Smart Vehicle Navigation System Using Hidden Markov Model and RFID Technology

Reza Malekian[1*], A. F. Kavishe[1], B. T. Maharaj[1], P. K. Gupta[1], G. Singh[2] and H. Waschefort[1]

[1]Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, South Africa

[2]Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat,Solan, HP 173 234, India

* Correspondence author: Reza Malekian, reza.malekian@ieee.org

A. F. Kavishe, kavishe@gmail.com

B. T. Maharaj, sunil.maharaj@up.ac.za

P. K. Gupta, pkgupta@ieee.org

G. Singh, drghanshyamsingh@gmail.com

H. Waschefort, Waschefort@gmail.com

Abstract

The road transport of dangerous goods has been the subject of research with increasing frequency in recent years. Global positioning system (GPS) based vehicle location devices are used to track vehicles in transit. However, this tracking technology suffers from inaccuracy and other limitations. In addition, real-time tracking of vehicles through areas shielded from GPS satellites is difficult. In this paper, the authors have addressed the implementation of a smart vehicle navigation system capable of using radio frequency identification based on information about navigation paths. For prediction of paths and accurate determination of navigation paths in advance, predictive algorithms have been used based on the hidden Markov model. At the core of the system there is an existing field programmable gate array board and hardware for collection of navigation data. A communication protocol and a database to store the driver's habit data have been designed. From the experimental results obtained, an accurate navigation path prediction is consistently achieved by the system. In addition, once-off disturbances to the driver habits have been filtered out successfully.

Keywords

Hidden Markov model, Navigation path, Prediction algorithms, RFID, Vehicle tracking

1. Introduction

Currently, the road transport of dangerous goods relies on tracking technology. In designing smart navigation systems Global Positioning System (GPS) data may be augmented with Wireless Fidelity (Wi-Fi) and Global System for Mobile comunications (GSM) signals to be used to provide location information of vehicles transporting goods and passengers [1]. At present these systems suffer from limitations such as reduced reliability in areas that are not permeated by the necessary GSM or Wi-Fi signals, or areas in which GPS satellites do not provide sufficient coverage. In this paper, we have designed and implemented a system that is capable of predicting the navigation path of a vehicle on the basis of a database built using the driver's existing driving practices. The problem addressed in this work is the implementation of a system capable of using Radio Frequency Identification (RFID) based information about navigation paths, in conjunction with predictive algorithms based on the hidden Markov model to accurately determine the vehicle navigation paths in advance. For predictive systems, current methods may be split into two main groups. Prediction based on historical data use either frequency based probabilistic models or Bayesian inference to determine future events [2]. In the absence of historical data, evolutionary or meta-heuristic algorithms are used to predict optimal navigation paths, such as the one used by [3]. These algorithms usually require a constraint to be placed on the system for effective prediction to take place. For example, if the prediction is for vehicle navigation, then the constraint could be finding the quickest possible navigation path. A genetic algorithm using this constraint is addressed by [4]. The design of the proposed smart navigation system will extend the predictive systems by using existing vehicle navigation information, gathered using RFID technology, to predict navigation paths without necessarily constraining the paths to the quickest or the shortest one.

A Markov process is a stochastic process in which one can make predictions about the future state of the process based on only its current state. These predictions would be as good as predictions made if one had been aware of the entire history of the process. Since the future of the system is dependent only on its current state, the process can be considered 'memoryless'. This 'memoryless' property of a process is called the Markov property, whereas a HMM is one in which the states, though known, are not directly visible to the observer. What is visible is the output, which is dependent on the current state (as a result of the underlying process assuming the Markov property). Therefore the sequence of observed output values provides information about the sequence of states. We have used the concept of HMM to design and implement the smart navigation system.

This paper is categorized into various sections. Sect. 2 discusses the literature survey and various techniques related to vehicle navigation path prediction and data acquisition. Sect. 3 focuses on the theoretical analysis and modeling aspect of the proposed system and also shows the various navigation path possibilities. Sect. 4 lists the design principles used in two different sections for hardware and software parts. Sect. 5 provides the detailed view of hardware and software implementations of the design. Sect. 6 consists the various experimental results of the implemented system. Finally, Sect. 7 concludes the work.

2. Literature Survey

The main aim of this work is to implement a system capable of predicting the navigation path of a vehicle according to a database built using the driver's habits and populated by data using the previous navigation path details. In addition, it is required that RFID devices and

sensors be used in gathering the data. Finally, the prediction algorithm needs to be based on the HMMl. In order to meet the requirements, an investigation into predictive techniques as well as data gathering methods has been undertaken. Furthermore, a detailed look at the nature of the HMM and its application in predictive algorithms. A discussion of the accumulated literature follows in the subsections below.

2.1. Hidden Markov model

It is known that in the HMM the sequence of observed output values provides information about the sequence of states. If modeled using an HMM, then the observer will only observe a sequence of output tokens directly. The underlying states and the state transition and emission probabilities are considered prior knowledge. Baum et al. [5] have described the model based on this information; the observer can attempt to infer the sequence of states that yielded the observed output sequence. The application of the HMM as a predictive algorithm has been used in biotechnical fields to study protein structures and genetics. Sonnhammer et al. [6] discussed a method of modeling and predicting the location and orientation of alpha helices in some forms of proteins, whereas Stanke and Wack [7] used the HMM for gene prediction. Despite the differences in application, the HMM is an acknowledged tool for predictive solutions to systems that can be modeled as Markov processes. Applying HMM to a vehicle navigation system requires only that the navigation path be represented as a Markov process as well. Ning et al. [8] proposed the route recommendation system architecture and the mathematical model for driving route prediction using K-means++ and Laplace smoothing technique.

2.2. Vehicle Navigation Path Prediction

Vehicle navigation paths are usually repetitive in nature due to natural constraints that limit the freedom of the driver. One of the most common natural constraints is time; most drivers attempt to reduce the amount of time spent traveling between their origin and destination. However the eventual navigation path that a vehicle driver decides on is influenced by emergent constraints from the road network and environment in general. Harsh weather conditions, poor traffic conditions and the availability of fuel will all factor into the decision-making of the driver and affect the eventual navigation path. The proposed system will collect driver habits over a period of time and, using this data, perform a static prediction on a vehicle's navigation path. The prediction is considered static, as it will not be updated as the vehicle is being driven along the navigation path. The possible navigation path will be modeled as an HMM whose parameters are derived from the driver's existing driving practices.

A number of methods can be considered for the prediction of vehicle navigation paths. Barth and Karbassi [9] have used a hierarchical tree data structure to perform real-time prediction on the navigation path that a vehicle will take for direct trips (source to destination). Their algorithm is recomputed as new data from the vehicle arrives while the vehicle is already in transit. However, their method is incapable of handling situations in which the vehicle is required to make an erroneous stop along the way (for example a fuel stop). Froehlich and Krumm [10] discussed an alternative method where details of vehicle's navigation path are collected and grouped by similarity. Each specific navigation path is assigned an index and stored. As the vehicle begins its journey, the navigation path progresses their algorithm attempts to match the current navigation path with an existing one. Although this allows for an initial prediction of the navigation path, the prediction is continuously updated as the

journey progresses. The nature of this algorithm implies that the presence of disturbances in the navigation path (such as unexpected stops or unexpected road works) will result in the creation of new navigation paths. These paths will affect the accuracy of future predicted navigation paths by increasing the time it takes for the matching algorithm to find an existing matching navigation path. Feng et al. [11] proposed a new method using a Kalman filter? to predict the reliable location of vehicles' next move. In their experiments they achieved a degree of location performance. They also quantitatively compared the prediction performance of the proposed method and neural network methods. Silva et al. [12] proposed a solution for navigation on congested roads by using smart phones. The proposed system first obtains the traffic information from a central system and then it guide the driver on the basis of obtained information. In Simmons et al. [13] proposed the usage of the HMM to perform predictions on a vehicle's navigation path. In their method, the historical driver data are gathered using GPS information. This is then used to supply parameters to the HMM. They were able to achieve results with acracy of above 98 % in most cases, although the navigation paths they tested had very few places in which choices were required. However, in their analysis it was shown that GPS data are not reliable because of noise and shielding from buildings in urban areas or tunnels. They had to include specialized algorithms to counteract the poor reliability of the GPS data.

The shortcomings of the systems described above create the gap that must be addressed by the proposed smart navigation system to be designed. The nature of the HMM allows for accurate predictions when presented with reliable data, as indicated by the results of Simmons et al. [13] . However, this navigation system is designed to gather data using RFID devices rather than GPS. The inclusion of a filtering algorithm on the gathered data will reduce the effect of occasional disturbances to the navigation paths that will be considered by the vehicle.

2.3. Acquisition of Navigation Paths

Using the prediction algorithms described above, data regarding vehicles' navigation paths have to be gathered. The data should include pertinent variables that can be used to construct a clear picture of the driver's decision-making process. To that effect, information such as the vehicle's speed, the time of day and the occurrence of precipitation must be gathered as the vehicle travels along a navigation path. In addition to the aforementioned variables, the position of the vehicle is also critical information for the algorithm. There are a number of ways to acquire this information. These include among others:

- Manual note-taking by the driver or an accompanying passenger.
- GSM signal tracking using cell tower triangulation.
- GPS signal tracking.
- RFID based tracking.

The global presence of mobile phones has led to a number of investigations as to their usability in tracking systems. Ye et al. [14] defined the relationship between various roads on the basis of social network analysis and then applied a route prediction algorithm to find the optimal route. The authors verified the efficiency of their algorithm by conducting various experiments.Quddus and Washington [15] developed a map-matching algorithm for finding the shortest path and vehicle trajectory. The proposed algorithm uses A* search algorithm for finding the shortest path. The accuracy of the proposed algorithm is obtained 98.9 % for every 30s GPS data. The authors considered the threshold of 1000m for finding the shortest

distance. Luo et al. [16] proposed a new path-finding query that finds the most frequent path during user-specified time periods. The authors conducted their experiments on GPS data obtained from 6000 taxis in Shanghai. Kansal et al. [17] discussed a sensor network for tracking using mobile phone devices. They mentioned the fact that the prevalence of mobile devices and the increased availably of GPS technology on them make them ideal nodes in a sensory network that focuses on the same GSM signals used for voice communication. Alternatively, exclusive GPS devices can be mounted on vehicles for the sole purpose of tracking. These devices can be integrated with the vehicle's electronics for power purposes and run indefinitely. Both GSM and GPS based systems suffer from noise and accuracy limitations, particularly in areas where the GSM towers of GPS satellites have limited or no coverage. This limits their potential usage as location data sources for the proposed prediction algorithm. However, RFID devices can be used to counteract this limitation in such areas.

3. Theoretical Analysis and Modelling

In order to design the proposed system, an understanding of the HMM and its application in predictive analysis is required. Details of HMM is discussed in the next section. Here, we will explain the nature of the model and highlight the specific features that will make it suitable for this system. In Fig. 1, a Markov process with random variable 'x' is shown. The transition probabilities between the three states of the variable are indicated as a_{ij} where 'i' refers to the preceding state and 'j' refers to the resulting state. The tokens yielded by the process (outputs at each state) are represented by the variable 'y'. The probability of a specific output, given that the system is at a specific state, is called the emission probability. These probabilities are indicated as $'b_m(y_n)'$. In this case, the 'm' represents the current state and the $'y'_n$ represents a particular output. The proposed work requires the use of the HMM to predict vehicle's navigation path which is necessary to model the driver environment as a stochastic process possessing the Markov property.

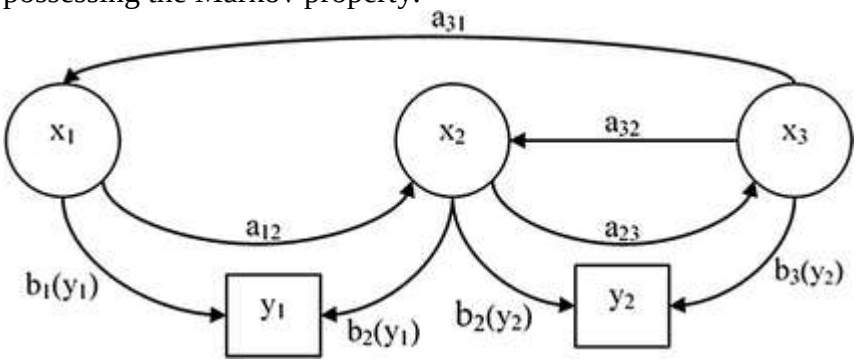

Fig. 1. Hidden Markov model

The real-world context of the navigation path prediction problem can be described as follows:

- A vehicle travels along a navigation path from point A to point B. It is assumed that this navigation path is frequently used but, there may be multiple navigation paths to travel between these two endpoints. Furthermore, it is assumed that the driver bases the decision on which navigation path to take on the environmental and traffic conditions at the time the path must be driven.
- Each vehicle that drives along this navigation path has a set of routing habits based on the decisions the drivers make, given specific environmental conditions. It is assumed that any other factors (such as the driver's health or the vehicle's status) are not statistically significant in the habitual analysis of the driver.

- Information about the vehicle's navigation paths and the external conditions at the time of the navigation is stored in an accessible database. Based on this context, a suitable stochastic process has to be determined. This would be the process modelled using the HMM. There are a number of ways to reduce this information into a Markov process. Two such methods are discussed below.

3.1. Navigation Paths as States

In this representation, the stochastic process settles at states that are defined as navigation paths. This means that transition probabilities indicate the probability of selecting a navigation path based on the present navigation paths and observations. As shown in Fig. 1, all possible navigation paths between points A and B are possible states of the Markov process. Information about the observations made on a specific day along a given navigation path of a vehicle is stored in a database. The navigation path information is linked to the driver's driving habits. The variables indicated as a_{ij} in the diagram represent the possibility of the driver selecting a navigation path, given the last one he has driven. The variable *bi* represents the probability of a specific observation being made (y_i), given the current state. For instance, if a vehicle travels along navigation path 3, the probability of driving navigation path 2 next is $a32$ in the figure. In addition, the probability of observation 2 being made is $b_3(y_2)$ where y is a variable indicating the different observations.

3.2. Vertices/Edges as States

In this representation, the stochastic process settles at states that represent points along the navigation path. These points could be intersections (vertices) or road sections (edges). The transition probabilities in this case would represent the chance of proceeding to a specific intersection (or road section), given that the vehicle is currently on a specified intersection (or road section). Figure 2 represents the road sections from x_1 to x_4 as states in the Markov process. A navigation path is stored in the database as a sequence of transitions from road to road. The transition probabilities between the road sections are expressed by variable *aij* in Fig. 2 above. For instance, given that the vehicle is on road section x_1 above, the probability of turning onto road $x2$ is $a12$ whereas the probability of driving on straight ahead would be a_{13} and the probability of turning onto road x_4 is a_{14}. It should be noted that this is functionally similar in most aspects for using the road intersections as states, rather than the road sections themselves. This is due to the fact that representing a navigation path as a sequence of road transitions is equivalent to using intersection transitions, and just as effective.

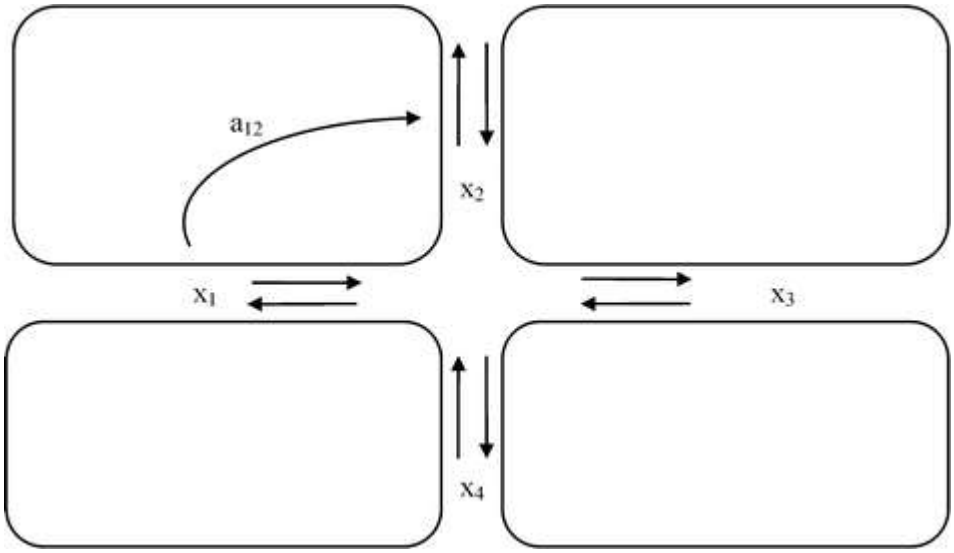

Fig. 2. Road sections as states

3.3. Selected Navigation Path Model

From the above two alternatives, for representation of the problem as a Markov process, we have considered that using the road intersections (vertices) as states in the algorithm would be the most effective way of representing the problem. Figure 3 shows a virtual road network with intersections marked as numbers. The arrows between the intersections indicate the allowable transitions between them. For example, it is possible to transition from vertex ’0’ to vertex ’1’ as well as from vertex ’1’ to vertex ’0’. This form of directed graph representation of the road network allows for easy translation into the HMM problem space. If we assume that the process by which a vehicle proceeds along a navigation path is a Markov process, then each vertex would represent a possible state in which the process can be at a specific time. The presence of at least one directed arrow between two states indicates the existence of a transition probability between the two. To expound on that point, there is a directed arrow between vertex ’1’ and vertex ’4’, which implies that there is a probability of transitioning between the two states (from 1 to 4). In addition, there are three directed arrows starting at vertex ’1’ (from 1 to 4 to 5 to 2 and back to 1), implying that it is possible to transition from vertex ’1’ and return to it.

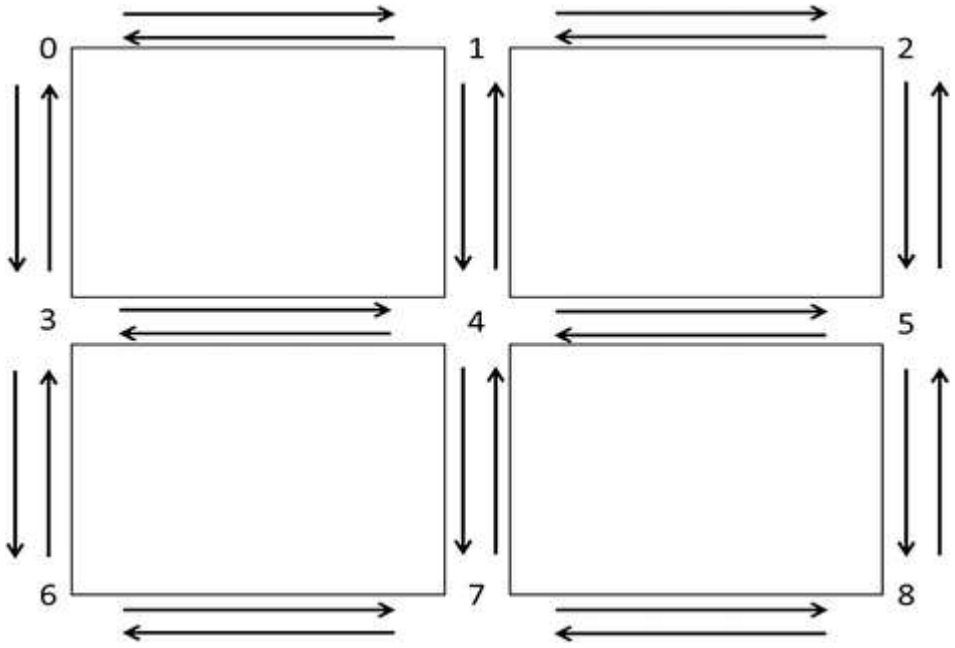

Fig. 3. Road network model

As the purpose of an algorithm is to determine the likely navigation paths, it is unlikely that a navigation path accurately predicted will require the vehicle to loop backwards onto the same intersection. Therefore, the definition of possible transitions is constricted, with the condition that the presence of exactly one directed arrow between two states indicates the existence of a transition probability between them. This would mean that it is not possible to transition from vertex ’1’ and return to it, as that requires more than 1 directed arrow. Having defined the states of the process, we can define the transition probabilities of the model. These probabilities indicate the likelihood of a vehicle to transition from the current state to a future one. Using Fig. 3, if the transition probability from vertex ’1’ to ’4’ is 60 %, from ’1’ to ’2’ is 35 % and from ’1’ to ’0’ is 5 %, then one can assert with high confidence that the vehicle at vertex ’1’ will most likely transition to vertex ’4’ next. These transition probabilities are derived from the accumulated existing driving practices of the driver.

Finally, translating the problem into the HMM which uses a-priori knowledge of emission probabilities (the probability of an observation being made given that the process is at a specific state), state transition probabilities and the initial probability of each state to compute the most likely sequence of states which results in the observations. This section of the HMM should be adapted in a novel fashion to suit the requirements of the work. For a road network based problem, possible observations include traffic, weather and time of day. None of these are directly caused by the current state of the process which affects their inclusion into the problem. To represent these two main design decisions were made.

- Three observations of a binary nature would be tracked, namely traffic (high or low), precipitation (rainy or dry) and daytime (day or night). These observations have been combined into a bit flag, allowing for eight possible combinations. For instance, the bit flag ’101’, which is the number 5 in the decimal system, would represent ’high traffic, dry, daytime’ if we use the order of observations as stated earlier and other possible combinations could be similarly designed.
- The observations would be represented as a list (containing the present observation, repeated) rather than a single observation. This stems from the nature of the HMM.

The number of states in the sequence predicted is directly linked to the number of available observations. If only one observation is available, only one state is predicted. However, with a list of observations, the sequence returned has a maximum length limited only by the length of the list of observations. By using in a maximally long list of observations, the resulting predicted sequence can be truncated based on another parameter (for example, when the sequence first reaches the destination state).

4. Design Methodology

In order to meet the stated objectives, a system consisting of a sensory platform and a predictive algorithm has to be designed. The possible interface of the proposed smart navigation system has been determined and preferable outputs have been considered. Based on these parameters, the boundary for the proposed smart navigation system has been defined and the top-level components of the system have been conceptualized and designed. These top-level components are broadly separated into a primarily hardware-based sensory platform and the software implementation of a predictive algorithm. The conceptual design of these two components is discussed in the following sub-sections.

4.1. Hardware-Centric Component

The hardware sensing platform has been designed to allow for the gathering of environmental and geographical data. A detailed hardware platform as represented in Figs. 4 and 5 consists of a sensory system comprising of an accelerometer, humidity sensor, temperature sensor and real-time clock capable of interrogating environmental conditions. The raw data output from these individual components would be used to determine the environmental and traffic conditions along the navigation path being recorded. The navigation path itself is determined by the sequence of tags read by the RFID reader in the vehicle. The real-time clock is included to reduce the complexity that the user would have to face when operating the system. In addition, it may not always be possible to recall the starting time of a navigation path with sufficient accuracy. Finally, the interface between the hardware and the computer running the algorithm was implemented as a direct cable, communicating using a serial link. This allows for direct viewing and analysis of collected data when the hardware platform is connected to a portable computer while the vehicle is on a navigation path. The entire hardware platform is designed around an FPGA controller.

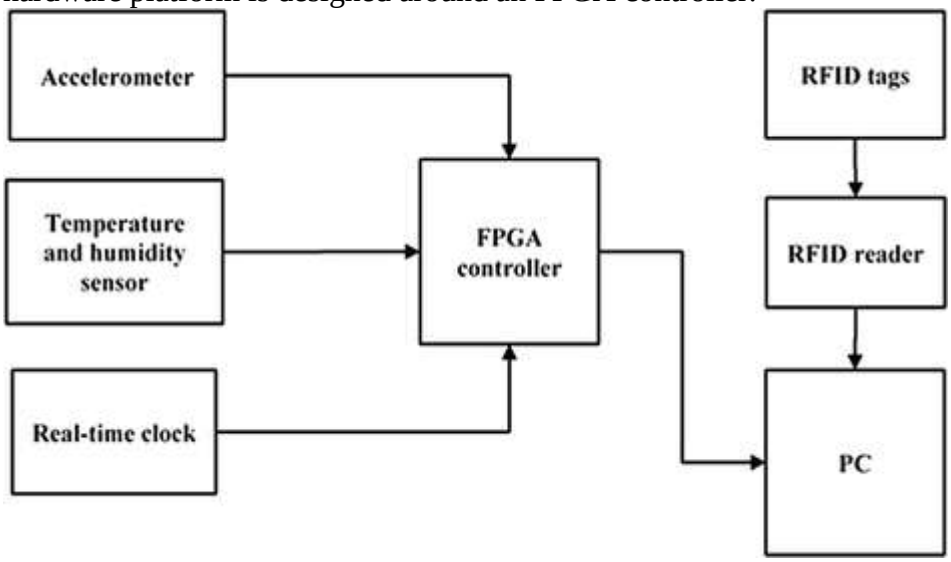

Fig. 4.Detailed hardware design of implemented system

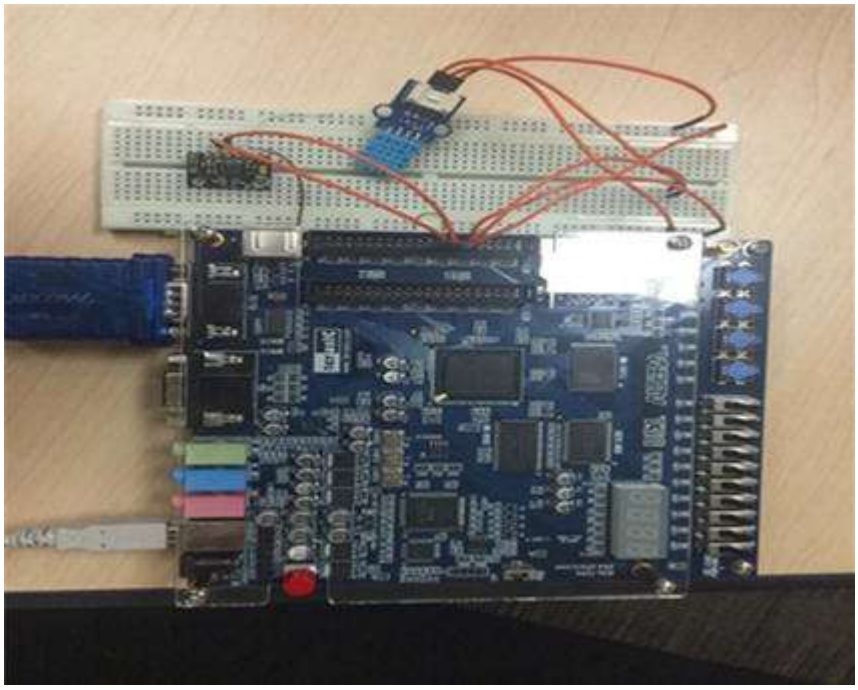

Fig. 5. Components used for designing the smart navigation system

4.2. Software System

The main component of the proposed navigation system as shown in Fig. 6, is the algorithm, which predicts a navigation path based on historical data collected by the hardware system. These historical data are stored in a database that can be accessed by the algorithm. Two software subsections complement the functionality of the predictive algorithm. (1) The filtering subsection is used to ensure that statistically significant data points have been used in the predictive algorithm to prevent unnecessary skewing of results due to once-off deviations from the norm. (2) The HMM is used to predict the navigation paths. Currently, two major algorithms are used to solve problems modelled by the HMM: (a) The forward-backward algorithm is used to determine the probability of all potential state transitions in the model that would have led to the observed outputs, (b) The Viterbi algorithm is used to determine the most likely sequence of states that would result in the observed outputs. In most cases the algorithm returns the same solution given the same input and conditions. However, the Viterbi algorithm is more representative to use with the proposed smart navigation system, which is why it is implemented as opposed to the forward-backward algorithm.

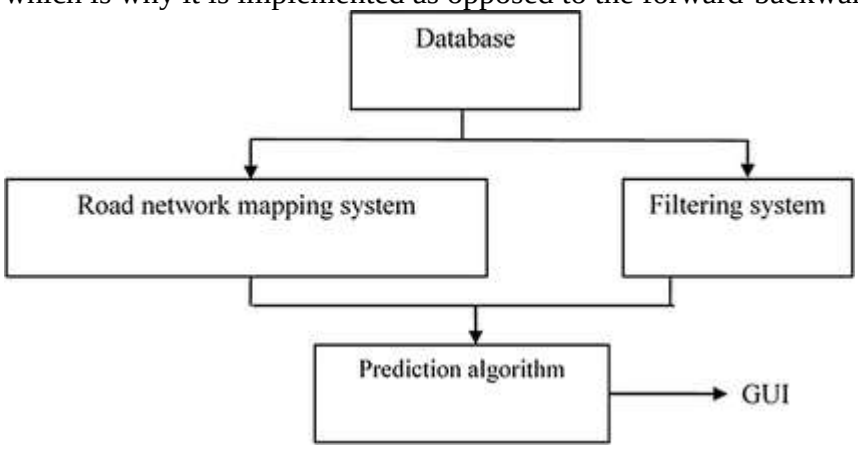

Fig. 6. Components of software systems

An exponentially weighted moving average (EWMA) filter has been used to smooth out spikes in the data points queried from the database before using them to compute state parameters for the prediction algorithm. The EWMA filter has an easily adjustable smoothness, making it ideal for situations in which the volume of data is not necessarily known beforehand. As for the mapping, an adjacency list has been used to store information about the roads between intersections. The adjacency list would be queried from the database and used to generate a graph with intersections as vertices and roads as directed edges between them.

5. Implementation

5.1. Hardware implementation

To implement the proposed smart vehicle navigation system as shown in Fig. 4, we have selected two communication protocols which are described as follows:

5.1.1. Universal Asynchronous Receiver/Transmitter (UART)

Communication between the FPGA and the computer on which the software performs is achieved using the UART protocol. This protocol allows for serial data transmission using a synchronized clock between the sender and receiver. The protocol transmits and receives data according to the set of standards described below:

- Idle - the line is high.
- Start - the line is set low (a ’0’ bit is sent).
- Data - the eight data bits are sent by setting the line to high for ’1’ and back to low for ’0’.
- Stop - the line is set high (a ’1’ bit is sent).
- There are two options:
    - Idle - keep the line high
    - Start - set the line low and send another byte.

The entire process is controlled by the UART clock speed, commonly called the baud rate. Baud (Bd) is the unit for symbol rate, effectively representing the number of distinct symbol changes per second on a digital signal. This clock is generated both by the emitter and receiver. As a result, the two clocks may fall out of synchronization, which would result in incorrect data or failed transmissions. To avoid such a type of scenario, the FPGA is coded to oversample the incoming signal at 16 times the baud rate. This allows for samples of the incoming data bits to be taken at times when the bit value is settled irrespective of clock synchronization status. For instance, in Fig. 7 the faster clock can be used to sample the slower one. If there are 16 ticks of the faster clock for each cycle of the slower one, one simply counts the number of ticks (rolling over from 15 to 0) and sample at a number that is suitable for the situation. To sample in the middle of the ’high’ cycle in the Fig. 6, the sample would be taken at tick counter ’0’.

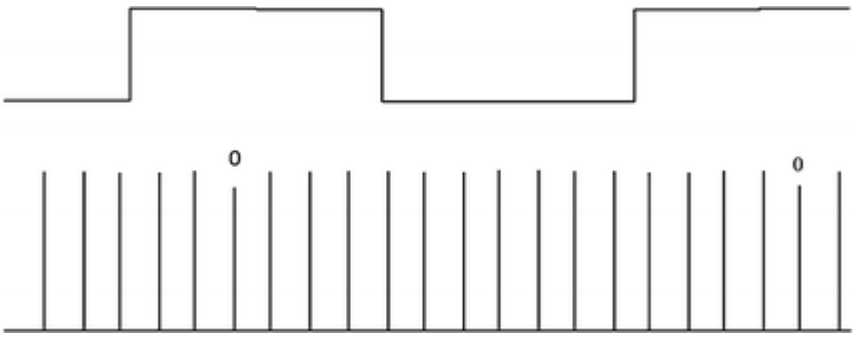

Fig. 7.Oversampling a slow clock

In the proposed navigation system the UART protocol is implemented as a finite state machine (FSM), shown in Fig. 8. The control signal 'rx' determines whether or not the UART is ready to receive a byte. The variable 'nbits' counts the number of bits received and after a full byte (8 bits) and the stop signal (1 bit) have been received, it returns to the idle state. The transmission FSM is fundamentally identical to the reception one. However, the direction of data flow is reversed.

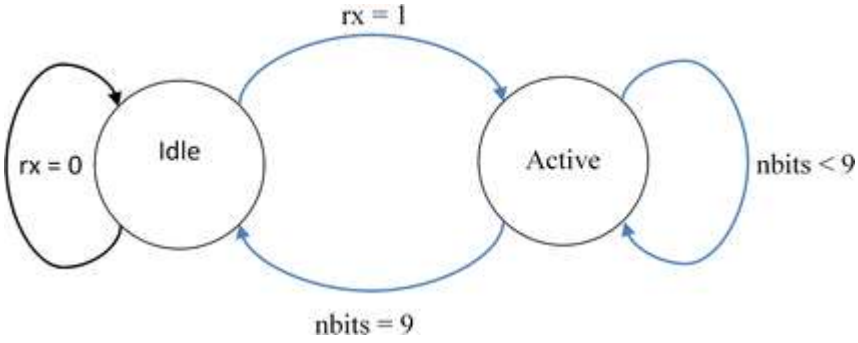

Fig. 8. UART reception FSM

5.1.2. Inter-Integrated Circuit (*I2C*)

Communication between the FPGA controller and the digital sensors uses the *I2C* protocol. The basic protocol considers two signal lines: (1) the data line, and (2) clock signal. For most *I2C* applications a maximum clock signal of 400 kHz is recommended. Considering the nature of the data being sent from the sensors to the FPGA, an *I2C* clock of 100 kHz is generated and used for the protocol. Because of to its dual lane nature, the protocol description is slightly more involved than the UART one. As shown in Fig.9, the data line is pulled high by an external resistor while the system is idling.

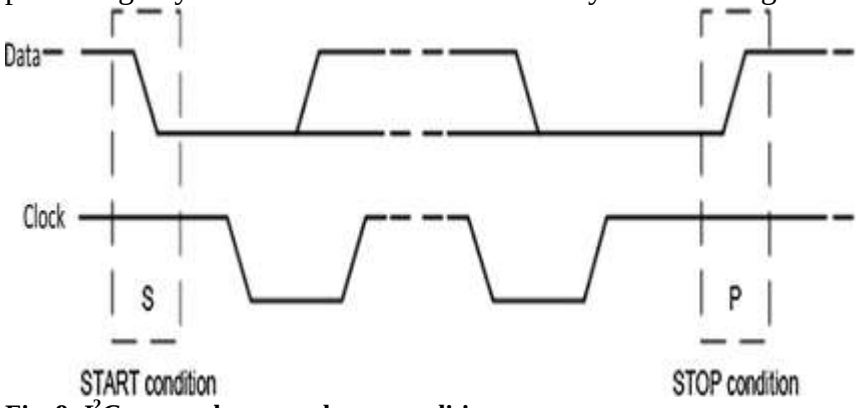

Fig. 9. I^2C protocol start and stop conditions

In between these two conditions, data are sent in bytes (exactly 8 bits long). Each transferred byte is followed by an acknowledgment condition, which is defined as the data line being pulled and held low (by the receiver) for the duration of a clock cycle 'high'. Based on this protocol and with small variations to its fundamental design, communication between the FPGA and sensory units is established and used to interrogate them as required. Protocol *I2C* is also implemented as FSM, shown in Fig. 10. Here, the system is initially in the idle state. As soon as the reset ('rx') button is set 'low', the system moves into the 'Start' state, which sets the 'start' condition on the data line. The next state is the 'Transfer' state in which the first byte on the stack (whose length is given by 'nbytes') is sent or received. Once the byte transfer is completed, an acknowledgment condition is set on the data line. Depending on whether there are more bytes to transfer or not the system then either transitions into the 'Stop' state or returns to the 'Transfer' state.

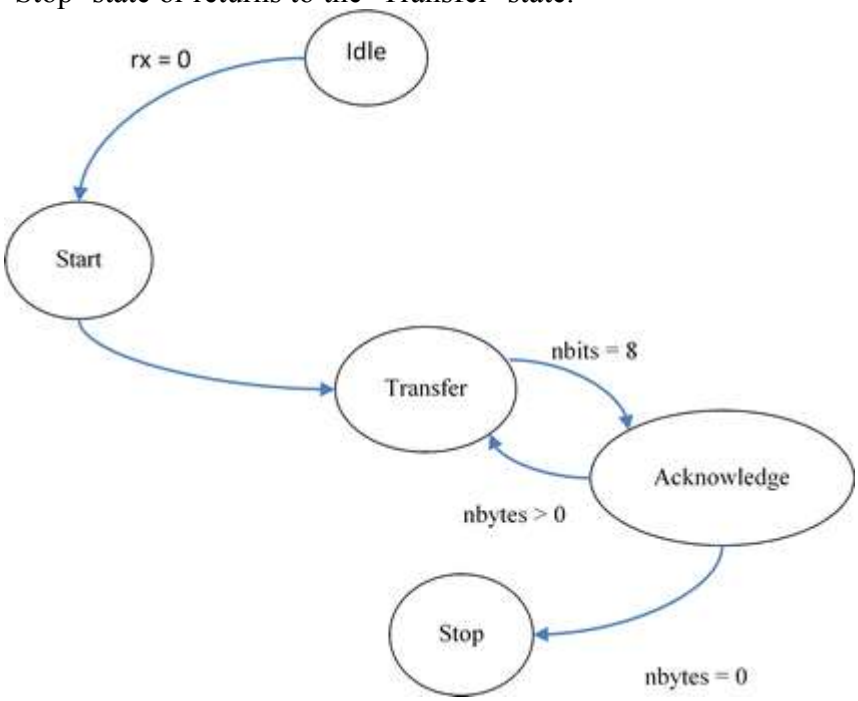

Fig. 10. I^2C FSM

5.2. Software implementation

The primary function of the software system is the prediction of the most likely navigation path for the navigation of the vehicle. As shown in Fig. 6, there are four subsections related to this prediction system, which are discussed as follows:

5.2.1. Driver habits database

The algorithm depends on information gathered about external conditions during the navigation of a vehicle, as well as the navigation path used between the endpoints of interest. This information is stored in a database, which is accessible by the algorithm. The schema of the database to store the received information is shown in Fig. 11. Here, the table representations only include the primary and foreign keys, whereas the arrows point towards the table being linked by the foreign keys. Based on this initial description of data, a relational database scheme is developed. To develop these schemas, ab SQLite database has been used. The programming language Python is used for the further implementation section. An Object Relational Mapper (ORM) system called peewee is used to facilitate database

communication within the software. The ORM layer allows for the representation of database tables and relationships as higher-level language classes and types. It encourages separation of low-level database management from high-level data usage, making it easy to switch out databases completely if the situation requires it.

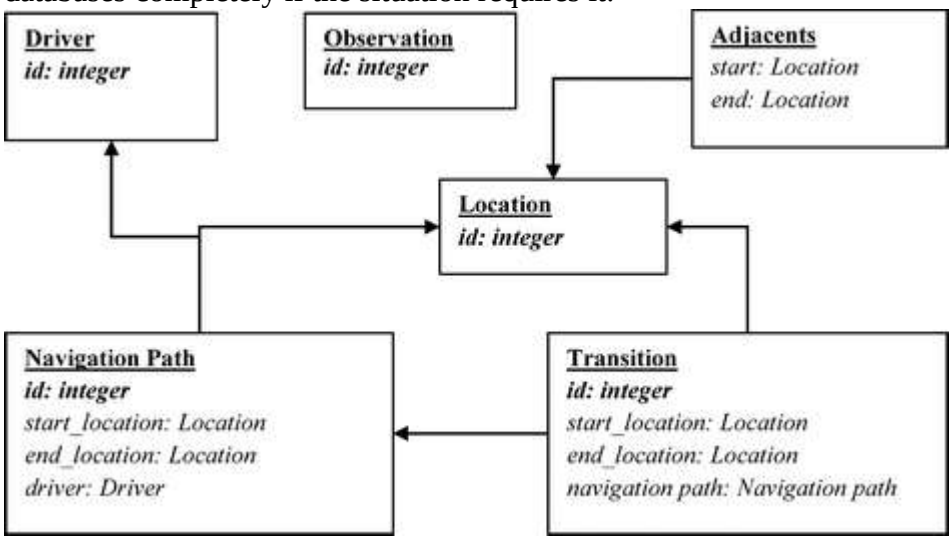

Fig. 11. Database schema showing major links

5.2.2. Road network map

The algorithm is required to predict vehicle's navigation path along a road network. This section of the design focuses on translating the real-world geographical map system into a mathematical construct that can be used by the algorithm. As discussed in the hardware implementation section, geographical data gathered by RFID devices strategically placed along the roads. This requirement of placing tags in advance allows for manual gathering of the geographical information relating to tagging locations. For instance, a tag placed along a certain road would be translated into the geographical coordinates of the terminal intersection of that road. Once the intersections had been tagged accordingly, an adjacency list is used to represent possible transitions between the intersections. The decision is made to use an adjacency list rather than an adjacency matrix because of the inherently sparse nature of road networks when represented as directed graphs. The 'sparse' property can be defined as the relationship between the average numbers of edges intersecting at a single vertex versus the total number of vertices in the graph.

For road networks, most intersections have four roads feeding them, whereas the number of intersections in an area with multiple navigation paths between two endpoints is assumed to be significantly higher than four. As the number of intersections (represented here by 'n') grows, the adjacency matrix size grows by a factor of $n2$
. The adjacency list is less space-efficient for small numbers of vertices, but as the scale of the problem grows, the upper limit to the size of the adjacency list approaches $n2$
as well. However, if the network is sufficiently sparse (for example, if there are at most four edges at a vertex), the adjacency list will be significantly more space-efficient, approaching only 4n in size.

Table 1. Table adjacency list

Vertex	Adjacents
0	1,3
1	0, 2, 4
2	1, 5
3	0, 4, 6
4	1, 3, 5, 7
5	2, 4, 8
6	3, 7
7	4, 6, 8
8	5, 7

Table 2. Adjacency matrix vertex versus adjacent

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	0	1	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0
2	0	1	0	0	0	1	0	0	0
3	1	0	0	0	1	0	1	0	0
4	1	0	0	1	0	1	0	1	0
5	0	0	1	0	1	0	0	0	1
6	0	0	0	1	0	0	0	1	0
7	0	0	0	0	1	0	1	0	1
8	0	0	0	0	0	1	0	1	0

Here, Tables 1 and 2 represents two road network models for Fig. 3. Note that the adjacency matrix representation has a large number of cells with no constituent data, whereas the adjacency list simply has shorter constituent lists as required.

5.2.3. Filtering

As the vehicle's navigation details are collected, it is expected that occasional deviations such as side trips to petrol stations, reroutes due to accidents or roadworks, human interference and many others from the normal path, may take place. To keep the efficiency of the prediction algorithm high, a filtering subsection has been designed to process the pertinent data that is prone to disturbance before using these data in the prediction algorithm. Here, the vehicle navigation details are represented as a sequence of transitions. If a vehicle navigates along the same navigation path 'n' times, there will be at least 'n' transitions from each intersection on the navigation path to the next one, for the intersection along the navigation path used.
Fig. 12 shows a short section of such a navigation path. As shown in Fig. 12, the vehicle has transitioned from vertex '3' to vertex '5' twice, and from vertex '3' to vertex '4' only once. Vertex '3' has three possible transitions becuse of the nature of the road network. In order to translate these data into transition probabilities, all the possible transitions are first considered:

$$tp_{3x}=1/adj(3)=1/3 \quad (1)$$

Equation 1 states that the initial transition probabilities for all transitions from state '3' are equal and a function of the number of adjacent intersections to state '3'. After this, the

individual transition probabilities are incremented by 1 for each transition in the database, which is shown in Eqs. 2 and 3. The resulting transition probability values are then normalized to sum up to 1 in Eqs. 4–6.

$$tp_{34}=tp_{3x}+1=4/3 \quad (2)$$

$$tp_{35}=tp_{3x}+2=7/3 \quad (3)$$

$$tp_{32norm}=tp_{32}/(tp_{32}+tp_{34}+tp_{35})=1/12 \quad (4)$$

$$tp_{34norm}=tp_{34}/(tp_{32}+tp_{34}+tp_{35})=1/3 \quad (5)$$

$$tp_{35norm}=tp_{35}/(tp_{32}+tp_{34}+tp_{35})=7/12 \quad (6)$$

The above calculations show that there is a 58.33 % chance of the vehicle transitioning to state 5 from state 3, a 33.33 % chance to transition to state 4 and an 8.33 % chance to transition to state 2. Whereas these values are intuitively acceptable, the nature of the HMM algorithm calls into question their reliability. The issue with this unfiltered transition data has strong confidence given to the transition probabilities after only 3 samples. It can also be seen that the transition probability from state ’3’ to ’2’, which should be approximately 33.33 %, is now 8.33 %. This low probability would have a large effect on the algorithm, where the probabilities are combined multiplicatively, and reduce all possible chances of a probable navigation path transition from ’3’ to ’2’ even when that transition may be the likeliest one based on the situation at the time of prediction.

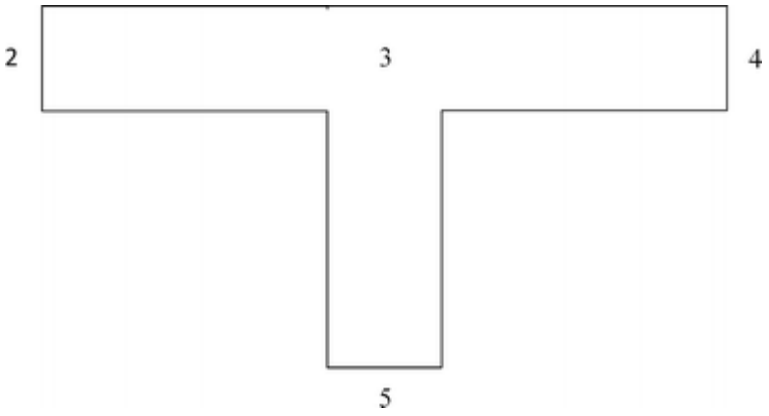

Fig. 12. Road section

Therefore, it is decided to implement an Exponential Weighted Moving Average (EWMA) filter to reduce the effect of single deviations and low sample sizes on the resulting transition probabilities. After three samples, there should ot be a significant difference between the new transition probabilities and the ones computed without any vehicle information and, after 100 samples, a few new deviations should not have a significant effect on the transition probabilities that have been established. A slow response of the EWMA filter would provide the required smoothing. The filter parameter (represented by a) could be parameterized, allowing for user control over the smoothness of the filter and its reactiveness to change.

With an ’α’ value of 0.9, the filter takes 0.9 of the existing transition value, and $(1 - 0.9)$ of the new value (set to be 1 in this case, as in the unfiltered case as well). This computation

takes place recursively for each new transition recorded. At the end, the normalization is done as before.

$tp_{34}=(0.9\times tp_{34})+0.1=0.4$ (7)

$tp_{35}=((0.9\times tp_{35})+0.1)\times 0.9)+0.1=0.46$ (8)

Normalizing using the normalization formulas given in Eqs. 4–6 for above results will be:

tp_{32} = 27.93%

tp_{34} = 33.52%

tp_{35} = 38.55%

These values still indicate the highest preference for the ’3’ to ’5’ transition and the lowest for the ’3’ to ’2’ transition. However, all the values are still relatively close to the original 33.33 % assigned to them by the nature of the road network. This is a closer representation of the truth, as the sample size does not give us the requisite confidence to deviate too far from the default, as was the case with the unfiltered values. The smaller range between the probabilities also means that the algorithm will be free to attempt all alternative values if the situation demands it.

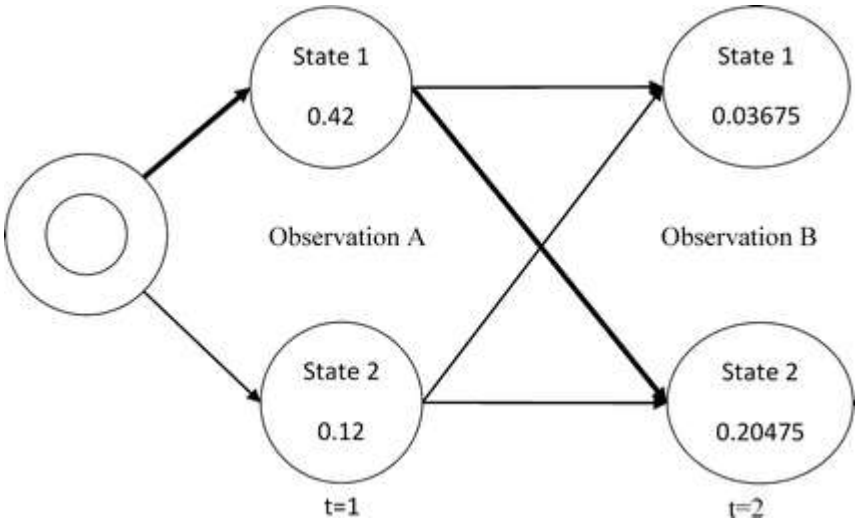

Fig. 13. Viterbi algorithm visualizer

5.2.4. HMM algorithm

As stated earlier, the Viterbi algorithm is used to determine the most likely sequence of intersections based on vehicle navigations and observations on the day of travel. This algorithm can be visualized as a trellis diagram, with the solution being the highest probability path through the trellis. The algorithm is visualized in Fig. 13 and a simple problem is postulated to demonstrate the algorithm’s process. For this we have considered the following parameters:

- Initial probabilities:
  - State 1 = 60 %

    - State 2 = 40 %

- Transition probabilities:
    - 1 to 1 = 35 %
    - 1 to 2 = 65 %
    - 2 to 1 = 55 %
    - 2 to 2 = 45 %

- Emission probabilities:
    - A given 1 = 70 %
    - A given 2 = 30 %
    - B given 1 = 25 %
    - B given 2 = 75 %

On the basis of the above probabilities, this algorithm proceeds as follows:

1.Initial path sections (t = 1). For these sections the computation is given by:

$$P_{initialforstate} \times P_{observation,givenstate}$$

- Path 1 = (0.6)×(0.7) = 0.42
- Path 2 = (0.4)×(0.3) = 0.12

1. Next observation (t = 2). For these sections the computation is given by:

$$P_{oldstate} \times Pt_{ransitionfromoldtonewstate} \times_{Pobservation,givennewstate}$$

- Path 11 = (0.42)×(0.35)×(0.25) = 0.03675
- Path 12 = (0.42)×(0.65)×(0.75) = 0.20475
- Path 21 = (0.12)×(0.55)×(0.25) = 0.0165
- Path 22 = (0.12)×(0.45)×(0.75) = 0.0405

Select the highest probability path to each state and preserve the path. For the above case those are Path 12 (probability 0.20475) and Path 11 (probability 0.03675). The remaining paths can be discarded. If there are additional observations, start with transitions from the remaining paths and proceed as indicated in number 2 above. This means that the next computations would be for Paths 121, 122, 111 and 112.

One aspect of the algorithm is the fact that probabilities are multiplied along the path. As probabilities are fractional quantities, their multiplication can result in small numbers that encroach on the precision limits of most computer systems. As a means of dealing with that, the Viterbi implementation used in this work uses the inverse logarithm of all probabilities in its calculations. This means that the values are added rather than multiplied, and, at the end, the Viterbi path is the one with the lowest inverse log probability. This effectively removes the need for handling extremely small numbers, as the inverse logs of probabilities will generally be large positive numbers and primarily using addition should keep them well within the precision limits of most commercial computer systems.

6. Experimental Analysis

This section represents the various experiments carried out using the above-mentioned hardware, software designs and algorithms. Details of these experiments and the results obtained are listed as follows:

6.1. Experiment 1: Prediction Accuracy

The objective of this experiment is to determine whether the algorithm can accurately predict the navigation path of the vehicle using the path database. To perform this experiment we used the GPS tracking android phone software (intelligent routing by FilterIS). In our observations we obtained the results shown in Table 3. From the data obtained it is clear that the algorithm correctly predicted the navigation path eight times out of 10, and for the two incorrect predictions the reasons are outside the scope of the problem.

Table 3Experiment 1 results

Test	Accurate	Possible reason for inaccuracy.
1	Yes	
2	No	Phone call diversion
3	Yes	
4	Yes	
5	Yes	
6	Yes	
7	No	Traffic lights out on major streets.
8	Yes	
9	Yes	
10	Yes	

6.2. Experiment 2: Heading Information Through Magnetic Compass

The objective of this experiment is to perform well in a test involving multiple direction changes. The vehicle is driven on a straight road for 100 meters; it makes a sharp right-hand turn and thn cotinues straight. Figure 14 depict three images, one on the first straight, the second while the turn is taking place and the third three seconds after the second turn when the vehicle is moving straight on the road. The degrees for Fig. 14 from left to right are: 97.12, 109.6 and 97.63. A deviation of less than one degree is shown.

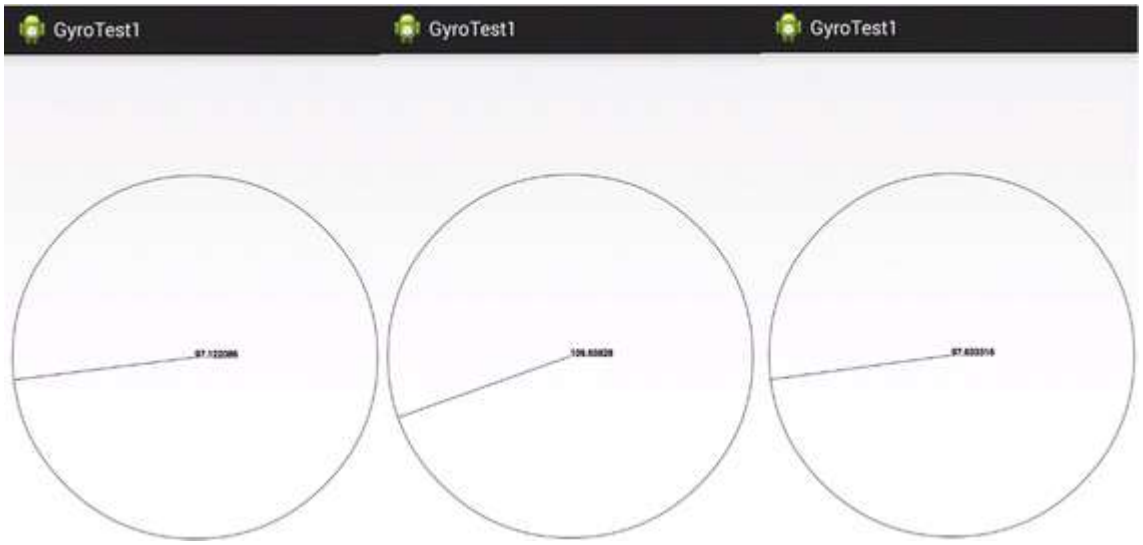

Fig. 14. Vehicle movement readings using magnetometer on a T-shaped road test

6.3. Experiment 3: RFID Tag Read Distance

The objective of this experiment is to determine the effective range of the RFID tags used in the system. To perform this experiment, we used RFID tags, a reader, and distance-measuring devices. In Table 4, most of the tags were unable to reach 5m. However, despite the rating for maximum distance on the tags being given as 3m, they could be all capable of being read from outside that range.

Table 4.Experiment 2 results

Tag ID	Read distance
123483	3.8
123444	5.2
123445	4.4
234545	4.6
123583	5.1
234511	3.4
165001	4.0
769696	4.2
123458	4.3

6.4. Experiment 4: RFID Devices Read Rate

This experiment sought to establish an upper limit on the vehicle speed that would allow for interrogation of the tags placed by the road. To perform this experiment, we used RFID tags and readers. Here, tags had a significantly larger acquisition rate, which improved the range in which the vehicle needed to be for a successful tag read. We considered a road distance of 700 m and all the tags were placed along it. Seve successful trips were recorded. A successful trip is defined as one in which all of the tags placed along the road are interrogated. The maximum speed of the vehicle across this experiment was recorded at 28.3 km/h, which is a marked improvement over the values obtained from the initial run of the experiment. The experimental runs are listed in Table 5 below.

Table 5. RFID read rate data

Navigation Path	Start	End	Duration	Distance	Speed (km/h)
1	19:47:38	19:51:38	0:04:00	700	10.50
2	20:09:39	20:12:04	0:02:25	700	17.38
3	20:23:27	20:25:25	0:01:58	700	21.36
4	20:36:50	20:38:43	0:01:53	700	22.30
5	20:48:29	20:50:03	0:01:34	700	26.81
6	21:05:12	21:06:41	0:01:29	700	28.31
7	21:18:52	21:20:30	0:01:38	700	25.71
Average			0:02:08	700	21.77

6.5. Experiment 5: GPS Location Estimation

The objective of this experiment was to obtain the latitude and longitude from the location listener and convey this to the map application. The camera position was constantly moved to the user's current location. A snapshot of the GPS location while on the move is shown in Fig. 15.

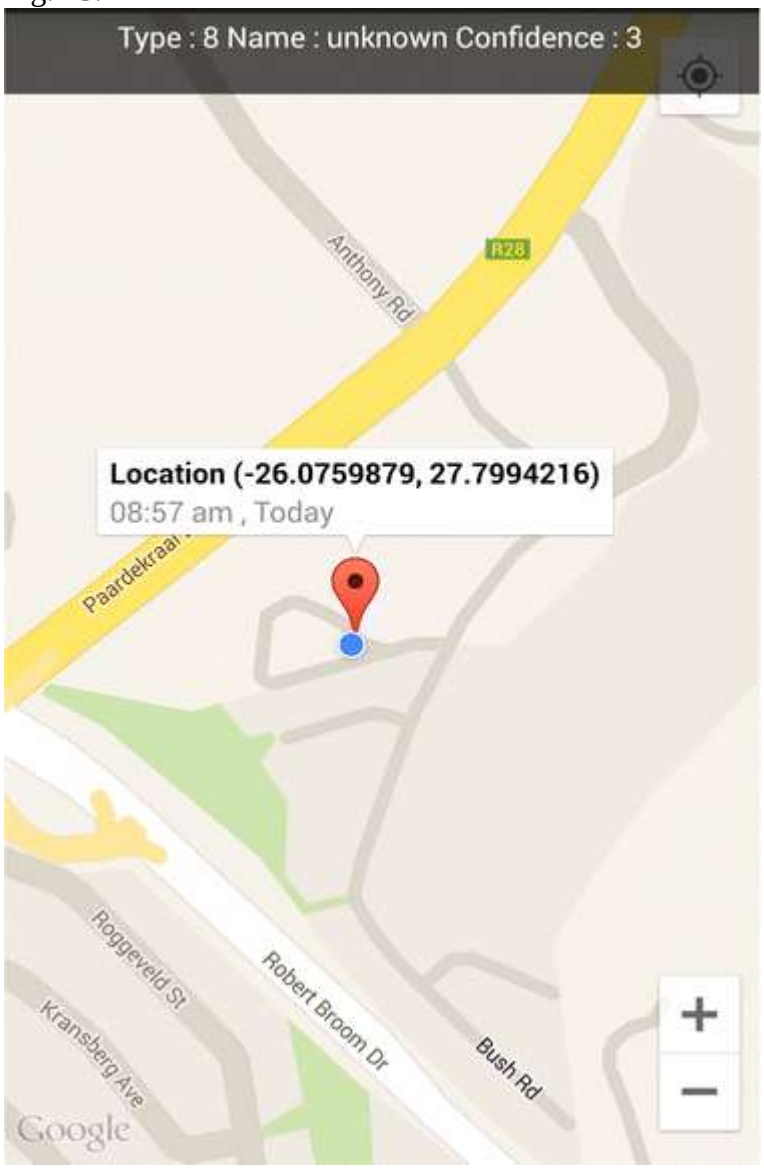

Fig. 15. Position estimation using GPS sensor

6.6. Experiment 6: Tag Detection Application

A reader is mounted underneath the car on the front side. Tags were placed on a straight section of an underground parking lot of a local mall 10 m apart. The vehicle traversed the tags at a speed of about 30 km/h. All four tags that were placed were detected by the reader and displayed with position data and timestamps. This is shown in Fig. 16.

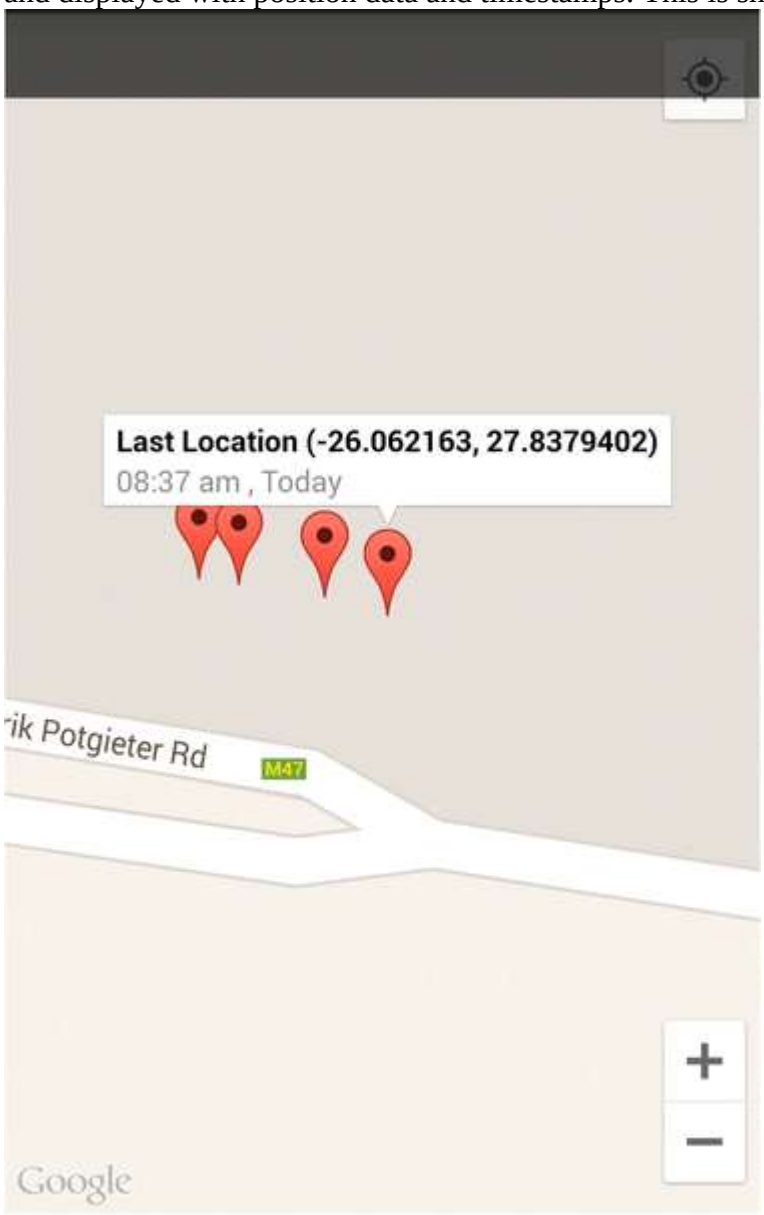

Fig. 16. Successful tag detection with location data and timestamps

6.7. Experiment 7: Total System Delay

This experiment measured the amount of time expended during the execution of the algorithm. Here, we used the prediction algorithm on a commercially available personal computer. We also used the stopwatch to record the working time of algorithm. In the results obtained it was found that the algorithm is much faster than the reaction time of the person holding the stopwatch.

6.8. Experiment 8: Filtering System

This experiment was designed to test the robustness of the filtering system in the face of disturbances. To perform this experiment we used a sample database with some pre-existing navigation paths and it was assumed that one of the navigation path would always be the same for a specific vehicle. Then, we added a new navigation path with a different transition from the common navigation path used by the vehicle and observed the resulting prediction. From our analysis, it is found that the predicted navigation paths were not affected by single instances as long as there were more than one existing navigation paths for the vehicle between the points of interest. In cases where there was only navigation path for the vehicle between the two endpoints, the addition of another navigation path would not be regarded as a disturbance because of the lack of information on what navigation path should be considered the norm.

7. Conclusion

The main objective of this research work is to implement a system capable of predicting a vehicle's navigation paths based on the existing driving practices of drvers by using an HMM inference algorithm. These existing driving practices are collected by a hardware-sensing platform and RFID technology. The software algorithms in this research work all turned out quite well. Implemented algorithms were rigorously tested and the entire prediction software suite was sbject to unit tests to prevent changes over time from affecting its core functionality. In addition, the UART communication module was well designed. No timing issue or data loss issues occurred throughout the implementation of work. The Viterbi algorithm, which is an inference algorithm for the HMM, was implemented using Python programming language. This algorithm provided the most likely sequence of states that a Markov process underwent to provide a given sequence of observations. For the data collection, sensors were acquired and attached to an FPGA controller. Communication protocols between the sensors and FPGA, as well as the FPGA and a computer, were designed. An RFID reader and tag set was used to gather geographical data as well as trigger the gathering of environmental data via the serial connection being bridged via the computer.

With the recent advancement in sensitivity and scale of smart-phone sensors, one future prospect for this project would be a fully mobile application, capable of collecting existing driving practices of a driver, storing the information, predicting navigation paths and tracking the vehicle in real-time using GPS where available and falling back on Bluetooth low energy devices, which are ideal because of the near universal presence of Bluetooth technology on modern smart phones.

References

1.Ning, Y., Zhong-qin, W., Malekian, R., Ru-chuan, W., & Abdullah, A. H. (2013). Design of accurate vehicle location system using RFID. *Electronics & Electrical Engineering*, *19*(8), 105–110.Google Scholar

2.Tebaldi, C., & West, M. (1998). Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, *93*(442), 557–573.MathSciNetCrossRefMATHGoogle Scholar

3.Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, *31*(12), 1985–2002.MathSciNetCrossRefMATHGoogle Scholar

4.Baker, B. M., & Ayechew, M. A. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, *30*(5), 787–800.MathSciNetCrossRefMATHGoogle Scholar

5.Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, *41*, 164–171.MathSciNetCrossRefMATHGoogle Scholar

6.Sonnhammer, Erik L. L., Heijne, Gunnar V., & Krogh, A. (1998). A hidden Markov model for predicting transmembrane helices in protein sequences. *Ismb*, *6*, 175–182.Google Scholar

7.Stanke, M., & Waack, S. (2003). Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, *19*, 215–225.CrossRefGoogle Scholar

8.Ning, Y., Zhong-qin, W., Malekian, R., Qiaomin, L., & Ru-chuan, W. (2015). A method for driving route predictions based on hidden Markov model. *Mathematical Problems in Engineering*, *2015*, 1–12.Google Scholar

9.Barth, M., & Karbassi, A. (2003). Vehicle route prediction and time of arrival estimation techniques for improved transportation system management." *Intelligent vehicles symposium*, (pp. 511–516).

10.Froehlich, J., & Krumm, J. (2008). Route prediction from trip observations. *SAE Technical Paper No. 2008-01-0201*.

11.Feng, H., Liu, C., Shu, Y., & Yang, O. W. (2015). Location prediction of vehicles in VANETs using a Kalman filter. *Wireless Personal Communications*, *80*(2), 543–559.CrossRefGoogle Scholar

12.De Silva, M. W. H. M., Konara, K. M. S. M., Karunarathne, I. R. A. I., Lal, K. K. U. P., & Wijesundara, M. (2014). An information system for vehicle navigation in congested road networks. *PNCTM 21*, *3*, 113–116.Google Scholar

13.Simmons, R., Browning, B., Zhang, Y., & Sadekar, V. (2006). Learning to predict driver route and destination intent. *Intelligent transportation systems conference (ITSX 06)*, (pp.127–132).

14.Ye, N., Wang, Z. Q., Malekian, R., Zhang, Y. Y., & Wang, R. C. (2015). A method of vehicle route prediction based on social network analysis. *Journal of Sensors*, *2015*, 1–9.CrossRefGoogle Scholar

15.Quddus, M., & Washington, S. (2015). Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies*, *55*, 328–339.CrossRefGoogle Scholar

16.Luo, W., Tan, H., Chen, L., & Ni, L. M. (2013). Finding time period-based most frequent path in big trajectory data. In *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, (pp. 713–724).

17.Kansal, A., Goraczko, M., & Zhao, F. (2007). Building a sensor network of mobile phones. In *Proceedings of the 6th international conference on Information processing in sensor networks(IPSN '07)*, (pp. 547–548).