Technical Report Documentation Page

Form DOT-F-1700.7

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Development of Micro Wireless Sensor Platforms for Collecting Data on Passenger-Freight Interactions		5. Report Date 6. Performing Organization Code
7. Author(s) Kyle Ying, Alireza Ameri, Ankit Trivedi, Dilip Ravindra, Darshan Patel, Mohammad Mozumdar		8. Performing Organization Report No.
9. Performing Organization Name and Address Electrical Engineering California State University, Long Beach 1250 Bellflower Blvd, Long Beach, CA 90840-8306		10. Work Unit No: 11. Contract or Grant No.
12. Sponsoring Agency Name and Address California Department of Transportation Sacramento, CA 95819		13. Type of Report and Period Covered Final Report 1-Jan-2015 to 31st- Dec-2015 14. Sponsoring Agency Code

15. Supplementary Notes

Supported by a grant from the US Department of Transportation, University Transportation Centers Program

16. Abstract

In this report, we propose an in-node microprocessor-based vehicle classification approach to analyze and determine the types of vehicles passing over a 3-axis magnetometer sensor. Our approach for vehicle classification utilizes J48 classification algorithm implemented in Weka (a machine learning software suite). J48 is a Quinlan's C4.5 algorithm, an extension of decision tree machine learning based on ID3 algorithm. The decision tree model is generated from a set of features extracted from vehicles passing over the 3-axis sensor. The features are attributes provided with correct classifications to the J48 training algorithm to generate a decision tree model with varying degrees of classification rates based on cross-validation. Ideally, using fewer attributes to generate the model allows for the highest computational efficiency due to fewer features needed to be calculated while minimalizing the tree with fewer branches. The generated tree model can then be easily implemented using nested if-loops in any language on a multitude of microprocessors. In addition, setting an adaptive baseline to negate the effects of the background magnetic field allows reuse of the same tree model in multiple environments. The result of our experiment shows that the vehicle classification system is effective and efficient with the accuracy at nearly 100%.

17. Keywords	18. Distribution Statement
anisotropic magnetoresistive (AMR) sensors; vehicle classification; machine learning algorithm	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Page
Unclassified		24	1

Development of Micro Wireless Sensor Platforms for Collecting Data on Passenger-Freight Interactions

Final Report

METRANS Project 14-06

February 2016

Principal Investigator:

Mohammad Mozumdar, Ph.D.

Graduate Students:

Kyle Ying (MS-EE)

Alireza Ameri (MS-EE)

Ankit Trivedi (MS-EE)

Dilip Ravindra (MS-EE)

Darshan Patel (MS-EE)

Department of Electrical Engineering

California State University, Long Beach

METRANS Transportation Center
USC | CSULB

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the accuracy of the data and information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, the California Department of Transportation and the METRANS Transportation Center in the interest of information exchange. The U.S. Government, the California Department of Transportation, and California State University, Long Beach assume no liability for the contents or use thereof. The contents do not necessarily reflect the official views or policies of the State of California, CSULB, or the Department of Transportation. This report does not constitute a standard, specification, or regulation.

ABSTRACT

Traditionally, pavement inductive loop sensors are used to collect real time traffic data for passenger-freight movement in roadways. This method, however, is expensive to install and maintain, and also requires an electronic control unit connected to the induction loop. In the last decade, significant improvements have been achieved in Micro-Electro-Mechanical System (MEMS) sensors domain with respect to size, cost and accuracy. Moreover, extreme miniaturization of RF transceivers and low power micro-controllers motivated the development of small and low power sensors and radio-equipped modules, which are now replacing traditional wired sensor systems. These modules can communicate with other similar modules to build an intelligent sensing network. Due to process miniaturization and low power consumption, these "sensor nodes" can potentially remain functional years. Motivated by these novel advances, we propose a wireless MEMS sensor based passenger-freight interaction detection framework for Intelligent Transportation Systems (ITS). The proposed system is mainly composed of two parts: the sensor nodes that contain wireless Magneto-Resistive (MR) sensors to detect passenger-freight vehicles and an Electronic Control Unit (ECU) to collect and generate the traffic data from sensor nodes' data such as vehicle detection, speed, and most importantly, classification. The sensor's primary function is to classify vehicles, which inherently includes detection. Through the use of machine learning algorithms, ECU can produce correct classification rates nearing 100%. Through the use of multiple sensors, the ECU can calculate and extrapolate the speed and level of congestion of the area where the sensors are installed. Our proposed solution will be significantly more cost effective than the traditional induction loop approach and is scalable to cover millions miles of roadways all over the US.

TABLE OF CONTENTS

Disclaimer iii

Abstract iv

Table of Contents v

Illustrations vi

Tables vi

Disclosure vii

Acknowledgements vii

1. Introduction 1

1.1. Outline 1

1.2. Problem 1

1.3. Scope 3

2. Related Works 5

3. Methodology 7

3.1. Small Scale Setup 9

3.2. Vehicle Classification 12

3.3. Classification Problems 18

4. Results 19

5. Conclusions 21

6. Implementation 22

References 23

ILLUSTRATIONS

Figure 1. Physical and Schematic Representation of Inductive Loop Sensors 3

Figure 2. Proposed Architecture for Smart Road Sensing Networks 7

Figure 3. Anistropic Magneto Resistive Sensor Schematic 9

Figure 4. System Test Bed with Lineup of Radio Controlled Cars 10

Figure 5. Threshold Detection with Vehicle Passing 12

Figure 6. Information Gain Resultant Binary Tree 15

Figure 7. Example Sensor Data Plot and Feature Extraction 17

Figure 8. X-axis Data Clipping due to Sensor Range 18

Figure 9. Graphical Representation of Machine Learning Trained Model 20

Figure 10. Miniature Sensor Node with Wireless Capabilities 22

TABLES

Table 1. Information Gain Calculation Example 15

Table 2. Initial Classification Rates with Various Features 17

Table 3. Classification Rates on Testbed 19

DISCLOSURE

This research project report was funded by a grant from the California Department of Transportation (Caltrans) through the Metrans Transportation Center (METRANS) under a cooperative agreement between the University of Southern California (USC) and California State University, Long Beach (CSULB).

ACKNOWLEDGEMENTS

The authors acknowledge support given from California Department of Transportation (Caltrans) through the METRANS under a cooperative agreement between the University of Southern California (USC) and California State University, Long Beach (CSULB). Authors would also like to thank undergraduate student Ramon Carneiro for helping us during Summer 2015.

1. INTRODUCTION

1.1. OUTLINE

This report is organized as follows: Section 1 talks about the issues with the current state of highway vehicular detection systems and technologies typically associated with such systems. Section 2 discusses in depth, the related classification work of previous papers with similar ideas and concepts for vehicular detection and classification. Section 3 presents our approach on how we decided to design a wireless sensor network, how we tackled the classification problem, and the background to our selected machine learning algorithm. Section 5 provides our results and how the algorithm's classification percentage rates potentially differ from real-world results. Section 5 contains our conclusions and plans for additional research. And lastly, Section 6 discusses our current approach for hardware implementation.

1.2. PROBLEM

Increasing highway traffic and inadequate construction of new highways across the US is causing the congestion level on our nation's roadways to spiral out of control. According to the US Department of Transportation, traffic on surface roads increased by 2% and highway traffic also increased by 33% between 1987 and 1997 [14]. It is estimated that due to the further demand for mobility, traffic will further increase by more than 40% by 2020. Hence, demands for improving and extending the existing road infrastructure, especially in the vicinity of megacities, are becoming a major concern that could cost billions of dollars. By efficiently using existing transportation networks and advanced traffic control management techniques, cost effective and environmentally friendly solutions can be achieved. Thus, there is an essential need for an affordable and environmentally friendly solution in order to maximize the capacity and

efficiency of existing transportation networks. Intelligent Transportation Systems (ITS) provide a solution that reuses the existing transportation network without the need to scrap the whole system. The goals of ITS are to provide an efficient solution for reducing travel time, easing delay and congestion, and improving safety. Advanced sensing, electronic surveillance, and traffic analysis and control are the main technologies employed by ITS to provide real time traffic information to policy makers and users of the transportation system. Moreover, different transportation service providers can use ITS data to monitor, route, and control traffic flow.

The success of these intelligent transportation systems significantly depends on the proper design, installation, and maintenance of the sensor unit of the system. Currently available traffic sensor systems such as: inductive loops, video, sonar, radar, magnetic and capacitive sensors, PVDF wire, and pneumatic treadle, are costly and use electricity from the power distribution network. Current system can cost thousands of dollars for each sensor (video, sonar, and radar) installed on electrical poles [14]. Moreover, costs for road-installed sensors (pavement) such as inductive, magnetic, PVDF wire, capacitive, and pneumatic treadle can span several thousand dollars each. Regardless, in-pavement sensors are still popular, due to their accuracy, ability to provide direct information with very little ambiguity, ability to monitor road conditions (i.e. presence of ice), and the fact that they don't require a human operator. In the US, we have millions of miles of highways but electrical power is not available across all these highways. Hence to collect traffic data in these areas, we need to have a system that would be inexpensive to install and maintain, has a small footprint, and can be deployed anywhere regardless of the direct power supply availability. Motivated by these novel causes, we suggest a sensing system based on a low-power Micro-Electro-Mechanical System (MEMS), which can be cost effective, easy to install, and remain operational without direct power supply for many years.

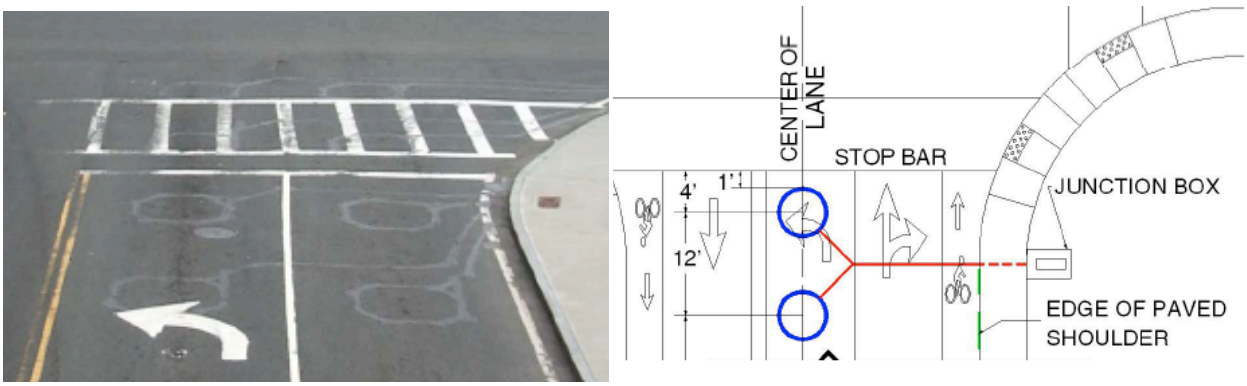

Figure 1. Physical and Schematic Representation of Inductive Loop Sensors

1.3. Scope

Inductive loop technologies are currently used in the United States; Fig. 1, placed during road construction or are sawed and placed in after. These inductive-loop traffic detectors are primarily used for the detection of vehicles; however, more advanced systems can classify vehicles in addition to detection. Regardless, these systems require relatively significant amounts of power for the generation of a magnetic field at relatively high frequencies for general usage. Many suggested systems propose to replace inductive loop detectors include image, acoustic, and infrared sensors to detect and classify vehicles. However, sensors that utilize sound may be adversely affected by Doppler effects due to the movement of vehicles. Image and infrared sensors are even more difficult to implement due to a number of environmental factors such as temperature, atmospheric conditions, and terrain variations that would require adjustments to the sensor system at every new installation. Line-of-sight issues also occur for the aforementioned sensors because of the need for a minimum distance between the sensor and the location where the vehicles would normally pass.

The anisotropic magneto resistive (AMR) sensor has immunity from all of the previous mentioned factors and shares the same basic concept as inductive-loop setups at a much lower

power usage while requiring the use of a much smaller area for operation. AMR sensors are not affected by weather conditions as much as infrared and image sensors. Furthermore, line-of-sight problems are a non-issue due to the close proximity of the sensor to the vehicles. Like inductive-loop sensors, a threshold is typically set such that pedestrians would not trigger detections or classifications. In addition, implementation of an adaptive baseline algorithm [1] would allow us to set our baseline to the background environmental magnetic field in order to use same classification model in multiple environments without the need for readjustment at each new sensor installation location.

Current implementations using AMR sensors for vehicle classification show successful classification rates between 60% and 98%. Most methods utilize a threshold detector to initiate a detection window where the features of a random vehicle are extracted before being put into a classification algorithm to determine that vehicle's class. These algorithms primarily utilize single or multiple novel features to classify most of the vehicles including but not limited to magnitude, Peak-Valley algorithm (PVA) [1], Hill-Pattern algorithms [2], and even Fast Fourier Transform (FFT) [3].

We propose a machine learning approach where the use of the decision-tree machine learning algorithm J48, an implementation of Quinlan's ID3, to generate our classification model. The research suggested by Lan [4, 5] uses an alternate machine learning algorithm called improved support vector machine (SVM). However, implementation of such a supervised learning model would be computationally inefficient for use in low-power systems. By training our classification model externally, using data gathered from a range of vehicles, we produced nearly 100% classification rates using the simplest of features gathered from a 3-axis AMR sensor in our test environment.

2. Related Works

The topic of vehicle detection and classification is not new. Many previous research groups have tackled this problem using their own methodologies to determine and produce an effective classification algorithm. Due to the relatively low power usage of AMR sensors that detect changes in the earth's magnetic field, most new research on vehicle classification utilizes these sensors in favor of the old induction loop sensors. The old technology uses relatively large amounts of power with the need to carve up large sections of the road for installation in addition to the potentially long power and communications wires to a nearby processor for detection and classification.

The work detailed in [6] and [7] uses two AMR sensors attached to a microcontroller. Their algorithm is based on a decision tree with the data obtained by extracting various features from the signal received from the AMR sensors. The extracted features include the calculation of the relative vehicle length, average vehicle energy, and the hill-pattern algorithm number of peaks. Their results show 95% classification rates for motorcycles as opposed to the relatively low 65-82% classification rates for other cars. In terms of vehicle classification algorithms, most utilize a method similar to this in that most topics attempt to find a single feature or set of features that can clearly differentiate multiple vehicles easily by observation. However, once there are more than three classification types, this method produces bad classification rates for at least one of the classes because many of the features begin to overlap and it is not intuitive to determine the correct thresholds for an increasing number of classes. For example, the process in [8] uses hill-pattern and Average-Bar as their primary selected features with four classifiers: passenger vehicles, vans/SUVs, trucks, and buses. Their classification rates range from 70-100% with the highest rate being for passenger vehicles. However, lower sampling rates change the

classification rates and most of the misclassifications occur with the larger three vehicles. Another similar method proposed in [9] uses normalized vehicle length, average energy, and hill-pattern peaks as their features to classify five different types of vehicles. Again, classification rates range from 66-100%, this time with the classifications favoring motorcycles and buses. We notice that with the intuitive methods in searching for the classification thresholds, results almost always tend to favor some vehicles with the rest having 10-20% lower classification rates.

The features are processed in a Filter-Filter Wrapper model using machine learning algorithms [10]. The model produces the differentiated features between the possible classifiers. Next, clustering SVM is then used to create the classification algorithm. The paper states that particle swarm optimization is also used to optimize the parameters of the SVM algorithm. According to the paper, the 10-fold cross validation results of 460 training samples produce a 99% overall classification rate. However, real world testing resulted in rates ranging from 91-100%. Even-though the real-world results are lower, these are rates are much higher than the previous non-machine learning methods. The improved SVM method described in [5], without the optimization used in [10], yields results around the range of 89-93%. An intelligent neural network classifier is proposed in [11] for vehicle classification. Overall, this method produced classification rates of 93% with individual classifications ranging from 89-100%.

We determined that most of the aforementioned machine learning algorithms would be very power-intensive due to computational requirements. Algorithms utilizing SVM or neural networks would athrequire a significant amount of computational time that could otherwise be spent in a microprocessor's low power mode or sleep. In addition, features such as FFT, area calculation, and PVA would need to use mathematic functions that would take a significant amount of time to compute. Our proposed method attempts to create a sensor network system

that could be extremely power efficient all the while using machine learning algorithms to generate a high classification rate model using the bare minimal amount of processing needed but also free from external supplementary computational devices such as a PC. Our proposed method utilizes several easy to extract features at a 75Hz sampling rate. We obtained greater than 98% classification rates using simple feature extraction along with the machine learning decision tree generating algorithm J48.

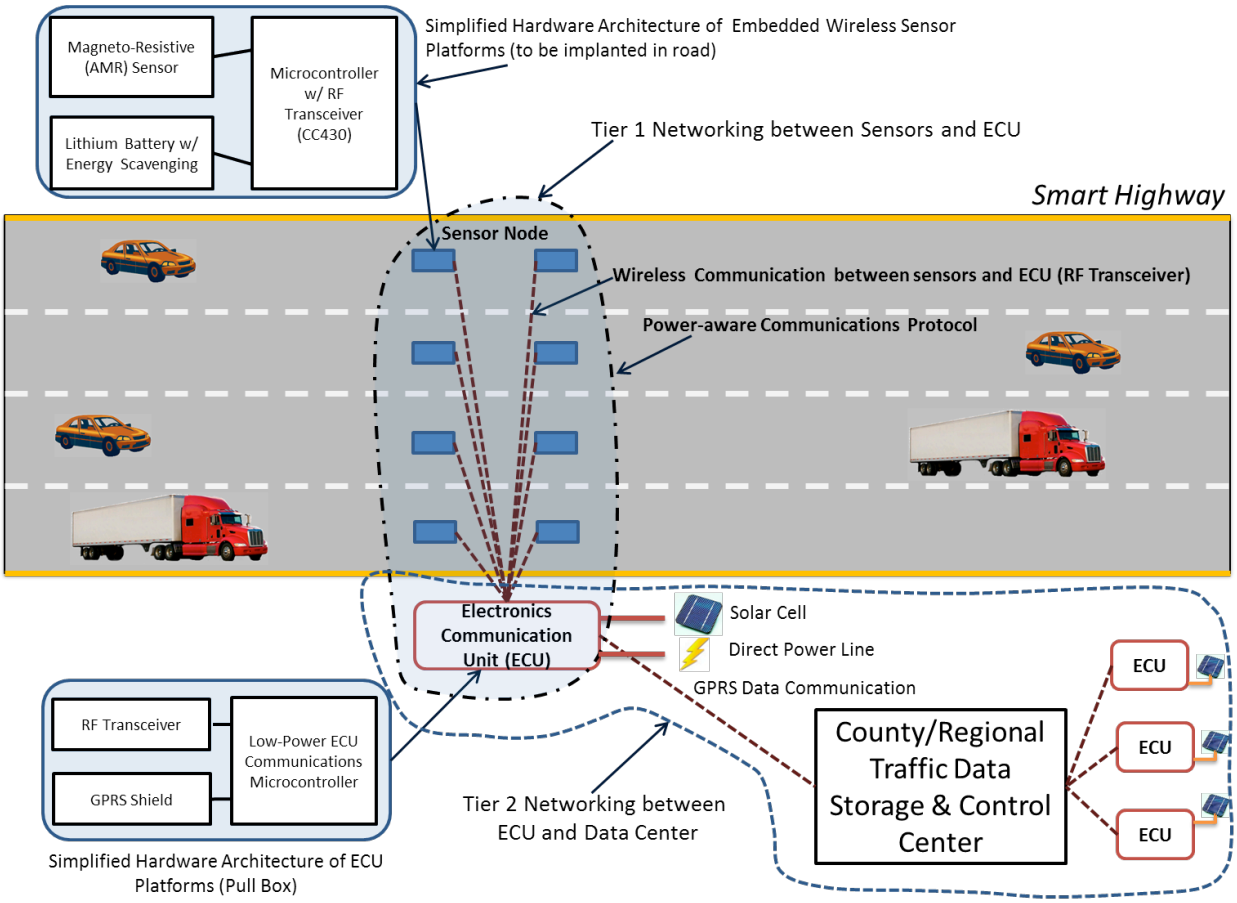

Figure 2. Proposed Architecture for Smart Road Sensing Networks

3. METHODOLOGY

A high-level system architecture of our proposed method is shown in Fig. 2. The system is composed of sensor nodes implanted into the pavement of the road. In each lane, two sensor

nodes will be implanted to detect vehicle presence, speed, and type. Each sensor node will be wirelessly connected with an ECU in a STAR networking topology. The ECU will be an on-site wireless transceiver and controller that will perform traffic data aggregation from the deployed sensor nodes. Furthermore, the ECU will be powered mainly from photovoltaic cells with a compact auxiliary battery. If a direct power supply is available, the ECU can be connected directly to an AC power line. Whenever a sensor node detects any vehicle event, such as arrival or departure, the node will report the event to the ECU with classification information. Once the ECU receives the traffic event packets from the sensors, it will store the data locally and after a certain interval, the ECU will transmit aggregated traffic data to the regional/county-wide transportation data center.

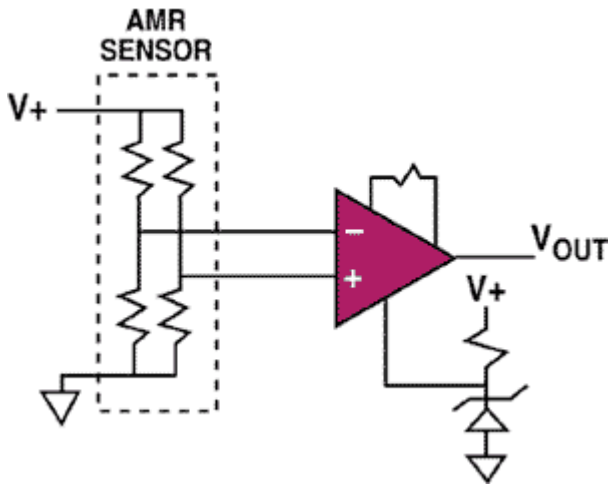

Figure 3. Anistropic Magneto Resistive Sensor Schematic

3.1. Small Scale Setup

Our in-node approach utilizes TI's CC430 microprocessor to process the data from a magnetic field sensor in addition to transmitting the classification data to the receiver or ECU. The sensor we used is a Honeywell HMC5883L magnetometer sensor utilizing AMR technologies. Most AMR sensors work using a Wheatstone bridge configuration in an operational amplifier as seen in Fig. 3. The bridge elements are variable resistors that alter with changes to the magnetic field. Converting this output voltage from the Op-amp using an analog-to-digital converter (ADC) allows for the microprocessor to work with data of the relative magnetic field. The HMC5883L supplies the magnetic field data on 3-axes in a 12-bit integer form from its integrated ADC for simple implementation, where most of the work in implementation will solely be based on the classification aspect. The sensors we are using are set to a sampling rate of 75Hz in continuous mode for accurate and consistent data.

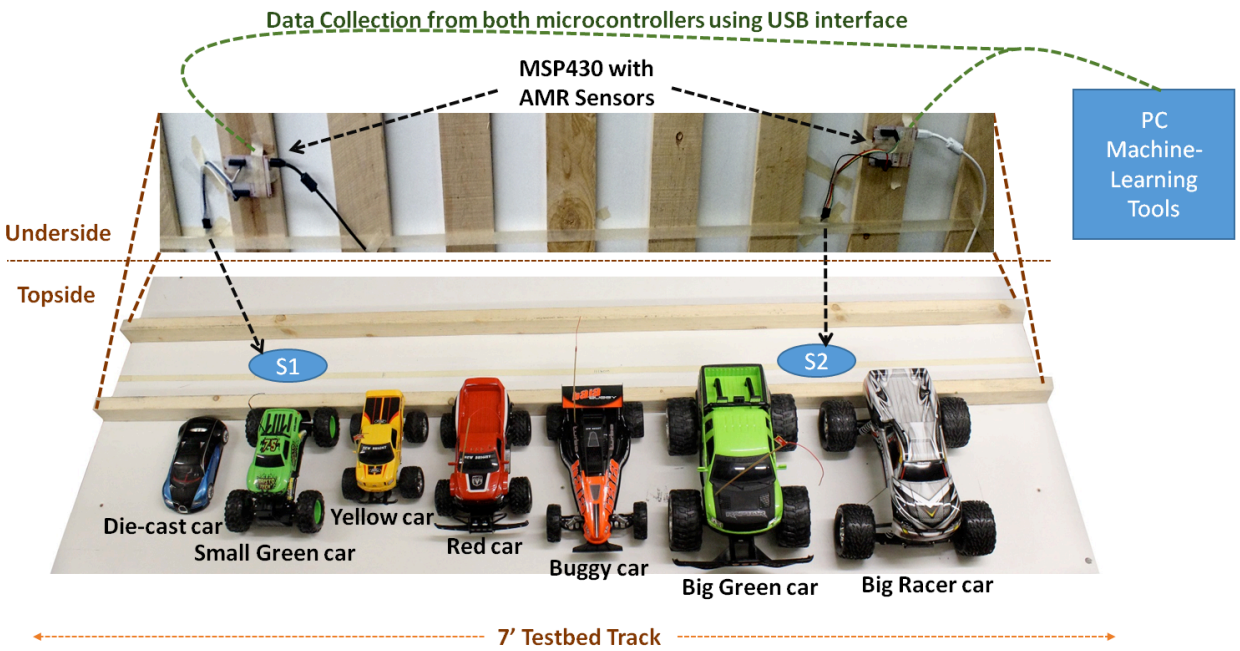

Figure 4. System Test Bed with Lineup of Radio Controlled Cars

Our small-scale test track with radio controlled (RC) cars that we use as an analog to real world vehicles are shown in Fig. 4. The test bed is 7' long and is composed of a wooden fence section underneath for rigidity and a large plastic sheet above to provide a smooth surface for the RC cars to travel over the sensors. Two wooden guide rails are used to keep the RC cars along the paths of the AMR sensors and to act similarly to marked lanes in the real world. The RC cars tend to stray from driving straight at higher speeds, likely due to poor wheel alignment. This causes the data gathered between the sensors to fluctuate wildly, thus requiring the railing. The points marked with the blue circles in Fig. 4 signify the locations where the two Honeywell AMR sensors are located, each attached to a microprocessor used for in-node classification.

To determine when a vehicle passes over our AMR sensor, we first have to determine the background magnetic field for where our sensor is placed; this is the baseline. In order to use our sensor system in multiple environments, the ability to reset baseline values in each new

environment is crucial. The adaptive baseline idea suggested by [1] gives us this ability. Once the baseline is determined, the values are subtracted from each raw axis data to effectively zero out the background magnetic noise floor. This offset level should be different at each sensor system providing a normalized dataset that allows us to use a single classification algorithm for multiple sensors.

$$mag = |x| + |z| \qquad (1)$$

Using the zeroed data on each of the 3 axes, our "magnitude" is computed by (1). This magnitude equation was chosen for computational efficiency as it needs to be calculated at each sample. The x-axis is parallel to the vehicle travel direction while the z-axis is pointing up out of the test bed. Y-axis information was not used for detection as it would likely pick up vehicles in adjacent lanes and cause false positives depending on the ferrous make-up of the adjacent vehicle.

Next, a threshold must be determined such that any ferromagnetic material that causes the magnitude to rise past will start the detection window for the vehicle. The threshold detection value should be adjusted such that only vehicles would be detected while lighter ferrous materials and other sources of noise will stay below the selected threshold. Fig. 5 shows how a passing vehicle would look like using magnitude w.r.t time. During the detection window, the processor initially extracts raw data used by the PC to calculate many vehicle features such as the minimum, maximum, mean, range, standard deviation, and the FFT of each of the axes' data. The best features according to the algorithm are then implemented in-node with no further assistance from the PC. With regards to the features used in this report, the Cartesian minimums, maximums, means, and ranges were shown to provide the highest information gains (discussed in the next section) and thus the higher computational features such as standard deviations and FFTs were

discarded to preserve power in ultra-low power applications. The features are calculated at the end of each detection window, therefore most of the processing and resultant power usage is at the end of each window when vehicle classification actually occurs.

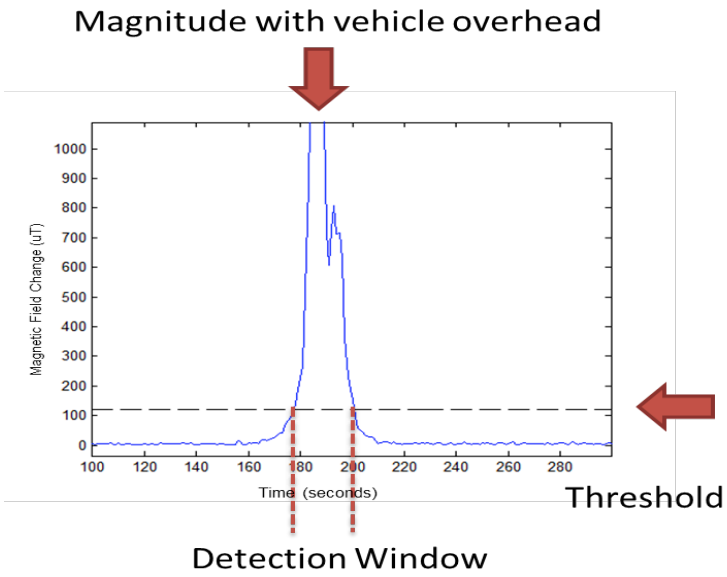

Figure 5. Threshold Detection with Vehicle Passing

3.2. VEHICLE CLASSIFICATION

In order to classify the types of vehicles, we use a machine learning algorithm called J48 [12]. J48 is an open-source Java implementation of Quinlan's C4.5 [13] algorithm, which in itself is an extension of his ID3 decision tree algorithm. J48 is primarily used in Weka [12], a free Java program used for machine learning developed by the University of Waikato. As described by the name, the J48/C4.5 algorithm outputs a decision tree based on a number of features or attributes used to train the final output tree model. The reason that we used a decision tree based machine learning algorithm is due to its simplicity for implementation as well as its computational efficiency. A decision tree can be easily implemented using nested if-loops in code and would be relatively fast for classification. Depending on the features used, most of the computational time

might be spent during the feature calculations used in the decision tree than in the classification tree itself.

ID3 and C4.5 utilize two concepts to determine the most optimal tree. The first is entropy and the second is information gain. The first entropy equation is described in (2), which is a recursive equation used in two separate levels of the information gain calculation. In this first case, |C| is the number of classes, |S| is the total number of training samples, and lastly $|C_i|$ is the number of training samples of that specific class.

$$H(S) = \sum_{i=1}^{|C|} \frac{|C_i|}{|S|} \log_2 \left(\frac{|C_i|}{|S|} \right) \qquad (2)$$

Next, a second entropy calculation needs to be performed as described in (3). Here |*a*| is the total number of different attributes or features used to identify a class which in our case, we select the extremities in magnitude. $|T_i|$ is the term for the number of elements of a specific attribute, and lastly |T| is the number of total attributes in all the samples or subset. In addition, this second entropy calls on the first entropy calculation (2). In this second case, |C| is the number of states that a specific attribute can be in and is typically two, one for when the state is the attribute and the other for when it is not. |S| is equal to $|T_i|$ and $|C_i|$ is equal to the number of samples that meet the criteria for a specific attribute and state.

$$H_a(S) = \sum_{i=1}^{|a|} \frac{|T_i|}{|T|} Info(T_i) \qquad (3)$$

Lastly, by subtracting (3) from (2) we can finally generate the information gain with respect to the classes and a single attribute (4).

$$IG(S) = H(S) - H_a(S) \qquad (4)$$

This equation is important because calculating the highest information gain for each attribute suggests which attribute would provide the best forks in our decision tree. At each fork, the information gain needs to be recalculated with the remaining samples on that branch of the tree. For example, if we start off with 100 samples and the highest information gain splits the samples into groups of 40 and 60, but one group contains more than one possible class, another fork is created below where information gain needs to be recalculated. The tree continues to calculate information gain for each fork until each branch reaches a single possible class at which point our decision tree model is completed.

In the case of numerical values, the whole set of numbers are first sorted from least to greatest, then are separated into two groups. The information gain is calculated between the two groups of numbers with the potential threshold value set at the split point of the sorted set. The split point can be chosen at any number between the two groups. Therefore, if the attribute set has n non-repeating numerical elements, then $n-1$ information gain calculations need to be completed. The threshold with the greatest information gain and its attribute is selected to be the conditional at that specific fork. All numbers greater than the threshold take one branch, otherwise they take the other. An example of the information gain values calculated for numerical attributes can be seen in Table 1. This example uses four attributes and the table shows that the highest information gain attribute is selected to be the attribute used to split the resulting binary tree at that respective node. In this particular case, the tree is built to the left side until it reaches the 5th node, then reverses to the last incompletely classified node (1st) before continuing to flesh out the tree to the right. The resultant tree for these information gain values is shown in Fig. 6.

Table 1. Information Gain Calculation Example

Node Num.	Information Gain Attribute				Best Attribute
	Min Y	***Max X***	***Mean Z***	***Range Z***	
1	0.9527	0.8002	0.9031	0.9868	***Range Z***
2	0.3339	0.9707	0.3978	0.3846	***Max X***
3	0.3060	0.8411	0.2155	0.3068	***Max X***
4	0	0.1735	0	0.0379	***Max X***
5	0.7108	0	0.7108	0.0570	***Min Y***
6	0.9317	0.1864	0.7574	0.4976	***Min Y***
7	0.2047	0.1601	0.3717	0.4249	***Range Z***
8	0.1386	0.4467	0.8798	0.0084	***Mean Z***
9	0.5724	0.3189	0.9625	0.8676	***Mean Z***
10	0	0	0.0066	0.6627	***Range Z***

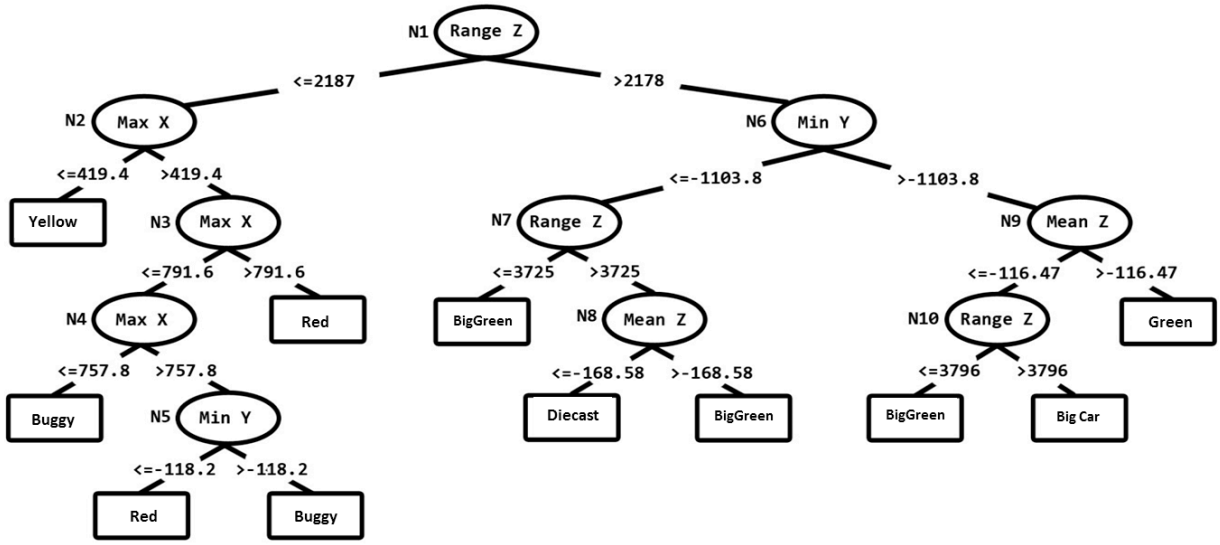

Figure 6. Information Gain Resultant Binary Tree

At this point there are multiple methods to determine how statistically accurate the output model is. One method is a percentage-split separating two thirds of the data for training and then generating the model using the final third for verification purposes. Another method is cross-validation. Using the cross-validation method in Weka for verification uses the whole training

sample set for generation of the model instead of two-thirds. The user stipulates some number of *k*-folds where *k* is the number of new models generated during validation. Weka splits the training data into *k* equal sets then uses *k-1* of those sets for generation of a new model and the remaining set for validation; this is repeated *k* times. The statistics generated after validation are averaged to generate a percentage for classification accuracy without changing the original model generated before cross-validation while using the whole sampling set.

By limiting the number of features or attributes available to Weka's J48 algorithm, we can generate even more efficient and accurate decision tree models according to cross-validation. The current implementation using information gain across a large number of attributes does not always generate the highest percentages during validation because the training process only has discrete information given to it during the validation process. The type of validation process chosen is important because it can potentially limit the amount of data given to the validation algorithm. A percentage split of the training model samples can be used for validation (percentage-split) or the whole set of samples can be used through generating additional split models and verifying after the fact (cross-validation). Regardless of the validation method, the output decision tree model generated is still the same even though the validation classification rates can be different. In addition, by forcing a limit on the number of attributes used to train the algorithm, we can determine which features are most important to the model in addition to eliminating features that may be computationally inefficient and may not be as useful as other features. This can be done manually using an intuitive process; however, we brute forced all combinations of *n* choose *k* attributes to determine the highest rates using as few *k* attributes as possible. A 10-fold cross-validation method was used and the resulting classification rates shown in Table 2.

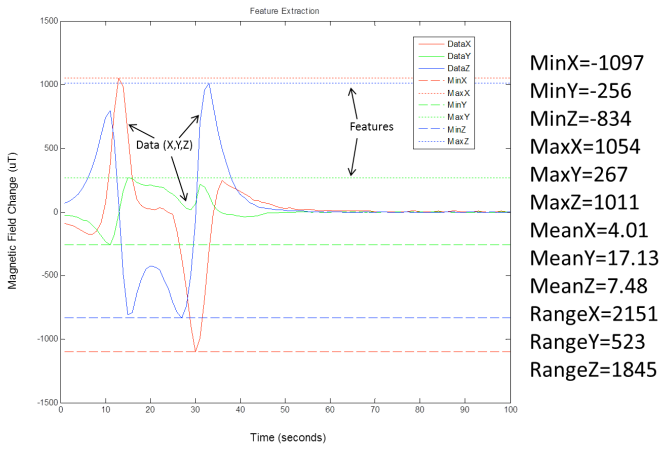

Figure 7. Example Sensor Data Plot and Feature Extraction

Table 2. Initial Classification Rates with Various Features

Initial Cross-Validation Percentages	
Number of Features (Attributes)	***Accuracy***
Two Features (*minx*, *maxx*)	94.00%
Three Features (*minx*, *maxx*, *maxz*)	98.00%
Four Features (*miny*, *maxx*, *meanz*, *rangez*)	98.86%
Five Features	98.86%

The total number of features was n=12, encompassing the *minimum*, *maximum*, *mean*, and *range* of data from all 3 axes from the AMR sensor during the vehicle detection window. An example of the data gathered during the detection window can be seen in Fig. 7. The detection window length is the number of samples of the longest training sample that stayed above the threshold. This naturally means that the smaller RC cars with lower ferrous content would be padded with low values towards the end of the detection window. We tried all combinations between two and five features to keep the resulting decision tree small. As we can see from the results in Table 2, we do not increase our classification rates above four features and thus decided to use those four features to generate the initial classification trees to keep the tree area small with fewer branches.

3.3. CLASSIFICATION PROBLEMS

Although these classification rates were high according to the cross-validation percentages, implementation of these features and their corresponding decision tree resulted in vastly different outcomes in real-world testing. We found out through testing that the *minx* feature, which is the minimum value of the magnetic field on the x-axis during the detection window, was actually similar for two different vehicles, the die-cast car and the big racer. As shown in Fig. 8, the *minx* is nearly identical between two cars because the results were clipped due to the limitations of our AMR sensor. The die-cast car has a very high metallic content in addition to a very low ride height. The clearance between the die-cast and the AMR sensor is easily less than one inch causing the clipping. The big racer, on the other hand, also generates a large magnetic field due to its large inductive motor generating a huge magnetic field. These two cars generate essentially the same minimum x-axis values even though the shapes are significantly different.

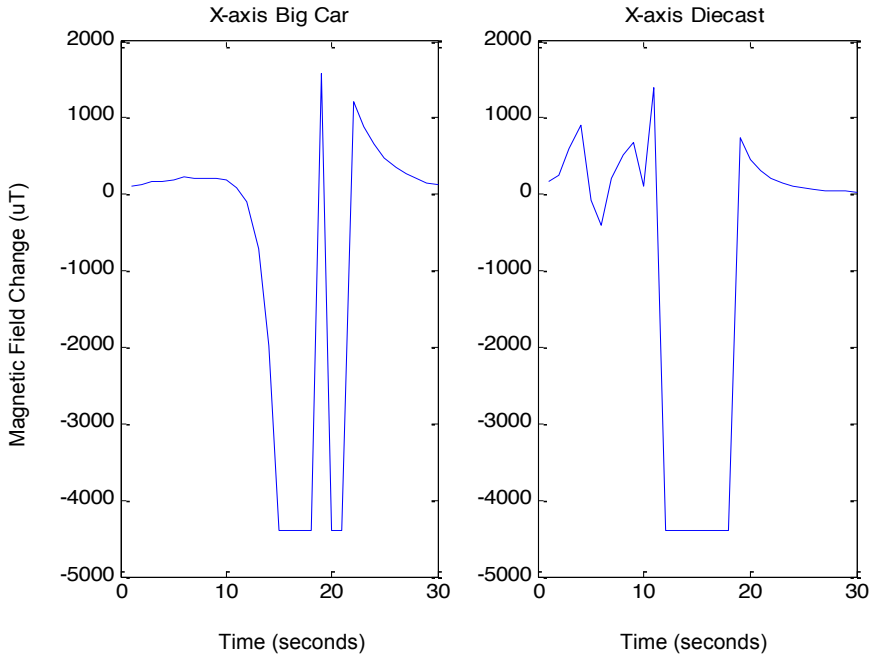

Figure 8. X-axis Data Clipping due to Sensor Range

4. RESULTS

We implemented the output decision tree models using the J48 algorithm in Weka along with our newly selected features from Table 3. We tested seven different classification algorithms in order to determine whether the classification rates using 10-fold cross-validation match the rates found on our test bed. A total of 30 runs were made on the test bed for each feature set with all 7 cars going over both sensors using the setup in Fig. 4. This effectively means 420 feature sets per algorithm were extracted between both sensors, totaling for 2940 real world sensor tests.

Table 3. Classification Rates on Testbed

Real world Classification Percentages		
Number of Features (Attributes)	***Cross-Val***	***Real-world***
Two Features (*minz*, *maxx*)	97.00%	95.71%
Two Features (*maxx*, *rangex*)	96.74%	96.67%
Two Features (*minx*, *rangex*)	98.00%	95.95%
Three Features (*minx*,*maxx*,*meany*)	98.00%	100%
Three Features (*miny*,*minz*,*maxx*)	97.86%	98.10%
Three Features (*minx*,*maxx*,*rangey*)	97.86%	99.52%
Three Features (*minz*,*maxx*,*rangex*)	97.86%	98.81%

We see immediately from the results of Table 3 that with the exception of one algorithm, the real world classification rates match the cross-validation percentages pretty closely (~2%). There were additional combinations of features that produced better cross-validation results but were omitted in favor of lowering the number of features that need to be processed. We can also tell from this table that generally, with a greater number of features used in the algorithm, the real-world classification rates tend to be higher than the suggested cross-validation percentage. Another observation is with fewer features, the misclassifications tend to focus on one or two of the seven classifiers. With three features and its higher classification rates, the misclassifications are largely random with no obvious pattern.

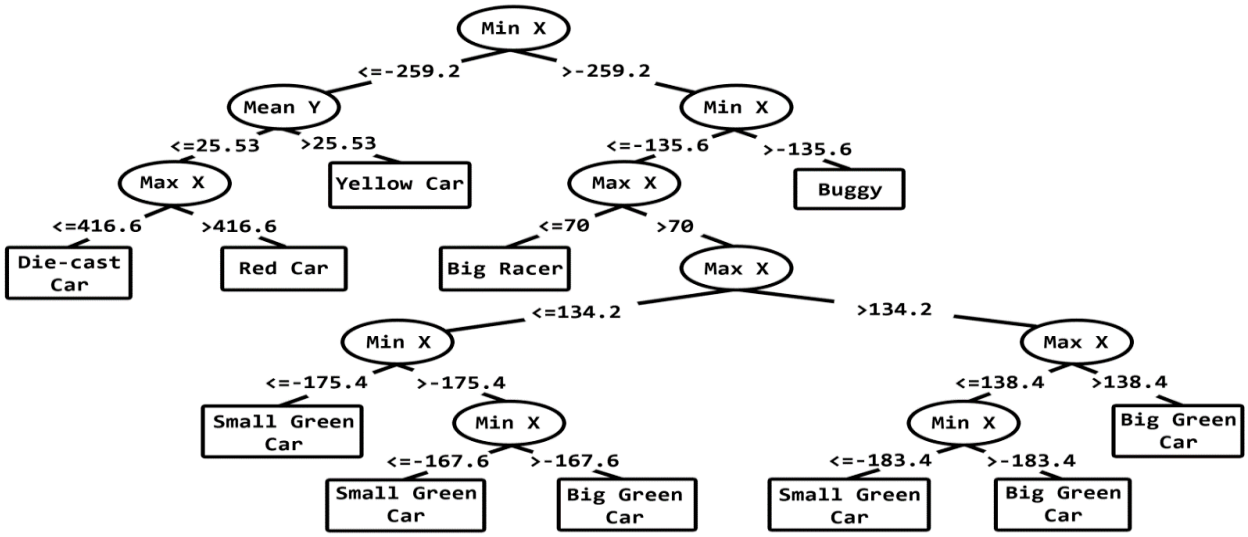

Figure 9. Graphical Representation of Machine Learning Trained Model

Now, we take a look at the resulting decision tree with the best overall testbed classification rate. Fig. 9 is the resulting output decision tree generated from the J48 algorithm based on the information gain splits. Again, the highest numerical information gain determines which feature and what threshold value gets decided to be at a certain node. The algorithm initially works its way down the left hand side of the tree until a single classifier is left then reverses back up the nodes until there are nodes with multiple classes remaining. The process is repeated until each branch has no more than one classifier.

This tree is the best to implement due to its simplicity and the fact that its size is small. With the addition of more features, the processing needed to calculate for the features and the potential size increase of the tree would result in more computational time. We found out that even with two features the size of the tree is still roughly the same therefore the addition of one feature for better results made sense.

5. CONCLUSIONS

The results shown in this report indicate that with the J48 machine learning algorithm, we can achieve extremely high vehicle classification rates given enough training samples to create a suitable decision-tree model. Even though testing was completed utilizing small-scale RC cars, this proof of concept would likely scale to the real-world vehicles as the only difference is the magnitude and length of the magnetic field generated from larger vehicles. In addition, with our focus on efficiency, this method can easily be implemented on wireless sensor networks deployed in the real world. Once the model is generated and programmed into a microprocessor with the correct thresholds, the simulated cross-validation results should be similar to the testbed results as we have shown.

Tradeoffs taken in this report such as computational and power efficiency with regards to classification accuracy will need to be further investigated in real world applications. The features selected in this report (Min X, Range Z, etc.) were selected due to high computational efficiency given the highly accurate classification results. Novel features such as hill-pattern counters and peak-valley algorithms require more computing power compared to simple Cartesian minimums and maximums. Given that most non-machine learning algorithms utilize novel features in their implementations, additional research into these features for this algorithm may produce even better results. The highly diverse range of vehicles on the road currently definitely surpasses seven distinct types and will likely require the use of these novel features to improve classification accuracy in the real world.

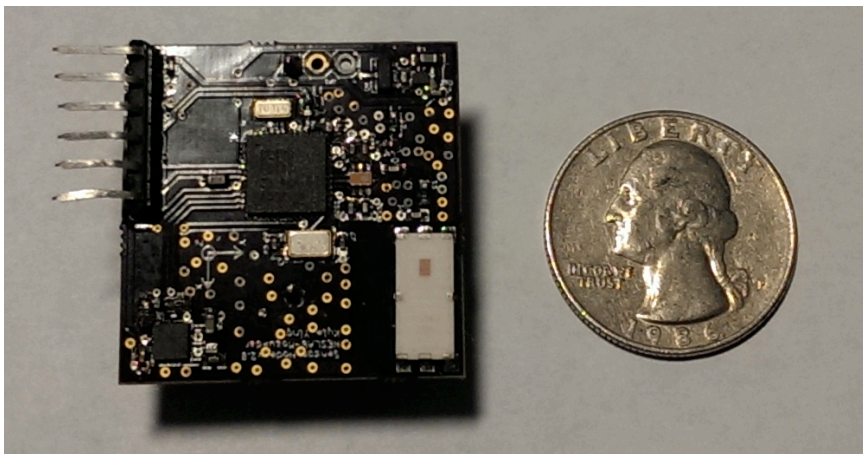

Figure 10. Miniature Sensor Node with Wireless Capabilities

6. IMPLEMENTATION

Utilizing the results shown with the J48 machine learning algorithm, we developed a system that utilizes the accuracy given by the resultant model to create a highly efficient sensor node. This node includes a microprocessor (CC430), an AMR sensor (HMC5883L) communicating through i2c, and an antenna needed to transmit and receive data. The firmware is written in C using TI's Code Composer Studio suite. Fig. 10 displays the developed prototype sensor node with a form factor of 30mm x 30mm or roughly the size of a quarter. This prototype is built for button cell battery operation in mind; however with long-term operation, the depth and volume of the unit will change depending on the size of the battery. This sensor alone is sufficient enough to detect, classify vehicles, and transmit or receive data between itself and the ECU or another sensor in the network. Furthermore, with the built in transceiver, the sensor node can also act as a simple ECU receiver as well. This sensor can output data through Serial UART and any application that can parse the serial data such as MATLAB can easily use the sensor information to calculate speed and congestion through the use of multiple sensors. However, with more complicated designs where more computational power needed, a larger, more powerful ECU with a stronger microprocessor may be necessary for niche applications.

REFERENCES

[1] Z. He et al., "A vehicle detection algorithm based on wireless magnetic sensor networks," in IEEE Int. Conf. on Information Science and Technology, Shenzhen, China, 2014, pp. 727-730.

[2] S.Y. Cheung et al., "Traffic measurement and vehicle classification with a single magnetic sensor," UC Berkeley, CA, Rep. UCB-ITS-PWP-2004-7, Sep. 2004.

[3] B. Yang, Y. Lei, "Vehicle detection and classification for low-speed congested traffic with anisotropic magnetoresistive sensor," IEEE Sensors J, vol. 15, no. 2, pp. 1132-1138, Feb. 2015.

[4] J. Lan, Y. Shi, "Vehicle detection and recognition based on a MEMS magnetic sensor," in IEEE Int. Conf. on Nano/Micro Engineered and Molecular Systems, Shenzhen, China, 2009, pp.404-408.

[5] J. Lan, et al., "Vehicle detection and classification by measuring and processing magnetic signal," Measurement, vol. 44, no. 1, pp. 174-180, 2011.

[6] S. Kaewkamnerd et al., "Vehicle classification based on magnetic sensor signal," in IEEE Int. Conf. on Information and Automation, Harbin, China, 2010, pp. 935-939.

[7] S. Kaewkamnerd et al., "Automatic vehicle classification using wireless magnetic sensor," in IEEE Int. Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Rende, Italy, 2009, pp. 420-424.

[8] I. Jolevshi et al., “Smart vehicle sensing and classification node with energy aware vehicle,” in IEEE Int. Conf. on Information Technology Interfaces, Dubronik, Croatia, 2011, pp.409-414.

[9] J. Chinrungrueng, S. Kaewkamnerd, “Wireless magnetic sensor network for collecting vehicle data,” in IEEE Sensors Conf., Christchurch, New Zealand, 2009, pp. 1792-1795.

[10] Y. He et al., “Vehicle classification method based on single-point magnetic sensor,” Procedia - Social and Behavioral Sciences, vol. 43, pp. 618-627. 2012.

[11] W. Zhang et al., “A distributed threshold algorithm for vehicle classification based on binary proximity sensors and intelligent neuron classifier,” Journal of Information Science and Engineering, vol. 26, pp.769-783. 2010.

[12] R. Bouckaert et al., (2014, Dec 16). WEKA Manual for Version 3-7-12 [Online] Available: www.cs.waikato.ac.nz/ml/weka/documentation.html

[13] J.R. Quinlan, C4.5: programs for machine learning, San Francisco, CA: Morgan Kaufmann Publishers, 1993.

[14] R.L. Gordon et al., (1996). TRAFFIC CONTROL SYSTEMS HANDBOOK-REVISED EDITION 1996 (No. FHWA-SA-95-032).