VALET

Smart Parking System
Mobile Application
Project

Goktug SELCUK
Department of Computer Science
Binghamton University
Binghamton, NY
gselcuk1@binghamton.edu

Gungor YOLAC
Department of Computer Science
Binghamton University
Binghamton, NY
gyolac1@binghamton.edu

ABSTRACT

VALET is a smart parking system that allows drivers to find available parking spots faster and easier. The main objective of developing an application like VALET is to propose a more convenient parking experience to it's users. We think that the presence of smart parking system applications like VALET is crucial for the majority of locations. Especially, metropolitan cities like New York and Istanbul tend to have parking problems due to their crowd level and number of active vehicles.

KEYWORDS

Smartphone, IOT , parking, lot, sensors , application, Smart Parking.

1 Introduction

In the last decade, there have been tremendous changes happening in Information Technologies. Most of IT developments affect people's lives directly. That is why, IoT is a significant area for both tech companies and nations all around the world. Time is very crucial for companies and also for the people. Most of those developments in the IoT area focus efficiency improvement for everything about people in their work hours and their spare times.

There are various extreme situations caused by the metropol life's necessities. High life standards in a high scaled and complex settled city depending on a functional transportation axis, side of a proper organized environment. Because the transportation network is the main base for a metropol,several solutions have been improved by the automotive industry since the very beginning of civilization. The point, vehicle and the road notions interacted, the 'traffic' issue came up as a civil life problem in the history. Traffic is a variable organic flow defined accordingly a density rate based on datas like population,the adjustment quality of the roads or the efficiency of the network etc. Of course, there are important attempts with the improvement of civilization; like public transportation,underground transportation variations, railway systems to prevent traffic problems. Unfortunately, there is still a huge personal car ratio on the roads in crowded cities.

The determination of the project establishes another important point in that case because one density caused another. So, in parallel to the described case a problem by lack of parking spots came up to the table. Again, solutions like underground car parks,private parking spots are used to correct this deficiency but in a living city, there is a critical circulation between spots and vehicles. The wrong parking incidents or instant park overs caused by lack of proper parking spots is an important factor that paralyzes the road network and causes more traffic or car accidents. That makes the stable status of a car as important as active status to provide a balance in traffic. Beside it, there is a serious amount of driving time for drivers to find an available parking spot. This is a huge waste of energy,fuel and also from a psychological perspective, It increases the stress level of metropol people which is already beholding in high levels.

Aim of the project is to head off that wasted energy and density by adjusting a plain and right parking spot network in a city. Documenting the proper parking spots and making their availability status accessible for drivers on the road is an attempt to direct drivers straight to their arrival point by avoiding the wasted energy caused by effort on finding a parking spot in a world consumed away by humans.

2. Hardware

To provide a service to find available parking spots easier, we had to build a hardware circuit to transfer data which includes the status of a specific parking spot to the users, who are drivers in this case. Our circuit consists of five different components:(i)Arduino Uno R3, (ii) NodeMCU ESP8266 WiFi Module, (iii) FC-51 Infrared Sensors, (iv) Jumper Cables and (v) Breadboard.

3. System Design

In the Valet project, the system was designed on multiple IR sensors for different parking spots and these spots' availability can be seen from the Blynk app. The IR Sensors detect the presence of any obstacle. After that, these sensors send the data to Arduino UNO. Ardunino is the microcontroller board which shares the data with other devices. In this project, NodeMCU ESP 8266 communicates with Arduino. NodeMCU takes the data which is coming from IR sensors and provides the connection with the Blynk app. Blynk app is the final destination for the data. Main aim in the Blynk app is demonstrating the data intelligibly for the users.

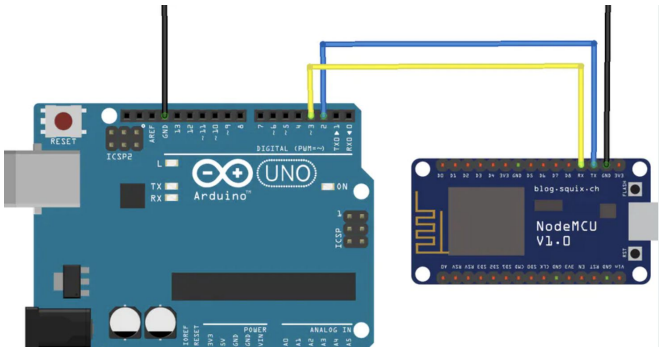

Figure 1: Arduino UNO and NodeMCU ESP8266 Serial Communication

In order to maintain the system, the connection between the Arduino UNO and Nodemcu ESP8266 is really important. The RX pin is used for receiving the data and the TX is used for sending the data between devices. The GND pins are connected to each other between devices and this connection is used to ground the circuit.

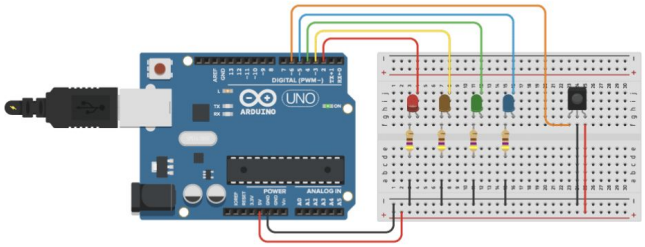

Figure 2: Arduino UNO and IR Sensors Serial Communication

In this project, there are five FC 51 InfraRed proximity sensors to check the availability of parking spots. Breadboard is a tool for designing and testing circuits [1]. Connecting multiple sensors is impossible without a breadboard. In this project, breadboard is used for connecting ground and digital out pins of the IR sensors. VCC pins are directly connected to different digital pins of the Arduino.

4. Software

Basically, VALET needs software for two main reasons.First of all, the code we implemented serves like a guide for our hardware components. Arduino Uno R3 needs to be guided to manage the data transmissions with both of the IR sensors and ESP8266 WiFi module. Without any failure, Arduino Uno R3 needs to receive the data which consists of the availability of parking spots . Otherwise, our main objective to develop this application will not be satisfied at all. In addition, after receiving the data correctly from sensors, Arduino is required to transmit these data to ESP8266 WiFi module without any errors. Therefore, the program implemented by us helps Arduino for taking the care of these responsibilities. In upcoming sections(4.2), this program is going to be examined in a detailed way.

Similar to Arduino Uno R3, the NodeMCU ESP8266 WiFi Module needs help from a program which manages the two-sided data transfer. First, the ESP8266 WiFi module must receive data regarding the status of parking spots without any failure. In the next step, it must read and categorize the data correctly before sending it to the mobile application. In section 4.3 the program of NodeMCU ESP8266 WiFi Module is going to be analyzed and described to the readers of this report.

In VALET, The second use of software is for showing free parking spots to users. We use Blynk IoT tool to establish an application which will show the status of a whole parking lot. The code of the Blynk tool was integrated in our program for ESP8266 WiFi module. In section 4.4, Blynk and the user interface of VALET is going to be examined elaborately.

4.1 Programming Language, Tools and Libraries

In this section, the programming language, tools and software-related libraries we used in VALET will be described. First of all the programming language is C for VALET since C is the most popular language for projects that include Arduino. Also the majority of documentations and examples are written in C so we wanted to take advantage of that.

Arduino IDE, Blynk Application, Blynk Arduino Library and SoftwareSerial.h header file are all of the tools and libraries that are used while developing VALET. Arduino IDE is the development environment we implemented the programs of Arduino Uno and ESP8266 WiFi Module. In addition to that, there are several drivers we downloaded for both Arduino and NodeMCU ESP 8266. These drivers allowed us to program these devices through Arduino IDE. These two devices connected directly to the PC.

Blynk is an application which allows developers to create a basic user interface for their IoT project. Blynk is reported elaborately in section 4.4. Blynk Arduino Library is required to use Blynk functions and definitions in our program, which is essential to transfer the data to the user interface. Finally, the SoftwareSerial.h is needed to expand the assets for the serial communication between hardware components. Normally, the Arduino hardware has built-in support for serial communication only on pins 0 and 1. With the aid of this header file, we were able to use other digital pins as well for serial communication. This is very crucial since we have 5 IR sensors and ESP8266 to connect with Arduino.

4.2 Arduino Program

The Arduino program is very important for VALET to operate correctly. As it is stated in previous sections, Arduino is responsible for receiving the data of parking spots from the IR sensors and transmitting those data to the NodeMCU. In this section we will go through the program.

In the beginning of the code, we included the SoftwareSerial.h header file to enable all of the pins of Arduino to communicate with other hardware components via serial communication. Then, we created a SoftwareSerial object which is a NodeMCU object. We assigned the values 2 and 3 as parameters of our object to declare that pin 2 and pin 3 are used to connect the NodeMCU to the Arduino. After those, there are some variable declarations. Five of them are strings and five of them are integers. The string

variables are required to store the status of a specific parking spot and since we have five sensors, there are five string variables. Similarly, the five integer variables are used to assign the pin number which we used to connect that sensor to Arduino. After these declarations, there are two arrays, sensorPins and sensorOutputs to store these declared variables for further use. Finally, there is a string variable to serve as our "message packet". "lotStatus" variable will be used to store all data of a parking lot to send it to the NodeMCU.

In our setup function, we make two calls, Serial.begin() and nodemcu.begin(). These calls are going to start the serial communication and boot the NodeMCU. Then, with the help of pinMode() function, we declare that all of the sensor variables(integer ones) will serve as input pins. The setup block has come to an end with those lines. Normally, in this code, we need to define functions to detect the occupancy status of each parking spot. Since we have 5 sensors, we need to define 5 functions for each sensor. However, to make it simpler we merged those five functions into one function. The get_Occupancy() [3] function will use the previously created arrays to iterate through them to detect the status of all five parking spots.

```
void get_Occupancy(){ // Function to iterate all sens

  for (int i = 0; i < 5; i = i + 1) {

    if( digitalRead(sensorPins[i]) == LOW)
    {
    sensorOutputs[i] = "F"; //F for FULL
    }

    else if( digitalRead(sensorPins[i]) == HIGH)
    {
    sensorOutputs[i] = "E";  //E for EMPTY
    delay(500);
    }

  }

}
```

Figure 3: get_Occupancy() function

Finally, in the loop block, we call the get_Occupancy function to take data of the parking spots. After adding all of the outputs to the lotStatus variable, we call println functions of both Serial and nodemcu. In the end, we erase all the data in lotStatus for future iterations of the loop.

4.3 NodeMCU Program

In the first part of our NodeMCU Program, there are four header files to include. ESP8266WiFi.h , BlynkSimpleEsp8266.h, SoftwareSerial.h and SimpleTimer.h. Then we have three different character arrays for authorization token of Blynk, local ID of WiFi and the WiFi password. Then we create a SimpleTimer object to use timer functions in our program. After that string variable "msg" and char variable "character" is created. "Character" will store the status of a single parking spot while "msg" stores the status of a group of parking spots(parking lot). Also, we create integer variables to store the final data to send to virtual pins of Blynk.

In the setup block, we start the serial communication with Serial.begin() then we also start Blynk with Blynk.begin function and we pass three parameters into that function. Those parameters are authorization token, wifi ID and wifi password. After these calls, we used timer.setInterval to make all 5 of our functions[4] that send data to each virtual pin in every second.

In the loop block, we control that if there is a serial communication. If there is, read the message coming from Arduino character by character using Serial.read(). Those characters will add up in each iteration to produce the complete data. When we come to the end of the data sent by Arduino, we split 5 values(0 or 255) from the whole data with an user defined function. After splitting, we convert these splitted strings to integers to send them to virtual pins.

```
void valuetoled1()
{
int sensor = led1; //take the converted integer
  Blynk.virtualWrite(V10, sensor); // and send it to virtual pin V10

}
```

Figure 4: Example function to send data to virtual pins(1 of 5)

4.4 UI with Blynk

All programs and applications need an user interface. We created a user interface using Blynk. Blynk is an application that allows developers to create basic and convenient user interfaces for IoT projects. Blynk connects with a device and receives the desired data. After that, with the help of various widgets, you are able to categorize and display those data easily and in a nice looking way. Developers also need to install the Arduino library for Blink to take help from Blynk in their Arduino projects.

In VALET, the device we connected with Blynk is our NodeMCU WiFi module. As it highlighted in the previous section, the NodeMCU program sends the data to the **virtual pins.** Each widget has a virtual pin. In our user interface, we used **tabs** and **LED** as our widgets. Tabs is to separate the parking spots in different locations and LED is used to represent a parking spot. In other words, each LED widget represents a specific parking spot. We assigned virtual pins (Ex: V10) to the LEDS and sent data to those virtual pins. If the received data says that a parking spot is occupied, the LED widget that represents that parking spot is turned on[5].

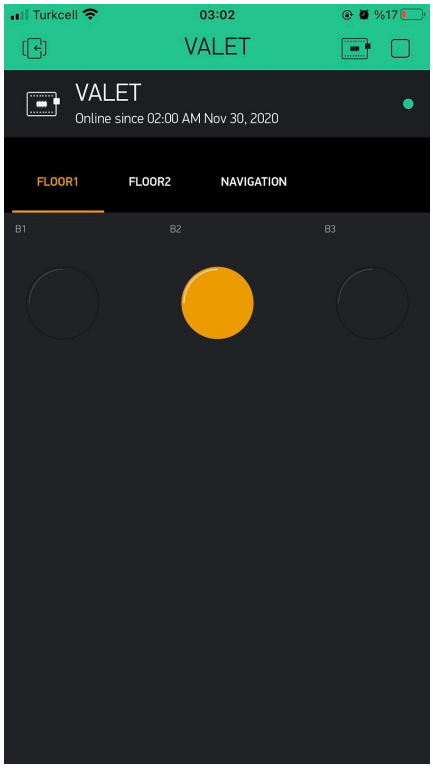

Figure 5: UI of VALET

Is it also possible to publish the application via Blynk but that operation is not free. Since we are students and this is a course project we are not able to publish VALET unfortunately.

5. Possible Improvements

As we experienced while using VALET, a navigation page is required to change between parking lots which are registered to the VALET system. Let's say, when an user goes to the Walmart, they should use a navigation method, like a menu, to choose that specific Walmart from it. Otherwise, the user needs to browse the location in the tabs and that can be frustrating. We gave our best to add that feature but we could not manage it before the deadline. Maybe removing Blynk usage from VALET and designing and implementing our own application can help on this issue.

Also, another feature that can be added to VALET is the payment system. There are some parking lots that require drivers to pay an amount that depends on the parking time. VALET can be updated to track the parking time, calculate the fee depending on that and offer a payment portal to easily make payments via the application.

6. Conclusion

In today's world, technologic improvements are affecting people's lives significantly. IoT has the most human interaction when we compare the other areas of Information Technologies. Not only there are various different types of IoT applications that people are using in their daily lives but also there are a lot of different projects that the companies are still working on. As most people realise, using these technologies can bring them not only a useful environment to make their work but also a lot of extra time.

Companies are still finding extraordinary ideas to bring our lives and people start to use countless different technologies day by day. That is why, giving the best user experience for users is so important for companies.

VALET is an example IoT project that helps significantly to people who use their car in crowded areas because sometimes, these people lose lots of time to searching for an available parking spot.

ACKNOWLEDGEMENTS

We would like to thank our professor Mo Sha in and our TA Junyang Shi for their endless help and support. We are grateful for leading and motivating us to create an IOT project as a solution to one of the real life problems we faced during our daily life.