

Article

Design of Low-Cost Vehicle Roll Angle Estimator Based on Kalman Filters and an Iot Architecture

Javier Garcia Guzman ^{1,†,*} , Lisardo Prieto Gonzalez ^{1,†} , Jonatan Pajares Redondo ^{2,†} ,
Susana Sanz Sanchez ^{2,†}  and Beatriz L. Boada ^{2,†} 

¹ Computer Science Department, Institute for Automotive Vehicle Safety (ISVA),
Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Madrid, Spain;
lpgonzal@inf.uc3m.es

² Mechanical Engineering Department, Institute for Automotive Vehicle Safety (ISVA),
Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911 Leganés, Madrid, Spain;
jopajare@ing.uc3m.es (J.P.R.); ssanz@ing.uc3m.es (S.S.S.); bboada@ing.uc3m.es (B.L.B.)

* Correspondence: jgarciag@inf.uc3m.es; Tel.: +34-91-624-6248

† These authors contributed equally to this work.

Received: 9 May 2018; Accepted: 31 May 2018; Published: 3 June 2018



Abstract: In recent years, there have been many advances in vehicle technologies based on the efficient use of real-time data provided by embedded sensors. Some of these technologies can help you avoid or reduce the severity of a crash such as the Roll Stability Control (RSC) systems for commercial vehicles. In RSC, several critical variables to consider such as sideslip or roll angle can only be directly measured using expensive equipment. These kind of devices would increase the price of commercial vehicles. Nevertheless, sideslip or roll angle or values can be estimated using MEMS sensors in combination with data fusion algorithms. The objectives stated for this research work consist of integrating roll angle estimators based on Linear and Unscented Kalman filters to evaluate the precision of the results obtained and determining the fulfillment of the hard real-time processing constraints to embed this kind of estimators in IoT architectures based on low-cost equipment able to be deployed in commercial vehicles. An experimental testbed composed of a van with two sets of low-cost kits was set up, the first one including a Raspberry Pi 3 Model B, and the other having an Intel Edison System on Chip. This experimental environment was tested under different conditions for comparison. The results obtained from low-cost experimental kits, based on IoT architectures and including estimators based on Kalman filters, provide accurate roll angle estimation. Also, these results show that the processing time to get the data and execute the estimations based on Kalman Filters fulfill hard real time constraints.

Keywords: real-time estimation; IoT; MEMS; low cost devices; Kalman Filter; vehicle dynamics; roll angle

1. Introduction

In recent years, there have been many advances in vehicle technologies based on the efficient use of real-time data provided by embedded sensors. Some of these technologies can help you avoid or reduce the severity of a crash such as the Roll Stability Control (RSC) systems for commercial vehicles [1–3]. These systems contribute to monitor and improve the vehicle stability, comfort and handling. This control functionality requires knowing relevant information of the vehicle dynamics, such as angular rates, angular positions, lateral and longitudinal acceleration [4–6].

Several related measures (lateral and longitudinal acceleration, yaw rate or roll rate) can be measured using low cost sensors, but others (i.e., sideslip or roll angle) need to be obtained using very

expensive equipment such as Global Positioning System (GPS) dual antenna; they cannot be measured using other sensors [7]. These kind of GPS devices would increase the price of commercial vehicles.

Nevertheless, sideslip or roll angles can be estimated using data provided by low-cost devices to solve the price problem in an efficient way. Several studies estimate sideslip applying data fusion techniques using low-cost GPS, inertial navigation systems, yaw rate and lateral acceleration [4,5].

In a similar way, roll angle can be estimated by the fusion of wheel angular speed, yaw rate, steering angle as proposed in [8] or fusing yaw rate, roll rate, lateral and longitudinal acceleration as in discussed in [7]. In most cases, the data required for this kind of proposals is obtained using a six-dimensional IMU [9]. The data fusion algorithms considered to estimate roll angle are based on different techniques implemented separately or jointly: Bayesian Filters [8], Kalman Filters [3,9,10], robust observers [6,11,12] and neural networks [3,7].

The Internet of Things (IoT) refers to the interconnection of uniquely-identifiable embedded devices within the Internet infrastructure [13]. IoT technologies are able to contribute to the effective implementation of real-time RSC systems embedded in commercial vehicles providing added value services, integrating a large amount of sensors, actuators, and communication devices (mobile devices, GPS devices, and embedded computers [14–16]) into vehicles.

The main elements to consider in a IoT architecture to implement embedded RSC systems for vehicles are [17]:

1. The first architectural component of IoT is the perception layer. It collects data using sensors, which are the most important drivers of the Internet of Things [18].
2. The next architectural component that we shall discuss is communication. The most common communication technologies for vehicular communications are Bluetooth, Zigbee, and WiFi [19].
3. The last architectural component integrates two kinds of software elements: middleware and applications. Middleware enhances interoperability of smart things and makes it easy to offer different kinds of services [15,16,20–23]. The applications include all the services notifying drivers roll risks situations and sending the appropriate orders to the vehicle active safety systems.

Nevertheless, it is necessary to consider specific requirements that IoT solution must satisfy to provide actual operational systems for RSC. These requirements include: (a) the data from the sensors in a high frequency sampling bases, at least at 50 Hz [14]; (b) in order to keep reasonable costs of commercial vehicles, it is necessary to obtain this information from low-cost sensors; (c) the data that cannot be provided directly by low-cost sensors, such as roll angle or side slip, must be estimated in a hard real time basis; (d) reduce the energy consumption to monitor the vehicle dynamics while the vehicle is running; and (e) the middleware must integrate in a synchronized, fault tolerant and reliable way the information coming from sensors and estimators.

Intel Edison and Raspberry Pi are examples of low-cost devices used to deploy IoT architectures in automotive research area [24,25]. These kinds of devices are widely used to implement IoT architectures due to their low price, flexibility and the knowledge available in the Internet to solve implementation problems [26–29]. These devices can be enriched with MEMS sensors (i.e., accelerometers and gyroscopes) to obtain the raw data required to properly estimate roll angle in real time [14,18].

As stated before, there are several types of roll angle estimators. In this research work, the estimators considered are based on Kalman techniques. A Linear Kalman filter can be used in applications that require real-time data fusion approaches because it is simple to implement, can be tailored to different sensor configurations, and provides accurate estimations in environments where the sensors provide raw data with noise [3]. Several practical applications of Kalman filters in research works in the automotive area are related to the estimation of vehicle roll angle [3,30,31], localization [32,33] and sideslip [10]. The Kalman filter is not able to handle curves as input. The Unscented Kalman filter introduces a workaround that transforms the curve to a bunch of points before executing the Kalman filter [34]. The difference between Linear and Unscented Kalman Filters is relatively small, but in the cases when

the cumulative effect of small errors can lead to relevant errors in the estimations provided (such as vehicle roll angle estimation), the use of an Unscented Kalman Filter can be appropriate [35].

When embedding roll angle estimators in commercial vehicles implementing IoT principles, it is essential to take into account that these software components must fulfill strong, real time processing restrictions. Previous research works [14] suggest that effective implementations of neural networks implemented in high-performance programming languages such as C++ are able to fulfill hard real-time restrictions. Even more so, the performance levels achieved indicate the possibility to embed, in the low-cost experimental kits, more complex estimators using a sensor fusion approach to obtain roll angle estimations closer to the actual values based on Kalman filters or combining neural networks and Kalman filters.

According to previous discussion, the objectives stated for this research work consist of integrating roll angle estimators based on Linear and Unscented Kalman filters to evaluate the precision of the results obtained and determining the fulfillment of the hard real-time processing constraints to embed this kind of estimators in IoT architectures based on low-cost equipment able to be deployed in commercial vehicles and enrich the current roll stability control systems deployed in actual vehicles. In this sense, the novelty of this research is related to the evaluation of roll angle estimation using Kalman Filters embedded in an IoT architecture using low-cost sensors and devices in real time conditions. Raspberry Pi and Intel Edison low-cost systems were considered. The performance and accuracy of the estimations obtained in these two low-cost kits are compared. Even more, practical considerations and lessons learnt implementing these kinds of architectures are provided.

The sections in this article are organized as follows. Section 2 presents the testbed designed for this research work, the hypothesis definition, the experiments specification including data gathering and analysis approach. This section also includes a discussion of the threats to validity associated to the experimental approach stated. Section 3 presents the results regarding the precision and performance of the estimators provided. Finally, Section 4, the discussion and conclusion of the results and the method are exposed.

2. Methodology

This section describes the experimental approach adopted to achieve the goals stated for this research work. Section 2.1 describes the IoT testbed for roll angle estimation based on the application of Kalman Filter. Section 2.2 enumerates the hypothesis to evaluate and the experiments defined for this purpose. Section 2.3 introduces the data gathered during the experiments execution and the data analysis methods proposed to analyze the results obtained. Finally, Section 4 summarizes the threats to validity related to this research work.

2.1. IoT Testbed for Roll Angle Estimation Based in Kalman Filter

The testbed designed for this research work was designed to be installed in any vehicle, but in the case of this research work, it was deployed in a Mercedes Sprinter van. This vehicle was considered because it is necessary to evaluate the accuracy of the estimations provided by the Kalman filters with the results presented in [3].

The considered testbed was composed of three experimental kits:

1. A ground truth kit using a VBOX 3i GPS dual antenna data logger with an IMU (Inertial Measurement Unit) from Racelogic. This sensors provide the reference measurements (roll angle roll and yaw, longitudinal and lateral acceleration).
2. A first low-cost experimental kit based on an Intel Edison chipset having connected a SparkFun “9 Degrees of Freedom” module.
3. A second low-cost kit using a Raspberry Pi 3 Model B with a Inertial Measurement Unit Shield. The technical specifications of this hardware elements are shown in Figure 1.




VBOX Kit		Raspberry Pi Kit		Intel Edison Kit	
					
Connected Sensors	VBOX 3i GPS dual antenna; IMU from Racelogic	Connected Sensors	BNO55 low-cost Inertial Measurement Unit Shield	Connected Sensors	SparkFun 9 Degrees of Freedom
Power consumption	Max. 5.5 Watts ²	Power consumption	5 V @ <1.5 W–6 W	Power consumption	3.3 V @ <1 W
Dimensions	170 × 121 × 41 mm ²	Dimensions	85.60 × 56.5 mm	Dimensions	35.5 × 25 mm
Angular rate range	±150°/s	Angular rate range	From ±125°/s–±2000°/s	Angular rate range	From ±245°/s–±2000°/s
Acceleration range	±1.7 g	Acceleration range	From ±2 g–±16 g	Acceleration range	From ±2 g–±16 g
Angular rate resolution	0.01°/s	Angular rate resolution	16 bits (From 0.003°/s for ±125°/s to 0.06°/s for ±2000°/s)	Angular rate resolution	16 bits (From 0.007°/s for ±245°/s to 0.06°/s for ±2000°/s)
Acceleration resolution	0.01 g	Acceleration resolution	14 bits (From 0.0002 g for ±2 g to 0.002 g for ±16 g)	Acceleration resolution	14 bits (From 0.0002 g for ±2 g to 0.002 g for ±16 g)
Price	> 16000 €	Price	63.2 €	Price	55,5 €

Figure 1. Technical specifications of hardware elements included in this study.

The testbed kit was deployed in a Mercedes Sprinter van. This deployment is shown in Figure 2.



Figure 2. Experimental testbed installed in Mercedes Sprinter van (1) with the ground truth kit (2, 3), Raspberry Pi and Intel Edison kits (4, 5).

The considered testbed can be described using the levels of a typical IoT architecture [17]: application, middleware, communication and perception layers. These layers are presented in the following sections and summarized in Figure 3. The software components stated for each level are developed in C++ to optimize the performance as stated in [14].

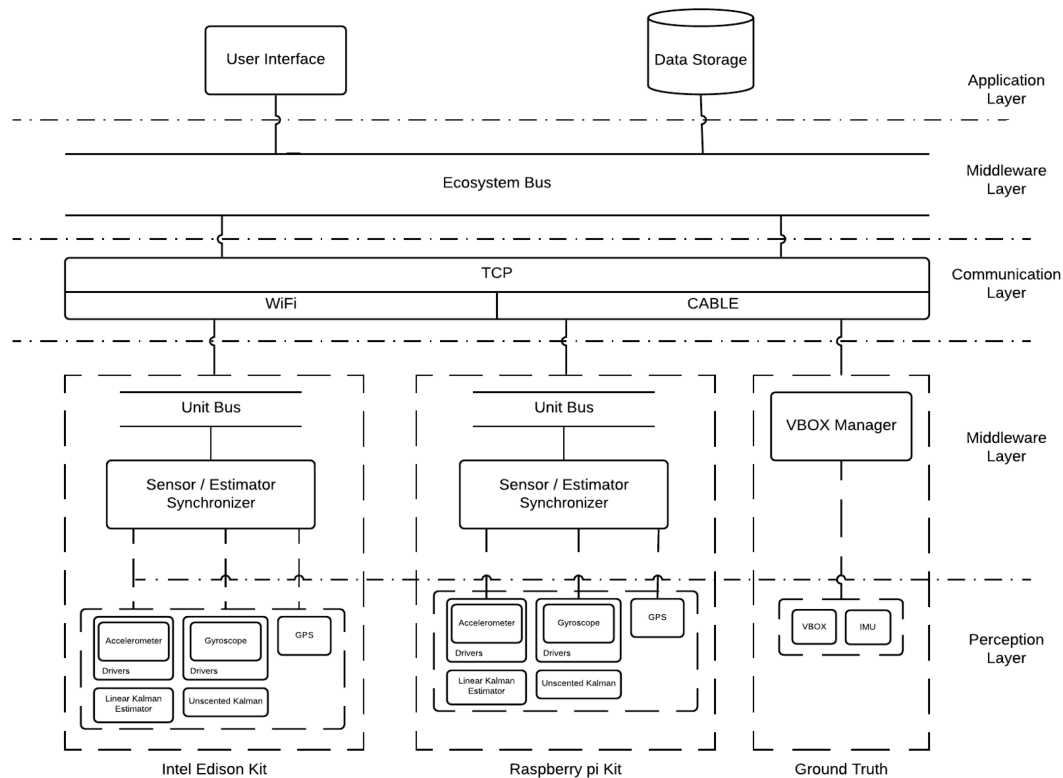


Figure 3. Testbed implemented IoT Architecture.

2.1.1. Application Layer

The application layer of this experimental kit is composed of a user interface that enables the experiments' launch and finalization. This application uses the middleware functionality to send the requests to the kits integrated to execute the experiments with a full synchronization. Even more, it is in charge of storing the data sent from the ground truth and low-cost experimental kits in files in the CSV format.

2.1.2. Middleware Layer

The software components considered in this layer provide the functionality required to execute the experiment and obtain the resulting data in a synchronized way. It is essential to obtain the raw data required to analyze the performance and accuracy regarding the roll-angle estimations using different types of Kalman filters.

The middleware components are organized in different devices:

1. There is an Ecosystem Bus in charge of coordinating the experiments among all the experimental kits connected to the testbed. This Ecosystem Manager is deployed in a small computer able to be located in the experimental vehicle. This component provides the functionalities to connect and disconnect the experimental kits. Even more, it is in charge of sending requests to the low-cost kits to: (a) start an experiment; (b) continue running an experiment; (c) stop an experiment; and (d) shutdown an experimental kit. Finally, the ecosystem manager sends the data to the application layer in order to proceed to its storage.
2. The low cost experimental kits have their own middleware layer composed of two components:

- (a) The Unit Bus is in charge of coordinating each experimental kits with the Ecosystem Manager that is in charge of coordinating the whole testbed. The functionalities provided by the Unit Bus are: (a) publish the experimental kit in the testbed; (b) receive the requests from the Ecosystem Bus; (c) send the data obtained by the kit sensors to the Ecosystem Bus; and (d) send to the sensors synchronizer the experiments start and stop signals for gathering appropriately the data from the sensors and roll angle estimators.
 - (b) The Sensors/Estimators Synchronizer obtains the data from the sensors and estimators with the required synchronization as it is shown in Figure 4. This component sends to the Unit Bus the data structure with the information gathered during the experiment when it receives the stop signal.
3. The middleware considered for the ground truth kit (VBOX based) consists of a software component (named VBOX Manager) that provides the functionality to manage the start/stop signals received from the Ecosystem Manager. This component also sends the data gathered from during the experiment execution. Due to the restrictions introduced by VBOX and Racelogic IMU manufacturers, the middleware component for the ground truth experimental kit is implemented in C#.

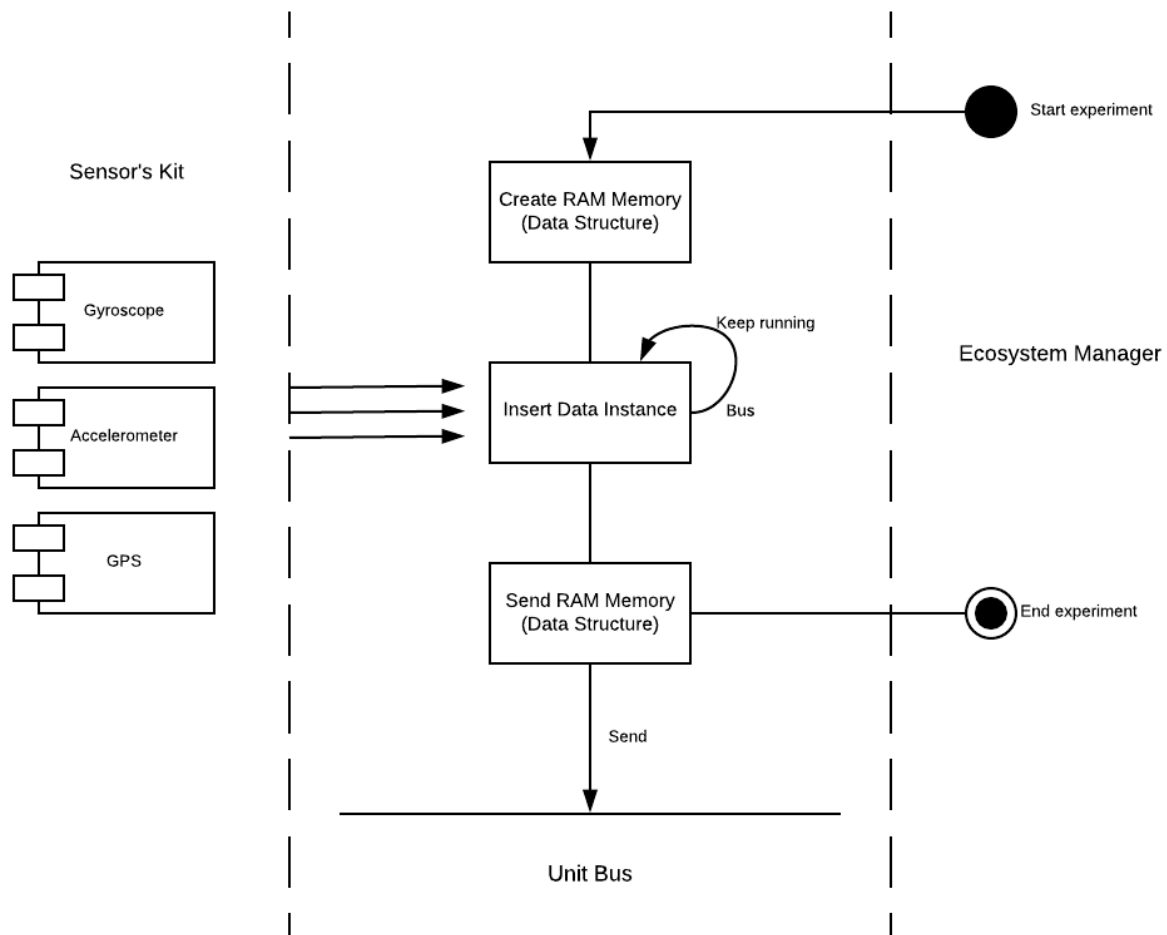


Figure 4. Sensors/Estimators Synchronizer functional diagram.

2.1.3. Communication Layer

The communications layer has the responsibility to ensure homogeneous and synchronous communication among the kits included in the experimental testbed when the experiments are executed. This layer is implemented as follows:

1. Ground truth kit communications. The connection of the sensors and the laptop where the ground truth middleware is installed are connected using a cable due to the communication interfaces provided by the Racelogic IMU and VBOX Dual Antenna. The VBOX Manager and the Ecosystem manager are deployed in the same laptop.
2. Low-cost experimental kit communications. The low-cost experimental kits (based on Intel Edison and Raspberry Pi 3) are connected to the Ecosystem Manager using a WiFi connection through a wireless (802.11 g) access point. The connection between the sensors and the Sensors/Estimators Managers is implemented using the GPIO ports provided by the Intel Edison and Raspberry Pi development boards.

At logic level, the communications between the low-cost experimental kits and the laptop where the Ecosystem Manager is deployed are managed using TCP sockets.

2.1.4. Perception Layer

This level is composed of the sensors and estimators connected to each experimental kit in the testbed:

1. The sensors considered in each experimental kit are implemented in a different way. As mentioned before the ground truth kit uses a Racelogic IMU and VBOX Dual Antenna. The drivers are provided by the manufacturers and used by the middleware element implemented in the VBOX Manager component. In the case of the low-cost experimental kits, each sensor (accelerometer and gyroscope) is managed through a driver implemented in C++ and used by the component that is deployed in the Intel Edison and Raspberry Pi development boards. These drivers gather the information from the sensors hardware using 50 Hz sampling rate.
2. The Roll Angle Estimator implements Linear and Unscented Kalman Filters to estimate roll angle in real time using a two Degree of Freedom (DoF) which represents the vehicle roll motion. A description of this model is presented in [35] and summarized in Section 2.1.5. As observation measurements required for both Kalman filters, lateral acceleration, roll rate and time are considered. Section 2.1.6 provides a more detailed description of this software component.
3. Finally, the NTP Client to assure the appropriate synchronization of the data gathered in the experimental kits included in the testbed, registers the actual date-time obtained from the GPS sensor in the hardware controller.

2.1.5. Vehicle Model

For a better understanding of this section and Section 2.1.6, Table 1 shows all the variables for the model and estimators presented in this work.

Table 1. Variables glossary.

Variable	Definition	Variable	Definition
\mathbf{x}_k	State vector	I_{xx}	Sprung mass moment of inertia
ϕ_k	Vehicle roll angle	m_s	Sprung mass
$\dot{\phi}_k$	Vehicle roll rate	h_{cr}	Sprung mass height about the roll axis
$a_{ym,k}$	Vehicle lateral acceleration	C_r	Total torsional damping
\mathbf{y}_k	Observation vector	K_r	Stiffness coefficient
k	Instant time	T_s	Sample time
g	Acceleration due to gravity	\mathbf{P}_k	Error covariance matrix
R	Covariance matrix of the measurement noise	Q	Process noise covariance matrix
K	Kalman gain matrix	H	Observation matrix

The model used in this research is presented in [35]. This model describes the roll vehicle motion in a discrete-time system (see Figure 5)

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d a_{ym,k} \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k \end{aligned} \quad (1)$$

where:

$$\mathbf{x}_k = [\phi_k \dot{\phi}_k]^T \quad (2)$$

$$\mathbf{A}_d = \begin{bmatrix} 1 + T_s & 0 \\ -\frac{K_r \cdot T_s}{I_{xx}} & 1 + \frac{C_r \cdot T_s}{I_{xx}} \end{bmatrix} \quad (3)$$

$$\mathbf{B}_d = \begin{bmatrix} 0 \\ \frac{m_s \cdot h_{cr} \cdot T_s}{I_{xx}} \end{bmatrix} \quad (4)$$

$$\mathbf{C} = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad (5)$$

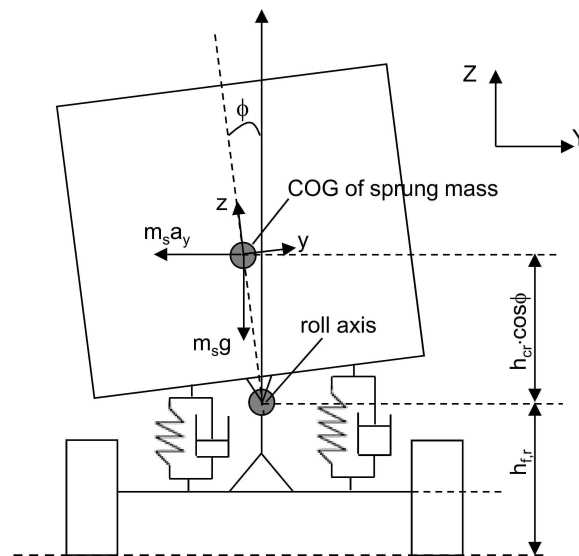


Figure 5. Vehicle roll model.

The vehicle model constant values are presented in Table 2.

Table 2. Vehicle model variables.

Variable	Definition	Value
I_{xx}	Sprung mass moment of inertia	751.16 kg m ²
m_s	Sprung mass	1700 kg
h_{cr}	Sprung mass height about the roll axis	0.25 m
C_r	Total torsional damping	3538.08 N m/rad
K_r	Stiffness coefficient	18,438.02 N/m ²

2.1.6. Roll Angle Estimators based on Kalman Filters

Two different roll angle estimators based on Kalman filters were developed to achieve the research work defined:

1. First, a linear Kalman estimator was developed. This estimator has as inputs the actual roll rate, time and lateral acceleration. As result, the software component provides an estimated value of the current roll angle. The formulas implemented for the calculations are presented below [10]:

- (a) Prediction of state:

$$\bar{\mathbf{x}}_{k|k-1} = \mathbf{A}_d \bar{\mathbf{x}}_{k-1|k-1} \quad (6)$$

- (b) Prediction of error covariance:

$$\mathbf{P}_{k|k-1} = \mathbf{A}_d \mathbf{P}_{k-1|k-1} \mathbf{A}_d^T + \mathbf{Q} \quad (7)$$

- (c) Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}^T [\mathbf{H} \mathbf{P}_{k|k-1} \mathbf{H}^T + \mathbf{R}]^{-1} \quad (8)$$

- (d) State estimation:

$$\bar{\mathbf{x}}_{k|k} = \bar{\mathbf{x}}_{k|k-1} + \mathbf{K}_k [\mathbf{y}_{measured} - \mathbf{H} \bar{\mathbf{x}}_{k|k-1}] \quad (9)$$

- (e) Error covariance estimation:

$$\mathbf{P}_{k|k} = [\mathbf{I} - \mathbf{K}_k \mathbf{H}] \mathbf{P}_{k|k-1} \quad (10)$$

In order to increase the performance required to fulfill hard real-time constraints, the software components embedded in low-cost devices have been optimized in the following way: (a) use temporary variables to store complex calculations that are used multiple times along the code without changing the values; (b) reduce the number of calculations when handling matrices by expanding and analyzing the values prone to change; and (c) optimize the memory and instantiation time by passing function arguments as reference instead of value copies.

2. Second, an Unscented Kalman estimator was designed. Similarly to the previous estimator the inputs are the actual roll rate, time and lateral acceleration. The formulas implemented for the calculations are shown below [10]:

- (a) Calculate weights:

$$\begin{aligned} W_0^m &= k / (n + k) \\ W_i^m &= 1 / 2(n + k); \quad i = 1, \dots, 2n \\ W_i^c &= k / (n + k) + (1 - \alpha^2 + \beta) \\ W_i^c &= 1 / 2(n + k); \quad i = 1, \dots, 2n \end{aligned} \quad (11)$$

where α is the distribution of the sampling points around the state mean, $\bar{\mathbf{x}}$, β is used to incorporate prior knowledge of the distribution of $\bar{\mathbf{x}}$, n is the dimension of $\bar{\mathbf{x}}$, and k is a scaling parameter:

$$k = \alpha^2(n + \epsilon) - n$$

ϵ is usually set to 0.

- (b) Calculate sigma points:

$$\bar{\mathbf{x}}_{k|k}^a = [\bar{\mathbf{x}}_{k|k} \quad \mathbf{0}] \quad (12)$$

$$\mathbf{P}_{k|k}^a = \begin{bmatrix} \mathbf{P}_{k|k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix} \quad (13)$$

$$\mathbf{X}_k^a = [\bar{\mathbf{x}}_k^a \quad \bar{\mathbf{x}}_k^a + (\sqrt{(n+k)\mathbf{P}_{xx}^a}) \quad \bar{\mathbf{x}}_k^a - (\sqrt{(n+k)\mathbf{P}_{xx}^a})] \quad (14)$$

- (c) Prediction of state:

$$\mathbf{X}_{i,k+1|k} = f(\mathbf{X}_{i,k|k}^a) \quad (15)$$

$$\tilde{x}_{k+1|k} = \sum_{i=0}^{2n} W_i^m \cdot X_{i,k|k} \quad (16)$$

(d) Prediction of error covariance:

$$\mathbf{P}_{k+1|k} = \sum_{i=0}^{2n} W_i^c \cdot (X_{i,k+1|k} - \tilde{x}_{k+1|k}) \cdot (X_{i,k+1|k} - \tilde{x}_{k+1|k})^T \quad (17)$$

(e) Prediction of observations:

$$\mathbf{Y}_{i,k+1|k} = h(X_{i,k|k}^a) \quad (18)$$

$$\tilde{y}_{k+1|k} = \sum_{i=0}^{2n} W_i^m \cdot \mathbf{Y}_{i,k+1|k} \quad (19)$$

(f) Innovation covariance:

$$\mathbf{P}_{YY,k+1|k} = R + \sum_{i=0}^{2n} W_i^c \cdot (\mathbf{Y}_{i,k+1|k} - \tilde{y}_{k+1|k}) \cdot (\mathbf{Y}_{i,k+1|k} - \tilde{y}_{k+1|k})^T \quad (20)$$

(g) Cross correlation matrix:

$$\mathbf{P}_{XY,k+1|k} = R + \sum_{i=0}^{2n} W_i^c \cdot (X_{i,k+1|k} - \tilde{x}_{k+1|k}) \cdot (\mathbf{Y}_{i,k+1|k} - \tilde{y}_{k+1|k})^T \quad (21)$$

(h) Kalman gain:

$$\mathbf{K}_{k+1} = \mathbf{P}_{XY,k+1|k} \mathbf{P}_{YY,k+1|k}^{-1} \quad (22)$$

(i) Error covariance estimation:

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1} \mathbf{P}_{XY,k+1|k} \mathbf{K}_{k+1}^T \quad (23)$$

(j) State estimation:

$$\tilde{x}_{k+1|k+1} = \tilde{x}_{k+1|k} + \mathbf{K}_{k+1} (\mathbf{y}_{meas} - \tilde{y}_{k+1|k}) \quad (24)$$

In this case, the Cholesky transform was implemented in a separate component due to enhance the reuse of this calculation in further research works. This estimated includes the same optimizations as the previous case, being more relevant due to the increased complexity of the calculations stated for Unscented Kalman Filters.

In this case, the values of Q and R are:

$$\mathbf{Q} = \begin{bmatrix} 5 & 0 \\ 0 & 10 \end{bmatrix} \quad (25)$$

$$\mathbf{R} = 0.01 \quad (26)$$

2.2. Experiments Specification

In order to evaluate the accuracy and performance of roll angle estimations provided by low-cost experimental kits, the following hypothesis were defined:

- H1: The roll angle estimation based on linear and unscented Kalman filters are similar than the actual roll angle values directly measured from the ground kit.

- H2: The performance of the roll angle estimation based on linear and unscented Kalman filters fulfills the constraints of hard real time processing providing, at least, results at a sampling rate of 50 Hz.

It was necessary to consider maneuvers such as regular driving situations, J-Turn and lane change to evaluate properly the previously stated hypothesis. Figure 6 summarizes the experiments defined.

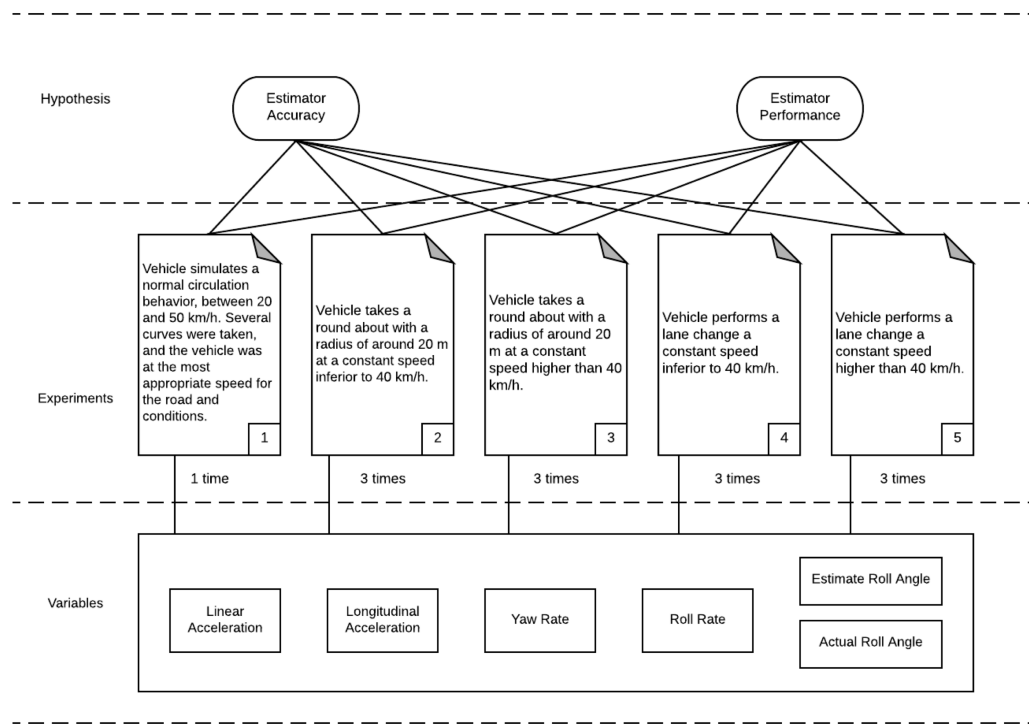


Figure 6. Defined experiments.

2.3. Data Processing

As defined in the experimental testbed, when the Stop experiment signal is invoked, the Ecosystem Manager stores the information gathered by each kit in a CSV file having a name that includes the execution date and time and experimental kit identifier. The information included in each file depends on the considered kit.

The ground truth provides the GPS coordinates where the measure is obtained, the measure time stamp, roll and yaw rate, longitudinal and lateral acceleration and the actual roll angle obtained from the VBOX dual antenna.

The low-cost experimental kits provide exactly the same value but the roll angle value corresponds to the data generated by the estimators based on linear and unscented Kalman filters.

The actual roll angle obtained from the ground truth kit was compared with the values calculated by the roll angle estimation based on linear and unscented Kalman filters to determine the accuracy of the values obtained from the estimators. The results obtained during the experiments execution are presented in Section 3.

2.4. Threats to Validity

Several threats must be considered in order to determine the validity of the results obtained in this research work.

Regarding the experiments definition, it is necessary to consider all the issues preventing the experiments replication and the results generalization. The considered issues were related to:

1. The road conditions. The road considered in this research work has not slope or gradient. Nevertheless the maneuvers were repeated in different directions at the speed specified for each experiment.
2. The vehicle conditions. The threat in this category is related to the equipment conditions. Considering the recommendations provided in [14], the Racelogic IMU and the low-cost sensors were located in the vehicle's center of mass.
3. The type of sensors and controllers considered. The threat in this category is related to the representativeness of the sensors embedded in the low-cost experimental kits. Nevertheless, it is important to remark that all the sensors considered are available on the market and they have an average price and quality. It can be foreseen that the sensors performance will improve in the coming years, so the results from the experiments execution can be better in the short term.

Regarding the experiments implementation, it is necessary to consider the relevant issues to prevent the errors introduction due to incorrect experiments execution. The considered issues in this area were:

1. The lack of precision in the measures obtained from the low-cost kits. As indicated in [14], the low-cost experimental kits provides data with the required precision. This precision is obtained when the corresponding calibrations are carried out in static conditions. Even more, to prevent errors from the specific sensors, two different units of each low-cost experimental kit type (Raspberry Pi and Intel Edison) were used.
2. The possible errors introduced by the sensor drivers and the middleware components considered in the IoT architecture designed to implement the experimental testbed. This threat was mitigated designing and implementing an automated unit testing plan to assure before the experiments execution, that these software components are free from critical bugs. The testing plan is automated executed before the deployment of the software components in the controller hardware in each low-cost experimental kit.
3. The possible errors in the execution of the maneuvers considered in each experiment. The mitigation of this threat consisted on the repetition of each experiment. Each maneuver was repeated, at least, three times.

3. Results

To perform the experimental validation corresponding to this research work, a Mercedes Sprinter van equipped with the low-cost devices and the Racelogic VBOX, as it is described in Section 2, was used. To evaluate both accuracy and processing time for the analyzed low-cost devices, two maneuvers were carried out, J-Turn and lane change. Also, a third test type was conducted in order to verify the devices behavior under normal circulation situations.

For the estimators presented in Section 2, the initial values for the state vector and the error covariance matrix are:

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (27)$$

$$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (28)$$

3.1. J-Turn

The first maneuver is a J-Turn, this maneuver is performed in a 22 m diameter roundabout at a speed close to 40 km/h (see Figure 7). Figures 8 and 9 show the roll angle estimated by the Raspberry Pi 3 Model B (Green), Intel Edison (Pink) and with the information provided by the VBOX IMU (Blue). The roll angle measured with the VBOX GPS dual antenna (Red) has been used as ground truth in

order to verify the accuracy of the devices and verify that the estimation is not affected by the low-cost components. In the three cases, estimations are very similar and the results show that the error comes principally from the estimator and not from the accuracy of the low-cost sensors. Also, in all the cases, the results show that there is a road bank in the path corresponding to the maneuver.

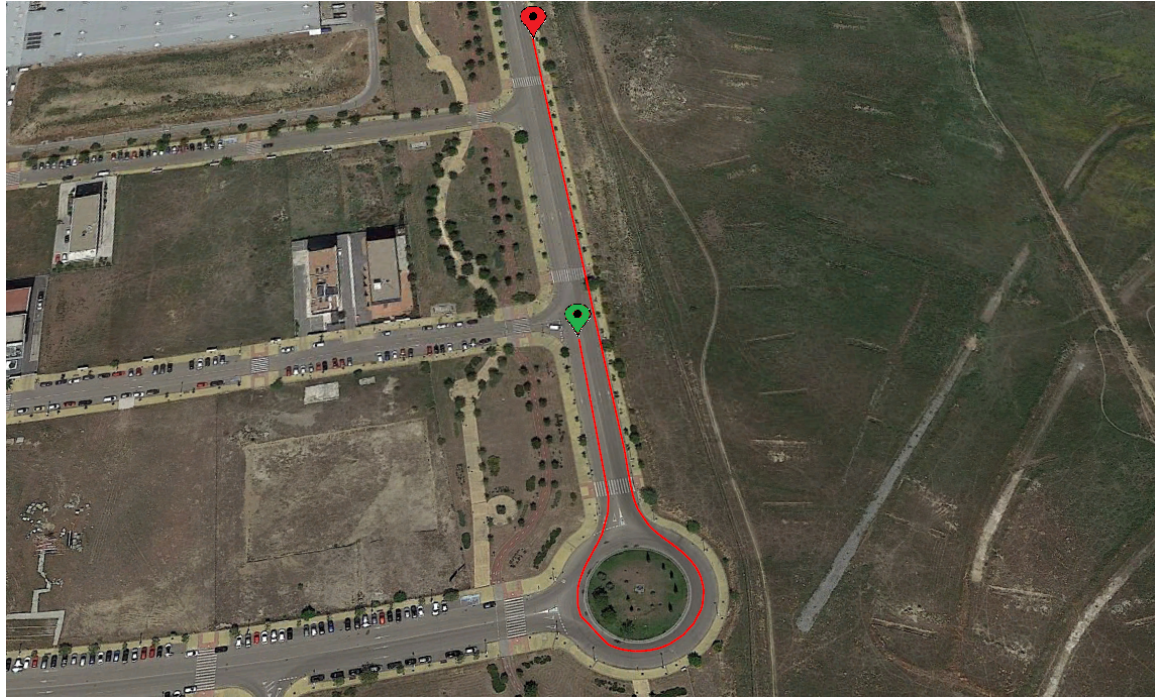


Figure 7. Test 1: J-turn trajectory (Map scale 1:2100 cm).

The root mean square (RMS), the norm and maximum errors have been calculated in order to quantify the accuracy of the estimation. The norm error has been calculated as follows [7]:

$$E_t = \frac{\varepsilon_t}{\sigma_t} \times 100, \quad (29)$$

where

$$\begin{aligned} \varepsilon_t^2 &= \int_0^T (\phi_{GT} - \phi_{lc})^2 dt \\ \sigma_t^2 &= \int_0^T (\phi_{GT} - \mu_{GT})^2 dt \end{aligned} \quad (30)$$

ϕ_{GT} represents the ground truth data, ϕ_{lc} represents the low-cost sensor data and μ_{GT} is the mean value of the ground truth data obtained during the period T.

Table 3 collects the previous values. The results show that errors are very similar in both devices and the estimated roll angle using VBOX IMU data. The norm error from Intel Edison is higher than the Raspberry Pi and the VBOX IMU (about 7%) but for the maximum error the data from Raspberry Pi is higher than the other two values (about 12°). With UKF the errors are smaller in the three devices (about 7% for norm error, 0.2° for RMS error and 0.3° for maximum error).

Table 3. Test 1: Errors of estimated roll angle using KF and UKF on Raspberry Pi and Intel Edison compared with the IMU from VBOX (ground truth).

	Roll Angle					
	Norm Error (%)		RMS Error (°)		Maximum Error (°)	
	KF	UKF	KF	UKF	KF	UKF
Raspberry Pi 3 Model B	43.38	42.29	2.41 ± 0.35	2.35 ± 0.04	19.32	19.06
Intel Edison	54.97	51.18	2.54 ± 0.21	5.37 ± 0.07	8.94	6.87
Racelogic VBOX IMU	48.89	45.51	2.33 ± 0.21	2.17 ± 0.32	6.38	6.66

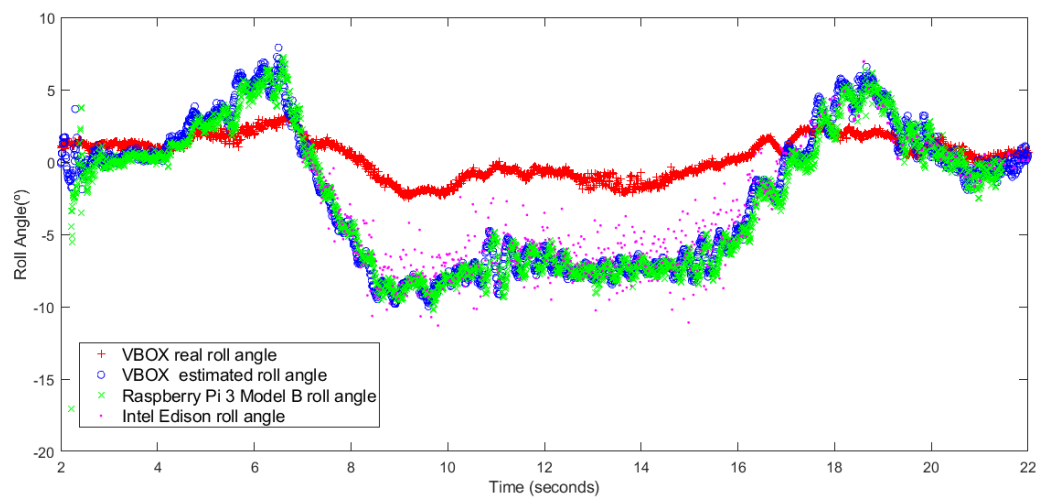


Figure 8. Test 1 (KF): Roll angle obtained from the dual antenna of VBOX (**red points**), estimated with Raspberry pi (**green points**), with Intel Edison (**pink points**) and with the IMU of VBOX (**blue points**).

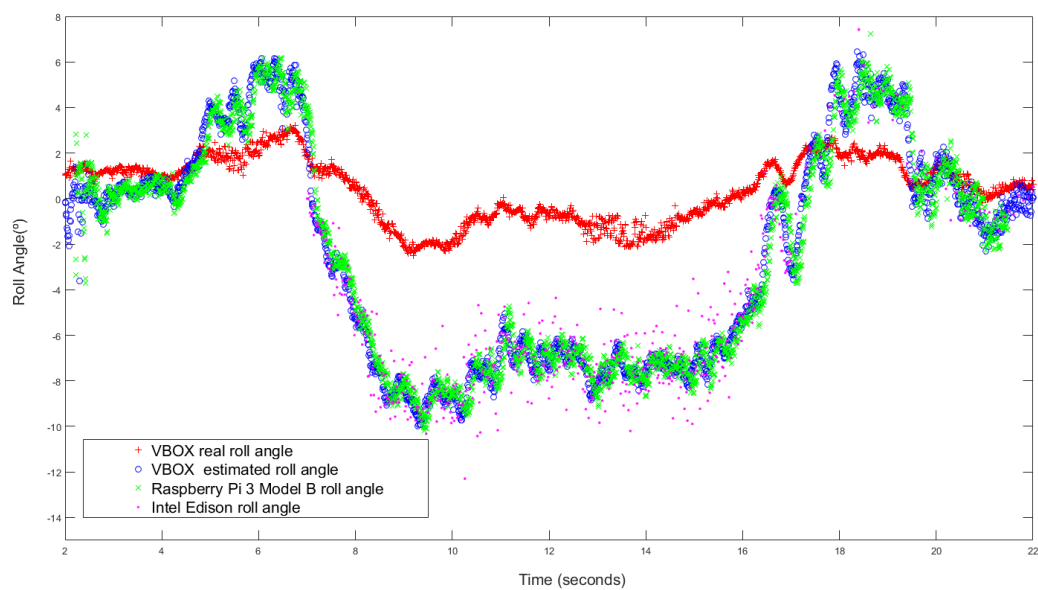


Figure 9. Test 1 (UKF): Roll angle obtained from the dual antenna of VBOX (**red points**), estimated with Raspberry pi (**green points**), with Intel Edison (**pink points**) and with the IMU of VBOX (**blue points**).

These two estimators must work under hard real time constraints. For the given case, 20 ms is the maximum processing time permitted to transform the inputs and to apply the estimator. This sampling rate (50 Hz) is forced by the low-cost sensors.

Table 4 contains both processing time for Raspberry Pi 3 Model B and Intel Edison when computing KF and UKF. The mean and maximum processing time are calculated in order to quantify both devices performance. The devices stability has been determined with the processing time mean deviation. Results show that processing times for Intel Edison are higher than the Raspberry Pi 3 model B ones. Concerning to estimators, the mean processing time is higher in the case of UKF than KF (27×10^{-6} s for Raspberry Pi 3 Model B and 46×10^{-6} s for Intel Edison). The difference between both devices are 3.5×10^{-6} s with KF and 23×10^{-6} s with UKF for the mean processing time, and 0.3×10^{-3} s with KF and 0.6×10^{-3} s with UKF for the maximum processing time. Concerning mean deviation, the difference is about 0.7×10^{-6} s with KF and 9×10^{-6} s with UKF.

Table 4. Test 1: Processing time using KF and UKF on Raspberry Pi and Intel Edison.

	Processing Time					
	Maximum (s)		Mean (s)		Mean Deviation (s)	
	KF	UKF	KF	UKF	KF	UKF
Raspberry Pi 3 Model B	0.29×10^{-3}	0.77×10^{-3}	1.82×10^{-6}	28.64×10^{-6}	0.96×10^{-6}	11.25×10^{-6}
Intel Edison	6×10^{-6}	100×10^{-6}	5.16×10^{-6}	51.83×10^{-6}	0.27×10^{-6}	2.55×10^{-6}

3.2. Lane Change

The second maneuver is a lane change, this was performed at a speed close to 50 km/h (see Figure 10). Figures 11 and 12 show the roll angle estimated by the Raspberry Pi 3 Model B (green), Intel Edison (pink) and with the information provided by the VBOX IMU (blue) using a KF and in an UKF respectively. As in the previous maneuver, the roll angle measured with the VBOX GPS dual antenna (red) has been used as ground truth in order to verify the devices accuracy and to assess that the resulting estimation is not affected by the low-cost sensors and devices. In the three cases, the estimation is very similar and the results show that the error comes mainly from the estimator itself and not from the accuracy of the low-cost sensors. Another source of this error may be due to irregularities of the road as it happens during the time period between 15 s and 20 s.

The root mean square (RMS), the norm and maximum errors have been calculated in order to quantify the accuracy of the estimation. Table 5 shows the results. Errors are higher for Raspberry Pi 3 Model B than for Intel Edison. Concerning the norm and RMS error, the difference is about 15% and 0.15° for Kalman Filter and 10% and 0.11° for UKF. For maximum error the difference is about 0.1° for KF and 2.2° for UKF. The results for the estimated roll angle using VBOX IMU data are very similar, and in some cases, they are higher than Intel Edison ones.

Table 5. Test 2: Errors of estimated roll angle using KF and UKF on Raspberry Pi and Intel Edison compared with the IMU from VBOX (ground truth).

	Roll Angle					
	Norm Error (%)		RMS Error ($^\circ$)		Maximum Error ($^\circ$)	
	KF	UKF	KF	UKF	KF	UKF
Raspberry Pi 3 Model B	120.1	100.2	0.76 ± 0.035	0.63 ± 0.058	3.21	2.76
Intel Edison	105.4	89.9	0.61 ± 0.051	0.52 ± 0.078	3.11	4.78
Racelogic VBOX IMU	106.1	104.23	0.66 ± 0.012	0.65 ± 0.05	2.3	3.38

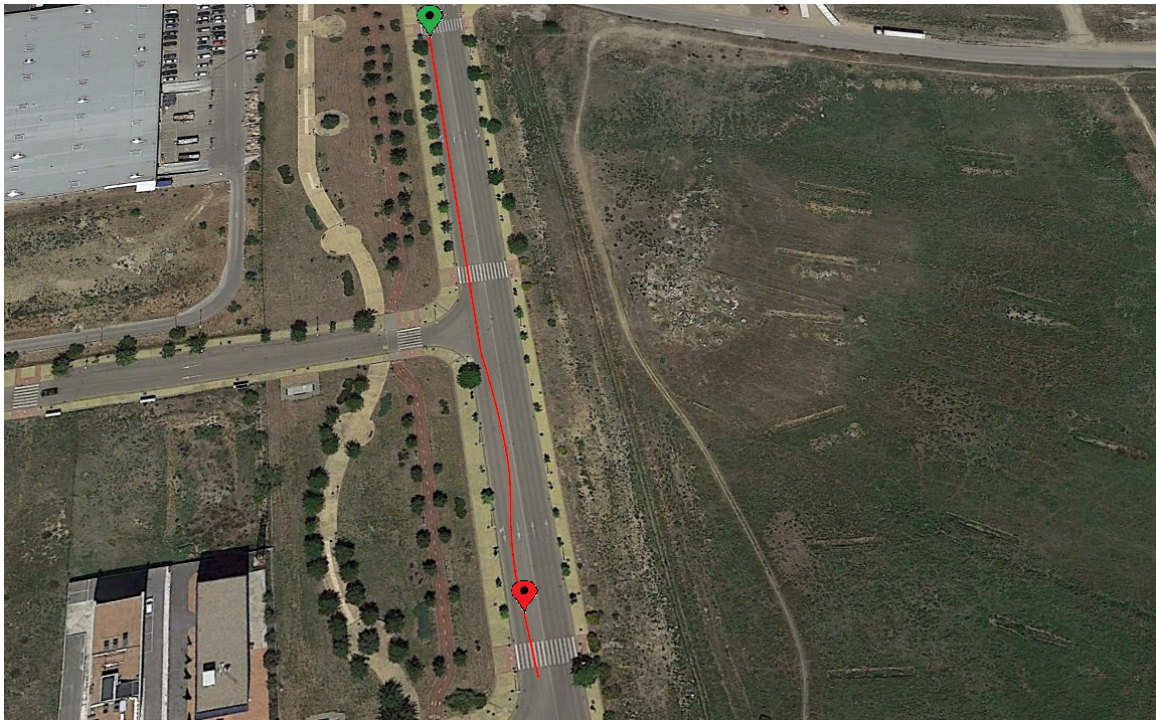


Figure 10. Test 2: Line change trajectory (Map scale 1:2100 cm).

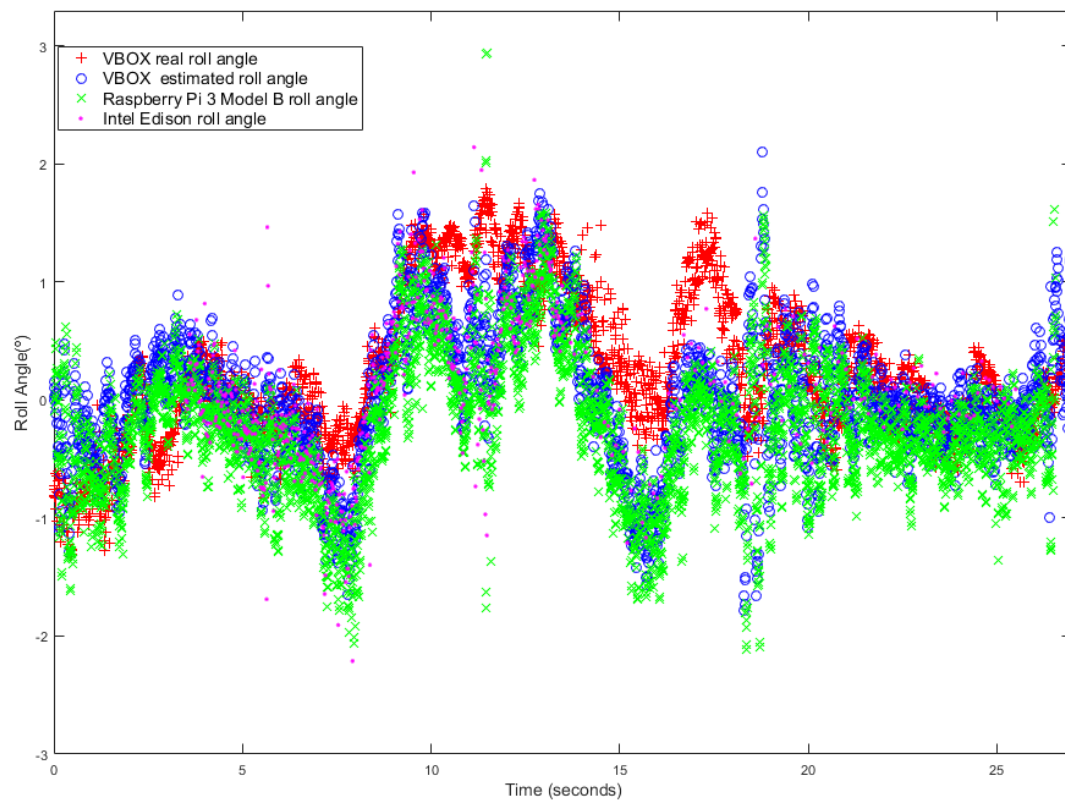


Figure 11. Test 2 (KF): Roll angle obtained from the dual antenna of VBOX (red points), estimated with Raspberry pi (green points), with Intel Edison (pink points) and with the IMU of VBOX (blue points).

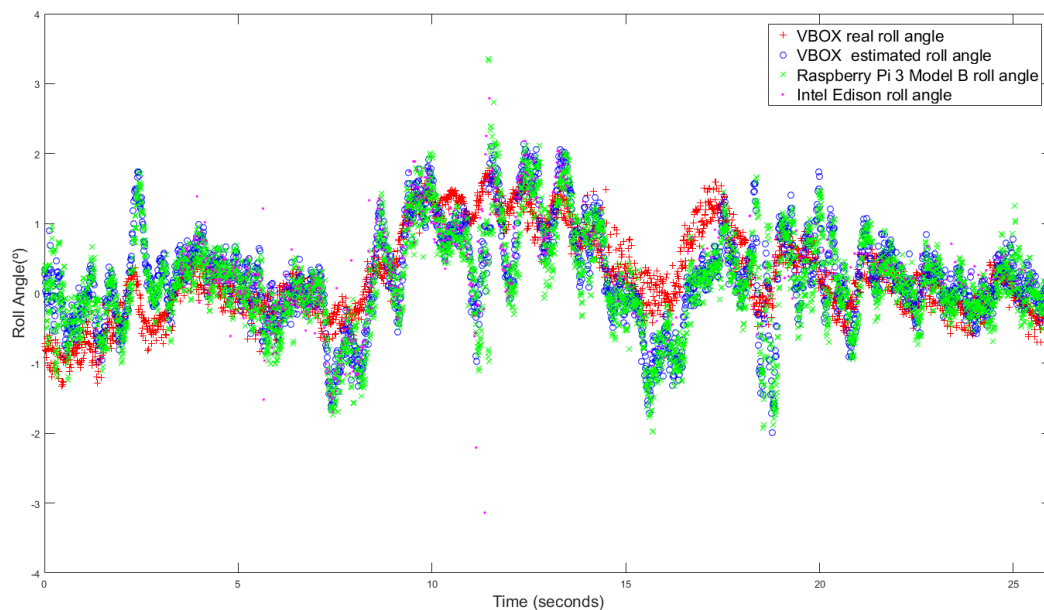


Figure 12. Test 2 (UKF): Roll angle obtained from the dual antenna of VBOX (red points), estimated with Raspberry pi (green points), with Intel Edison (pink points) and with the IMU of VBOX (blue points).

Similarly to the previous maneuver, Table 6 shows both processing time for Raspberry Pi 3 Model B and Intel Edison when computing KF and UKF. The mean and maximum processing time are calculated in order to quantify the performance of both devices. The stability of the devices has been calculated through the mean deviation of the processing time. Results show that processing times for Intel Edison are lower than the Raspberry Pi 3 model B ones. Concerning to estimators, the mean processing time is higher for UKF than for KF (30×10^{-6} s for Raspberry Pi 3 Model B and 45×10^{-6} s for Intel Edison). The difference between both devices are 1.25×10^{-6} s with KF and 11×10^{-6} s with UKF for the mean processing time and 14×10^{-3} s with KF and 6.5×10^{-3} s with UKF for the maximum processing time. Concerning mean deviation, the difference is about 10×10^{-6} s with KF and 24×10^{-6} s with UKF.

Table 6. Test 2: Processing time using KF and UKF on Raspberry Pi and Intel Edison.

	Processing Time					
	Maximum (s)		Mean (s)		Mean Deviation (s)	
	KF	UKF	KF	UKF	KF	UKF
Raspberry Pi 3 Model B	14.21×10^{-3}	6.76×10^{-3}	6.93×10^{-6}	374.77×10^{-6}	10.54×10^{-6}	26.93×10^{-6}
Intel Edison	88×10^{-6}	120×10^{-6}	5.25×10^{-6}	51.93×10^{-6}	0.41×10^{-6}	2.65×10^{-6}

3.3. Standard Circulation

The last test is performed taking low and medium velocity with typical maneuvers and smooth movements. This test is carried out under usual circulation conditions. Figures 13 and 14 show the roll angle estimated by the Raspberry Pi 3 Model B (green), Intel Edison (pink) and with the information provided by the VBOX IMU (blue), by using a KF and an UKF respectively. The roll angle measured with the VBOX GPS dual antenna (Red) has been used as ground truth in order to verify the accuracy of the devices and to verify that the resulting estimation is not affected by the low-cost sensors nor devices. In all the three cases, the error is very similar and higher than in previous tests. These devices are very sensitive to the noise [14] and this test type is prone to noise.

The root mean square (RMS), the norm and maximum errors have been calculated in order to quantify the accuracy of the estimation. In Table 7 the results are given, the errors are very similar except

for the maximum error, where the Intel Edison has a 55.81° of error for KF and 43.68° for UKF, this error is due to the noise that causes atypical data. Concerning the norm and RMS error, the difference is about 0.06% and 0.12° for Kalman Filter and 4% and 0.15° better for UKF for Intel Edison and for Raspberry Pi the difference is about 5% and 0.17° for Kalman Filter and 7% and 0.18° better for UKF. The results of the estimated roll angle using VBOX IMU data are very similar, and in some test are higher than the both Raspberry Pi and Intel Edison.

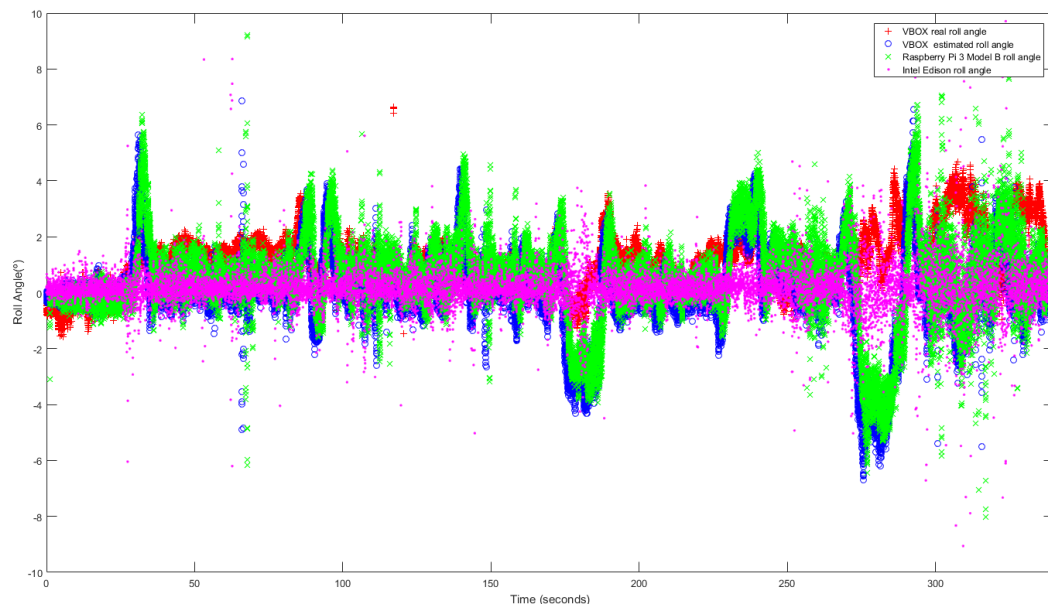


Figure 13. Test 3 (KF): Roll angle obtained from the dual antenna of VBOX (red points), estimated with Raspberry pi (green points), with Intel Edison (pink points) and with the IMU of VBOX (blue points).

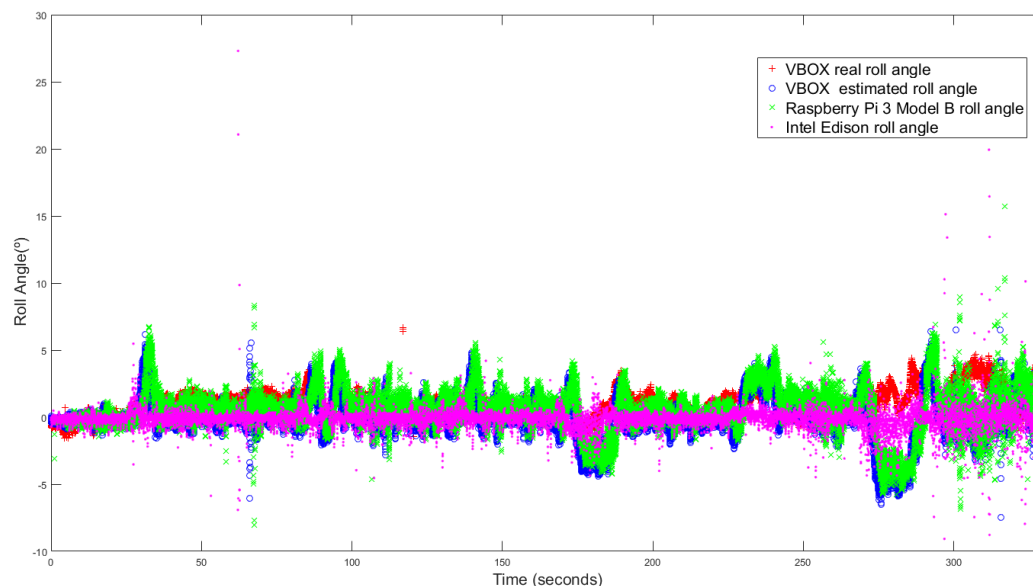


Figure 14. Test 3 (UKF): Roll angle obtained from the dual antenna of VBOX (red points), estimated with Raspberry pi (green points), with Intel Edison (pink points) and with the IMU of VBOX (blue points).

Table 7. Test 3: Errors of estimated roll angle using KF and UKF on Raspberry Pi and Intel Edison compared with the IMU from VBOX (ground truth).

	Roll Angle					
	Norm Error (%)		RMS Error (°)		Maximum Error (°)	
	KF	UKF	KF	UKF	KF	UKF
Raspberry Pi 3 Model B	193.86	186.67	1.86	1.79	15.67	14.33
Intel Edison	198.97	189.46	1.91	1.82	55.81	43.68
Racelogic VBOX IMU	198.91	193.34	2.03	1.97	9.9	8.91

Similarly to the previous maneuver, Table 8 shows both processing time for Raspberry Pi 3 Model B and Intel Edison when computing KF and UKF. The mean and maximum processing time are calculated in order to quantify the performance of both devices. The stability of the devices has been calculated through the mean deviation of the processing time. Results show that processing times for Intel Edison are lower than the Raspberry Pi 3 model B ones. Concerning to estimators, the mean processing time is higher for UKF than for KF (26×10^{-6} s for Raspberry Pi 3 Model B and 47×10^{-6} s for Intel Edison). The difference between both devices are 3.2×10^{-6} s with KF and 24×10^{-6} s with UKF for the mean processing time and 6×10^{-3} s with KF and 7.7×10^{-3} s with UKF for the maximum processing time. Concerning mean deviation, the difference is about 1×10^{-6} s with KF and 9×10^{-6} s with UKF.

Table 8. Test 3: Processing time using KF and UKF on Raspberry Pi and Intel Edison.

	Processing Time					
	Maximum (s)		Mean (s)		Mean Deviation (s)	
	KF	UKF	KF	UKF	KF	UKF
Raspberry Pi 3 Model B	6.41×10^{-3}	8.436×10^{-3}	2.05×10^{-6}	28.49×10^{-6}	1.33×10^{-6}	12.27×10^{-6}
Intel Edison	0.054×10^{-6}	0.71×10^{-6}	5.21×10^{-6}	52.10×10^{-6}	0.35×10^{-6}	3.11×10^{-6}

4. Discussion

The results discussion is focused on the performance of the low-cost devices and the accuracy of the estimations obtained.

4.1. Accuracy

As expected, the roll angle estimations provided by the Unscented Kalman are better than the ones obtained using Linear Kalman. The RMS error for all the tests obtained by both estimators in comparison with the ground truth values are between 0.1 and 2.2 degrees. So, it can be concluded that the estimations are accurate enough for the expected use in RSC.

Nevertheless, it is necessary to remark that the results calculated by these estimators are less precise than the provided by Neural Network based estimators. However, Kalman based estimators do not need a previous training and are tight to an specific model allowing its direct usage in a variety of vehicles.

Also, it can be indicated that Kalman Filters based estimators are less sensitive to noise than Neural Networks based ones.

Finally, as future work, it is planned to create more complex estimators based on a combination of Neural Networks and Kalman Filters because the performance of software components implementing these techniques in low-cost devices seems to be high enough to support both of them respecting hard real time constraints.

4.2. Processing Capability

The temporal performance and real time constraints are main aspects to consider in order to integrate estimators and controllers in embedded low-cost devices. The results show that the processing time to get the data, execute its normalization, perform the roll angle estimation via ANN and the denormalization of the outcome, is four orders of magnitude lower than the required sample rate threshold of 20 ms.

The average processing times are 5.16×10^{-6} s (KF) and 51.83×10^{-6} s (UKF) for Intel Edison and 1.82×10^{-6} s (KF) and 28.64×10^{-6} s (UKF) for Raspberry Pi 3 Model B.

This performance will permit the integration of Neural Networks and Kalman filters to reduce the noise present in data gathered from low cost sensors.

Several optimizations were carried out to enhance this performance, essentially:

- Usage of standard libraries from C++, which allow a straightforward compilation in almost any development board.
- Reduction of the number of operations when handling matrices by expanding and analyzing the specific resulting values prone to change (algorithmic optimization).
- Optimization of memory usage and instantiation time by passing function arguments as reference instead of value copies, and by multiple revisions of source code to keep it clean and simple.

5. Conclusions

As results presented in Section 3 indicate, low-cost experimental kits based on IoT architectures and including estimators based on Kalman filters provide accurate roll angle estimation. Moreover, an effective design and development of software components applying these techniques is essential to fulfill hard real time restrictions when embedding this components in low cost devices such as Raspberry Pi 3 Model B and Intel Edison to estimate roll angle. Scalable, versatile, maintainable and efficient IoT architectures for commercial vehicles can be implemented using the software design and results obtained in this research work. As stated before, the performance levels demonstrated in this research work envisage that more complex estimators based on the combination of Kalman filters, Neural Networks and other techniques such as deep learning algorithms can provide better results.

Author Contributions: J.G.G., L.P.G., J.P.R., S.S.S. and B.L.B. proposed ideas; L.P.G., J.G.G. designed the IOT architecture, communications and software components; L.P.G., J.P.R. and B.L.B. developed the hardware setup; J.G.G., L.P.G., J.P.R., S.S.S. and B.L.B. conceived and designed the tests; J.G.G., J.P.R., S.S.S. and B.L.B. analyzed the data; and J.G.G., L.P.G., J.P.R. and B.L.B. wrote the paper.

Funding: This research was funded by Spanish Government through the project TRA2013-48030-C2-1-R.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GPS	Global Positioning System
IMU	Inertial Measurement Unit
IoT	Internet Of Things
RMS	Root Mean Square

References

1. Boada, M.J.L.; Boada, B.L.; AMunoz, A.; Díaz, V. Integrated control of front-wheel steering and front braking forces on the basis of fuzzy logic. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2006**, *220*, 253–267. [[CrossRef](#)]
2. Riofrio, A.; Sanz, S.; Boada, M.J.L.; Boada, B.L. A LQR-Based Controller with Estimation of Road Bank for Improving Vehicle Lateral and Rollover Stability via Active Suspension. *Sensors* **2017**, *17*, 2318. [[CrossRef](#)] [[PubMed](#)]

3. Vargas-Melendez, L.; Boada, B.L.; Boada, M.J.L.; Gauchía, A.; Díaz, V. A Sensor Fusion Method Based on an Integrated Neural Network and Kalman Filter for Vehicle Roll Angle Estimation. *Sensors* **2016**, *16*, 1400. [[CrossRef](#)] [[PubMed](#)]
4. Guo, J.; Luo, Y.; Li, K.; Dai, Y. Coordinated path-following and direct yaw-moment control of autonomous electric vehicles with sideslip angle estimation. *Mech. Syst. Signal Proc.* **2018**, *105*, 183–199. [[CrossRef](#)]
5. Strano, S.; Terzo, M. Vehicle sideslip angle estimation via a Riccati equation based nonlinear filter. *Meccanica* **2017**, *52*, 3513–3529. [[CrossRef](#)]
6. Zhang, C.; Chen, Q.; Qiu, J. Robust H_∞ filtering for vehicle sideslip angle estimation with sampled-data measurements. *Trans. Inst. Meas. Control* **2017**, *39*, 1059–1070. [[CrossRef](#)]
7. Boada, B.L.; Boada, M.J.L.; Vargas-Melendez, L.; Diaz, V. A robust observer based on H_∞ filtering with parameter uncertainties combined with Neural Networks for estimation of vehicle roll angle. *Mech. Syst. Signal Proc.* **2018**, *99*, 611–623. [[CrossRef](#)]
8. Jo, K.; Chu, K.; Sunwoo, M. Interacting Multiple Model Filter-Based Sensor Fusion of GPS With In-Vehicle Sensors for Real-Time Vehicle Positioning. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 329–343. [[CrossRef](#)]
9. Jiang, G.; Liu, L.; Guo, C.; Chen, J.; Muhammad, F.; Miao, X. A novel fusion algorithm for estimation of the side-slip angle and the roll angle of a vehicle with optimized key parameters. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2017**, *231*, 161–174. [[CrossRef](#)]
10. Boada, B.L.; Boada, M.J.L.; Diaz, V. Vehicle sideslip angle measurement based on sensor data fusion using an integrated ANFIS and an Unscented Kalman Filter algorithm. *Mech. Syst. Signal Proc.* **2016**, *72–73*, 832–845. [[CrossRef](#)]
11. Zhao, L.; Liu, Z. Vehicle Velocity and Roll Angle Estimation with Road and Friction Adaptation for Four-Wheel Independent Drive Electric Vehicle. *Math. Prob. Eng.* **2014**, *2015*, 11. [[CrossRef](#)]
12. Zhang, H.; Huang, X.; Wang, J.; Karimi, H.R. Robust energy-to-peak sideslip angle estimation with applications to ground vehicles. *Mechatronics* **2015**, *30*, 338–347. [[CrossRef](#)]
13. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [[CrossRef](#)]
14. Pajares Redondo, J.; Prieto González, L.; García Guzman, J.; L Boada, B.; Díaz, V. VEHIOt: Design and Evaluation of an IoT Architecture Based on Low-Cost Devices to Be Embedded in Production Vehicles. *Sensors* **2018**, *18*, 486. [[CrossRef](#)] [[PubMed](#)]
15. Sangiovanni-Vincentelli, A.; Di Natale, M. Di Natale Embedded System Design for Automotive Applications. *Computer* **2007**, *40*, 42–51. [[CrossRef](#)]
16. Chakraborty, S.; Lukaszewicz, M.; Buckl, C.; Fahmy, S.; Chang, N.; Park, S.; Adlkofer, H. Embedded systems and software challenges in electric vehicles. *Des. Autom. Test Eur. Conf. Exhib.* **2011**, 242–429. [[CrossRef](#)]
17. Sethi, P.; Sarangi, S.R. Internet of things: Architectures, protocols, and applications. *J. Elect. Comput. Eng.* **2017**, *2017*. [[CrossRef](#)]
18. Pieri, F.; Zambelli, C.; Nannini, A.; Olivo, P.; Saponara, S. Limits of sensing and storage electronic components for high-reliable and safety-critical automotive applications. *Int. Conf. Elect. Electron. Technol. Autom.* **2017**. [[CrossRef](#)]
19. Sheng, Z.; Yang, S.; Yu, Y.; Vasilakos, A.; Mccann, J.; Leung, K. A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. *IEEE Wirel. Commun.* **2013**, *20*, 91–98. [[CrossRef](#)]
20. Razzaque, M.A.; Milojevic-Jevric, M.; Palade, A.; Clarke, S. Middleware for internet of things: A survey. *IEEE Int. Things J.* **2016**, *3*, 70–95. [[CrossRef](#)]
21. Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. *IEEE Int. Things J.* **2016**, *3*, 854–864. [[CrossRef](#)]
22. He, W.; Yan, G.; Li, D. Developing vehicular data cloud services in the IoT environment. *IEEE Trans. Ind. Inf.* **2014**, *10*, 1587–1595. [[CrossRef](#)]
23. Hermans, T.; Ramaekers, P.; Denil, J.; De Meulenaere, P.; Anthonis, J. Incorporation of AUTOSAR in an Embedded Systems Development Process: A Case Study. In Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications, Oulu, Finland, 30 August–2 September 2011; pp. 247–250.
24. Ambroz, M. Raspberry Pi as a low-cost data acquisition system for human powered vehicles. *Measurement* **2017**, *100*, 7–18. [[CrossRef](#)]
25. Umakirthika, D.; Pushparani, P.; Rajkumar, M.V. Internet of Things in Vehicle Safety-Obstacle Detection and Alert System. *Int. J. Eng. Comput. Sci.* **2018**, *7*, 23540–23551. [[CrossRef](#)]

26. Intel®Edison Compute Module IoT. Available online: <https://ark.intel.com/products/84572/Intel-Edison-Compute-Module-IoT> (accessed on 31 May 2018).
27. SparkFun Block for Intel®Edison-9 Degrees of Freedom. Available online: <https://www.sparkfun.com/products/13033> (accessed on 31 May 2018).
28. SparkFun 9DOF Block for Edison CPP Library. Available online: https://github.com/sparkfun/SparkFun_9DOF_Block_for_Edison_CPP_Library (accessed on 31 May 2018).
29. Raspberry, P. Available online: https://github.com/adafruit/Adafruit_Python_BNO055 (accessed on 31 May 2018).
30. Nam, K.; Oh, S.; Fujimoto, H.; Hori, Y. Estimation of sideslip and roll angles of electric vehicles using lateral tire force sensors through RLS and Kalman filter approaches. *IEEE Trans. Ind. Electron.* **2013**, *60*, 988–1000. [CrossRef]
31. Chun, D.; Stol, K. Vehicle Motion Estimation Using Low-Cost Optical Flow and Sensor Fusion. In Proceedings of the 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Auckland, New Zealand, 28–30 November 2012; pp. 507–512.
32. Zhang, S.; Yu, S.; Liu, C.; Yuan, X.; Liu, S. A dual-linear kalman filter for real-time orientation determination system using low-cost MEMS sensors. *Sensors* **2016**, *16*, 264. [CrossRef] [PubMed]
33. Wan, E.A.; Van Der Merwe, R. The Unscented Kalman Filter for Nonlinear Estimation. In Proceedings of the Adaptive Systems for Signal Processing, Communications, and Control Symposium, Lake Louise, AB, Canada, 4 October 2000; pp. 153–158.
34. Hong, S.; Lee, C.; Borrelli, F.; Hedrick, J.K. A novel approach for vehicle inertial parameter identification using a dual Kalman filter. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 151–161. [CrossRef]
35. Vargas-Melendez, L.; Boada, B.L.; Boada, M.J.L.; Gauchia, A.; Diaz, V. Sensor Fusion Based on an Integrated Neural Network and Probability Density Function (PDF) Dual Kalman Filter for On-Line Estimation of Vehicle Parameters and States. *Sensors* **2017**, *17*, 987. [CrossRef] [PubMed]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).