Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Informe de Práctica

CC5901 – Práctica Profesional II

Empresa:	Departamento de Ingeniería Eléctrica Universidad de Chile
Supervisor:	Sandra Céspedes, Ph.D.
Alumno:	Manuel Olguín
Carrera:	Ingeniería Civil en Computación
RUT:	18.274.982 – 6
E-Mail:	molguin@dcc.uchile.cl
Tel:	+56 9 7463 6997

4 de abril de 2017
Santiago, Chile.

1. Certificado de la Empresa

2. Observaciones

Índice

Índice de figuras

3. Resumen

El presente documento detalla el trabajo realizado entre Agosto y Octubre del 2016 por el alumno Manuel Olguín en el Grupo de Investigación en Wireless Networking del Departamento de Ingeniería Eléctrica de la Universidad de Chile. Dicho trabajo se realizó bajo la supervisión de la profesora Sandra Céspedes, y consistió en la modelación de un sistema de alarma temprana para ciclistas en el contexto de un Sistema de Transporte Inteligente.

4. Introducción

4.1. Lugar de Trabajo

El Grupo de Investigación en Wireless Networking (en adelante, *el Grupo*) es un conjunto de estudiantes y profesionales del área de Tecnologías de la Información y Comunicaciones (en adelante, *TICs* [1]) liderado por la profesora Sandra Céspedes [2], docente del Departamento de Ingeniería Eléctrica de la Universidad de Chile (en adelante, *DIE* [3]). El Grupo desarrolla sus labores de investigación y desarrollo en el Laboratorio de Comunicaciones Avanzadas, ubicado en el 5º piso del departamento del DIE ubicado en Av. Tupper 2007, Santiago.

4.2. Equipo de Trabajo

Los miembros del grupo son estudiantes de pre- y posgrado, además de profesionales ligados a las TICs, de diversos orígenes, especialidades e incluso nacionalidades. En específico, el trabajo realizado por el practicante fue desarrollado bajo la supervisión de la doctora Sandra Céspedes y en colaboración con un estudiante de pregrado del DIE.

4.3. Software y Conceptos Importantes

El trabajo se realizó principalmente en C++ y Python, utilizando software especializado para la modelación y estudio de redes inalámbricas, redes de transporte y la integración de ambas en Sistemas de Transporte Inteligente.

4.3.1. Sistemas de Transporte Inteligente

Los sistemas de transporte conforman la columna vertebral de nuestras ciudades, contribuyendo directamente al desarrollo de la sociedad urbana. Un sistema de transporte bien diseñado y eficiente permite el desplazamiento rápido y cómodo de personas y bienes; en cambio, uno ineficiente genera grandes problemas, alargando los tiempos de viaje y aumentando la contaminación atmosférica.

Los Sistemas de Transporte Inteligente (en adelante *ITS*, por sus siglas en inglés – *Intelligent Transportation Systems*) surgen como una respuesta a la necesidad de optimización y modernización de los sistemas de transporte existentes. La Unión Europea define a los ITS como aplicaciones avanzadas que, sin incorporar inteligencia como tal, pretenden proveer servicios innovadores relacionados con distintos modos de transporte y de administración de tráfico, que además otorgan información a los usuarios, permitiéndoles utilizar el sistema de transporte de manera más segura, coordinada e inteligente [4]. De acuerdo al Departamento de Transportes de los EEUU, estos sistemas se pueden dividir en dos grandes categorías [5]:

Sistemas de Infraestructura Inteligente Tienen como enfoque el manejo de los sistemas de transporte a niveles macro, y la transmisión de información oportuna a los usuarios. Esta categoría incluye, entre otros, sistemas de advertencia y señalización dinámica en ruta (ya sea a través de pantallas o sistemas de comunicación inalámbrica), sistemas de pago electrónico y de coordinación del flujo de tráfico.

Sistemas de Vehículos Inteligentes Engloba todo aquello relacionado con la automatización y optimización de la operación de un vehículo. Dentro de esta categoría se incluyen sistemas de advertencia y prevención de colisiones, de asistencia al conductor — por ejemplo, sistemas de navegación — y control autónomo de vehículos.

El factor común entre ambas categorías es la necesidad de extraer información en tiempo real desde el entorno, la cual debe procesarse y en muchos casos generar una respuesta a transmitir al usuario. Para este fin, actualmente se utilizan tecnologías de comunicación inalámbricas, tanto de área local (los estándares incluídos en la familia WLAN, IEEE 802.11), como de área personal (WPAN, IEEE 802.15) [6, 7, 8].

4.3.2. OMNeT++

Objective Modular Network Testbed in C++, mejor conocido como OMNeT++ [9], es un *framework* y librería para la simulación de sistemas de eventos discretos, principalmente utilizado para la simulación y análisis de redes de comunicaciones. Consiste en un conjunto de módulos y librerías en C++, además de un entorno de desarrollo integrado (en adelante, *IDE* por sus siglas en inglés – *Integrated Development Environment*) y una interfaz gráfica, los cuales posibilitan la construcción de simulaciones dinámicas y extensibles tanto de sistemas cableados como inalámbricos.

OMNeT++ es software libre, de fuente abierta, y altamente popular en la comunidad académica.

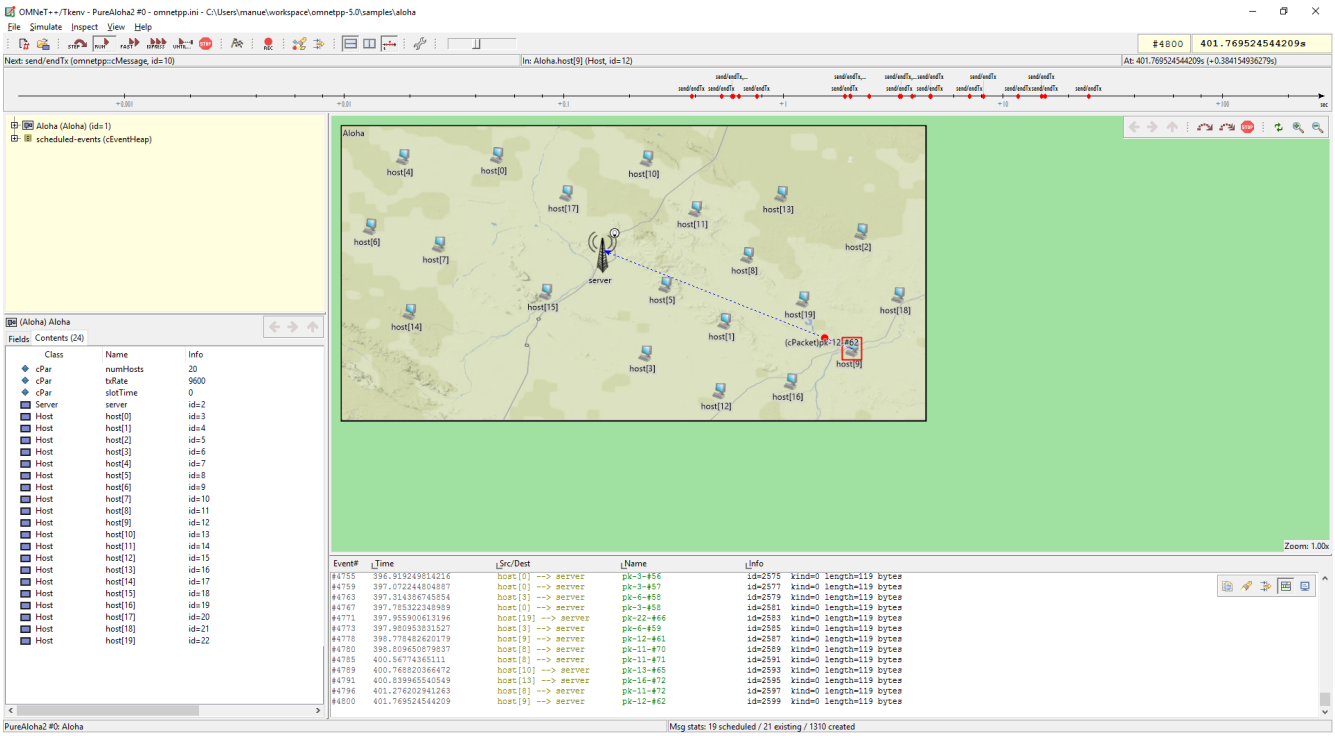

Figura 1: Entorno gráfico de simulación de OMNeT++

4.3.3. SUMO

SUMO (*Simulation of Urban MObility*) es un simulador de tráfico vehicular de fuente abierta desarrollado originalmente por el Instituto de Sistemas de Transporte del Centro Aeroespacial Alemán (DLR, por sus siglas en alemán – *Deutsches Zentrum für Luft- und Raumfahrt e.V.*) [10]. Es un simulador microscópico, en el cual cada vehículo es simulado de manera particular, moviéndose individualmente por la red.

4.3.4. VEINS

VEhicles In Network Simulation, *VEINS*, es un framework de fuente abierta para la simulación, tanto en su aspecto de comunicaciones como de transporte, de Sistemas de Transporte Inteligente.

4.4. Situación Previa

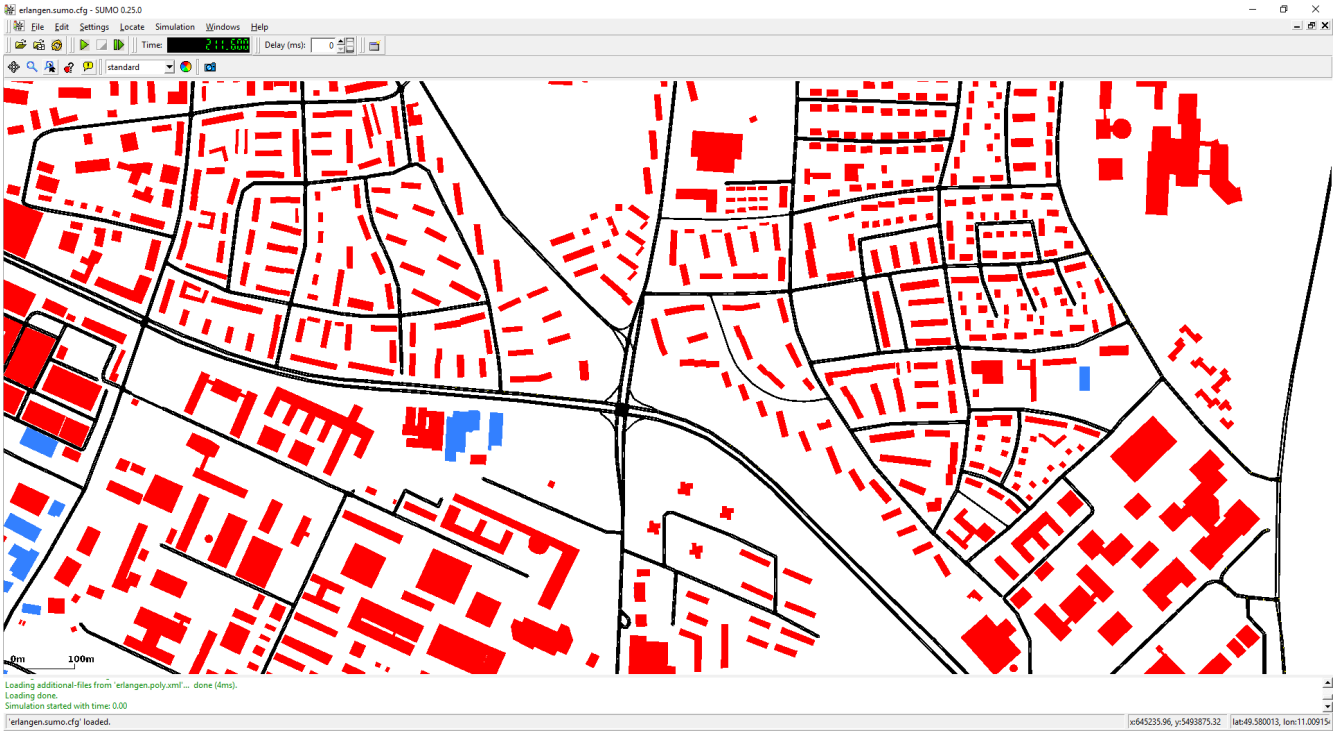

Figura 2: Simulación de una red en SUMO

5. Trabajo Realizado

El trabajo realizado tenía como fin último la elaboración de un prototipo de un sistema de alerta temprana para ciclistas en un ITS. El fin de este sistema era el de predecir, mediante un modelo de probabilidades bayesianas descrito por Liebner *et al.* en [11], la probabilidad de un giro sorpresivo de un vehículo en un cruce, y notificar esto a ciclistas cercanos a dicho cruce (se asumieron ciclistas y vehículos en el contexto de un ITS, dotados de capacidades de comunicación inalámbrica).

Este trabajo se dividió entonces en dos partes:

1. Estudio y comprensión del software especializado OMNeT++, SUMO y VEINS, y la posterior elaboración de un prototipo simple para ilustrar la factibilidad del proyecto.
2. Implementación del sistema, utilizando como base el prototipo anterior y utilizando registros de tráfico real.

Cabe destacar que las labores del practicante abarcaron la implementación del modelo vehicular y la integración del modelo predictivo a este. El estudio y la implementación del modelo descrito en [11] quedó a cargo de Sebastián Piña M., alumno del DIE y compañero de equipo del practicante.

5.1. Estudio del software y elaboración de un prototipo inicial simple

El primer mes de la práctica se enfocó totalmente en el estudio del software especializado, enfocado principalmente en VEINS y la elaboración de modelos de sistemas de transporte inteligentes utilizando el framework. Esto además significó el aprendizaje de C++ por parte del practicante. Con el fin de realizar un aprendizaje didáctico y aplicado, se implementó además un prototipo simple de un cruce vehicular con carriles para ciclistas.

5.1.1. Prototipo inicial

El prototipo inicial se construyó en base al trabajo previamente realizado por Javiera Born (descrito en la sección 4.4). Este consistió en una simulación muy simple, compuesta por un vehículo motorizado y una bicicleta, los cuales interactúan en un cruce. A continuación se detalla la elaboración de este prototipo, y los resultados obtenidos para la siguiente etapa del trabajo de práctica.

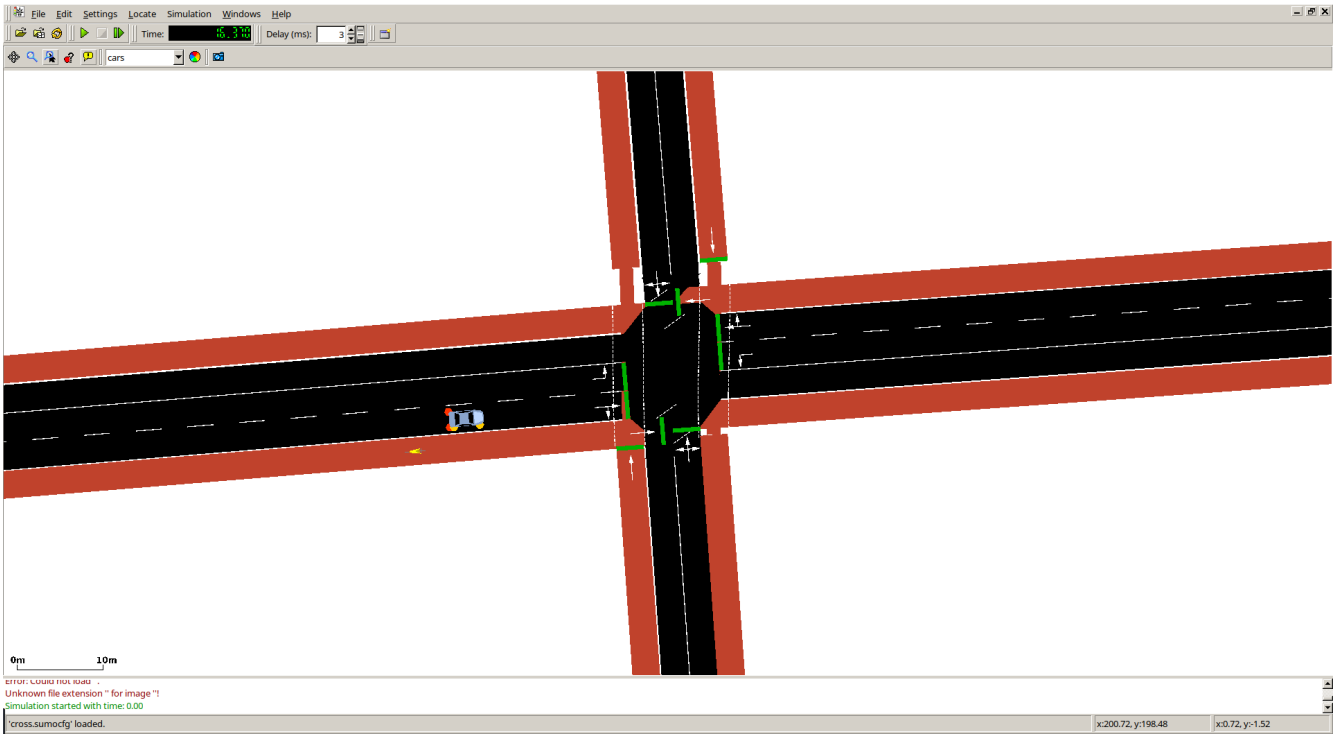

Figura 3: Prototipo inicial en ejecución

5.1.2. Parámetros físicos de la simulación

El aspecto de transporte de la simulación no presenta grandes cambios respecto a lo desarrollado anteriormente por J. Born – se reutilzó el cruce vehicular simple compuesto por una vía de tres pistas y otra de dos, ambas con ciclovías a los costados. Se modificaron los flujos vehiculares para que consistieran únicamente en un automóvil y una bicicleta, y para sincronizar su llegada al cruce y poder simular la detención de la bici en

reacción a un mensaje de viraje emitido por el automóvil (detalles de la implementación de esto en la sección 5.1.3).

5.1.3. Simulación de comunicación inalámbrica y lógica?

La lógica de la simulación se implementó en un módulo de OMNeT++ llamado *BikeManeuver*. El comportamiento de ambos vehículos fue implementado en este único módulo; al iniciar la simulación, se verifica (entre otros parámetros) el identificador del vehículo, en base a lo cual se asigna un valor a un *flag* que determina el comportamiento del módulo en el resto de la simulación.

Cada vehículo fue modelado como una máquina de estados finita, los cuales cambian de estado en respuesta a estímulos exteriores y/o interiores (la figura 4 ilustra estos diagramas de estado). A continuación se detalla el comportamiento de ambos, y el código 1 expone un extracto del código del módulo.

Bicicleta El comportamiento de esta es extremadamente simple. El módulo espera hasta recibir un mensaje de advertencia desde el vehículo; al recibirlo, verifica la distancia actual de la bicicleta al cruce. De ser esta menor a una distancia D establecida, la bicicleta se detiene hasta recibir un segundo mensaje del automóvil, el cual confirma que este último ha finalizado su giro. Finalmente, la bicicleta resume su trayectoria hasta terminar la simulación.

Automóvil El automóvil tiene un comportamiente ligeramente más complicado, ya que debe constantmente estar monitoreando su distancia al cruce para determinar cuando emitir los mensajes que espera la bicicleta. Para esto, al principio de la simulación se programa un "automensaje" (*selfbeacon* en inglés) a ser enviado por el módulo a si mismo en el siguiente instante de simulación ($0{,}1$ más tarde). Al recibir dicho mensaje, se programa un nuevo *selfbeacon* y se verifica si la distancia del vehículo al cruce es menor o igual a la distancia D previamente mencionada. De ser así, se emite el primer mensaje de advertencia, y se cambia el estado a "girando", hasta detectar (mediante la misma técnica anterior de mediciones periódicas) que se encuentra nuevamente fuera del cruce (distancia $\geq D$), momento que se emite el segundo y último mensaje y se vuelve a la operación normal.

Cabe notar que la técnica de uso de *selfbeacons* para realizar operaciones periódicas en OMNeT++ es altamente común, y fue uno de los principales aprendizajes de esta estapa del trabajo de práctica [12].

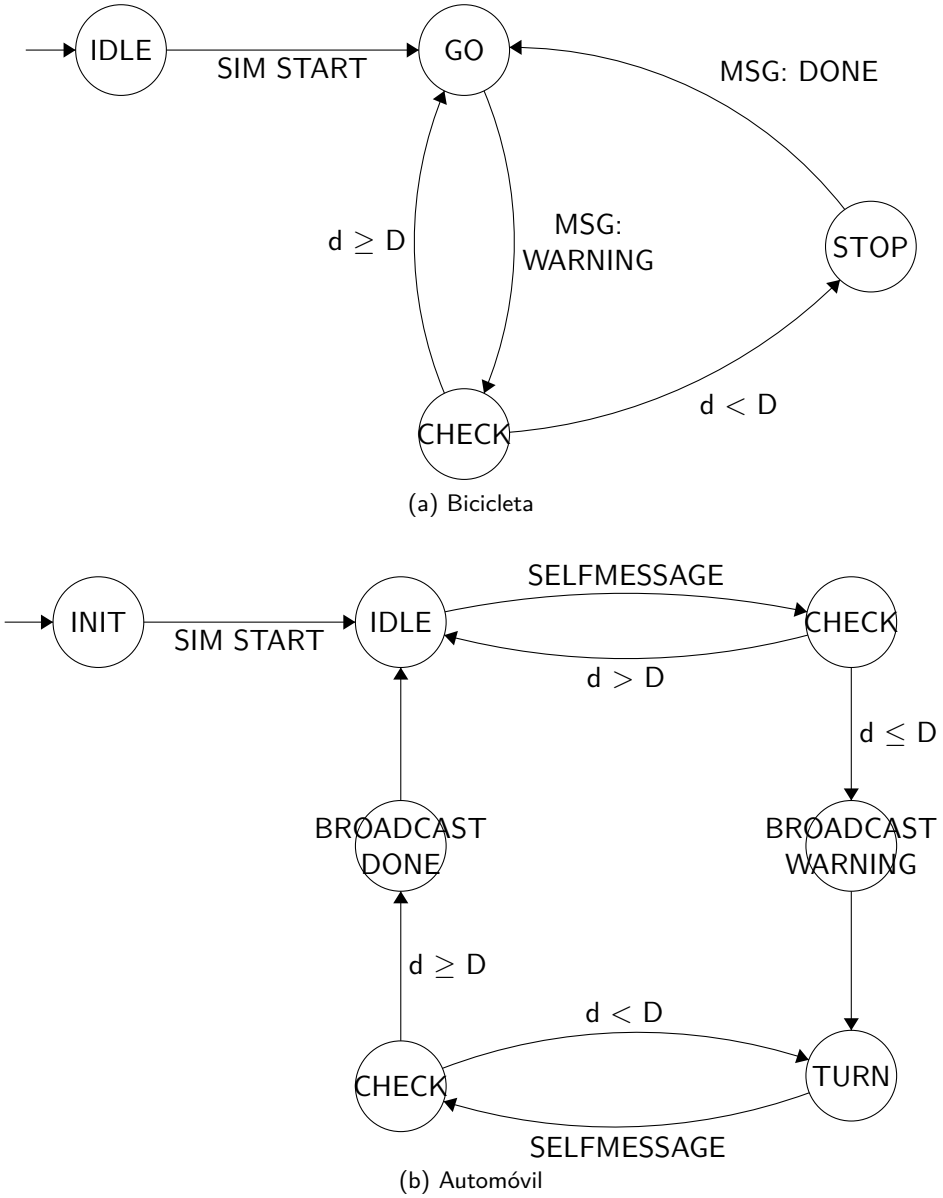

Figura 4: Diagramas de estado de bicicleta y automóvil. Notar que se representan también estados transitorios, para los cuales no se estimó necesario crear un estado "real" en la implementación.

5.2. Modelación del sistema de alarma temprana

6. Conclusiones

Referencias

[1] *Tecnologías de la Información y Comunicaciones* - https://es.wikipedia.org/wiki/Tecnolog%C3%ADas_de_la_informaci%C3%B3n_y_la_comunicaci%C3%B3n

[2] *Sandra Céspedes U.* - http://www.cec.uchile.cl/~scespedes/

[3] *Departamento de Ingeniería Eléctrica* - http://die.cl/

[4] Directive 2010/40/EU of the European Parliament and of the Council *on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport*, 2010 O.J. L 207/1

[5] U.S. Department of Transportation *Office of the Assistant Secretary for Research and Technology (OST-R)* http://www.itsoverview.its.dot.gov/ (*04/2017*)

[6] D. J. Dailey, K. McFarland and J. L. Garrison *Experimental study of 802.11 based networking for vehicular management and safety*, Intelligent Vehicles Symposium (IV), 2010 IEEE, San Diego, CA, 2010, pp. 1209-1213.
doi: 10.1109/IVS.2010.5547955

[7] W. Xiong; X. Hu; T. Jiang. *Measurement and Characterization of Link Quality for IEEE 802.15.4-compliant Wireless Sensor Networks in Vehicular Communications*, IEEE Transactions on Industrial Informatics , vol.PP, no.99, pp.1-1
doi: 10.1109/TII.2015.2499121

[8] D. Jiang and L. Delgrossi. *IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments*, Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE, Singapore, 2008, pp. 2036-2040.
doi: 10.1109/VETECS.2008.458

[9] OMNeT++ Discrete Event Simulator. *An extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators*.
https://OMNeTpp.org/ (*04/2017*)

[10] SUMO – Simulation of Urban MObility. *A free and open traffic simulation suite which is available since 2001.*
http://sumo.dlr.de (*4/2017*)

[11] M. Liebner, F. Klanner, M. Baumann, C. Ruhhammer and C. Stiller. *Velocity-Based Driver Intent Inference at Urban Intersections in the Presence of Preceding Vehicles*, IEEE Intelligent Transportation Systems Magazine, vol. 5, no. 2, pp. 10-21, Summer 2013.
doi: 10.1109/MITS.2013.2246291

[12] OMNeT++ Simulation Manual *4.7.1 Self-Messages*
https://omnetpp.org/doc/omnetpp/manual/#sec:simple-modules:self-messages (*4/2017*)

7. Anexos

```
FSM_Switch(carFSM)
{
case FSM_Exit(CAR_INIT):
{
    // init simulation: go to car idle state and schedule selfbeacon for
    // periodic checks of distance to junction
    FSM_Goto(carFSM, CAR_IDLE);
    scheduleAt(simTime() + ping_interval, selfbeacon);
    break;
}
case FSM_Exit(CAR_IDLE):
{
    // got a selfbeacon?
    if (msg == selfbeacon)
    {
        // "transitory" state: check distance to crossing.
        // if we are too far away then do nothing apart from scheduling another
        // selfbeacon, otherwise send warning and change state
        if (traci->getDistance(
                traci->junction("cluster_0__0_0__2_0__4_0__6").getPosition(),
                mobility->getCurrentPosition(), false) > 30)
        {
            // too far away, schedule next check
            scheduleAt(simTime() + ping_interval, selfbeacon);
            break;
        }

        // close to the crossing, send warning and start turn
        t_channel channel = dataOnSch ? type_SCH : type_CCH;
        turnmsg = prepareWSM("data", dataLengthBits, channel, dataPriority, -1, 2);
        turnmsg->setWsmData(warning);
        sendWSM(turnmsg);
        scheduleAt(simTime() + ping_interval, selfbeacon);
        FSM_Goto(carFSM, CAR_TURNING);
    }
    break;
}
case FSM_Exit(CAR_TURNING):
{
    // got selfbeacon?
    if (msg == selfbeacon)
    {
        // "transitory" state: check distance to crossing.
        // are we still turning? then do nothing apart from scheduling selfbeacon
        // otherwise, broadcast and return to normal operation
        if (traci->getDistance(
                traci->junction("cluster_0__0_0__2_0__4_0__6").getPosition(),
                mobility->getCurrentPosition(), false) < 30)
        {
```

```
50              // still turning, do nothing and schedule next check
51              scheduleAt(simTime() + ping_interval, selfbeacon);
52              break;
53          }
54
55          // donde turning, let the bike know!
56          t_channel channel = dataOnSch ? type_SCH : type_CCH;
57          turnmsg = prepareWSM("data", dataLengthBits, channel, dataPriority, -1, 2);
58          turnmsg->setWsmData(warning);
59          sendWSM(turnmsg);
60          scheduleAt(simTime() + ping_interval, selfbeacon);
61          FSM_Goto(carFSM, CAR_IDLE);
62      }
63      break;
64  }
65  }
```

Código 1: Extracto del código encargado del comportamiento del automóvil.