Car Tracking With Encryption

Yogesh Dhavale
33 Schubert St.
Binghamton, N.Y. 13905
(313)-413-8540
ydhaval1@binghamton.edu

Vamshi Kolanu
2 Bellevue Ave
Binghamton, N.Y. 13905
(607)-768-6980
vamshi.kolanu@gmail.com

Joe Chamish
13 Clarke St.
Binghamton, N.Y. 13905
(646)-644-0562
jchamis1@binghamton.edu

Aleksandr Gorbachev
42 vestal Ave
Binghamton, N.Y. 13905
(740)-590-7587
menus12@mail.ru

ABSTRACT

With the increase in population of our world over the last few decades, daily commute has become very crucial issue. In most of the modern counties, people prefer their personal vehicle for daily commute. According to United States Bureau of Transportation statistics, there are 255 million registered vehicles on road and out of which there are 183 million are classified as light duty vehicle. With such a huge number of vehicles on road, traffic has become very obvious problem in major cities around the world. There is another very crucial problem which has however, not given much attention to. And that problem is finding a parking spot!

Parking your vehicle has not kept up with today's technology and there's still a lot of wasted time looking for a parking spot. However, there hasn't been a system designed like ours because most parking don't find it practical include technology. The reason that technology hasn't been incorporated into parking lots is because of the high price of the sensors. Since, the price of sensors has drop dramatically it has become practical to install sensors to improve our daily lives.

In this paper, we assess the effectiveness of the sensors and use encryption to secure the sensors data. That sensors provide the website with its content of spots being available or occupied. We used the hardware built into the sensor to gage if there's a car or not and the data from the sensor gets sent to a base station. The base station forwards the serial data to the server which displays a webpage. The webpage has each car's location with a unique id and will be green if the space is free, otherwise it will be red.

CCS Concepts

• Information Sensor → Base Station • Forwarding Serial Information → Server Parsed Information to Website.

Keywords

Car tracking; Wireless sensor networks; Embedded systems; Wireless security; TinyOS; Encryption ; TelosB; Wireless motes.

1. INTRODUCTION

When looking for a parking spot in a large parking lot it could take a lot of time to find a opening parking space. The traditional way when looking for a parking spot is searching each row and column until the driver find a spot. This approach is linear because the driver starts at the beginning of the parking lot and searches until he finds a spot or until he reaches the end of the parking lot. In the case that there isn't a spots in the parking lot the driver would have wasted his time searching through the whole parking lot. We thought that there had to be way to solve this problem using sensors and integrating it with a graphical user interface.

Our solution will reducing the amount of time searching for a parking spot because we implemented series of sensors and display the information on webpage. The webpage will be viewed by user who are looking for a parking spot and it will greatly reduce the amount of time for them to find a parking spot. Our approach is different compared to traditional parking lot sensor system. In a traditional parking lot sensor system they have a display before the user enters the parking lot, so the user can see an open spot on the display and drive to the open parking space. There's a major problem with this method because if a driver is currently going to the open parking space and another driver pulls up to the display they will think that the spot is open. When in reality the open spot is about to be occupied. This is a timing issue because we have one driver heading to the open spot, while we have another driver thinking that the spot is open. Since, there's only one display the second driver would have to go back to the display to find another open spot or just search for a spot without the display. We thought of this problem and decided to develop a webpage so that if this problem would occur the second driver would just have to look at the website for another free parking space.

2. DESIGN

The approach we took in designing our system was to divide it into four modules and those modules are parking lot sensors, base station, server, and single webpage. We decided to use this

modular design because it made it easier for each person to work on each part without affecting the other module. Then we just need to connect the modules together with standard protocols like HTTP, serial forwarding, and radio transmission.

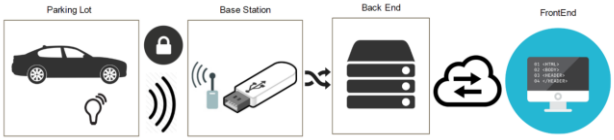

Figure 1. Block diagram of car tracking with encryption

3. IMPLEMENTATION

The implementation of "Car Tracking with Encryption" is four-fold. The Sensor mote, Base station, Back-end, and Front-end. Following sections explain the implementation of each part.

3.1 Sensor motes

The sensor mote is very important component of the system. The key responsibility of the sensor mote is to detect the presence of a car in the parking lot and send this information over to the Base station. The sensor mote with the help of sensor detects whether a car is parked over a particular parking spot or not. In this project, we used TelosB mote as a sensing device. For detecting the presence of a car we can use different sensors such as proximity, infra-red, etc. We decided to use Hamamatsu 1087 photodiode sensor because this sensor is on-board on TelosB mote and it serves our purpose.

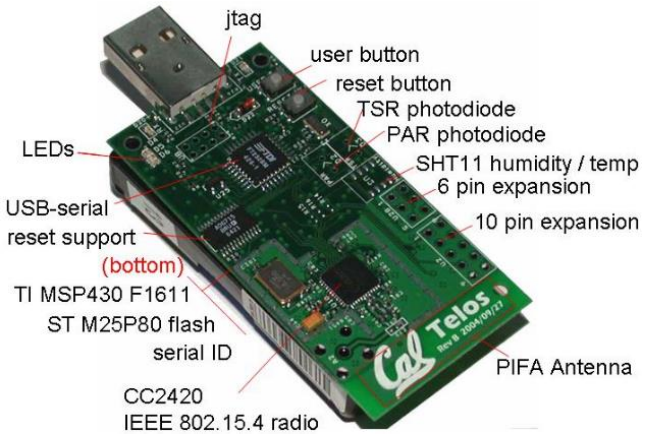

Figure 2. TelosB mote

The notion behind using photodiode is that when a car is parked over a sensor mote, it blocks the light from falling on the photodiode. The program running on TelosB detects this to indicate that a car is parked over a particular parking lot. When a car is drove away, the sensor detects by sensing the light that car is no longer parked over. The sensor mote sends the information about the presence of a car to Base station wirelessly. The sensor mote program is developed using TinyOS modules and components. The following figure shows the module interface of the sensor mote program.

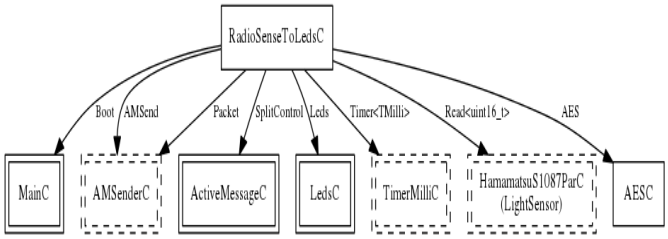

Figure 3. Module interface of sensor mote program

The table below gives a brief details about each module used in sensor mote

Module	Description
MainC	Performs the boot up operation
AMSenderC & ActiveMessageC	Prepares a payload and transmits it across to Base station
LedsC	Indicates the transmission activity on TelosB mote
TimerMilliC	Generates 1 second periodic event to read the photodiode value
HamamatsuS1087ParC	Reads the photodiode value
AESC	Encrypt the message using 16 byte encryption key

Table1. A brief description of each module used in sensor mote.

The data to be sent from a sensor mote to the base station is 3 byte long. The first two byte indicates the node id of the sensor mote which is set while downloading the code on TelosB and the third byte indicates the presence of a car. It would be either 0 or 1. 1 means parking lot is available and 0 means parking lot is not available.

A notable feature of sensor mote is that the communication between sensor mote and base station is encrypted based on AES128 encryption standard. Today, so much research is going on wireless sensor networks and its application in real world and security has been one of the important aspect of the research. As a part of understanding this aspect of wireless sensor network, we decided to use encryption between sensor mote and base station communication. The sensor mote encrypts the transmitting message as per AES128 standard 16 byte encryption key. The same key is used at the base station to decrypt the received message. Thus, the length of message sent to base station is 16 byte. So, for each parking lot a dedicated sensor mote can be used to indicate the availability of it. In our implementation, we decided to use 3 sensor mote and each one is sending to base station its node ID and the availability of the parking lot.

3.2 Base Station

The base station program runs on TelosB mote. It is responsible for receiving the encrypted message from the sensor motes.

It received 16 byte encrypted message from all sensor motes. After receiving each message, it decrypts it using the same 16 byte AES128 standard encryption key used by each sensor mote. It is connected to the workstation where the back-end program is

running over a serial port. The module interface of base station program is as shown below.

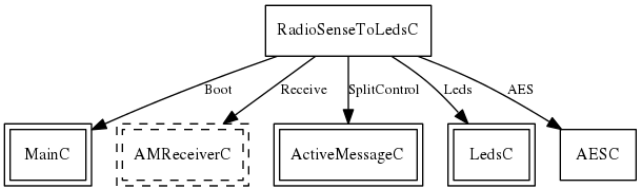

Figure 4. Module interface of base station program

The module interface of base station is quite similar to the sensor mote. The table below gives a brief description about each module used in base station program.

Module	Description
MainC	Performs the boot up operation
AMReceiverC & ActiveMessageC	Prepares a payload and transmits it across to Base station
LedsC	Indicates the transmission activity on TelosB mote
AESC	Decrypt the message using 16 byte decryption key

Table2. A brief description of each module used in sensor mote.

The base station wall powered device hence power consumption is not an issue for it.

3.3 Back-End Program

Back-end implemented using C# language on the top of .NET 4.5 platform. The application itself is multi-threaded: main thread keep read information from serial port while second thread is essentially responsible for web requests.

The application has to be started from command line with three parameters: serial port name, serial port baud rate and web port number. Exact usage as follows: SerialReader.exe <com port> <baud rate> <web port>.

Internally back-end application has dictionary structure "nodes" which is updated every time it receives information from the base station. Once any information received on the serial port, Port_DataReceived event is triggered. Event handler is responsible for attach received data to existing string of text and match the regular expression "Decrypt [0-9] + [0-1]" which looks for decrypted part of text and matches the node number (0-...) and node state (0 or 1).

Once internal state is updated and the second thread receives web request, it will generate web page with JSON object that includes all available node numbers and their respective states.

3.4 Front-End Program

Web interface has been implemented using Bootstrap, JQuery, html, Ajax. Web interface interacts with the backend by making Ajax calls. Backend sends a JSON object for every Ajax request as a response. The structure of the JSON object is { node_number : "node_state"}.

Figure 5. Front-End User Interface

Node-number indicates the position of the car and node-state is used to know the status of the parking lot (i.e. available or parked). Web interface iterates through the JSON object to display the statuses of the parking lots and it updates the results for every 2 seconds.

In the interface, green cars indicate that the slots are available for parking and Red cars show the taken spots. This interface is compatible for both desktop as well as mobile.

4. EVALUATION

We have successfully evaluated our project by using 3 sensor motes and the base station. The each sensor mote is planted at the parking lot to indicate the availability of it. We tested it in day light time for convenience. The video for the live demonstration of our project is available at this link-

https://www.youtube.com/watch?v=M0te2WJ_BHo

5. CONCLUSION

In this paper, we proved that car tracking with encryption works and implemented a website to display the information accurately to the client. The process that was involved to create our solution started by formulating the problem of traditional parking and coming up with a better solution. Our project has exceeded our expectations by using more than one mote to handle multiple parking spaces. We created a model parking lot with toy cars and out of cardboard to prove that this implementation would be possible in a real world setting. We have evaluated our project in the actual parking lot during daylight and at night to assess the accuracy of our application. We have create a system for parking lots that has never be created because our system uses a website to display open parking spaces.

6. ACKNOWLEDGEMENT

We would like to thank professor Mo Sha for all the help getting starting with TinyOS and giving our team a copy of the virtual machine. The PowerPoint slides on TinyOS were very usefully in starting and finishing our project, and that we're grateful. We also thank our classmates for listening to presentation and giving us feedback about our project.

7. Limitations

The limitations of the system would be not knowing how many sensors the base station could handle because we have tested the system with 3 motes but we don't know the amount of motes per

base station. Another concern is how many clients the server could handle because we are using an Ajax call every 20 seconds to update the available parking spaces and adding more clients would increase the network load, which could break our system. The final limitation of our system would be radio interference of the surrounding area because this could delay parking spots from transmitting or could block the signal from reaching the base station. This would cause the base station to lost track of that mote. Even though, we have some limitations these problems aren't major and could be solved with more work done on this project.

8. Future Work

In the future we would like to deploy a small scale implementation and grow it larger to handle more drivers. We know that we could make the motes and base station more reliable with acknowledgements because adding this feature would ensure the delivery of the status of the mote. Another feature we would like to develop would be an enclosure for the motes and base station, so they could stand up to the weather conditions. Another future task is understanding how many motes per base station because we need to know the limitation of the base station. Finally we would need a dedicated server per parking lot to relay the information to the website, because we are currently using a laptop and a server would help us scale our system.

9. REFERENCES

[1] Levis, P., & Gay, D. (n.d.). TinyOS APIs. *TinyOS Programming*, 241-251. doi:10.1017/cbo9780511626609.016

[2] Levis, P., & Gay, D. (n.d.). Advanced components. TinyOS Programming, 129-144. doi:10.1017/cbo9780511626609.010

[3] Levis, P., & Gay, D. (2009). TinyOS Programming. doi:10.1017/cbo9780511626609