“AUTOMATIC PARKING USING SEMAPHORE”

PROJECT REPORT

Submitted for the course: Operating Systems (CSE2005)

By

Shivansh Chadha (Team Leader) **19BCT0142**

Dananjay Murugesh **19BCT0138**

Amara Hemant Kumar **19BCT0126**

Slot: G1

Name of faculty: Kalyanaraman P

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

June, 2020

ACKNOWLEDGEMENT

We express gratitude to our teacher, **Prof. Kalyanaraman P**, for his guidance and suggestions that helped me to complete the project on time. Words are inadequate to express my gratitude to the faculty and those who encouraged and supported us during the project.

Introduction

This presentation focuses on the Parking problem which is generally seen in places like shopping malls, tourist points and marketplaces. People spend a huge amount of their time to park their vehicles, so we developed an Automatic Parking System. We have made use of Semaphores and First Come First Serve (FCFS) Synchronization so that we can avoid the deadlock situation for which vehicle should the machine park first. We have implemented our system using C language. .

Techniques Used

We are incorporating two main concepts of operating systems in our project which are:

1. **Semaphore:** A semaphore, in its most basic form, is a protected integer variable that can facilitate and restrict access to shared resources in a multi-processing environment. The two most common kinds of semaphores are counting semaphores and binary semaphores. Counting semaphores represent multiple resources, while binary semaphores, as the name implies, represents two possible states (generally 0 or 1; locked or unlocked).

wait() was called (meaning *to decrement*) and signal() was called V (meaning *to increment*).

In our project we will use semaphore to ensure that only one vehicle gets to be parked at one time and all other have to wait until their turns come. As getting two vehichle (process) in the automatic machine will cause a deadlock that's why to avoid deadlock we are using semaphore.

2. **FCFS Synchronization:** To avoid deadlock other process have to wait until the vehicle in the automatic machine gets parked. So, to synchronize the order and avoid any confusion we are using First Come First Serve Synchronization which will ensure that the vehicles align in the queue according to the order they came.

Literature Survey

Semaphore is a visual method of communication that involves signaling the alphabet or numbers by the hand holding of 2 flags in specific positions. It has been described as '**Optical telegraph**'. The flags are colored differently, depending on whether the signal is sent over the land or across the sea. Red and yellow flags (the Oscar flags) are used at sea and are similar to our Surf Lifesaving flags.

The system was developed in France, in 1790 by Claude Chappe and his brothers. This was the time of the French Revolution and there was a great need for the government to be able to quickly communicate orders and to receive information. Their first message, on March 2, 1791 was sent a distance of 10 miles and read: "If you succeed, you will soon bask in glory". They used black and white flags initially, as well as clocks, codebooks and telescopes.

Many automatic parking systems have been designed in the present scenario. Now we are going to take a look the work conducted by Tuan Le-Anh and M.B.M. De Koster. They have come up with an automated vehicle guiding system which can simplify the task of parking. Initially they have fixed a parking area, and they decide where the next vehicle should be parked using scheduling and semaphore.

The parking allotment is done on first come first served basis. They have used the vehicle position system to determine the free slots in the parking area. Vehicle locations have to loaded as soon as possible to get quick responses for the new vehicles approaching the parking which has also been handled by their static vehicle positioning strategy. This system can also be handled using semaphore which we have used.

METHODOLOGY

Our project is based on the concept of semaphores and scheduling. The vehicles enter the parking system in two queues. Only one vehicle can enter the parking system/machine at a time.

P and V Semaphores are used to avoid deadlock whenever a vehicle enter into machine function P is called which decrements the value of S (constant) to 0 and until S equal's one no other vehicle can enter the machine and when vehicle get parked it calls function V which increments S to 1 so now other vehicle can also enter the machine.

We have done this for two-wheelers and four-wheeler separately. Then the vehicles reach the automatic parking machines and are left in the machine to start the process of automatic parking. The machine automatically parks the vehicles. This can be explained as: The parking machine is a critical section and the vehicles are processes.

The parking entrance of 2 rows acts as a lock Only one process can enter the critical section/shared memory at a time in each row one four-wheeler and one two-wheeler.

The processes (here vehicles) are then parked (scheduled) on the basis of first come first serve and semaphore. The processes the leave the shared memory and next vehicle can enter the parking area.

Features

We have implemented the following in our program:

1. Arrival Of Vehicles
2. Total No. of Vehicles Parked
3. Total No. of Two-Wheelers Parked
4. Total No. of Four-Wheelers Parked
5. Display the Order of Vehicle Parked
6. Display the Wait Queue
7. Departure of the Vehicle

Block Diagram for Vehicle Entrance

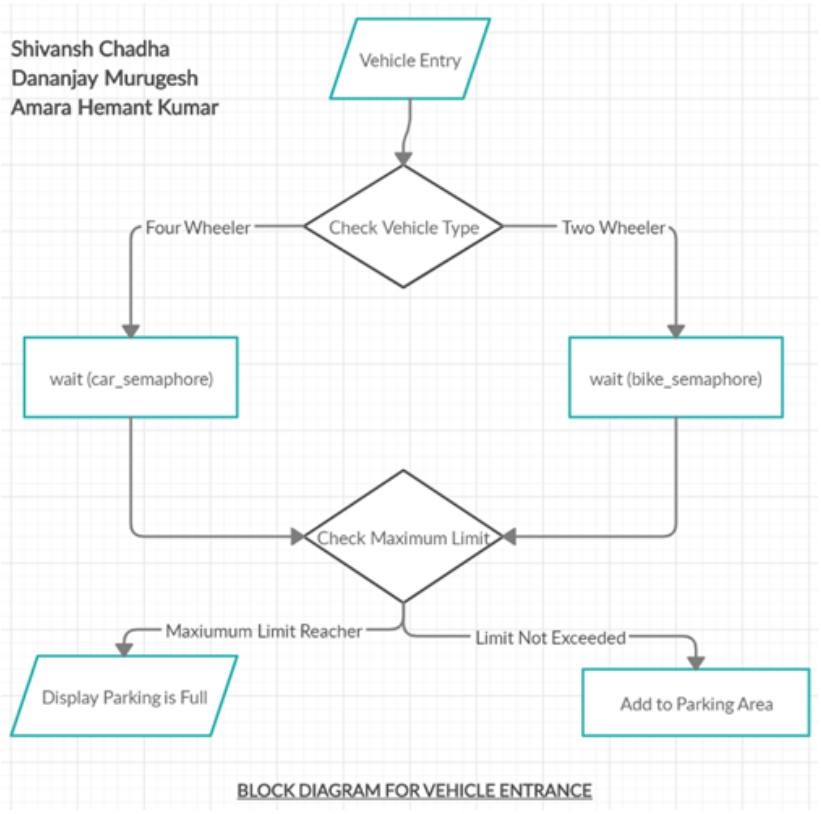

Figure 3.1: Block Diagram for Vehicles Entrance

Block Diagram for Vehicle Exit

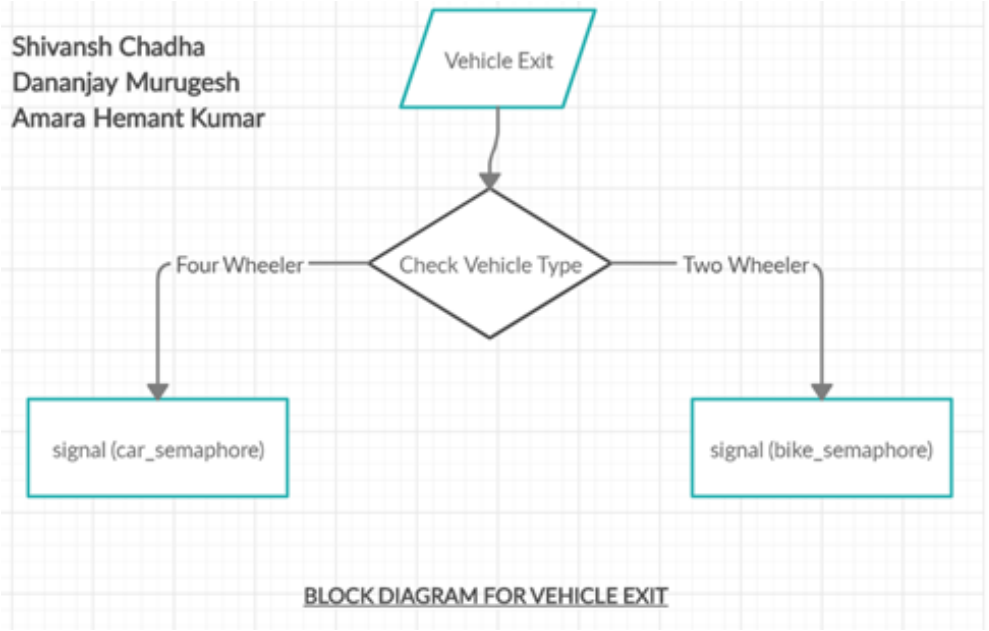

Figure 3.2: Block Diagram for Vehicles Exit

RESULTS

It has become a major problem to find place for parking cars in public places such as market places or malls or any famous place. Based on the concept of semaphore and scheduling we have solved this problem.

Using P and V Semaphores, P semaphore will decrease the value of S and V will increase the value of S. Every vehicle will be given priority on the basis of first come first service.

This way the traffic problem will be solved everybody will get place for parking vehicle both two-wheeler and four-wheeler at proper places according to their sizes.

IMPLEMENTATION

Our implementation plan was very simple we first created our Arrival and Departure function and then used an array to Display the order of vehicle parked. Semaphore was used to stop other vehicle for making an entry on the parking machine when other vehicle was still in the process of parking. FCFS synchronization was implemented to store the vehicles in a first come first serve manner and their vehicle number is stored in wait queue to avoid confusion.

REASON FOR CHOOSING THIS TOPIC

- Many Researchers and Engineers before us provided solution for the Parking System but all those system were costly, here we are using semaphores which will reduce the cost.

- From this project we learnt a real-life scenario in which semaphores can be used. There are many such scenarios which can be efficiently handled by semaphores. Synchronization is required in many places around us, which tells us about the importance of semaphores.

- We can develop a program which will manage more than one system at a time so that we can implement it to bigger parking spaces to save more time.

OUTCOMES OF OUR PROJECT

1. Cost efficient as it uses very simple techniques and can be used in any type of machines.

2. It can be implemented in real world and solve the real issues.

3. It will save time for people as they just need to place their vehicle on the machine and then they are free to go.

4. Traffic will be reduced inside the parking spaces.

ALGORITHM

Input a number 'choice' to select various options

- Choice-1: Enter Vehicle Type (Car/Scooter), Vehicle Number, If empty slot is available. Vehicle will be Parked else Error will be displayed
- Choice-2: Displays total number of vehicles parked
- Choice-3: Displays total Cars parked.
- Choice-4: Displays total Scooters parked.
- Choice-5: Displays order in which vehicles are parked.
- Choice-6: Enter Vehicle Type (Car/Scooter), Vehicle Number, the entered vehicle will be removed from the parking slot else Error.
- Choice-7: Displays the Wait Queue
- Choice-8: Exit

Code:

```
#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <semaphore.h>
#define CAR 1
#define SCOOTER 2
#define MAX 10
int k=1, top =-1;
int waitqueue[MAX], parkinfo[4][10], vehcount, carcount,
scootercount;
////////////////////////////////////////////////////////////////////////////
struct vehicle{
   int num ;
   int row ;
   int col ;
   int type ;
};
struct vehicle * add ( int t, int num, int row, int col )
{
   struct vehicle *v ;
   v = ( struct vehicle * ) malloc ( sizeof ( struct vehicle ) ) ;
   v -> type = t ;
   v -> row = row ;
   v -> col = col ;
   if(k==0){
      printf("Another Car in Process. Please Wait!!");
      if(top == MAX-1)
      {
         printf("\n Wait Queue is full!!");
      }
      else
      {
         top=top+1;
         waitqueue[top]=num;
      }
   }
   if ( t == CAR && k==1)
   {
         P();
        carcount++ ;
```

```
        //delay(10000);
        parkinfo[row][col] = num ;
        vehcount++ ;
         V();
   }
   else if(t== SCOOTER && k==1)
        {
            P();
           scootercount++ ;
           parkinfo[row][col] = num ;
           V();
         vehcount++ ;
        }
   return v ;
}
void changecol ( struct vehicle *v ){
   v -> col = v -> col - 1 ;
}
void del ( struct vehicle *v ){
   int c ;
   for ( c = v -> col ; c < 9 ; c++ )
        parkinfo[v -> row][c] = parkinfo[v -> row][c+1] ;
   parkinfo[v -> row][c] = 0 ;
   if ( v -> type == CAR )
      carcount-- ;
   else
      scootercount-- ;
   vehcount-- ;
}
////////////////////////////////////////////////////////////////////////////////
void P(){
   printf("Vehicle is Parking.\n");
   if (k == 0) {
      printf("Wait!! Another Car In Process!!");
      // add process to queue
      delay(10000);
   }
   else
   k = k - 1;
}
void V(){
```

```
    k = k + 1;
}
////////////////////////////////////////////////////////////////////////////////
void getfreerowcol ( int type, int *arr ){
    int r, c, fromrow = 0, torow = 2 ;
    if ( type == SCOOTER ){
        fromrow += 2 ;
        torow += 2 ;
    }
    for ( r = fromrow ; r < torow ; r++ ){
        for ( c = 0 ; c < 10 ; c++ ){
            if ( parkinfo[r][c] == 0 ){
                arr[0] = r ;
                arr[1] = c ;
                return ;
            }
        }
    }
    if ( r == 2 || r == 4 ){
        arr[0] = -1 ;
        arr[1] = -1 ;
    }
}
void getrcbyinfo ( int type, int num, int *arr )
{
    int r, c, fromrow = 0, torow = 2 ;

    if ( type == SCOOTER )
    {
        fromrow += 2 ;
        torow += 2 ;
    }

    for ( r = fromrow ; r < torow ; r++ )
    {
        for ( c = 0 ; c < 10 ; c++ )
        {
            if ( parkinfo[r][c] == num )
            {
                arr[0] = r ;
                arr[1] = c ;
```

```
            return ;
          }
        }
    }

    if ( r == 2 || r == 4 )
    {
        arr[0] = -1 ;
        arr[1] = -1 ;
    }
}
////////////////////////////////////////////////////////////////////////////////
void delay(int n)
{
    int ms=1000*n;
    clock_t start_time=clock();
    printf("Another car in process,wait\n");
    while(clock()<start_time+ms);
    //printf("Now you go\n");

}
////////////////////////////////////////////////////////////////////////////////
void display( )
{
    int r, c ;

    printf ( "Cars ->\n" ) ;

    for ( r = 0 ; r < 4 ; r++ )
    {
        if ( r == 2 )
            printf ( "Scooters ->\n" ) ;

        for ( c = 0 ; c < 10 ; c++ )
          printf ( "%d\t", parkinfo[r][c]);
        printf ( "\n" ) ;
    }
}
void display_wq()
{
    int i;
```

```
   if(top==-1)
   {
      printf("\nWait Queue is empty!!");
   }
   else
   {
      printf("\nWait Queue is...\n");
      for(i=top;i>=0;--i)
         printf("%d\t",waitqueue[i]);
   }
}
////////////////////////////////////////////////////////////////////////////////
void main( ){
   int choice, type, number, row = 0, col = 0, i, tarr[2], finish = 1;
   struct vehicle *v ;
   struct vehicle *car[2][10] ;
   struct vehicle *scooter[2][10] ;

   while ( finish ){
      printf ( "\nAUTOMATIC PARKING\n" ) ;
      printf ( "1. Arrival of a vehicle\n" ) ;
      printf ( "2. Total no. of vehicles parked\n" ) ;
      printf ( "3. Total no. of cars parked\n" ) ;
      printf ( "4. Total no. of scooters parked\n" ) ;
      printf ( "5. Display order in which vehicles are parked\n" ) ;
      printf ( "6. Departure of vehicle\n" ) ;
      printf ( "7. Display the Wait Queue\n" ) ;
      printf ( "8. Exit\n" ) ;

      scanf ( "%d", &choice ) ;
      switch ( choice )
      {
         case  1:printf ( "\nAdd: \n" ) ;
               type = 0 ;
               while ( type != CAR && type != SCOOTER )
               {
                  printf ( "Enter vehicle type (1 for Car / 2 for Scooter ):
\n" ) ;
                  scanf ( "%d", &type ) ;
                  if ( type != CAR && type != SCOOTER )
                     printf ( "\nInvalid vehicle type.\n" ) ;
```

```
            }
            printf ( "Enter vehicle number: " ) ;
            scanf ( "%d", &number ) ;
            /* add cars' data */
            if ( type == CAR || type == SCOOTER )
               {
                  getfreerowcol ( type, tarr ) ;

                  if ( tarr[0] != -1 && tarr[1] != -1 )
                  {
                     row = tarr[0] ;
                     col = tarr[1] ;

                     if ( type == CAR )
                        car[row][col] =  add ( type, number, row, col ) ;
                     else
                        scooter[row - 2][col] = add ( type, number, row,
col ) ;
                  }
                  else
                  {
                     if ( type == CAR )
                        printf ( "\nNo parking slot free to park a car\n"
) ;
                     else
                        printf ( "\nNo parking slot free to park a
scooter\n" ) ;
                  }
               }
               else
               {
                  printf ( "Invalid type\n" ) ;
                  break ;
               }
               printf ( "\nPress any key to continue..." ) ;
               getch( ) ;
               break ;

         case  2:printf ( "Total vehicles parked: %d\n", vehcount ) ;
            printf ( "\nPress any key to continue..." ) ;
            getch( ) ;
```

```
            break ;

        case  3:printf ( "Total cars parked: %d\n", carcount ) ;
            printf ( "\nPress any key to continue..." ) ;
            getch( ) ;
            break ;

        case  4:printf ( "Total scooters parked: %d\n", scootercount )
;
            printf ( "\nPress any key to continue..." ) ;
            getch( ) ;
            break ;

        case  5:printf ( "Display\n" ) ;
            display( ) ;
            printf ( "\nPress any key to continue..." ) ;
            getch( ) ;
            break ;

        case  6:printf ( "Departure\n" ) ;
            type = 0 ;
            while ( type != CAR && type != SCOOTER )
            {
               printf ( "Enter vehicle type (1 for Car / 2 for Scooter ):
\n" ) ;
               scanf ( "%d", &type ) ;
               if ( type != CAR && type != SCOOTER )
                  printf ( "\nInvalid vehicle type.\n" ) ;
            }
            printf ( "Enter number: "  ) ;
            scanf ( "%d", &number ) ;
            if ( type == CAR || type == SCOOTER ){
               getrcbyinfo ( type, number, tarr ) ;
               if ( tarr[0] != -1 && tarr[1] != -1 ){
                  col = tarr [1] ;
                  if ( type == CAR ){
                     row = tarr [0] ;
                     del ( car [row][col] ) ;
                     for ( i = col ; i < 9 ; i++ ){
                        car[row][i] = car[row][i + 1] ;
                        }
```

```
                    car[row][i] = NULL ;
                    }
                    else{
                       row = tarr[0] - 2 ;
                       if ( ! ( row < 0 ) ){
                          del ( scooter[row][col] ) ;
                          for ( i = col ; i < 9 ; i++ ){
                             scooter[row][i] = scooter[row][i + 1] ;
                             }
                          scooter[row][i] = NULL ;
                       }
                    }
                 }
                 else{
                    if ( type == CAR )
                       printf ( "\nInvalid car number, or a car with such
number has not been parked here.\n" ) ;
                    else
                       printf ( "\nInvalid scooter number, or a scooter
with such number has not been parked here.\n" ) ;
                 }
              }
              printf ( "\nPress any key to continue..." ) ;
              getch( ) ;
                break ;

        case 7:printf ( "Display\n" ) ;
              display_wq( ) ;
              printf ( "\nPress any key to continue..." ) ;
              getch( ) ;
              break ;

        case  8:exit(0);
     }
  }
}
```

Input and Output

```
Car Parking
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
1

Add:
Enter vehicle type (1 for Car / 2 for Scooter ):
1
Enter vehicle number: 1807
Car is Parking.

Press any key to continue...
Car Parking
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
1

Add:
Enter vehicle type (1 for Car / 2 for Scooter ):
2
Enter vehicle number: 2212
```

```
Enter vehicle number: 2212
Another Car in Process. Please Wait!!
Press any key to continue...
Car Parking
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
3
Total cars parked: 1

Press any key to continue...
Car Parking
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
4
Total scooters parked: 0

Press any key to continue...
Car Parking
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
```

```
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
5
Display
Cars ->
1807    0       0       0       0       0       0       0       0   0
0       0       0       0       0       0       0       0       0   0
Scooters ->
0       0       0       0       0       0       0       0       0   0
0       0       0       0       0       0       0       0       0   0

Press any key to continue...
Car Parking
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
7
Display

Wait Queue is...
2212
Press any key to continue...
Car Parking
1. Arrival of a vehicle
```

```
1. Arrival of a vehicle
2. Total no. of vehicles parked
3. Total no. of cars parked
4. Total no. of scooters parked
5. Display order in which vehicles are parked
6. Departure of vehicle
7. Display the Wait Queue
8. Exit
8
```

CONCLUSION

We have handled the parking problem using semaphore. Practical implementation of parking system can also be implemented using semaphores. Some more improvement can be done in our project as every vehicle has to wait until there is space available. If the problems is handled, this can be a good solution for the actual parking situations.
From this project we learnt a real-life scenario in which semaphores can be used. There are many such scenarios which can be efficiently handled by semaphores. Synchronization is required in many places around us, which tells us about the importance of semaphores.

REFERENCES

Lee, J. H. (2012). U.S. Patent No. 8,229,645. Washington, DC: U.S. Patent and Trademark Office.

Jung, H. G., Kim, D. S., Yoon, P. J., & Kim, J. (2006, August). Structure analysis-based parking slot marking recognition for semi-automatic parking system. In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR) (pp. 384-393). Springer, Berlin, Heidelberg.

Gomaa, H. (1984). A software design method for real-time systems. Communications of the ACM, 27(9), 938-949.

Sarkar, V. (1988, June). Synchronization using counting semaphores. In Proceedings of the 2nd international conference on Supercomputing (pp. 627-637). ACM.

Sirithinaphong, T., & Chamnongthai, K. (1999). The recognition of car license plate for automatic parking system. In Signal Processing and Its Applications, 1999. ISSPA'99. Proceedings of the Fifth International Symposium on (Vol. 1, pp. 455-457). IEEE.

Mehdipour, E., Mehdipour, F., Mehdipour, F., & Imani, K. (1986). U.S. Patent No. 4,603,390. Washington, DC: U.S. Patent and Trademark Office.

Burns, L. D. (2013). Sustainable mobility: a vision of our transport future. Nature, 497(7448), 181.

Rao, Y. R. (2017). Automatic smart parking system using Internet of Things (IOT). Int. J. Eng. Tech. Sci. Res, 4, 2394-3386.

Vlahogianni, E. I., Kepaptsoglou, K., Tsetsos, V., & Karlaftis, M. G. (2016). A real-time parking prediction system for smart cities. Journal of Intelligent Transportation Systems, 20(2), 192-204.

Kahn, P. R., Kinsolving, A., Vogel, D., & Christensen, M. A. (2018). U.S. Patent Application No. 10/043,388.

Volz, C. M. (2017). U.S. Patent Application No. 15/390,028.

Shahzad, A., Choi, J. Y., Xiong, N., Kim, Y. G., & Lee, M. (2018). Centralized Connectivity for Multiwireless Edge Computing and Cellular Platform: A Smart Vehicle Parking System. Wireless Communications and Mobile Computing, 2018.

Bales, J., Hafner, M., Smith, K., & Lavoie, E. M. (2018). U.S. Patent No. 9,981,656. Washington, DC: U.S. Patent and Trademark Office.