V",volt);
  lcd_gotoxy(1,1);
  printf(lcd_putc,"Time_:_%02d:%02d:%02d_",hrs,min,sec);
  lcd_gotoxy(1,2);
  printf(lcd_putc,"Bat_Char:_%2.2fV",volt);
  display_date_hour();
}
  else break;
}
//=============================================================================
//                          Indictor for Battery Charging
//=============================================================================
void batt_c(void)
{
  batt_level_0();
  xdelay();
  batt_level_1();
  xdelay();
  batt_level_2();
  xdelay();
  batt_level_3();
  xdelay();
```

```
  batt_level_4();
  xdelay();
  batt_level_5();
  xdelay();
  batt_level_6();
  xdelay();
  batt_level_7();
  xdelay();
  batt_level_8();
  xdelay();
  batt_level_9();
  xdelay();
  batt_level_10();
  xdelay();
}
//=============================================================================
//                          Subroutine for Display Time
//=============================================================================
void display_date_hour(void)
{
  ds1307_read_date_time();
  sec = gca_ds1307_regs[DS1307_SECONDS_REG];
  min = gca_ds1307_regs[DS1307_MINUTES_REG];
  hrs = gca_ds1307_regs[DS1307_HOURS_REG];
  day = gca_ds1307_regs[DS1307_DAY_OF_WEEK_REG];
  date = gca_ds1307_regs[DS1307_DATE_REG];
  month = gca_ds1307_regs[DS1307_MONTH_REG];
  yr = gca_ds1307_regs[DS1307_YEAR_REG];
  printf("\n\rDate :_%d/\%d/\%02d____Time :_\%d:\%d:\%02d\n\r",month,date,yr, hrs,min,sec);
}
//=============================================================================
//                          Subroutine for checking battery level
//=============================================================================
void batt_level(void)
{
  int16 value;
  float volt;
  if(input(DET_C))
  {
    output_bit(OUT1,0);
    batt_c();
  }
  else
  {
    delay_ms(1000);
    value=read_ADC();
    printf("\n\rAnalog_adc_value :_%ld",value);
    volt=8.0653463016*Vbc*(float)value;
    printf("\n\rBattery_Voltage_:_%2.2f_V",volt);
    lcd_gotoxy(1,1);
    printf(lcd_putc,"Time_:_%02d:%02d:%02d_",hrs,min,sec);
    lcd_gotoxy(1,2);
    printf(lcd_putc,"Bat_Volt :_%2.2fV",volt);
    display_date_hour();
    if (volt>=25.5)
    {
      batt_level_10();
      wait=0;
```

```
  output_bit(OUT1,1);
}
if (volt<25.5 && volt>=25)
{
  batt_level_9();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<25 && volt>=24.5)
{
  batt_level_8();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<24.5 && volt>=24)
{
  batt_level_7();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<24 && volt>=23.7)
{
  batt_level_6();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<23.7 && volt>=23.4)
{
  batt_level_5();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<23.4 && volt>=23.1)
{
  batt_level_4();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<23.1 && volt>=22.8)
{
  batt_level_3();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<22.8 && volt>=22.5)
{
  batt_level_2();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<22.5 && volt>=22)
{
  batt_level_1();
  wait=0;
  output_bit(OUT1,1);
}
if (volt<22)
{
```

```
      if (wait>30)
        {
          batt_level_0();
          output_bit(OUT1,0);
        }
     }
     wait=wait+1;
  }
}
//=============================================================================
//                                  Main Program
//=============================================================================
void main(void)
{
  lcd_init();
  setup_adc_ports(RA0_ANALOG);
  setup_adc(ADC_CLOCK_INTERNAL);
  set_adc_channel(0);
  while(TRUE)
  {
    batt_level();
  }
}
//=============================================================================
//                                  End Program
//=============================================================================
```

Appendix C

Schematics and Printed Circuit Board

C.1 Schematic and Printed Circuit Board of CPU

The Micro-controller DSPIC 30F4011 has been use for control to balance a MIP, we can program this CPU via connector " Program" and connect to ICD2 Programmer.
This board have two channels of encoder inputs for capture the position and velocity of the motor left and right, it also can receive the RF signal from remote control and decode the signal by IC HT12D. We can send data out via serial interface and connect to PC for data logger. The schematic and the PCB of this CPU as shown Below

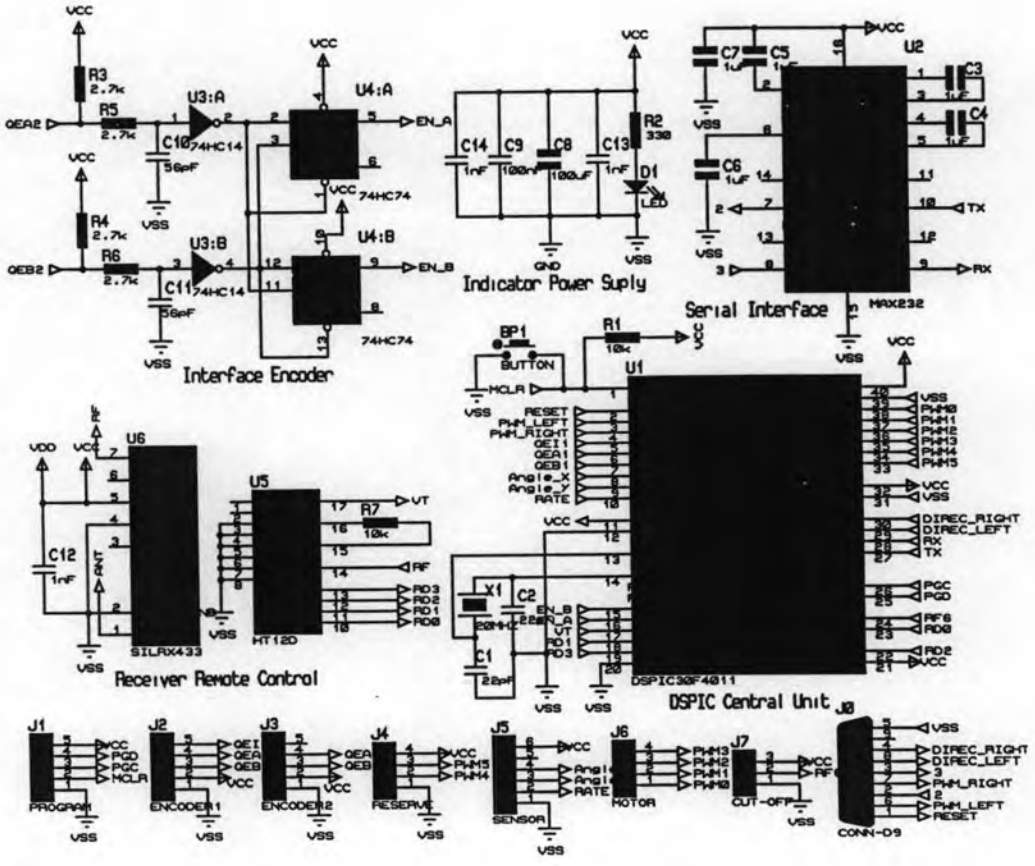

Figure C.1: The schematic of the CPU DSPIC30F4011 for the MIP.

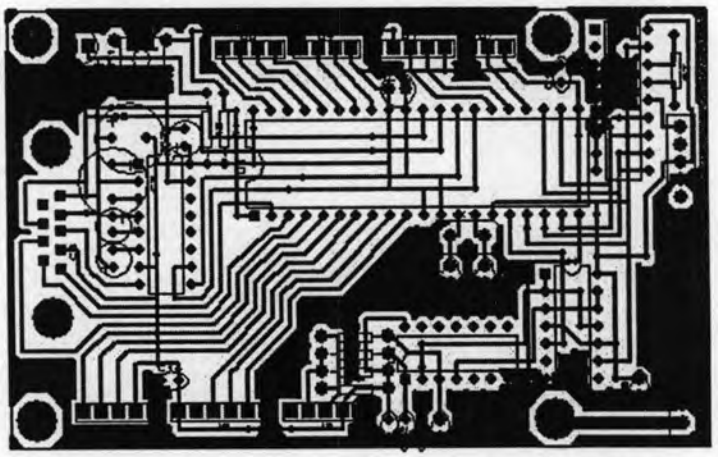

Figure C.2: The printed circuit board of the CPU DSPIC30F4011.

C.2 Schematic and Printed Circuit Board of Motors Drives

To drive the motor, we need the circuit that can work under 24V with 3A, thus LMD18200 is used for drive our motors.

In this board also have the circuit to cut-off the power supply when the battery is lower than 22V or when the battery is charging.

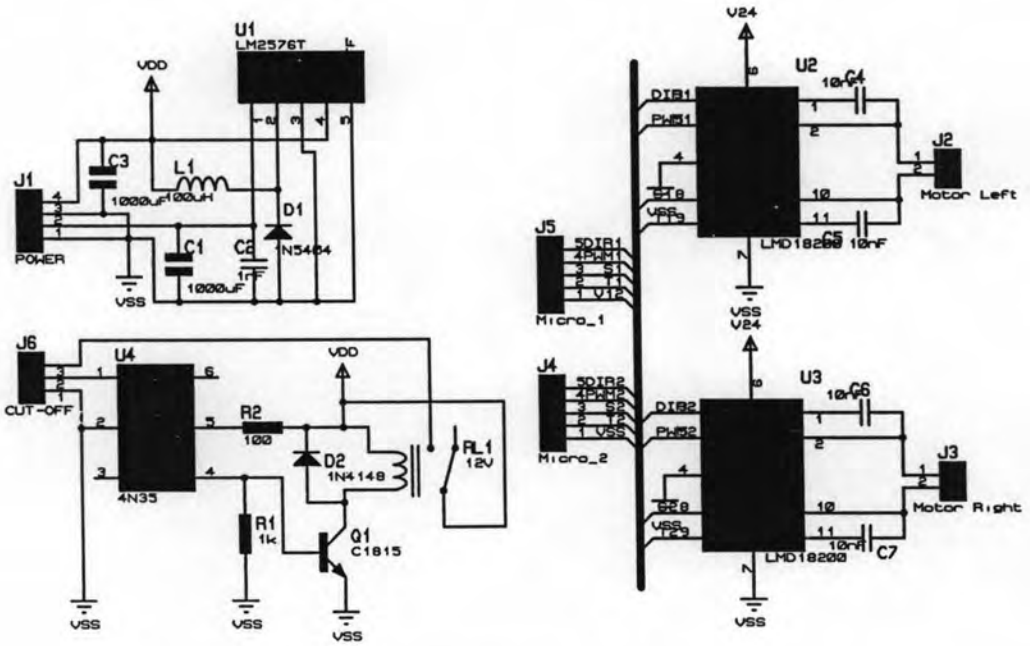

Figure C.3: The schematic of motor drive for the MIP.

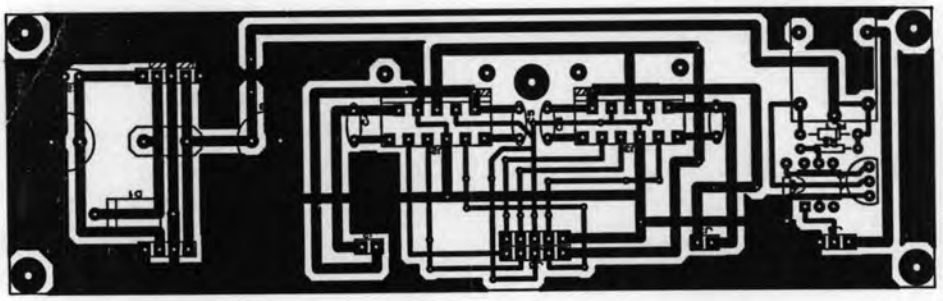

Figure C.4: The printed circuit board of motor drive.

C.3 Schematic and Printed Circuit Board of Monitoring Battery

To protect our battery, we need the circuit to cut-off the power supply when the battery is too low, thus we have designed a circuit below for monitoring the battery.

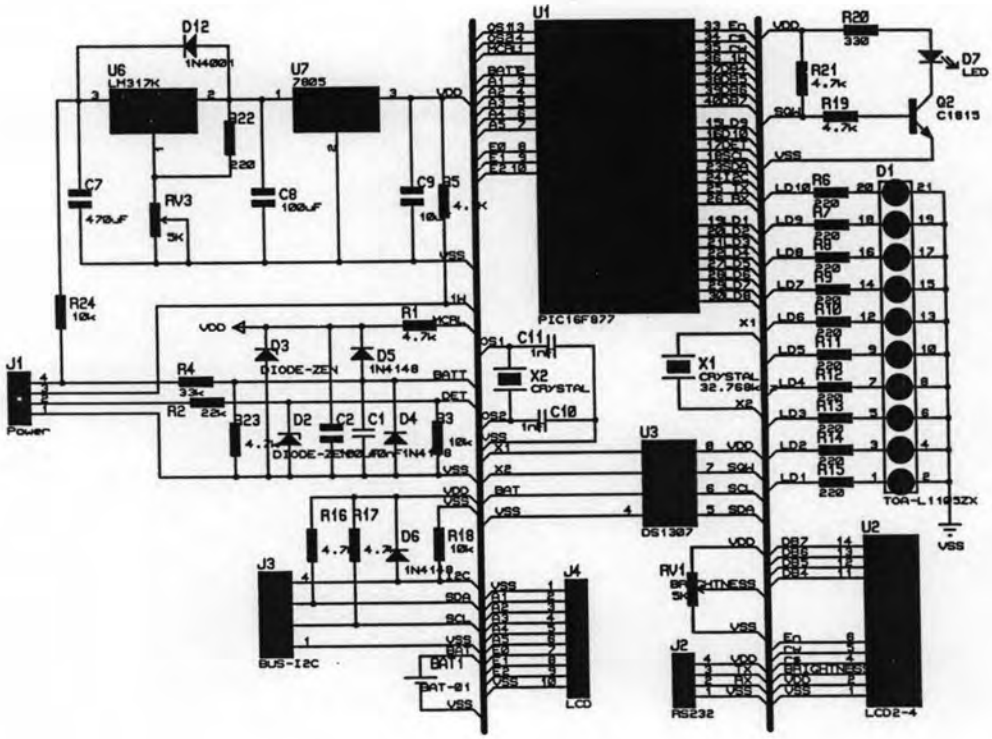

Figure C.5: The schematic of monitoring battery for the MIP.

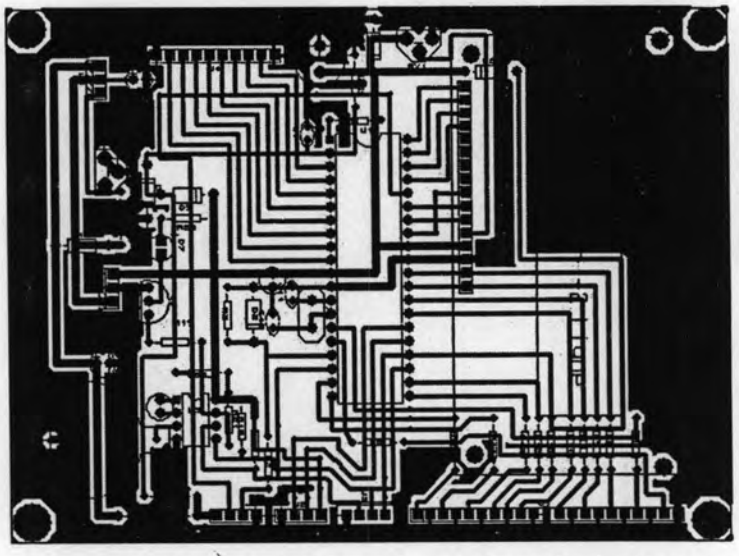

Figure C.6: The printed circuit board of monitoring battery.

C.4 Schematic and Printed Circuit Board of Power Supply Cut-Off

When the Kalman filter calibrate, we don't want our motor run, we need to wait around 3 seconds. Thus, we have designed a small circuit to cut-off power during the Kalman calibration.

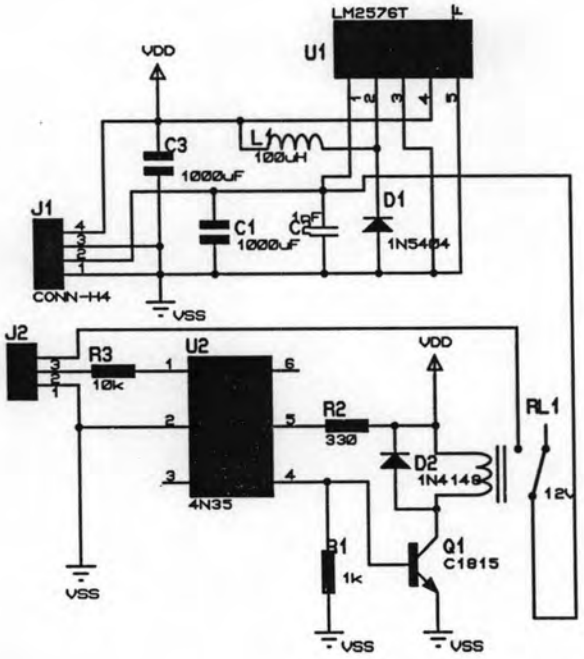

Figure C.7: The schematic of motor's power supply cut-off for the MIP.

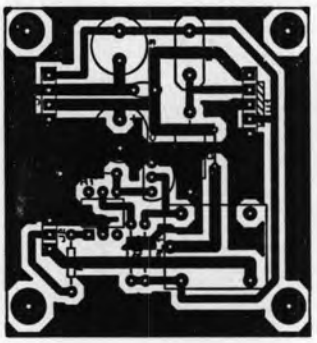

Figure C.8: The printed circuit board of motor's power supply cut-off.

C.5 Schematic and Printed Circuit Board of Sensors Interface

Accelerometer has some noisy, so we need to use analog filter to reduce some noise before we connect to micro-controller. This is a simple circuit for filter signal from accelerometer.

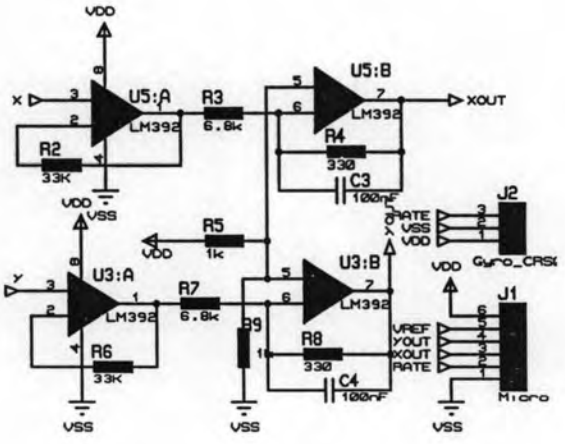

Figure C.9: The schematic of interface for gyro and accelerometer.

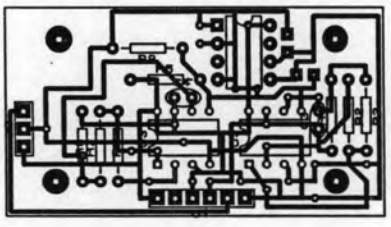

Figure C.10: The printed circuit board of interface for gyro and accelerometer.

C.6 Schematic and Printed Circuit Board of Remote Control

This schematic use for control the MIP via wireless, it possible to control to move forward, backward, turn left and turn right. We have used the encoder HT12E to code the signal before we transmit the signal via RF module from radiometrix.

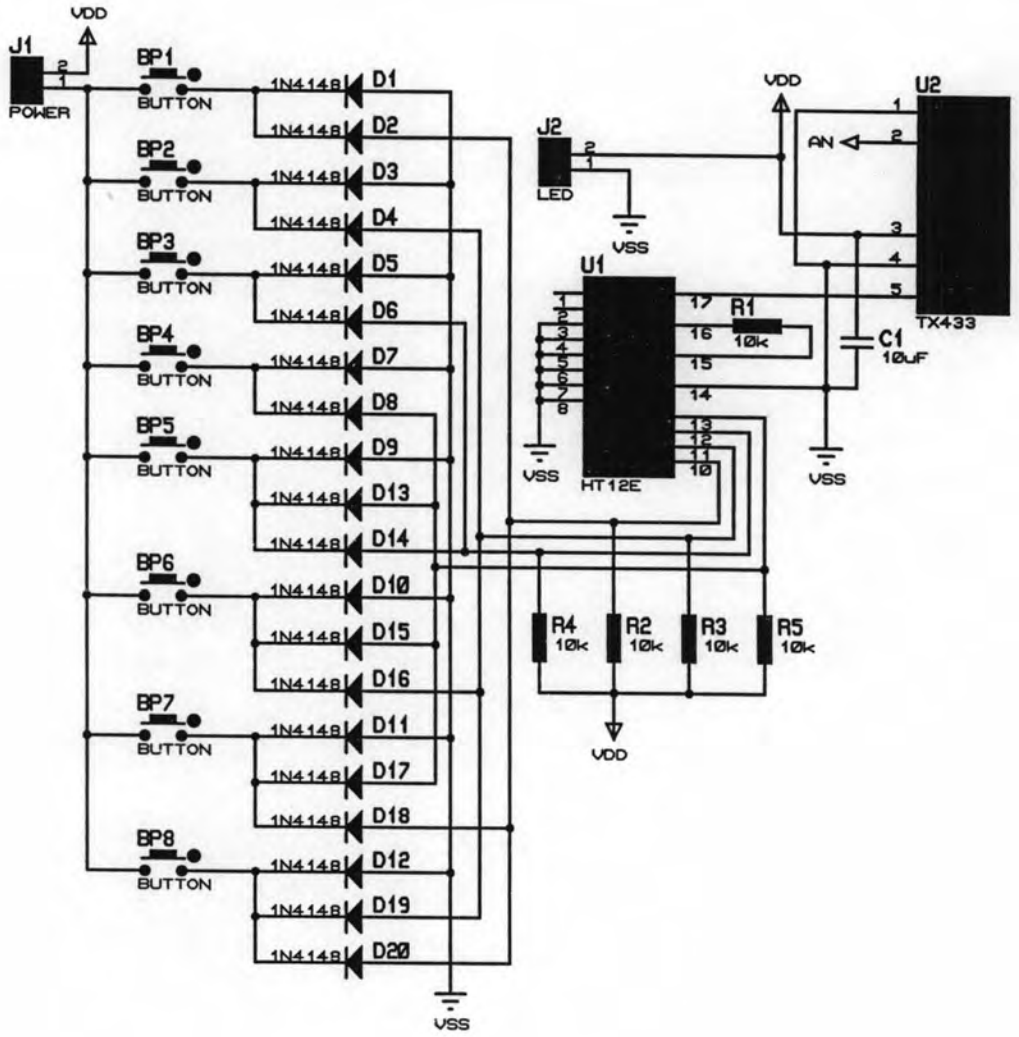

Figure C.11: The schematic of remote control for the MIP.

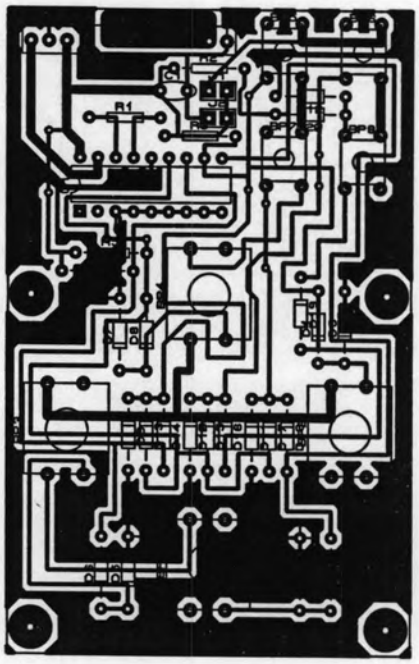

Figure C.12: The printed circuit board of a remote control.

หอสมุดกลาง ศูนย์วิทยทรัพยากร จุฬาลงกรณ์มหาวิทยาลัย

Biography

Mr. Lychek Keo was born in 1978, in Prey Veng, Cambodia. He received the B. E. joint degree in Electrical Engineering from the Institute of Technology of Cambodia (ITC), Phnom Penh, Cambodia and the École Supérieure d'Ingénieurs en Electronique et Electrotechnique (ESIEE), Amiens, France, in 2000. Since September, 2000, he has been a lecturer in power electronic and feedback control system at the Institute of Technology of Cambodia. In October 2004, he was rewarded AUN/SEED-Net scholarships for the Master's Degree Program to study at Chulalongkorn University. His research interests are in the area of control systems, robotics and electronics.