Computer Science and Information Technology 1(4): 276-279, 2013 http://www.hrpub.org
DOI: 10.13189/csit.2013.010406

Park-A-Lot: An Automated Parking Management System

Kuo-pao Yang*, Ghassan Alkadi, Bishwas Gautam, Arjun Sharma, Darshan Amatya, Sylvia Charchut, Matthew Jones

Computer Science and Industrial Technology, Southeastern Louisiana University, Hammond, LA 70402
*Corresponding Author: kyang@selu.edu

Abstract This paper describes the architecture and design of Part-A-lot, an automated parking management system. It explains the working dynamics of this prototype system and its communication with the website to find a parking spot before arriving at the destination. This system proposes a solution for urban parking problems. It minimizes the hassles of existing issues and provides an implementable model for parking lots in an urban setting.

Keywords Parking Management System, Ultrasonic Sensor, Arduino, Data Server, Web Server

1. Introduction

Imagine a scenario that you look at your smart phone to find a parking spot close to your destination, drive there, and just park. You do not waste your time and money to drive around in order to find one parking spot amidst a sea of cars. We develop the Part-A-Lot, the parking lot monitoring system [1], which would not only help us, but also people in urban parking scenarios.

According to a survey by Department of Transportation in 2007, there were estimated 254.4 million registered passenger vehicles in United States, with the number still increasing at a very rapid rate. According to United States Department of Energy, the rate of motorization in 2007 peaked at 842.6 vehicles per 1000 people. To easily find an unoccupied parking space in a large car park is a problem for many drivers. With the increasing growth of automotive industry, the demand for intelligent parking service is expected to grow rapidly in the near future [2].

With the increasing volume of automobiles, as the number of student increases in the campus area, the possibility of finding the best location for the parking spot decreases. During the peak hour of campus, it becomes even harder to find a parking spot in the whole campus and the possibility of missing classes or appointments increases. This problem persists in urban areas too and people lose time and fuel while driving around looking for a parking spot [3]. Higher density of consumer vehicles demands an implementation of automated parking lots whose information can be easily sustained and monitored [4]. This emerging service will provide automatic management of parking lots by accurate monitoring and making that information available to customers and facility administrators.

We cannot avoid this problem by simply adding more parking spots. We need to efficiently manage what parking spot we already have. To avoid the case of having to spend more money on building more parking lots and help guide students to the proper parking spot [5] so that they can efficiently make use of time they would otherwise spend searching for empty parking spot. Some of the possible solutions available in the market provides services to find empty parking spot but can be monitored through the server itself. It would help a lot if the monitoring could be done by everyone online even using smart phones or any other web capable device. Current technology facilitates the opening and closing of gates only after swiping of the card but it would help a lot if the gate would automatically open as the vehicle approaches. Thus it is useful to have technical solutions which can provide information on parking space occupancy.

There are many other systems to monitor and maintain parking lots and many technologies such as ultrasonic distance sensors [6] [7], magnetic sensors [8], image processing [9][10] or even hybrid [11] of these technologies have been research upon by many other scholars.

So the solution we came up with a new monitor system, Park-A-Lot, which comes with its own platform-independent web-page interface that is easily accessible using any devices like iPhone, computer or android and can be used anywhere with the access to web. It is based on Ultrasonic sensors. It provides occupancy information for car park users and helps them to place the car in a very efficient way.

2. Park-A-Lot System Workflow

The Park-A-Lot is a complete parking manage system, which gives a web interface to monitor a parking lot. This system also has a parking lot entrance control, which opens and closes the gate as a vehicle enters the lot. The system

detects the occupancy of a parking space using ultrasonic sensors.

The whole process of sending the data from the sensors and its analysis is done in four steps shown in Figure 1. First the ultrasonic sensors feed the data into the Arduino board which then calculates if the spot is empty or available. If the space is taken it sends the data to the web server. The gate opens or closes based on the sensor designated to the gate. At the same time a client can look at the web interface of the system which interprets the data in the web server to show the availability of the parking space.

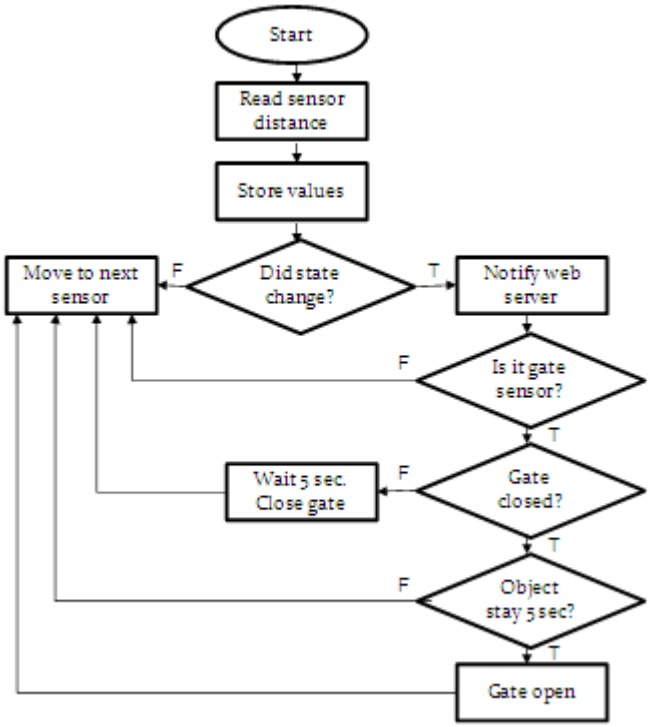

Figure 1. Park-A-Lot System Workflow

3. Implementation of Parking Monitor Service

This system consists of three ultrasonic gate sensors that feed data into the Arduino board. To increase the accuracy, the sensors take ten distance reading and average them. This method of taking ten values makes sure that small dust particles or other small object that might come in front of the sensor do not affect the reading that is taken into consideration while making logical decision for the availability of the parking lot shown in Figure 2. The Arduino board as shown in Figure 3 is a microcontroller. Using Ethernet shield to lay the foundation of inter-server communication, the motor shield is used to control the entrance of the parking lot. The web site makes this system display the information live to anyone who needs to park.

After the sensor agent gets the distance, the Arduino board which has the microcontroller checks if the data provided fulfills the requirement for the processor to declare the parking spot as taken. The processing unit goes through data provided by each sensor one at a time. The process though increases the processing time, it makes sure that all the sensors are not working at the same time and sending data to the processor increasing the load to the processor. While going through each sensor it updates the data about the sensors to the data-server using the Ethernet shield.

```
void setAndCheckValues(){
    if(averageDistance <= 50 && averageDistance!=0){
        //Send changes to the website
        parkState[echoPinCount] = 1;
        sendChanges();
        //Gate sensor to open motor
        if(averageDistance <= 30 && echoPinCount ==
                GateSensor){
            openMotor();
            GateOpened = true;
        }
    }
    else{
        //Send changes to the website
        parkState[echoPinCount] = 0;
        sendChanges();
        //Gate sensor to close motor
        if(echoPinCount == GateSensor){
            if(GateOpened){
                delay(2000);
                closeMotor();
                GateOpened = false;
            }
        }
    }
}
```

Figure 2. Arduino Program Logic and Code

Figure 3. Arduino Mega Board

The data communication agent consists of two parts. The first component is the Ethernet shield that is attached to the Arduino board. It provides the system ability to connect to the data-server and use the network connection provided to it. When the device loads up the Ethernet shield checks if a DHCP is available to configure its IP address, if available it gets an IP address from the DHCP but if the DHCP is

unavailable. It sets itself a manual IP that is encoded in the actual code. After successfully getting the network connection, the connection is used by the processing Agent to send the data to the data server. The Xively service is used as a data service provider to connect sensor derived data to the web. The system is configured to use a Xively account that was created by the team. So, after the Arduino has the information about the account in the Xively, the Arduino sends the information about the parking space to the web service provider.

The Xively data server has its own API which can be used to retrieve the data that was sent by the processing unit. The website makes a restful request to the Xively service which in turn returns a XML with the corresponding data for the parking lot. The website then parses the XML shown in Figure 4 which is retrieved from the web service and then makes the logical decision to show the parking space as either occupied or empty.

```
$xml = simplexml_load_file($completeurl);
if(($xml->environment->status)=="live"){
      echo "<p style='color:green'>System Live and
            running</p><br/>";
      $i=1;
      foreach ($xml->environment->data as $values) {
            if(($values->current_value)!=0){
                  echo "<div   style ='background:red' id='box".$i."'>";
                  echo "<h2>Spot".$i++."</h2>";
                  echo($values->current_value);
                  echo   "</div>";
            }
            else{
                  echo "<div style='background:green' id='box".$i."'>";
                  echo "<h2>Spot".$i++."</h2>";
                  echo($values->current_value);
                  echo   "</div>";
             }
      }
 }
else {
      echo "<p style='color:red'>System Frozen please try again
            later</p><br/>";
}
```

Figure 4. PHP Code to Parse Xml Received from Xively

The page which shows the information about the parking lot refreshes every six seconds to make sure that the client is viewing the live data. In the web interface bootstrap has been implemented to make sure that the website is responsive in any device and adjusts to any screen size. This increases the platform independence of the system. The web interface makes the system available through any web or Smartphone interface with internet access. Figure 5 shows the spot 2 is taken.

4. Implementation of Entrance Control Service

The entrance of the parking lot has a gate which is hooked up with ultrasonic distance sensor. The sensor sends an average distance from ten readings to the processing unit which is used in the monitor service.

The processing unit makes a logical decision to open the gate if it finds an object is in front of the sensor for an interval of 5 seconds and close it if it moves away from sensor, after 5 second interval to ensure that the car is not damaged when the gate closes. Whenever an object stays in front of the sensors, the bipolar stepper motor controlled by motor shield rotates 90° to the clockwise direction, and the gate opens. Whenever the object leaves the sensor, the motor shield ensures that the bipolar stepper motor rotates 90° in anti-clockwise direction, so that the gate closes.

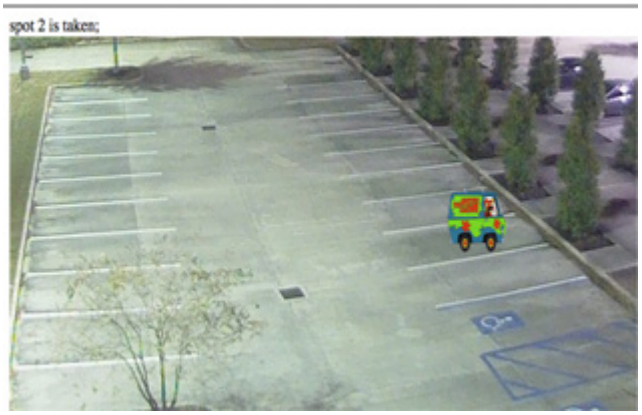

Figure 5. Web Interface Displays Spot 2 is Taken

5. Discussion

This parking management system is developed using ultrasonic sensors, Arduino board, motor shield and Ethernet shield. The development of this prototype system is capable of sending the status of the parking lot to the web interface after processing the information in the Arduino board with the help of internet connection in the Ethernet shield. A parking gate is also implemented with the help of motor shield and a sample gate arm.

This system is capable of reading the values of the ultrasonic sensor echo distances, takes their average, and manipulates the average to decide whether the parking spot that the particular represented sensor is free or taken. The threshold distance taken for the demonstration purpose was 50 cm, so any object within the range would be treated as a parked object. A threshold distance of 30cm and time threshold of 5 seconds is set for the gate sensor. In order for the gate sensor to open, the sensor should continuously trigger parked state for 5 seconds. A delay of 5 seconds is set to the gate closing event to avoid any accident or collision situation.

Upon change of the status of the parking spot, the information regarding the status is sent to the Xively data server. The Xively data server would send the data to the Park-A-Lot website, and the data would get processed in the

web site to output visual for the end user. The whole system has a lag time of 18 seconds because of the time lapse when switching between sensors in the program loop. The Arduino Mega 2560 does not support asynchronous processes. Therefore the values from the sensors are read in a loop moving from one to another.

6. Conclusion

This prototype system is automated which can be efficiently implemented into a real parking lot. This system can be extended to have a feature to count the number of vehicles entering the gate to keep track of number of parking spots available in the parking lot. Further, a camera can be integrated into the system to take picture of the vehicle or its registration plate when it enters the gate for security purposes. This system has a high potential of being an automated system that requires less maintenance and employee costs. The gathered information can be used in a distributed or centralized way to evaluate other meaningful metrics such as duration of parking, automatic billing and payment to the benefit of users and administrators.

REFERENCES

[1] J. Chinrungrueng, S. Dumnin, and R. Pongthornseri. iParking: a Parking Management Framework, Proceedings of the 11th International IEEE Conference on ITS Telecommunications (ITST), St. Petersburg, FL, 63-68, August, 2011.

[2] S. Mathur, S. Kaul, M. Gruteser, and W. Trappe. ParkNet: a Mobile Sensor Network for Harvesting Real Time Vehicular Parking Information, Proceedings of the MobiHoc S3 Workshop on MobiHoc S3 (MobiHoc S3 '09), New York, 25-28, 2009.

[3] A. Klappenecker, H. Lee, and , J. Welch. Finding Available Parking Spaces Made Easy, Proceedings of the 6th International Workshop on Foundations of Mobile Computing (DIALM-POMC '10), Cambridge, MA, 49-52, September, 2010.

[4] S. Banerjee, P. Choudekar, M. Muju. Real Time Car Parking System Using Image Processing, Proceedings of the 3rd International IEEE Conference on Electronics Computer Technology (ICECT), Kanyakumari, India, Vol. 2, 99-103, April, 2011.

[5] J. Martin. A Solution to the Campus Parking Problem, Proceedings of the 22nd Annual ACM SIGUCCS Conference on User Services (SIGUCCS '94), New York, 31-33, 1994.

[6] W. Park, B. Kim, D. Seo, D. Kim, and K. Lee. Parking Space Detection Using Ultrasonic Sensor in Parking Assistance System, Proceedings of the IEEE Conference on Intelligent Vehicles Symposium, Eindhoven, Netherlands 1039-1044, June, 2008.

[7] J. Propst, K. Poole, and J. Hallstrom. An Embedded Sensing Approach to Monitoring Parking Lot Occupancy, Proceedings of the 50th Annual Southeast Regional Conference (ACM-SE '12), Tuscaloosa, AL, 309-314, March, 2012.

[8] J. Wolff, T. Heuer, H. Gao, M. Weinmann, S. Voit, and U. Hartmann. Parking Monitor System Based on Magnetic Field Sensor, Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference (ITSC '06), Toronto, Canada, 1275-1279, September, 2006.

[9] L. Chen, J. Hsieh, W. Lai, C. Wu, and S. Chen. Vision-Based Vehicle Surveillance and Parking Lot Management Using Multiple Cameras, Proceedings of the 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), Darmstadt, Germany, 631-634, October, 2010.

[10] S. Nath, A. Deshpande, Y. Ke, P. Gibbons, B. Karp, and S. Seshan. IrisNet: An Architecture for Internet-Scale Sensing Services, Proceedings of the 29th International Conference on Very Large Data Bases (VLDB '2003), Berlin, Germany, Vol. 29, 1137-1140, 2003.

[11] S. Lee, D. Yoon, and A. Ghosh. Intelligent Parking Lot Application Using Wireless Sensor Networks, Proceedings of the International IEEE Symposium on Collaborative Technologies and Systems (CTS 2008), Irvine, CA, 48-57, May, 2008.