10. Anexos

Actuadores

```
function act_init(INICIOKERNELS)

% Initialize TrueTime kernel
ttInitKernel('prioFP'); % nbrOfInputs, nbrOfOutputs, fixed priority

% Create task data (local memory)

data.period = INICIOKERNELS.T;
data.Nbif=INICIOKERNELS.Nbif;
data.iteract = 0;
data.itercont=0;
data.total_iter = INICIOKERNELS.ITERACIONES;
data.flag=1;
data.nodata = 0;
data.count=1;
data.horizonte = INICIOKERNELS.horizonte;

%Variables TFM
data.ax=[INICIOKERNELS.ax INICIOKERNELS.ax INICIOKERNELS.ax INICIOKERNELS.ax
    INICIOKERNELS.ax];
data.delta=[INICIOKERNELS.delta INICIOKERNELS.delta INICIOKERNELS.delta ...
INICIOKERNELS.delta INICIOKERNELS.delta];



offset = 0;
ttCreatePeriodicTask('act_task', offset, data.period, 'act_code',data);

function [exectime, data] = act_code(seg, data)

switch seg

        case 1
            if data.iteraux == 10
                    data.iteraux = 0;
            end

        data.iteract = data.iteract + 1;
        data.iteraux = data.iteraux + 1;
        exectime = 0;
        case 2
            msgrcv = ttGetMsg;
            if isempty(msgrcv)
                disp(["No msg en actuador" string(ttCurrentTime)]);
                if (data.flag==1 && data.iteract==1)
                    data.flag=0;
                else
                    data.nodata = 1;
                    data.count = data.count + 1;
            end
        else
            disp(["Si hay msg en actuador" string(ttCurrentTime)])
            data.count = 1;
            data.ax = msgrcv.ax;
            data.delta = msgrcv.delta;
            data.itersens=msgrcv.itersens;
            data.itercont=msgrcv.itercont;
        end
        exectime = 0;

    case 6
        if(data.nodata == 0)
            ttAnalogOut(1, data.ax(data.iteraux));
            ttAnalogOut(2, data.delta(data.iteraux));
            data.count = data.iteraux;
            disp(['Aplico␣DREKF' string(ttCurrentTime)]);
        elseif(data.count<=data.horizonte)
            data.nodata=0;
            disp(['Aplico␣predicci n' string(data.count) string(ttCurrentTime)
    ])
            ttAnalogOut(1, data.ax(data.count));
            ttAnalogOut(2, data.delta(data.count));
        else
            disp(['Aplico␣predicci n' string(data.horizonte) string(
    ttCurrentTime)])
            data.nodata=0;
            ttAnalogOut(1, data.ax(end));
            ttAnalogOut(2, data.delta(end));
        end
        exectime = 0;

    case 10
        exectime = -1;

    otherwise
        exectime = 0;
end
```

Sensores

```
function sens_init(INICIOKERNELS)

ttInitKernel('prioFP');

period = INICIOKERNELS.T;                          % sampling period
data.itersens = 0;                                 % iteration in sensor device
data.total_iter = INICIOKERNELS.ITERACIONES;       % number of iterations
offset = 0;

data.lossc = INICIOKERNELS.lossc;

ttCreatePeriodicTask('sens_task', offset, period, 'sens_code',data);

function [exectime, data] = sens_code(seg, data)
switch seg
        case 1
            data.itersens = data.itersens+1;
            exectime = 0;
        case 2
            data.vx_limpia = ttAnalogIn(1);
            data.vy_limpia = ttAnalogIn(2);
            data.x_limpia = ttAnalogIn(3);
            data.y_limpia = ttAnalogIn(4);
            data.psi_limpia = ttAnalogIn(5);
            data.psid_limpia = ttAnalogIn(6);
            exectime = 0;
        case 6
            ttAnalogOut(1, data.vx_limpia)
            ttAnalogOut(2, data.vy_limpia)
            ttAnalogOut(3, data.x_limpia)
            ttAnalogOut(4, data.y_limpia)
            ttAnalogOut(5, data.psi_limpia)
            ttAnalogOut(6, data.psid_limpia)
            exectime = 0;
        case 7
            if(data.lossc(data.itersens) == 1)
                disp(['mensaje enviado desde sensor' string(ttCurrentTime)])
                msgenv.itersens = data.itersens;
                msgenv.vx = data.vx_limpia;
                msgenv.vy = data.vy_limpia;
                msgenv.x = data.x_limpia;
                msgenv.y = data.y_limpia;
                msgenv.psi = data.psi_limpia;
                msgenv.psid = data.psid_limpia;
                ttSendMsg(2, msgenv, 8);
            else
                disp(['mensaje␣no␣enviado␣desde␣sensor' string(ttCurrentTime)])
        end
        exectime = 0;
    case 10
        exectime = -1;
    otherwise
        exectime = 0;
end
```

Controlador

```
function slow_ctrl_init(INICIOKERNELS)



ttInitKernel('prioFP');


data.period = INICIOKERNELS.T;
data.Nbif = INICIOKERNELS.Nbif;
data.nodata = 0;
data.itercont = 0;
data.total_iter = INICIOKERNELS.ITERACIONES;
data.flag=1;
data.horizonte = INICIOKERNELS.horizonte;

data.retardoCA = INICIOKERNELS.retardoCA;

data.losca = INICIOKERNELS.losca;

data.z_EKF = [INICIOKERNELS.vx; 0; INICIOKERNELS.x; INICIOKERNELS.y;
    INICIOKERNELS.psi; INICIOKERNELS.psid];

s=tf('s');

%Acciones de control iniciales
data.ax = INICIOKERNELS.ax;
data.delta = INICIOKERNELS.delta;
                        % measurement noises
data.w_ekf=INICIOKERNELS.w_ekf;
data.v_ekf=INICIOKERNELS.v_ekf;
                          % EKF matrices
data.P=INICIOKERNELS.P;
data.Q=INICIOKERNELS.Q;
data.R=INICIOKERNELS.R2;
data.vhMdl=INICIOKERNELS.vhMdl;
data.trMdl=INICIOKERNELS.trMdl;
data.dt=INICIOKERNELS.dt;
data.T=INICIOKERNELS.T;
data.M=INICIOKERNELS.M;
data.psidrefn=INICIOKERNELS.psidrefn;
data.L=INICIOKERNELS.L;
data.Kp=INICIOKERNELS.Kp;
data.gamma=INICIOKERNELS.gamma;
data.ipp=INICIOKERNELS.ipp;
data.xref_tot=INICIOKERNELS.xref_tot;
data.yref_tot=INICIOKERNELS.yref_tot;
data.xrefnac=[];
data.yrefnac=[];
data.ddac=[];
data.psidrefnac=[];
data.xrefn = INICIOKERNELS.xref_tot(1);
data.yrefn = INICIOKERNELS.yref_tot(1);


offset = (data.period)/1.01;
ttCreatePeriodicTask('slow_ctrl', offset, data.period, 'slow_ctrl_code',data);

function [exectime, data] = slow_ctrl_code(seg, data)

switch seg

    case 1
        if data.flag==1
            data.flag=0;
            ttSetNextSegment ( 10 )
        else
            data.itercont = data.itercont + 1;
        end
        exectime=0;

        case 2
            msgrcv = ttGetMsg;
        if isempty(msgrcv)
            disp (['no␣hay␣msg␣en␣Controlador' string(ttCurrentTime)])
            data.nodata = 1;
            data.y_ekf(:,data.itercont) = [data.z_EKF(1,data.itercont,1);...
            data.z_EKF(3,data.itercont,1); data.z_EKF(4,data.itercont,1);...
            data.z_EKF(5,data.itercont,1)];
        else
            data.nodata = 0;
            data.itersens = msgrcv.itersens;
            data.vx = msgrcv.vx;
            data.vy = msgrcv.vy;
            data.x = msgrcv.x;
            data.y = msgrcv.y;
            data.psi = msgrcv.psi;
            data.psid = msgrcv.psid;
            data.y_ekf(:,data.itercont) = [data.vx; data.x; data.y; data.psi];
            disp (['si␣hay␣msg␣en␣Controlador' string(ttCurrentTime)])
        end
        exectime = 0;
    case 3

    data.u_ekf(:,data.itercont) = [data.ax(data.itercont,1), data.delta(data
.itercont,1)];
    % FILTRO DE KALMAN
    [data.z_EKF(:,data.itercont,2), data.P] = EKF(data.z_EKF(:,data.itercont
,1),...
    data.w_ekf, data.v_ekf, data.P, data.y_ekf(:,data.itercont),...
    data.Q, data.R, data.u_ekf(:,data.itercont),data.vhMdl,...
    data.trMdl, data.dt, data.itercont, data.M,data.nodata);

    data.vxe=data.z_EKF(1,data.itercont,2);      % para calcular delta(kk+1)
    data.vxef=data.vxe;            % para calcular psidrefn en path planning
    %vye(kk+1)=z_EKF(2);
    data.xe=data.z_EKF(3,data.itercont,2);
    data.xecf=data.xe;              % para calcular dist_x en path planning
    data.ye=data.z_EKF(4,data.itercont,2);
    data.yecf=data.ye;             % para calcular dist_y en path planning
    data.psie=data.z_EKF(5,data.itercont,2);
    data.psief=data.psie;           % para calcular alfa y psirefn en plath
planning
    data.pside=data.z_EKF(6,data.itercont,2);    % para calcular delta(kk+1)


    %C lculo acciones de control
    data.ax(data.itercont+1,1)=data.ax(data.itercont,1);
    data.delta(data.itercont+1,1) = atan2(data.psidrefn*data.L,data.vxe)+...
    (data.psidrefn-data.pside)*data.Kp*data.gamma;

    %hStepEstimation

    %Utilizar esta linea de c digo en el caso de desactivar hstep
    %data.z_EKF(:,data.itercont+1,1) = data.z_EKF(:,data.itercont,2);

    [data.z_EKF,data.ax,data.delta] = hStepEstimation(data.z_EKF,data.ax,
data.delta,...
    data.vhMdl, data.trMdl,data.horizonte,data.itercont,...
    data.ipp,data.xref_tot,data.yref_tot,data.dt,...
    data.w_ekf,data.xrefn,data.yrefn);

    %% Path Planning.
    data.flag_pp = true;

    data.look_ahead = 5;
    while(data.flag_pp && data.ipp<1200)
```

```
        data.ipp = data.ipp+1;
        data.dist_x = abs(data.xecf - data.xref_tot(data.ipp));
        data.dist_y = abs(data.yecf - data.yref_tot(data.ipp));
        data.dist_total = sqrt(data.dist_x*data.dist_x + data.dist_y*data.
dist_y);
        if(data.dist_total > data.look_ahead)
            data.xrefn = data.xref_tot(data.ipp);
            data.yrefn = data.yref_tot(data.ipp);
            data.flag_pp = false;
            data.ipp = data.ipp-1;
        end
    end




    ttAnalogOut(1,data.xrefn);
    ttAnalogOut(2,data.yrefn);
    data.xrefnac=[data.xrefnac; data.xrefn];
    data.yrefnac=[data.yrefnac; data.yrefn];

    data.dd=sqrt((data.yrefn-data.yecf)^2+(data.xrefn-data.xecf)^2);
    data.ddac=[data.ddac data.dd];
    data.alfa=atan2((data.yrefn-data.yecf),(data.xrefn-data.xecf))-data.
psief;

    data.psidrefn=(2*data.vxef*sin(data.alfa))/data.dd;

    data.psidrefnac=[data.psidrefnac; data.psidrefn];
    data.psirefn=data.psief+data.T*data.psidrefn;

    exectime = data.retardoCA(data.itercont);
    %exectime=0;

    case 7
    if(data.losca(data.itercont)==1)
        disp(['mensaje␣enviado␣desde␣el␣controlador' string(ttCurrentTime)])
        msgenv.ax = data.ax(data.itercont+1,:);
                msgenv.delta = data.delta(data.itercont+1,:);
                msgenv.itersens=data.itersens;
                msgenv.itercont=data.itercont;
                ttSendMsg(3, msgenv, 8);
    else
        disp('mensaje␣no␣enviado␣desde␣el␣controlador')
    end
            exectime = 0;
        case 10
        exectime = -1;

        otherwise
                exectime = 0;
end
end

%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [mx_km1, P_km1] = EKF(mx_k, mw_k, mv_k, P_k, y_kp1, Q, R, u_ekf, vhMdl
    ,...
trMdl, dt, kk, M, nodata)

  % EKF calculations
  xDim = size(mx_k,1);
  mx_kp1= f_BicycleModel(mx_k, mw_k, u_ekf, vhMdl, trMdl, dt, kk); % Prediction
  mx_kp1;
  A = numerical_jac_x(mx_k, mw_k, u_ekf, vhMdl, trMdl, dt, kk);
  L = numerical_jac_w(mx_k, mw_k, u_ekf, vhMdl, trMdl, dt, kk);
  P_kp1 = A*P_k*A' + L*Q*L';
  if (mod(kk+1,M)==0 && nodata ==0)
    disp(['Correcci n_DREKF!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!' string(ttCurrentTime)
    ])
    my_kp1 = h_BicycleModel(mx_kp1, mv_k, kk);
    H = numerical_jac_x_2(mx_kp1, mv_k, kk);
    M = numerical_jac_w_2(mx_kp1, mv_k, kk);
    P12 = P_kp1*H';
    K = P12*inv((H*P12)+(M*R*M'));
    mx_km1 = mx_kp1 + K*(y_kp1 - my_kp1);  % Correction
    yy=(y_kp1 - my_kp1);
    P_km1 = K*R*K' + (eye(xDim)-K*H)*P_kp1*(eye(xDim)-K*H)';
    mx_km1;
  else
    disp('<>0')
    mx_km1 = mx_kp1;  % Shift
```

```
        P_km1 = P_kp1;
    end
end

function [z_EKF,u1,u2] = hStepEstimation(z_EKF,u1,u2,vhMdl,trMdl,...
horizonte,itercont,ipp,xref_tot,yref_tot,dt,w,xrefn,yrefn)

    %Estimaci n h pasos en el futuro
    z_EKF(:,itercont+1,1) = z_EKF(:,itercont,2);
    kk=itercont;
    Kp = 1;
    gamma=0.55;
    %Modelo del proceso
    for i=2:horizonte

        % get state and control actions
        Vx=z_EKF(1,itercont+1,i-1);
        Vy=z_EKF(2,itercont+1,i-1);
        X=z_EKF(3,itercont+1,i-1);
        Y=z_EKF(4,itercont+1,i-1);
        psi=z_EKF(5,itercont+1,i-1);
        wz=z_EKF(6,itercont+1,i-1);

        ax=u1(itercont+1,i-1);
        delta=u2(itercont+1,i-1);

        % extract parameters
        a=vhMdl(1);
        b=vhMdl(2);
        m=vhMdl(3);
        I=vhMdl(4);
        L = a +b;

        Caf=trMdl(1);
        Car=trMdl(2);

        % compute lateral tire forces
        Fyf = -Caf * atan(( (Vy + wz*a)/max(Vx, 5*0.44704) ) - delta);
        Fyr = -Car * atan( (Vy - wz*b)/max(Vx, 5*0.44704) );

        % compute next state Hacer prueba con ruido y sin ruido
        Vx_next = Vx + dt*(ax + w(1,kk));
        Vy_next = Vy + dt*(tan(delta)*(ax - wz*Vy) + (Fyf/cos(delta) + Fyr)/m -
    wz*Vx +w(2,kk));
        X_next = X + dt*(Vx*cos(psi) - Vy*sin(psi) + w(3,kk));
        Y_next = Y + dt*(Vx*sin(psi) + Vy*cos(psi) + w(4,kk));
        psi_next = psi + dt*(wz + w(5,kk));
        wz_next = wz + dt*(m*a/I*tan(delta)*(ax-wz*Vy) + a*Fyf/(I*cos(delta)) -
    b*Fyr/I + w(6,kk));

%Sin ruido
%          Vx_next = Vx + dt*(ax);
%          Vy_next = Vy + dt*(tan(delta)*(ax - wz*Vy) + (Fyf/cos(delta) + Fyr)/m
    - wz*Vx);
%          X_next = X + dt*(Vx*cos(psi) - Vy*sin(psi));
%          Y_next = Y + dt*(Vx*sin(psi) + Vy*cos(psi));
%          psi_next = psi + dt*(wz);
%          wz_next = wz + dt*(m*a/I*tan(delta)*(ax-wz*Vy) + a*Fyf/(I*cos(delta))
    - b*Fyr/I);

        z_EKF(:,itercont+1,i) = [Vx_next; Vy_next; X_next; Y_next; psi_next;
    wz_next];

        vxe=z_EKF(1,itercont+1,i);
        vxef=vxe;
        xe=z_EKF(3,itercont+1,i);
        xecf=xe;
        ye=z_EKF(4,itercont+1,i);
        yecf=ye;
        psie=z_EKF(5,itercont+1,i);
        psief=psie;
        pside=z_EKF(6,itercont+1,i);

        %Path planing
        flag_pp = true;

        look_ahead = 5;
        while(flag_pp && ipp<1200)
            ipp = ipp+1;
            dist_x = abs(xecf - xref_tot(ipp));
            dist_y = abs(yecf - yref_tot(ipp));
            dist_total = sqrt(dist_x*dist_x + dist_y*dist_y);

            if(dist_total > look_ahead)
                xrefn = xref_tot(ipp);
                yrefn = yref_tot(ipp);
                flag_pp = false;
                ipp = ipp-1;
            end
        end
        if ipp == 1200
           ipp = 1;
        end

        dd=sqrt((yrefn-yecf)^2+(xrefn-xecf)^2);
        alfa=atan2((yrefn-yecf),(xrefn-xecf))-psief;
        psidrefn=(2*vxef*sin(alfa))/dd;
        psirefn=psief+dt*psidrefn;

        %Calcular acci n de control futura
        u1(itercont+1,i)=u1(itercont+1,i-1); %ax
        u2(itercont+1,i) = atan2(psidrefn*L,vxe)+ (psidrefn-pside)*Kp*gamma; %
    delta
    end


end

function [z_next] = f_BicycleModel(z, w, u, vhMdl, trMdl, dt, kk)

  Vx=z(1,1);
  Vy=z(2,1);
  X=z(3,1);
  Y=z(4,1);
  psi=z(5,1);
  wz=z(6,1);

  ax=u(1);
  delta=u(2);

  a=vhMdl(1);
  b=vhMdl(2);
  m=vhMdl(3);
  I=vhMdl(4);

  Caf=trMdl(1);
  Car=trMdl(2);

  Fyf = -Caf * atan(( (Vy + wz*a)/max(Vx, 5*0.44704) ) - delta);
  Fyr = -Car * atan( (Vy - wz*b)/max(Vx, 5*0.44704) );

  Vx_next = Vx + dt*(ax + w(1,kk));
  Vy_next = Vy + dt*(tan(delta)*(ax - wz*Vy) + (Fyf/cos(delta) + Fyr)/m - wz*Vx
    +w(2,kk));
  X_next = X + dt*(Vx*cos(psi) - Vy*sin(psi) + w(3,kk));
  Y_next = Y + dt*(Vx*sin(psi) + Vy*cos(psi) + w(4,kk));
  psi_next = psi + dt*(wz + w(5,kk));
  wz_next = wz + dt*(m*a/I*tan(delta)*(ax-wz*Vy) + a*Fyf/(I*cos(delta)) - b*Fyr/
    I + w(6,kk));


  z_next = [Vx_next; Vy_next; X_next; Y_next; psi_next; wz_next];
end


function [y_next] = h_BicycleModel(z, v, kk)

  % compute next output
  y_next = [z(1,1)+v(1,kk); z(3,1)+v(2,kk); z(4,1)+v(3,kk); z(5,1)+v(4,kk)];
end


% Functions to compute Jacobian matrices
function [jac]=numerical_jac_x(x, w, u_ekf, vhMdl, trMdl, dt, kk)

  y = f_BicycleModel(x, w, u_ekf, vhMdl, trMdl, dt, kk);
  jac=zeros(size(y,1),size(x,1));
  eps=1e-5;
  xp=x;
  for i=1:size(x,1)
    xp(i,1) = x(i,1) + eps/2.0;
    yhi = f_BicycleModel(xp, w, u_ekf, vhMdl, trMdl, dt, kk);
    xp(i,1) = x(i,1) - eps/2.0;
    ylo = f_BicycleModel(xp, w, u_ekf, vhMdl, trMdl, dt, kk);
    xp(i,1) = x(i,1);
    jac(:,i) = (yhi-ylo)/eps;
  end
end

 function [jac]=numerical_jac_w(x, w, u_ekf, vhMdl, trMdl, dt, kk)

  y = f_BicycleModel(x, w, u_ekf, vhMdl, trMdl, dt, kk);
  jac=zeros(size(y,1),size(w,1));
  eps=1e-5;
  wp=w;
  for i=1:size(w,1)
    wp(i,kk) = w(i,kk) + eps/2.0;
    yhi = f_BicycleModel(x, wp, u_ekf, vhMdl, trMdl, dt, kk);
    wp(i,kk) = w(i,kk) - eps/2.0;
    ylo = f_BicycleModel(x, wp, u_ekf, vhMdl, trMdl, dt, kk);
    wp(i,1) = w(i,kk);
    jac(:,i) = (yhi-ylo)/eps;
  end
 end

function [jac]=numerical_jac_x_2(x, v, kk)

  y = h_BicycleModel(x, v, kk);
  jac=zeros(size(y,1),size(x,1));
  eps=1e-5;
  xp=x;
  for i=1:size(x,1)
    xp(i,1) = x(i,1) + eps/2.0;
    yhi = h_BicycleModel(xp, v, kk);
    xp(i,1) = x(i,1) - eps/2.0;
    ylo = h_BicycleModel(xp, v, kk);
    xp(i,1) = x(i,1);
    jac(:,i) = (yhi-ylo)/eps;
  end
end

function [jac]=numerical_jac_w_2(x, v, kk)

  y = h_BicycleModel(x, v, kk);
  jac=zeros(size(y,1),size(v,1));
  eps=1e-5;
  vp=v;
  for i=1:size(v,1)
    %vp(i,1) = v(i,1) + eps/2.0;
    vp(i,kk) = v(i,kk) + eps/2.0;
    yhi = h_BicycleModel(x, vp, kk);
    %vp(i,1) = v(i,1) - eps/2.0;
    vp(i,kk) = v(i,kk) - eps/2.0;
    ylo = h_BicycleModel(x, vp, kk);
    %vp(i,1) = v(i,1);
    vp(i,kk) = v(i,kk);
    jac(:,i) = (yhi-ylo)/eps;
  end
end
```

Programa principal

```
%%% EXECUTE THIS CODE THE FIRST TIME TO GENERATE THE GLOBAL
VARIABLE ”INICIOKERNELS” (FOR INITIALIZATIONS) %%%
clc;
clear all;

tic
global INICIOKERNELS;


for simu=20:10:20
%Variables TFM
titulo = ”h = ” + int2str(simu);
%titulo = ”P rdidas 25% y retardos”
INICIOKERNELS.horizonte=simu;
INICIOKERNELS.flag_ini=1;
INICIOKERNELS.M=10; %Multiplicidad en el filtro
INICIOKERNELS.Kp=1.0;
INICIOKERNELS.gamma=0.55;
INICIOKERNELS.x(1)=0.0;
INICIOKERNELS.y(1)=79;
INICIOKERNELS.vx(1)=5;
INICIOKERNELS.vy(1)=0;
INICIOKERNELS.psi(1)=273*pi/180;
INICIOKERNELS.psid(1)=0;
INICIOKERNELS.ax(1)=0.05;
INICIOKERNELS.delta=0;
INICIOKERNELS.ay(1)=−0.001;
INICIOKERNELS.vxe(1)=INICIOKERNELS.vx(1);
INICIOKERNELS.pside(1)=INICIOKERNELS.psid(1);
INICIOKERNELS.psidd(1)=0;
INICIOKERNELS.vxf=INICIOKERNELS.vx(1);
INICIOKERNELS.vyf=INICIOKERNELS.vy(1);
INICIOKERNELS.ayf=INICIOKERNELS.ay(1);
INICIOKERNELS.psif=INICIOKERNELS.psi(1);
INICIOKERNELS.xcf = INICIOKERNELS.x(1);
INICIOKERNELS.ycf = INICIOKERNELS.y(1);
INICIOKERNELS.psidf=0.1;
INICIOKERNELS.axf=INICIOKERNELS.ax(1);
%Par metros rajmani
INICIOKERNELS.lf=1.2;
INICIOKERNELS.lr=1.65;
INICIOKERNELS.m=1800;
INICIOKERNELS.Caf=140000;
INICIOKERNELS.Car=120000;
INICIOKERNELS.Iz=3270;
INICIOKERNELS.a=1.20;
INICIOKERNELS.b=1.65;
INICIOKERNELS.L=INICIOKERNELS.a+INICIOKERNELS.b;
INICIOKERNELS.ftire_stiffness=140000.0;
INICIOKERNELS.rtire_stiffness=120000.0;
INICIOKERNELS.mass=1800.0;
INICIOKERNELS.iz=3270.0;
%Para el EKF
INICIOKERNELS.vhMdl=[INICIOKERNELS.a,INICIOKERNELS.b,INICIOKERNELS.mass,
    INICIOKERNELS.iz];
INICIOKERNELS.trMdl=[INICIOKERNELS.Caf,INICIOKERNELS.Car];
load meas_noise
load proc_noise
INICIOKERNELS.v_ekf=v_ekf;
INICIOKERNELS.w_ekf=w_ekf;
var_gps=1.0e-6;
var_v=1.0e-4;
var_psi=1.0e-6;
var_ax=1.0e-4;
var_delta=1.0e-4;
var_noise=1.0e-4;
INICIOKERNELS.var1 = [var_ax, var_delta, var_noise, var_noise, var_noise,
    var_noise];
INICIOKERNELS.var2 = [var_v, var_gps, var_gps, var_psi];
INICIOKERNELS.P=eye(6);
INICIOKERNELS.Q=diag(INICIOKERNELS.var1);
INICIOKERNELS.R2=diag(INICIOKERNELS.var2);

INICIOKERNELS.ITERACIONES = 5500;      % number of iterations
INICIOKERNELS.T = 0.01;   % actuation period
INICIOKERNELS.NT = INICIOKERNELS.T*INICIOKERNELS.M; % sampling period
INICIOKERNELS.Nbif = INICIOKERNELS.M;      % multiplicity
% Sampling time
INICIOKERNELS.dt=INICIOKERNELS.T;

load ref_soriano_coche %Carga xref_tot yref_tot cada 0.1 metros en cuadrado 20 x
    20 metros
factor_escala = 4; %escalamos las referencias
xref_tot = xref_tot*factor_escala;
yref_tot = yref_tot*factor_escala;
psiref_tot = [zeros(400,1);(pi/2)*ones(400,1);pi*ones(400,1);(3*pi/2)*ones
    (400,1)];
%
INICIOKERNELS.xref=xref_tot;
INICIOKERNELS.yref=yref_tot;
INICIOKERNELS.xref_tot=[INICIOKERNELS.xref(600:800);INICIOKERNELS.xref(1:800)
    ;...
INICIOKERNELS.xref(1:200)];
INICIOKERNELS.yref_tot=[INICIOKERNELS.yref(600:800);INICIOKERNELS.yref(1:800)
    ;...
INICIOKERNELS.yref(1:200)];

for hh=1:length(INICIOKERNELS.xref_tot)-1
    INICIOKERNELS.dref(hh)=sqrt((INICIOKERNELS.xref_tot(hh+1)-...
    INICIOKERNELS.xref_tot(hh))^2+(INICIOKERNELS.yref_tot(hh+1)-...
    INICIOKERNELS.yref_tot(hh))^2);
end



INICIOKERNELS.psirefn=270*pi/180;
INICIOKERNELS.psidrefn=0.1;
INICIOKERNELS.km = 1;
INICIOKERNELS.ipp=1;

% delays
load retardos_009.mat

INICIOKERNELS.retardoCA=retardo;

% dropouts
load perdidas_15_doblecero.mat

INICIOKERNELS.lossc=lossc;

INICIOKERNELS.losca=losca;

load w_ekf_sim.mat

sim ('Control_en_red.slx');

ind = 1;

x = ans.x';
x=x(3:length(x)-1);
y = ans.y';
y=y(3:length(y)-1);
xrefnac = ans.xrefn';
xrefnac = xrefnac(3:length(xrefnac)-1);
yrefnac = ans.yrefn';
yrefnac = yrefnac(3:length(yrefnac)-1);

distp2p=(sqrt((xrefnac-x).^2+(yrefnac-y).^2));

distbuena=zeros(1,length(distp2p));
for i=1:length(x)
    distbuena(i)=100000;
    for j=1:length(xrefnac)
        distbuena(i)=min(distbuena(i),(sqrt((xrefnac(j)-x(i))^2+(yrefnac(j)-y(i)
    )^2)));
    end
end

figure
plot(distp2p)
hold on
plot(distbuena)
xlabel("Iteraciones");
ylabel("Dist(m)");
legend("Dist. ref din mica","Dif. con la trayectoria");
title(titulo)

% ndice J1
J1mod=0;
for i=ind:length(xrefnac)
    J1mod=J1mod+distbuena(i);
end

% ndice J2
J2mod=distbuena(ind);
for i=ind:length(xrefnac)
    J2mod=max(J2mod,distbuena(i));
end

J1mod
J2mod
desv = std2(distp2p)

% Ploteos;

figure
plot(xref_tot,yref_tot,'b');
xlabel("X(m)");
ylabel("Y(m)");
hold on
```

```
plot(x,y,'—r','MarkerSize',0.5);
legend("Referencia","Trayectoria");
title(titulo);

filename = strcat("Simulacion_15_",int2str(simu));
save(filename,"J1mod","J2mod","desv")
end
toc
```

Esquema Simulink

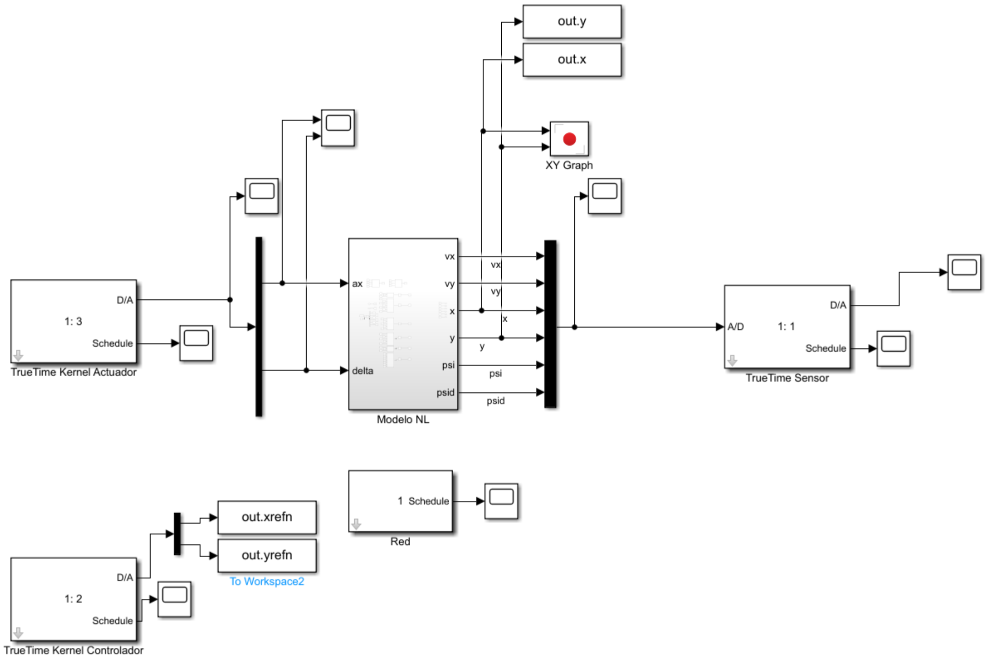

Figura 55: Esquema Simulink

Referencias

[1] Á. Cuenca, W. Zhan, J. Salt, J. Alcaina, C. Tang, M. Tomizuka, A remote control strategy for an autonomous vehicle with slow sensor using Kalman filtering and dual-rate control, Sensors 19 (13) (2019). doi:10.3390/s19132983.
URL https://www.mdpi.com/1424-8220/19/13/2983

[2] J. Salt Ducajú, J. Salt Llobregat, Á. Cuenca, M. Tomizuka, Autonomous ground vehicle lane-keeping LPV model-based control: Dual-rate state estimation and comparison of different real-time control strategies, Sensors 21 (4) (2021). doi:10.3390/s21041531.
URL https://www.mdpi.com/1424-8220/21/4/1531

[3] R. Carbonell, Á. Cuenca, V. Casanova, R. Pizá, J. Salt, Dual-rate extended Kalman filter based path-following motion control for an unmanned ground vehicle. Realistic simulation, Sensors 21 (22) (2021).
URL https://www.mdpi.com/1424-8220/21/22/7557

[4] Ford Motor Company, Ficha técnica del vehículo 2017 lincoln mkz (January 2017).
URL https://es.lincoln.com/img/lincoln/ES_140329.pdf

[5] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, C. Peng, Networked control systems: A survey of trends and techniques, IEEE/CAA Journal of Automatica Sinica 7 (1) (2020) 1–17. doi:10.1109/JAS.2019.1911651.

[6] A. Cervin, D. Henriksson, M. Ohlin, Truetime 2.0 reference manual (April 2016).
URL http://archive.control.lth.se/media/Research/Tools/TrueTime/report_2016-02-10.pdf

[7] ai2, Áreas de investigación ai2 (January 2015).
URL https://www.ai2.upv.es/area-de-control-de-procesos/

[8] A. González, Á. Cuenca, J. Salt, J. Jacobs, Robust stability analysis of an energy-efficient control in a networked control system with application to unmanned ground vehicles, Information Sciences 578 (2021) 64–84. doi:https://doi.org/10.1016/j.ins.2021.07.016.
URL https://www.sciencedirect.com/science/article/pii/S0020025521007064

[9] J. Alcaina, A. Cuenca, J. Salt, V.Casanova, R. Pizá, Delay-independent dual-rate pid controller for a packet-based networked control system, Information Sciences 484 (2019) 27–43. doi:https://doi.org/10.1016/j.ins.2019.01.059.
URL https://www.sciencedirect.com/science/article/pii/S0020025519300726

[10] R. Rajamani, Vehicle Dynamics and Control, Springer (2012). doi:10.1007/978-1-4614-1433-9.

[11] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Oren-stein, J. Paefgen, I. Penn, S. Thrun Junior, The Stanford entry in the Urban Challenge, Journal of Field Robotics 25 (9) (2008) 569–597. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20258, doi:https://doi.org/10.1002/rob.20258.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20258

[12] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, How, Real-time motion planning with applications to autonomous urban driving, IEEE Transactions on Control Systems Technology 17 (5) (2009) 1105–1118. doi:10.1109/TCST.2008.2012116.

[13] A. Sala, Predicción en sistemas dinámicos lineales: observador óptimo (filtro de Kalman) (2018).

URL http://personales.upv.es/asala/DocenciaOnline/material/FiltroKalman.pdf