CMPE341 OPERATING SYSTEMS

PROJECT REPORT

Project No:1

Smart Indoor Parking Lot Management System using MATLAB

Submitted by:

Asuman Çağla Hepyükselenler (CMPE)- 20181901054

Figen Demir (CMPE)- 20201701020

Hatice Yaren Bulut (CMPE)- 20191710013

Samet Kartlar (CMPE)- 20201706016

Project Supervisor:

Assoc.Prof. Taner Arsan

Project Mentors:

Assoc.Prof. Nima Jafari Navimipour

Associate Prof, Rahim Dehkhargani,

Assistant. Prof. Seyed-Sajad Ahmadpour

Dr. Maryam Zaker

Faculty of Engineering and Natural Sciences

Kadir Has University

Fall 2023

TABLE OF CONTENTS

1 INTRODUCTION ……………………………………………………………… 3

1.1 Purpose of the Project ……………………………………………………....... 3

1.2 Usage of the Project…………………………………………….................. 4

2 OPERATING SYSTEM INTERACTION……………………………………… 3

2.1 Circuit and Operating System Connection………………………………..... 4

2.2 Circuit Components ……………………………………………………...... 8

2.3 Three Important Stages...

3 MAIN FINDINGS ……………………………………………………………. 9

3.1 Simulation Setup……………………………………………………………….. 9

3.2 Explaining the MATLAB Function Codes……………………………….......... 11

12

4 DISCUSSION………………………………………………………………

14

5 CONCLUSION………………………………………………………………

15

6 REFERENCES…………………………………………………………………

16

1 INTRODUCTION

With the increasing number of vehicles in today's rapidly developing and automated world, parking a car has become a big problem. It is very difficult to find a parking space, especially in crowded and closed areas. Problems such as loss of time experienced by vehicle owners and traffic congestion in the city made it necessary to find an effective solution to these problems. We aimed to find a systematic and effective solution to these problems with the smart parking garage management system.

1.1 PURPOSE OF THE PROJECT

This project aims to increase efficiency in indoor car parks, provide a better experience for vehicle owners, provide an effective management tool for car park operators, and offer an environmentally friendly approach and increase efficiency by optimizing the city's traffic flow. In this project, it is aimed to create a simple but effective and efficient system using MATLAB software.

1.2 USAGE OF THE PROJECT

The Smart Parking Management System was developed by writing a function in MATLAB and creating an interface with circuit elements. In a parking area with a Smart Parking System installed, incoming and outgoing traffic can be automatically controlled and synchronized. MATLAB's vehicle counting mechanism accurately tracks and counts incoming vehicles, ensuring accuracy in low-light conditions and providing real-time data on parking lot entry. In this way, drivers benefit from a fast and easy parking experience.

2 OPERATING SYSTEM INTERACTION

2.1 Interaction Between Circuit and Operating System

The functions set up through MATLAB can be considered as inputs to the main control unit of the system. They generate commands and assertions that at some point act as the system's microcontroller. The function receives input and output information from the circuit and processes the input and output states that the circuit elements receive through external triggering. This is how communication between the circuit elements and the operating system takes place. MATLAB functions play an active role in managing the data received from the circuit and establishing a connection with the operating system.

Buttons and Sensors:

The buttons and sensors in the circuit are responsible for monitoring changes associated with vehicle input and output by sensing input and output operations, with external triggering. At each of the input and output actions, MATLAB functions, the processing unit of the system that receives the command from the button, receives the signal and initiates the necessary actions according to the input or output status.

Seven Segments and LED:

LEDs and 2 seven-segment indicators are used to visually display vehicle entries. When viewed from the outside, these indicators inform the user whether the parking lot is empty or full. The connection of this information with the operating system is realized through the function codes installed in MATLAB. The function controls the status of the indicators and ensures that

2.1 CIRCUIT COMPONENTS

The circuit elements that interact with MATLAB functions and their internal features can be explained as follows:

DATA STORE:

In the block where the temporarily received data is stored, the information of the temporary variables and intermediate results are kept.

MEMORY BLOCK:

The memory block is used for tracking empty parking spaces. Each of these blocks, which keeps the free space information in memory, consists of 2 bits. In the information stored in arrays in the codes used in MATLAB functions, for example in the expression [1 0], the occupancy of the first parking space is expressed as 1 and the occupancy of the second parking space is expressed as 0.

PULSE SENDING BLOCK:

It is the circuit element that sends a pulse to the counter every time input and output operations are performed. Thanks to Pulsela, the counter blocks are updated and information is transmitted to the functions used in MATLAB.

COUNTER BLOCKS:

Flip-Flop

A flip-flop is a circuit element used in digital electronics and is often used to store information. This circuit element is a storage unit that has two basic states and can change its output with a control signal. They are generally used in computer memories, counter circuits, and digital systems.

Flip-flops come in several different types, but the most basic types include the RS flip-flop (Reset-Set flip-flop), D flip-flop (Data flip-flop), and JK flip-flop. These flip-flops change their output under certain conditions, which allows them to be used to store and process information in digital circuits.

Flip-flops can be found in electronic counters, memory elements, data loggers, and many other applications. Each type of flip-flop has a specific control logic and operating principle, so the preferred type is chosen depending on the application requirements.

JK Flip Flop

“JK flip flop operates on sequential logic principle, where the output is dependent not only on the current inputs but also on the previous state.”[1] It is an improved version of the RS (Reset-Set) flip-flop and offers more flexible use. A JK Flip Flop's core circuitry is made up of combination of logic gates, see Figure 1.1.

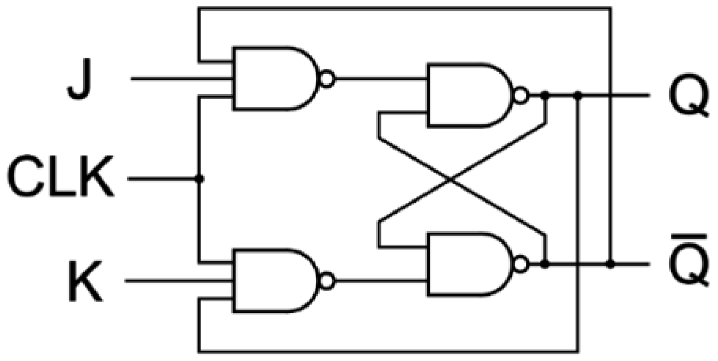


Figure 1.1 [2]

The main features of JK flip-flop are:

- J (Set) Input: This input turns the output of the flip-flop into "Set" state.
- K (Reset) Input: This input turns the output of the flip-flop to "Reset" state.
- Output (Inputs Q and Q'): The JK flip-flop has a pair of outputs: Q and Q' (inverter output). It varies depending on the Q, J and K inputs, and can also “toggle” depending on specific control logic.

Excitation table and the symbol of JK flip flop is given below as Table 1.1 and Figure 1.2, respectively.

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 1.1

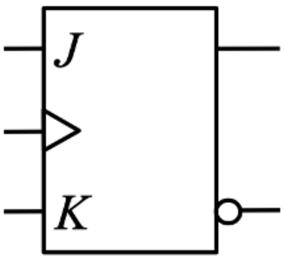


Figure 1.2

Characteristic equation of JK flip flop is given below as equation (1).

There are 2 counters in the circuit. Counters with 4 bits in total are created with 4 JK type Flip-Flops. In the circuit, each time the input and output operation occurs, it undertakes the task of updating the counter with pulse transmission. By taking 2 inputs, 4 outputs are given, which are given to the decoder element of the circuit. 16 different data can be represented.

BUTTONS FOR ENTER AND EXIT:

These are the circuit elements that allow input and output operations to be triggered manually by the user. These elements also transmit information to MATLAB functions when pressed.

SEVEN-SEGMENT DISPLAY BLOCKS:

Seven Segment Display

Seven-segment display is a type of display usually used to display numbers and some letters. This type of display usually consists of seven light-emitting diodes (LEDs), with each diode representing part of a region or segment.

Each segment is often referred to as a, b, c, d, e, f, and g, and combinations of these segments are used to form a number or letter as shown below in Figure 2.1.

Seven-segment displays are widely used in digital clocks, digital measuring instruments, microcontroller-based projects and many other applications. These indicators are usually combined with a simple control circuit and use a certain number of pins to control each segment. Each pin represents a specific segment, and by sending appropriate signals to these pins, the desired number or letter can be displayed.

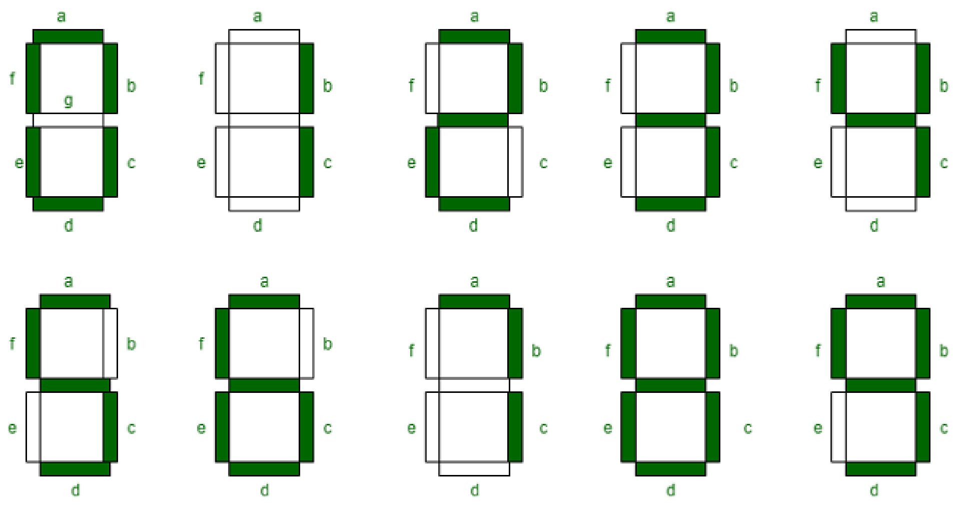


Figure 2.1 [3]

There are a total of 2 seven-segment display screens in the circuit. They are created to visually display the vehicle inputs and outputs in the circuit. The data updated in MATLAB functions are transmitted to the blocks and the user visually sees the occupancy and vacancy status.

MATLAB FUNCTION BLOCK:

It is the main block that provides the connection between the circuit and the software. It receives data from other elements of the circuit such as buttons and counters, updates them and transfers this data to functions.

BCD (BINARY CODED DECIMAL)

By using a decoder in the circuit, the data coming in decimally with 1s and 0s is converted into a format that the seven-segment can understand. The numerically incoming information receives input by converting the four-digit data input to decimal. The input is done in BCD and the value converted in BCD is given to the output, that is, to the seven segments. Seven segments with seven outputs (from a to g) output how the number in binary base will appear on a seven-segment display after the number is converted. Each digit of the BCD number interacts with one of the corresponding output segments. Each BCD input also has a truth table associated with the segments and the data in this table is related to which segment is activated.

Each segment of the 7-segment display

A: BCD[3] + BCD[2] + ~BCD[1] + BCD[0]

B: ~BCD[3] + BCD[2] + BCD[1] + ~BCD[0]

C: BCD[3] + ~BCD[2] + BCD[1] + BCD[0]

D: BCD[3] + BCD[2] + BCD[1] + ~BCD[0]

E: ~BCD[3] + BCD[2] + BCD[1] + BCD[0]

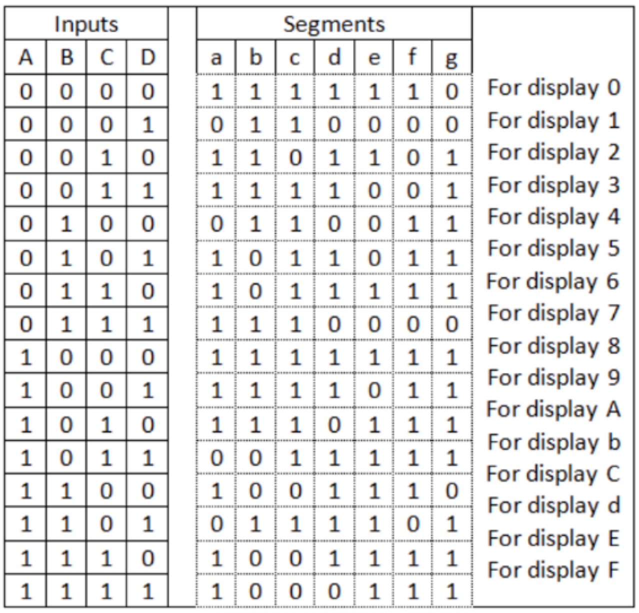
F: BCD[3] + ~BCD[2] + BCD[1] + BCD[0]

G: BCD[3] + BCD[2] + BCD[1] + BCD[0]

is integrated into the segment by a transformation. In addition, there are control signals and these signals determine which segment is active. In summary, in this block of the circuit, BCD to seven segments are transmitted.

When the vehicle enters, the enter button is manually activated and this information is quickly transmitted to MATLAB functions. The functions check for empty parking spaces in memory. In the codes within the function, code blocks related to occupancy and vacancy states have been created, when the function block of the circuit communicates with the functions, whichever state is examined in the code block is activated and control is provided.

Truth table of the decoder for common cathode type is given as Table 2.1 below.



Inputs					Segments							
A	B	C	D		a	b	c	d	e	f	g	
0	0	0	0		1	1	1	1	1	1	0	For display 0
0	0	0	1		0	1	1	0	0	0	0	For display 1
0	0	1	0		1	1	0	1	1	0	1	For display 2
0	0	1	1		1	1	1	1	0	0	1	For display 3
0	1	0	0		0	1	1	0	0	1	1	For display 4
0	1	0	1		1	0	1	1	0	1	1	For display 5
0	1	1	0		1	0	1	1	1	1	1	For display 6
0	1	1	1		1	1	1	0	0	0	0	For display 7
1	0	0	0		1	1	1	1	1	1	1	For display 8
1	0	0	1		1	1	1	1	0	1	1	For display 9
1	0	1	0		1	1	1	0	1	1	1	For display A
1	0	1	1		0	0	1	1	1	1	1	For display b
1	1	0	0		1	0	0	1	1	1	0	For display C
1	1	0	1		0	1	1	1	1	0	1	For display d
1	1	1	0		1	0	0	1	1	1	1	For display E
1	1	1	1		1	0	0	0	1	1	1	For display F

Table 2.1[4]

2.2 THREE IMPORTANT STAGES

Car Count:

This section consists of operations that track and update the number of vehicles in the parking lot. If any vehicle comes to the parking lot, one of the empty parking spaces must be occupied. At this stage, the current status of the parking lot should be constantly updated and instant information should be stored in the circuit.

Vehicle Entry:

At this stage, what will happen if any vehicle enters the parking lot is observed. Vehicle entry should be constantly simulated according to the occupancy rate of the parking lot. Since the total vehicle entry is limited, the capacity can be kept under control.

Vehicle Parking:

At this stage, the occupancy rates of the places in the parking lot should be constantly checked and visualized. LEDs and seven segments provide visual warnings to the user according to occupancy or vacancy status. On the other hand, when the user logs in and out manually, the process is done at this stage with the help of buttons.

The system gives customers a visual depiction of the number of cars in the parking lot as well as the number of occupied and vacant parking spaces when these three steps are combined. Simultaneously, it facilitates manual vehicle entry and exit, so aiding in parking lot control and administration.

3 MAIN FINDINGS

3.1 Simulation Setup

If a car enters the parking lot, that is, if the 'Enter' button is pressed; The first function updates the variable 'up_down' to 0.

If a car exits, that is, if the 'exit' button is pressed, the first function updates the 'up_down' variable to 1.

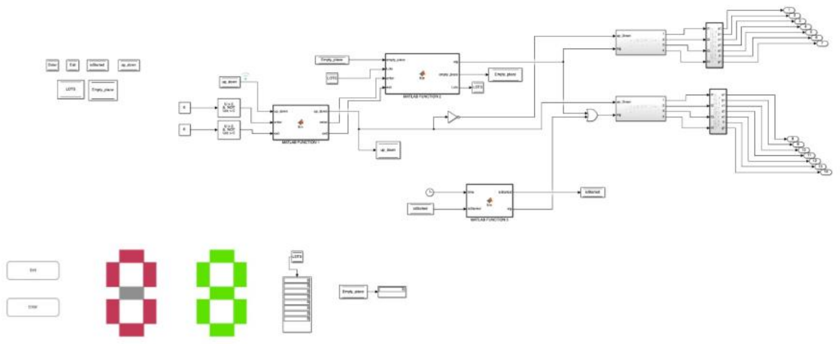
This 'up_down' variable controls the counters. The up_down value is transmitted to the counter circuit created for occupied parking areas using 'not gate'. However, it is directly transmitted to the counter created to count empty parking spaces. The reason for this is that if one increases, the other will decrease. Also, it is connected to the 'up_down Data Store Write' block and thus the up_down data is updated. Enter and exit values are sent to the input of the second function.

In the second function, if the 'enter' value is 1, that is, if there is a car entry, and there is an empty parking space (Empty_place>0) available, the empty parking space is found among the 8 available parking spaces and its value is updated to 1 which means the parking lot is now full. If there is an exit from the car park, that is, if the 'exit' value is 1, a full parking space is found among the 8 car park spaces, then the value of this space is updated to 0, which means the parking lot is now empty

At the same time, in the second function, the 'sig' value indicating the signal is changed from 0 to 1 for both vehicle input and vehicle exit. This value of 1 indicates that there is a change in the parking areas. Therefore, the 'sig' value is updated to 1 at each vehicle entry and exit. Then, this signal is sent to the Clk inputs of 4 JK flip-flops located in the 4-bit up/down counters.

Empty parking lots should be displayed as 8 at the beginning of the simulation. What the third function does is to increase the number displayed in the seven-segment from 0 to 8. This function was created because the counter of this segment will not receive a signal initially. This function assigns the current time of the simulation to the variable 'time'. A new variable 'isStarted' is initialized and it corresponds to the number empty parking lots. From the moment the simulation starts, the algorithm checks every 100 milliseconds whether the number of isStarted is reached 8 or not, and if this number is less than 8, it increases this number and, in every increment, it sends a signal to the counter circuit that used for counting the number of empty parking lots and stops sending the signal when isStarted variable reaches 8.

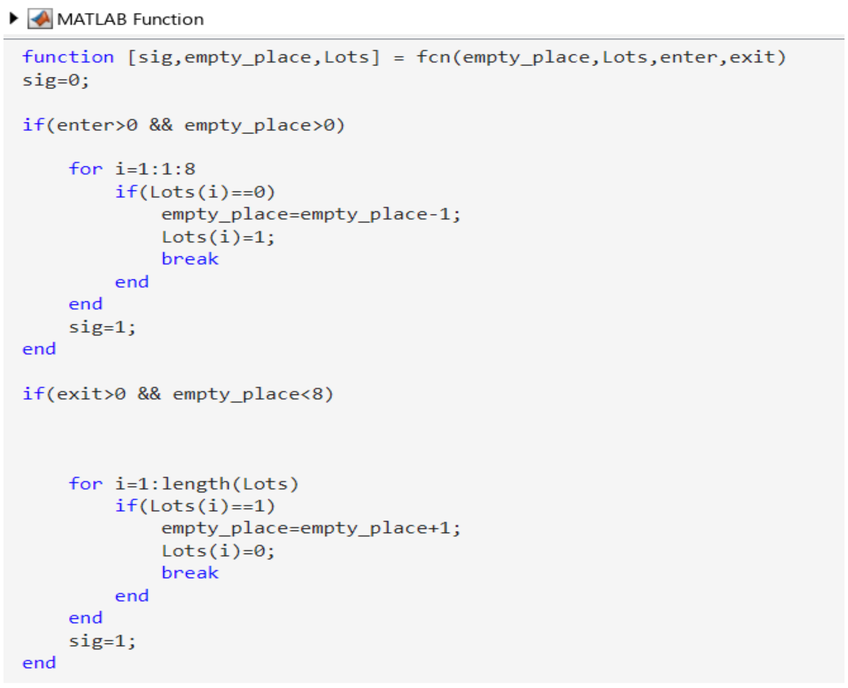
Initial circuit diagram is given as Figure below



Figure

After that, BCD to seven-segment decoder analyzes the BCD input that comes from the counter and provides an output to the seven-segment display by brightening the appropriate segments.

3.2 Explaining the MATLAB Function Codes



```
function [sig,empty_place,Lots] = fcn(empty_place,Lots,enter,exit)
sig=0;

if(enter>0 && empty_place>0)

    for i=1:1:8
        if(Lots(i)==0)
            empty_place=empty_place-1;
            Lots(i)=1;
            break
        end
    end
    sig=1;
end

if(exit>0 && empty_place<8)



    for i=1:length(Lots)
        if(Lots(i)==1)
            empty_place=empty_place+1;
            Lots(i)=0;
            break
        end
    end
    sig=1;
end
```

This MATLAB code defines a function named `fcn` that manages an indoor parking lot system based on entry and exit events. This is a break down the key components:

1. **Function Signature:**

function [sig, empty_place, Lots] = fcn(empty_place, Lots, enter, exit)

- This line declares a function named **fcn** with four input parameters (‘**empty_place**’, ‘**Lots**’, ‘**enter**’, ‘**exit**’) and three output values (‘**sig**’, ‘**empty_place**’, ‘**Lots**’).

2. **Variable Initialization:**

sig = 0;

- Initializes the ‘**sig**’ variable to 0. This variable is used to signal whether a parking operation (entry or exit) has occurred.

3. **Checking for Car Entry:**

if (enter > 0 && empty_place > 0)

- Checks if there are cars trying to enter (‘**enter > 0**’) and there are available parking spaces (‘**empty_place > 0**’).

```
for i = 1:1:8

  if (Lots(i) == 0)

    empty_place = empty_place - 1;

    Lots(i) = 1;
```

```
        break
    end
end
```

- If conditions are met, it iterates through parking spaces (‘**Lots**’) to find the first empty space (‘**Lots(i) == 0’**). Once found, it decreases the ‘**empty_place**’ count, marks the space as occupied (‘**Lots(i) = 1’**), and breaks out of the loop. It sets ‘**sig’** to 1, indicating a successful operation.

4. Checking for Car Exit:

```
if (exit > 0 && empty_place < 8)
```

- Checks if there are cars trying to exit (‘**exit > 0’**) and there are occupied parking spaces (‘**empty_place < 8’**).

```
for i = 1:length(Lots)
    if (Lots(i) == 1)
        empty_place = empty_place + 1;
        Lots(i) = 0;
        break
    end
end
```

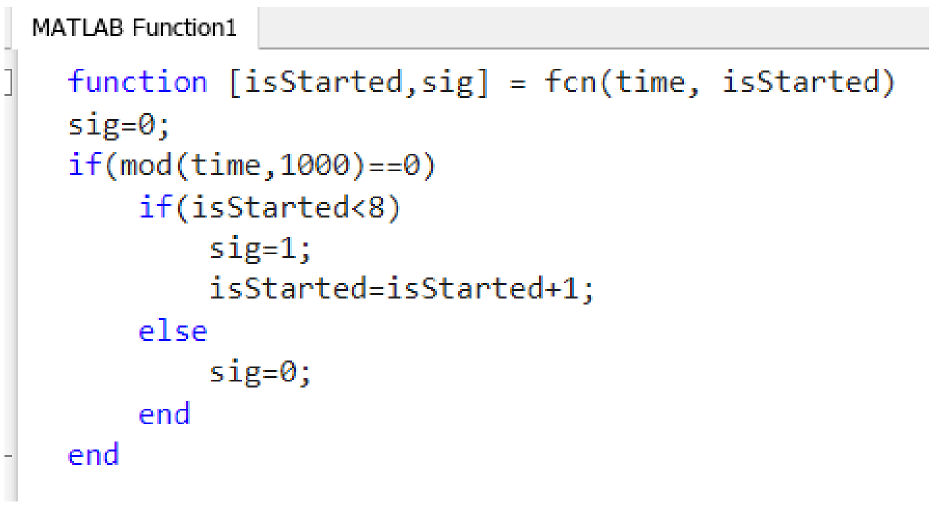
- If conditions are met, it iterates through parking spaces to find the first occupied space (‘**Lots(i) == 1’**). Once found, it increases the ‘**empty_place**’ count, marks the space as empty (‘**Lots(i) = 0’**), and breaks out of the loop. It also sets ‘**sig’** to 1.

5. Output:

- The function returns three values: sig (1 if a car entered or exited successfully, 0 otherwise), empty_place (the updated count of available parking spaces), and Lots (the updated array representing the status of each parking space).

In summary, this MATLAB function simulates cars entering and exiting a parking lot, updating the status of parking spaces accordingly.

Matlab Function 1


MATLAB Function1

```
function [isStarted,sig] = fcn(time, isStarted)
sig=0;
if(mod(time,1000)==0)
    if(isStarted<8)
        sig=1;
        isStarted=isStarted+1;
    else
        sig=0;
    end
end
```

1. Initialization:

- Set `sig` to 0.

2. Check if `time` is a multiple of 1000:

- If not, return with unchanged values for `isStarted` and `sig`.
- If yes, proceed to the next step.

3. Check if `isStarted` is less than 8:

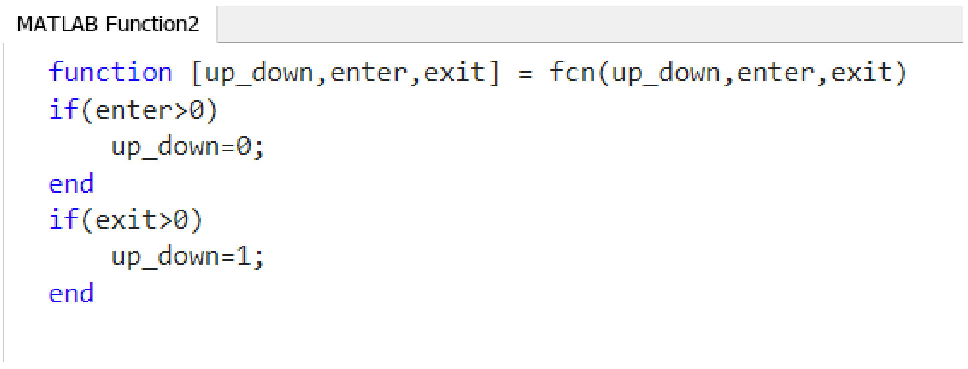
- If yes, set `sig` to 1 and increment `isStarted` by 1.
- If no (if `isStarted` is already 8 or greater), set `sig` to 0.

4. Return the updated values:

- Return the updated values of `isStarted` and `sig`.

In summary, the function increments a counter (`isStarted`) by 1 whenever `time` is a multiple of 1000, up to a maximum of 8 times. If the counter is already at 8 or greater, it does not increment further and sets `sig` to 0.

Matlab Function 2



```
function [up_down,enter,exit] = fcn(up_down,enter,exit)
if(enter>0)
    up_down=0;
end
if(exit>0)
    up_down=1;
end
```

1. **Check if enter is greater than 0:**

- If enter is greater than 0, set up_down to 0.

2. **Check if exit is greater than 0:**

- If exit is greater than 0, set up_down to 1.

3. **Return the updated values:**

- The function returns the updated values of up_down, enter, and exit.

In summary, this function appears to control the value of up_down based on the values of enter and exit. If enter is greater than 0, up_down is set to 0. If exit is greater than 0, up_down is set to 1. If neither enter nor exit is greater than 0, up_down retains its original value.

4 DISCUSSION

MATLAB was used in the development of the Smart Indoor Parking Lot Management System project, which provides a smart parking management system that integrates digital logic circuits and operating systems. Three primary phases of this project—vehicle entrance, counting, and parking spot distribution—are executed seamlessly in conjunction with the operating system. Vehicle counting updates the number of cars and saves them in a database or temporary memory, whereas vehicle entry saves the incoming data to the operating system. The operating system's temporary memory is where parking space allocation keeps track of the locations of the cars. In addition to effectively completing the project, this integrated strategy helps engineers become more adept at handling problems with less resources. The vehicle meter, memory, entry and exit buttons, five LED lights, and other parts show how well-thought-out and user-friendly the system is. Furthermore, it is highlighted that the project used cost-effective solutions in an inexpensive manner. This initiative offers an example for creative and economical problem-solving and is a significant step toward the development of smart parking management systems in the future.

5 CONCLUSION

In conclusion, the MATLAB project Smart Indoor Parking Lot Management System effectively combines real-world uses for digital logic circuits and operating systems. This project exemplifies an engineering competence that targets efficient communication and synchronization between the operating system and hardware components in order to increase problem-solving abilities in limited circumstances. The operating system treats vehicle entrance as login data, and pertinent details like the license plate number and entry time are noted. The operating system has a function called "vehicle counting" that tracks the number of cars coming and going and updates information in a database or temporary memory. The operating system uses its temporary memory to store variables that represent the locations of parked cars. A thorough design is demonstrated by the addition of components including memory for unoccupied parking spots, car counter, entrance and exit buttons, and five LED lights. Two seven-segment displays that indicate the number of occupied and empty parking spots are also included, and a translator is used to translate the parking space numbers into BCD code. This multifaceted strategy is in line with the goal of addressing the project's main objectives and fostering the growth of engineering problem-solving abilities. Budget-friendly solutions are used to highlight how useful and economical the system is. Consequently, the Smart Indoor Parking Lot Management System presents a comprehensive amalgamation of digital reasoning, operational frameworks, and efficient correspondence, furnishing a novel and economical resolution for smart parking administration.

6 REFERENCES

1- https://byjus.com/gate/jk-flip-flop/
2- https://dcaclab.com/blog/j-k-flip-flop-explained-in-detail/
3- https://www.geeksforgeeks.org/seven-segment-displays/
4- https://electronics-fun.com/7-segment-hex-decoder/
5- Radhakrishnan, N., Yeluri, A., & Singh, G. (2022). Smart Parking System using MATLAB.ResearchGate. https://www.researchgate.net/publication/366320845_Smart_Parking_System_using_MATLAB
6- Automated Parking Valet - MATLAB & Simulink. (n.d.). https://www.mathworks.com/help/driving/ug/automated-parking-valet.html
7- GeeksforGeeks. (2019, November 25). BCD to 7 segment Decoder. https://www.geeksforgeeks.org/bcd-to-7-segment-decoder/
8- Integrated Circuits - MATLAB & Simulink. (n.d.). https://www.mathworks.com/help/sps/integrated-circuits.html