A

Project Report on

Implementation of Car Parking System using VEGA Processor

Submitted

in partial fulfillment of the requirements for
the award of the degree of

BACHELOR OF TECHNOLOGY

in

Electronics & Communication Engineering

By

Maganti Shanmukha Sri Datta – 20R11A0494

under the supervision of

Dr. P. Vijai Bhaskar

Professor, ECE, Dean Academics

Department of Electronics and Communication Engineering
GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY
(UGC Autonomous)

(Accredited by NBA, Accredited by NAAC with A+ grade, Approved by AICTE, New Delhi and Affiliated to JNTUH)
Cheeryal (V), Keesara (M), Medchal Dist, Hyderabad– 501 301, Telangana State

2023-2024

GEETHANJALI COLLEGE OF ENGINEERING AND TECHNOLOGY

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the project report titled **Implementation of Car Parking System using VEGA Processor** being submitted by **Maganti Shanmukha Sri Datta** bearing hall ticket number **20R11A0494** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in ***Electronics & Communication Engineering*** is a record of bonafide work carried out under my supervision.

Dr. P. Vijai Bhaskar
Professor, ECE, Dean Academics

Dr. G Sreelakshmi
HoD - ECE

Internal Examiner

External Examinar

ACKNOWLEDGEMENT

I **Maganti Shanmukha Sri Datta,** student of ECE department of Geethanjali College of Engineering and Technology would like to convey heartfelt thanks to **Dr. S. Udaya Kumar**, Principal of the college for the inspiration and encouragement given to me to move ahead in the completion of this project.

I am highly grateful to **Dr. G. Sreelakshmi**, Head of the Department of **Electronics and Communication Engineering** of **GCET** for the support extended.

I am very happy to be supervised by **Dr. P. Vijai Bhaskar, Professor, ECE, Dean Academics** for his able guidance during the completion of the proposed work successfully.

I am also thankful for the members of the project review committee for the timely suggestions which helped a lot to complete our project work as per schedule.

With Regards

Maganti Shanmukha Sri Datta (20R11A0494)

TABLE OF CONTENTS

Abstract

In today's rapidly advancing technological world, cars have become an essential necessity, whether they are electric or mechanical, with their demand increasing worldwide due to the growing population. However, this increase in vehicles has caused one of the biggest issues in the parking sector, which is a problem that many people are affected by. While manual car parking systems are in place, issues persist in crowded locations such as multiplexes and shopping malls, where traffic congestion becomes a challenge to manage through traditional labor-intensive methods. To address these challenges, modern technologies like Internet of Things (IoT) models and frameworks come into play. This project aims to reduce human efforts and offer multiple benefits compared to traditional parking and toll management methods. The IoT model integrates various hardware components, such as RFID (Radio Frequency Identification) cards, IR (Infra-Red) sensors, Servo motors, and an ESP 8266 WiFi module. Moreover, it leverages critical software's such as Arduino IDE, Ada fruit IO, and IFTTT. When users interact with the Smart Car Parking System, their data is processed and automatically converted into a user-friendly digital format and helps for the users and parking management authorities improving user experience by providing an ideal and trouble-free process for vehicle entry and exit. IoT-based parking systems can offer real-time solutions regarding parking availability in different areas. Through mobile apps or digital signage, drivers can access this information, allowing them to locate available parking spaces more efficiently. This int urn reduces traffic congestion caused by traffic drivers circling around in search of parking spots. This Technique saves time and effort in the modern world.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS USED

IoT	-	**I**nternet **o**f **T**hings
RFID	-	**R**adio **F**requency **ID**entification
CDAC	-	**C**entre for **D**evelopment of **A**dvanced **C**omputing
RISC	-	**R**educed **I**nstruction **S**et **C**omputing
FPGA	-	**F**ield **P**rogrammable **G**ate **A**rray
NFS	-	**N**etwork **F**ile **S**ystem
MPU	-	**M**emory **P**rotection **U**nit
PWM	-	**P**ulse **W**idth **M**odulation
SPI	-	**S**erial **P**eripheral **I**nterface
UART	-	**U**niversal **A**synchronous **R**eceiver / **T**ransmitter
IO	-	**I**nput **O**utput
RAM	-	**R**andom **A**ccess **M**emory
GPIO	-	**G**eneral **P**urpose **I**nput **O**utput
SoC	-	**S**ystem **o**n **C**hip
SRAM	-	**S**tatic **R**andom **A**ccess **M**emory
DC	-	**D**irect **C**urrent
IR	-	**I**nfra**R**ed
IFTTT	-	**I**f **T**his **T**hen **T**hat
URL	-	**U**niform **R**esource **L**ocator
UID	-	**U**nique **I**dentification **N**umber

CHAPTER 1 – INTRODUCTION

In an era of rapid urbanization and technological advancement, the management of parking spaces has become a critical challenge in urban areas worldwide. Traditional parking systems are often plagued by inefficiencies, leading to traffic congestion, wasted time, and environmental pollution. Smart Parking Systems offer a promising solution to address these issues by leveraging cutting-edge technologies such as Internet of Things (IoT), data analytics, and automation to optimize parking space utilization and enhance the overall parking experience for users.

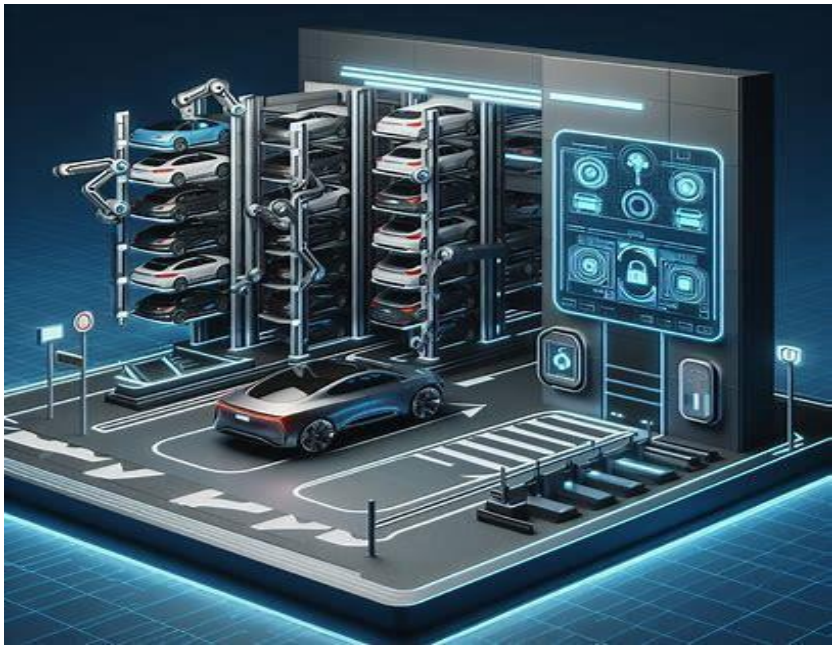


Fig 1.1 : Smart Car Parking System

1.1 Understanding the Need for Smart Parking Systems

- **Urbanization and Congestion:** With the continuous influx of people into urban areas, the demand for parking spaces has surged, exacerbating issues related to congestion and traffic flow.

- **Inefficient Use of Parking Space:** Conventional parking systems often suffer from inefficiencies such as underutilization of parking spaces, difficulty in finding available spots, and lack of real-time information.
- **Environmental Impact:** The excessive circling of vehicles in search of parking not only contributes to traffic congestion but also leads to increased emissions, air pollution, and fuel consumption.

1.2 Key Features and Components of Smart Parking Systems

- **Sensor Technology:** Smart parking systems deploy various sensor technologies such as ultrasonic sensors, infrared sensors, and magnetic sensors to detect the presence of vehicles in parking spaces accurately.
- **Data Analytics:** Advanced data analytics algorithms process the data collected from sensors to provide real-time insights into parking space availability, occupancy patterns, and demand forecasts.
- **Mobile Applications:** User-friendly mobile applications enable drivers to access real-time information about available parking spaces, reserve parking spots in advance, and navigate to the nearest vacant spot seamlessly.
- **Automated Payment Systems:** Integration with payment gateways allows for cashless transactions, enabling users to pay for parking fees conveniently through mobile apps or RFID-enabled cards.
- **IoT Connectivity:** Smart parking systems leverage IoT connectivity to facilitate communication between sensors, mobile applications, and backend servers, enabling seamless data exchange and system integration.
- **Infrastructure Integration:** Integration with existing infrastructure such as parking meters, street lights, and traffic signals enables holistic management of urban mobility and parking resources.

1.3 Benefits of Smart Parking Systems

- **Improved User Experience:** Smart parking systems streamline the parking process, reducing the time and effort required to find parking spaces, thus enhancing the overall user experience for drivers.
- **Optimal Space Utilization:** By providing real-time insights into parking space availability and occupancy, smart parking systems optimize the utilization of parking resources, minimizing wastage and maximizing efficiency.
- **Reduced Traffic Congestion:** The efficient allocation of parking spaces helps reduce the need for vehicles to circle in search of parking, thereby alleviating traffic congestion and improving traffic flow.
- **Environmental Sustainability:** By minimizing unnecessary vehicle movements and idling, smart parking systems contribute to reducing greenhouse gas emissions, air pollution, and fuel consumption, promoting environmental sustainability.
- **Data-driven Decision Making:** The rich data collected by smart parking systems enables city planners and administrators to make informed decisions regarding urban mobility planning, infrastructure development, and policy formulation.

Fig 1.2 : Smart Car Parking System with IoT

1.4 Challenges and Future Directions

- **Initial Investment Costs:** The deployment of smart parking systems involves significant upfront costs for infrastructure setup, sensor installation, and technology integration, posing a barrier to adoption for some municipalities and parking operators.
- **Data Privacy and Security:** Collecting and processing sensitive data such as vehicle movements and user information raises concerns regarding data privacy and cybersecurity, necessitating robust measures to safeguard against unauthorized access and misuse.
- **Scalability and Interoperability:** Ensuring the scalability and interoperability of smart parking systems is crucial for their widespread adoption and integration with existing urban infrastructure, requiring standardized protocols and open data formats.
- **Emerging Technologies:** The continued advancement of technologies such as artificial intelligence, edge computing, and 5G connectivity holds the potential to further enhance the capabilities and performance of smart parking systems, opening up new possibilities for innovation and improvement.

Fig 1.3 : Traffic Congestion

CHAPTER 2 – LITERATURE SURVEY

Smart car parking systems have emerged as a transformative solution to alleviate the challenges associated with traditional parking management. This section provides a comprehensive review of existing literature, discussing key aspects such as Overview of Smart Parking System, technologies, RFID Technology, benefits, challenges, and future directions.

2.1 Overview of Smart Parking System

The ideal of creating a Smart City is now becoming possible with the emergence of the Internet of Things. One of the key issues that smart cities relate to are car parking facilities and traffic management systems[11]. In present day cities finding an available parking spot is always difficult for drivers, and it tends to become harder with ever increasing number of private car users. This situation can be seen as an opportunity for smart cities to undertake actions in order enhance the efficiency their parking resources thus leading to reduction in searching times, traffic congestion and road accidents. Problems pertaining to parking and traffic congestion can be solved if the drivers can be informed in advance about the availability of parking spaces at and around their intended destination. Recent advances in creating low-cost, low-power embedded systems are helping developers to build new applications for Internet of Things. Followed by the developments in sensor technology, many modern cities have opted for deploying various IoT based systems in and around the cities for the purpose of monitoring. A recent survey performed by the International Parking Institute [12] reflects an increase in number of innovative ideas related to parking systems. At present there are certain parking systems[13] that claim to citizens of delivering real time information about available parking spaces. Such systems require efficient sensors to be deployed in the parking areas for monitoring the occupancy as well as quick data processing units in order to gain practical insights from data collected over various sources

Fig 2.1 : Overview of Smart Parking System

2.2 Technologies

Smart parking systems leverage a variety of technologies to enable efficient parking space management. These include sensor technologies such as ultrasonic sensors, infrared sensors, and magnetic sensors for real-time detection of vehicle presence [15, 17, 19]. Additionally, integration with Internet of Things (IoT) platforms facilitates seamless communication between sensors, mobile applications, and backend servers [20, 22]. Data analytics algorithms process the collected data to provide actionable insights into parking space availability, occupancy patterns, and demand forecasts [16, 18, 21].

2.3 RFID Technology

Radio Frequency IDentification is an emerging technology with a vast array of applications in many industries and helps to identify animate or inanimate objects through radio waves [8]. It was invented in the years 1930-1940 during World War II to identify allied from enemy planes but it was stilling unknown for commercial applications until the 1980s [9]. Nowadays, RFID has an important role in many fields due to its various benefits and features such as supply chain management including production and delivery of products, object tracking, smart parking systems, etc.

The main components of an RFID system are [10]: • **readers or interrogators.** • **tags or transponders.** The readers can identify the tags attached to an object, an animal, or a person within a given radio frequency range through radio waves without the need of human intervention or data entry for tracing purposes. The principle is to emit a radio frequency signal towards tags which in return use the energy issued from the transmitted reader signal to reflect their response [10].

RFID uses different frequency ranges for the user purpose:

- The low-frequency range is between 125-134 kHz.
- The High-frequency range is 13.56 MHZ.
- Ultra-high-frequency range is 908 MHZ.
- Microwave frequency range is 2.4 GHz.

Fig 2.2 : RFID Technology

2.4 Benefits and Impacts

Smart car parking systems offer a multitude of benefits to both users and urban environments. By optimizing parking space utilization and reducing the need for vehicles to circle in search of parking, these systems alleviate traffic congestion and improve traffic flow [14, 19, 22]. Furthermore, the reduction in vehicle idling and unnecessary movements leads to decreased emissions, promoting environmental sustainability [15, 17, 23]. Enhanced data-driven decision-making enables city planners to develop more efficient urban mobility strategies, contributing to overall urban development [20,21].

2.5 Challenges faced

Despite their potential benefits, smart parking systems face several challenges that warrant consideration. Initial investment costs associated with infrastructure setup, sensor installation, and technology integration pose barriers to adoption for some municipalities and parking operators [15, 18, 22]. Data privacy and security concerns arise due to the collection and processing of sensitive information, necessitating robust measures to safeguard against unauthorized access and misuse [16, 19, 21]. Interoperability issues and scalability challenges also need to be addressed to ensure seamless integration with existing urban infrastructure [14, 17, 20].

2.6 Future Directions

The future of smart car parking systems lies in continued technological innovation and strategic planning. Advancements in artificial intelligence, edge computing, and 5G connectivity hold the potential to further enhance the capabilities and performance of these systems [16, 18, 23]. Standardization efforts and collaboration among stakeholders are crucial to addressing interoperability challenges and fostering widespread adoption [15, 19, 22]. Moreover, emphasis should be placed on research and development initiatives aimed at addressing emerging needs and optimizing system efficiency [17, 21]

CHAPTER 3 – PROPOSED METHODOLOGY

In the proposed methodology, we used VEGA Processor, which is a new processor.

3.1 Introduction to VEGA Processors

Centre for Development of Advanced Computing (CDAC) has successfully developed VEGA series of microprocessors in soft IP form, viz. 32-bit single-core (in-order), 64-bit single-core (in-order & out-of-order), 64-bit dual-core (out-of-order), and 64-bit quad-core (out-of-order). These high performance processors are based on the open source RISC-V Instruction Set Architecture with Multilevel Caches, Memory Management Unit and Coherent Interconnect. The Processor IP have been integrated with a wide range of indigenously developed Silicon proven system and peripheral IPs and ported on Field Programmable Gate Array (FPGA) development platforms, Linux/FreeRTOS ported, and performance demonstrated by benchmarking and executing various applications. Face detection, object detection, seamless image merging, and media server are some of the applications successfully ported. Also, Linux with X-Windows (GUI based Linux) and Network File System (NFS) was booted successfully with monitor, keyboard, and mouse interfaced to the 64-bit single-core processor ported on FPGA Development Platform.

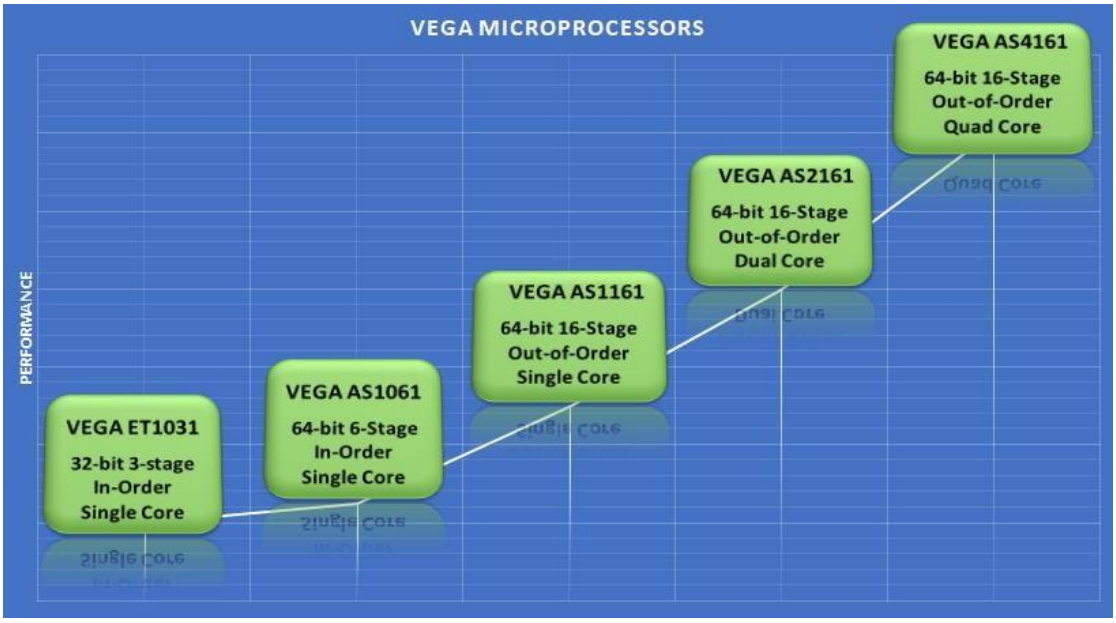


Fig 3.1 : Different types of VEGA Series Microprocessor

3.2 VEGA ET1031 Processor

In this project, we are going to use one of the VEGA SERIES Processor which is ET1031 Processor. VEGA ET1031 is a compact and efficient 3-stage in-order 32-bit RISC-V processor core which features separate instructions and data bus to increase performance. This Microprocessor can be used as an effective work horse in low power IoT applications.

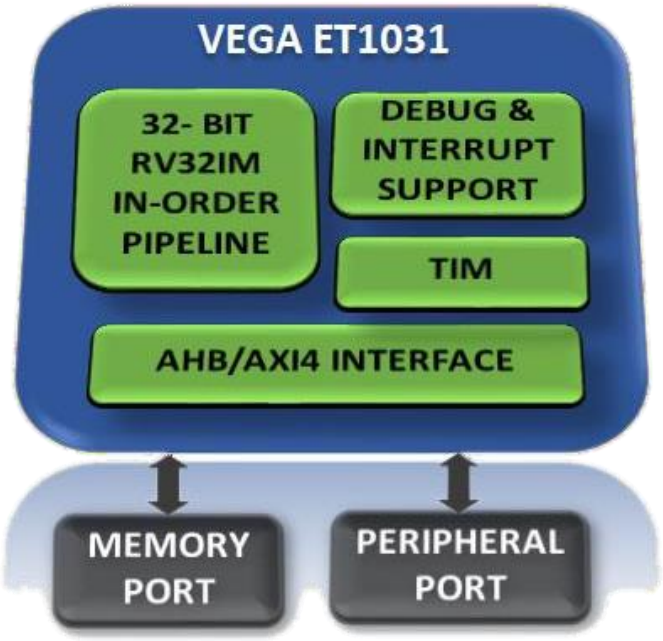


Fig 3.2 : Architecture of VEGA ET1031 Processor

Key Features of VEGA ET1031 Processor

- RISC-V (RV32IM) Instruction Set Architecture
- It supports Superscalar Architecture
- Open source, flexible and customisable
- 3 Stage in-order pipeline implementation
- Harvard architecture (separate instructions and data buses)
- High performance multiply/divide unit
- Configurable AXI4 or AHB external interface
- Optimal MPU (Memory Protection Unit)

- Platform Level Interrupt Controller
  - Up to 127IRQs
  - Low interrupt latency
- Vectored Interrupt Support
- Advanced Integrated Debug Controller

Applications of VEGA ET1031 Processor

- Sensor fusion
- Smart Meter
- System supervisors
- Remote sensors
- Small IoT devices
- Wearable devices
- Toy and electronic education equipment
- Legacy 8/16-bit applications
- Industrial networking

3.3 Hardware Requirements

3.3.1 VEGA ARIES V2.0 Board

The VEGA Microprocessor is a indigenous developed processor, similar to popular boards like Arduino, Raspberry Pi, and NodeMCU. Among various VEGA processor variants, we are specifically utilizing the ET1031 processor. The ET1031 is a 32- bit, 3-stage, in-order RISC-V processor core designed to be compact and efficient[4]. Its low power consumption makes it ideal for reliable use in Internet of Things (IoT) applications such as Sensor fusion, Smart meter, Remote sensors, and wearable devices. To facilitate development with this microprocessor, a platform called ARIES v2.0 is employed. The ARIES v2.0 is based on THEAJS32 ASIC, which operates at a frequency of 100MHz[4].

Fig 3.3 : VEGA ARIES V2.0 Board

Controller	THEJAS32
SRAM	256KB
Flash	2MB
Input Voltage	7 – 12 V
PWM Pins	8
Analog Input Pins	4
SPI	3
UART	3
I2C	2
GPIOs	32
DC Current Per I/O Pin	12 mA
IO Voltage	3.3 V
Clock Speed	100 MHz
Length	78 mm
Width	66 mm

Table 3.1 : Technical Specifications of VEGA ARIES V2.0 Board

Processor	VEGA ET1031
RAM	256 KB
PWMs	8
Timer	3
SPI	4
GPIO	32
UART	3
I2C	3
Frequency	100 MHz
IO Voltage	3.3 V
Core Voltage	1.2 V
Package	LQFP 128

Table 3.2 : Technical Specifications of System on Chip THEJA S32

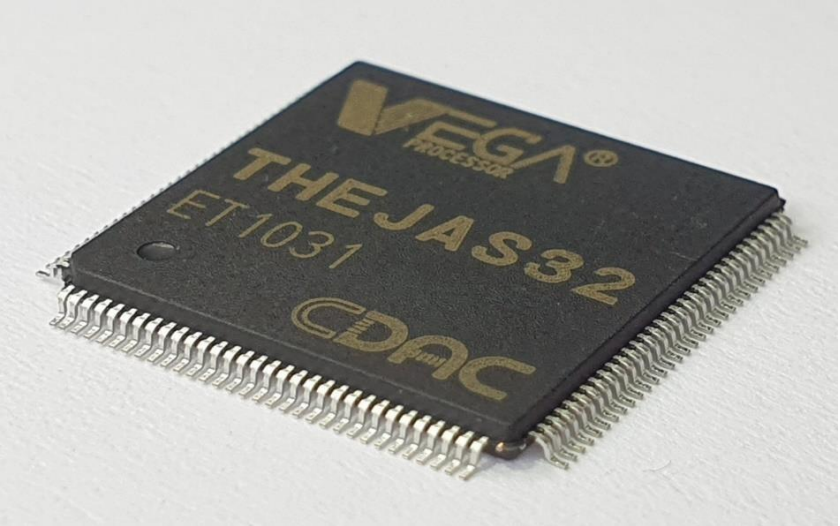


Fig 3.4 : System on Chip THEJA S32

There are advantages of choosing VEGA Processor than Arduino which is shown below as a comparison:

Basis	**VEGA V2.0**	**Arduino UNO**
Architecture	RISC - V	AVR
Microcontroller	THEJA S32	ATMega328
GPIO Pins	32	14
Clock Frequency	100 MHz	16 MHz
PWM Pins	8	6
SRAM	256 KB	2 KB
Flash Memory	2 MB	32 KB
UARTs	3	1
DC Per GPIO Pin	12 mA	40 mA
IO Voltage	3.3 V	5 V
I2C	2	1

Table 3.3 : Comparison between VEGA V2.0 and Arduino UNO

3.3.2 ESP01 Wifi Module

The ESP WiFi module is a essential part manufactured by Espressif Systems, designed for use in both the ESP8266 and ESP32 families. These modules serve as Wi-Fi connectivity solutions and find widespread usage in IoT applications due to their affordability, low power consumption, and user-friendly nature[6]. They are highly versatile and are extensively applied in various fields, including smart devices, sensor networks, home automation, and industrial IoT implementations.

Fig 3.5 : ESP01 Wifi Module

The ESP modules come equipped with an array of integrated peripherals, such as UART (Serial), I2C, SPI, ADC, DAC, and PWM, among others. These additional features enable seamless sensor interfacing, external device control, and facilitate seamless communication with other devices.

3.3.3 RFID (Radio- Frequency Identification) Tags and Readers

RFID tags can be classified into three main groups: passive tags, active tags, and battery-assist tags.A fixed UID (Unique Identification) value for passive tags is predetermined by the manufacturer and cannot be altered. In contrast, active tags come with programmable memory, allowing read and write operations. Battery-assist tags are a combination of active and passive tags[6]. In this model, the RFID reader module employed is the MFRC522, which operates at a frequency of 13.56MHz[2]. The RFID system in use utilizes passive tags. These tags are scanned both during entry and exit using the RFID reader.

Fig 3.6 : MFRC522 Reader and Tags

3.3.4 Servo Motor

Servo motors are employed in applications requiring rotary motion. They act as actuators with a feedback device and control circuitry. The control circuit receives a control signal, often in the form of pulse-width modulation (PWM), to determine the desired position or speed of the motor shaft. The feedback device supplies information about the current position, enabling the control circuit to adjust the motor's output to achieve the desired position.

Fig 3.7 : Servo Motor

In this specific scenario, the servo motor is utilized to raise the barrier upon reading an RFID tag from the receiver. The barrier will only open if the tag's code matches the one provided by the administrator; otherwise, the barriers will remain closed.

3.3.5 Infrared (IR) Sensors

IR Sensors have played a significant role in advancing IoT technology by offering valuable capabilities in various applications such as smart parking system. These sensors have been seamlessly integrated into connected devices, granting users the ability to control them remotely through mobile applications or voice assistants. Alongside other sensing technologies like cameras and ultrasonic sensors, IR sensors are now utilized in IoT systems for object detection and tracking purposes. They excel at detecting the presence of objects, measuring distances, and facilitating localization in parking management system sent applications.

Fig 3.8 : Infrared (IR) Sensor Module

3.4 Software Requirements

3.4.1 Arduino IDE

The Arduino IDE serves as a userfriendly tool designed to streamline the programming process for Arduino boards and various other boards[3]. It also enables coding for VEGA processors, facilitating this by importing VEGA Aries boards. In order to program the specific VEGA board successfully, one needs to configure the IDE settings to match the appropriate board type, processor, and serial port.

Steps to use VEGA Board using Arduino IDE Software:

- Download and Install Arduino IDE 1.8.19
  https://downloads.arduino.cc/arduino-1.8.19-windows.exe
  Open Arduino IDE
- Open File->Preferences

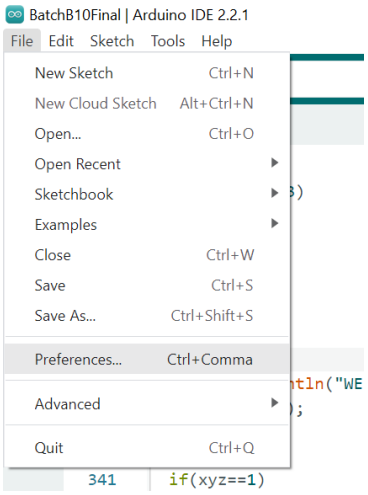


Fig 3.9 : Step 1

Add JSON in https://gitlab.com/riscv-vega/vega-arduino/-/raw/main/package_vega_index.json , Press OK

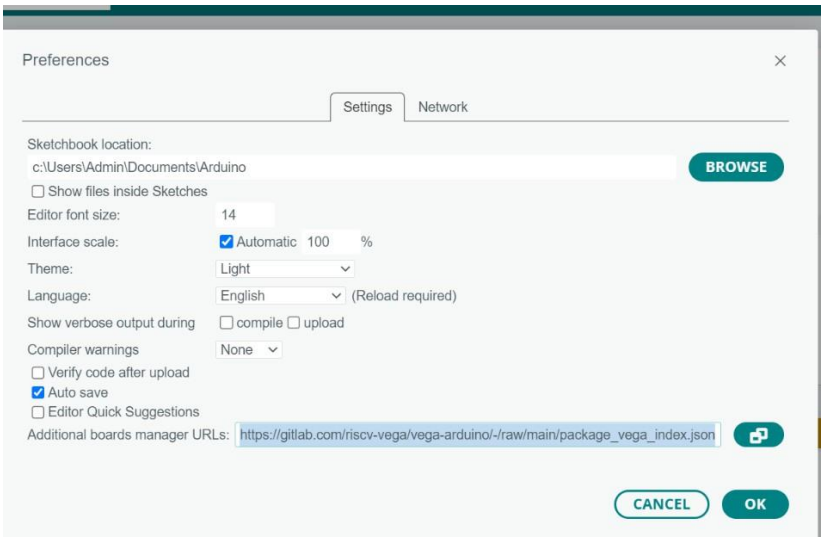


Fig 3.10: Step 2

Install "VEGA ARIES Boards"

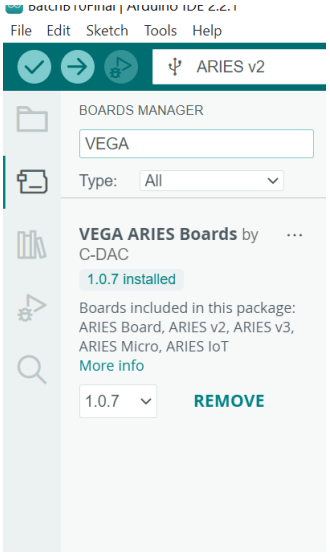


Fig 3.11 : Step 3

- **Select Tools->Board->VEGA Processor: ARIES Boards->ARIES v2**

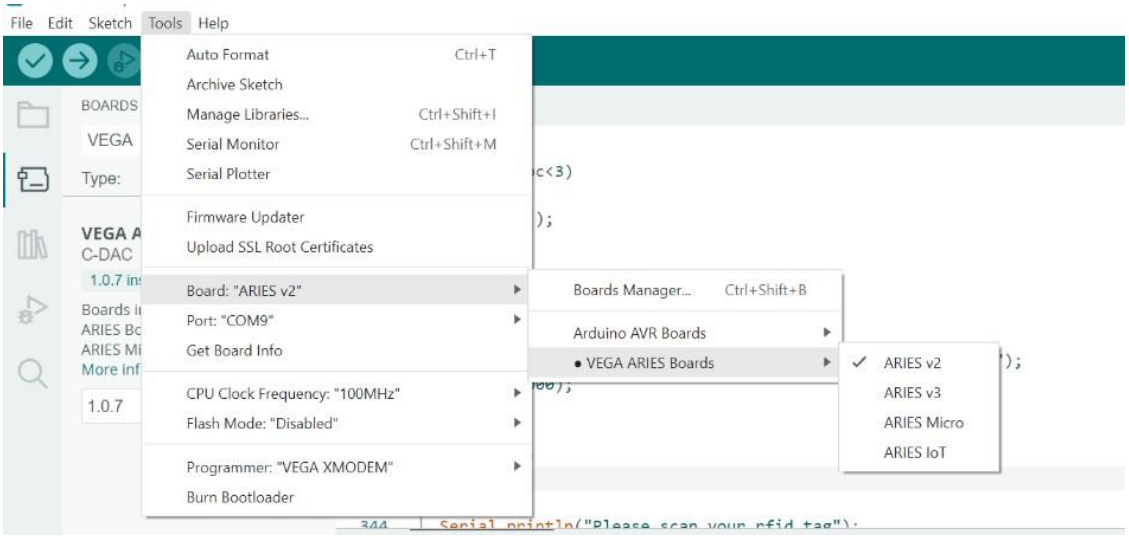


Fig 3.12 : Step 4

If

BOOT-SEL jumper (J12) is shorted :

a) Select Flash Mode -> Enabled

b) Select Tools -> Programmer -> VEGA FLASHER

Else

a) Select Flash Mode -> Disabled

b) Select Tools -> Programmer -> VEGA XMODEM

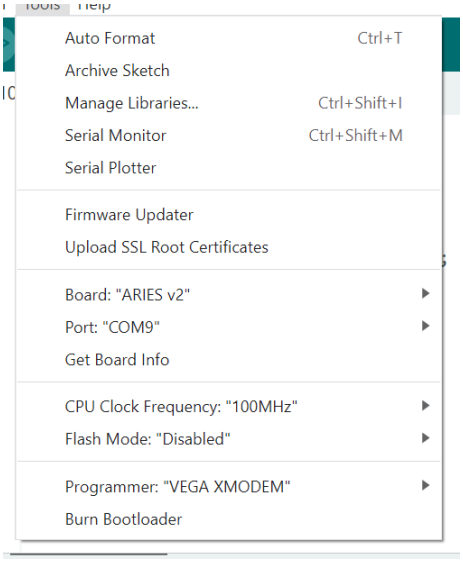


Fig 3.13 : Step 5

- Select 115200 baud in Serial Monitor

3.4.2 Adafruit IO

Ada fruit IO is a cloud-based server platform designed for IoT projects, offering various data visualizations like toggle and stream, which support MQTT (Message Queuing Telemetry Transport) and ATCommand protocols. The VEGA V2.0 ARIES can be connected to the ESP Wifi module using AT Commands. Adafruit IO simplifies device communication, data management, and visualization, making it user-friendly for creating IoT applications. Additionally, its advanced features cater to the requirements of experienced IoT developers.

Fig 3.14 : Adafruit IO

3.4.3 IFTTT (If This Then That)

IFTTT, a widely used webbased tool and app, empowers users to create automated tasks and workflows that connect various online tools and devices. In IFTTT, an applet comprises a trigger and an action, allowing users to customize and configure specific actions based on chosen triggers. Among the available notification options in IFTTT, there are SMS notifications and Android SMS notifications. When a trigger is activated, IFTTT sends an SMS notification, while the Android SMS notification is sent to the designated mobile number.

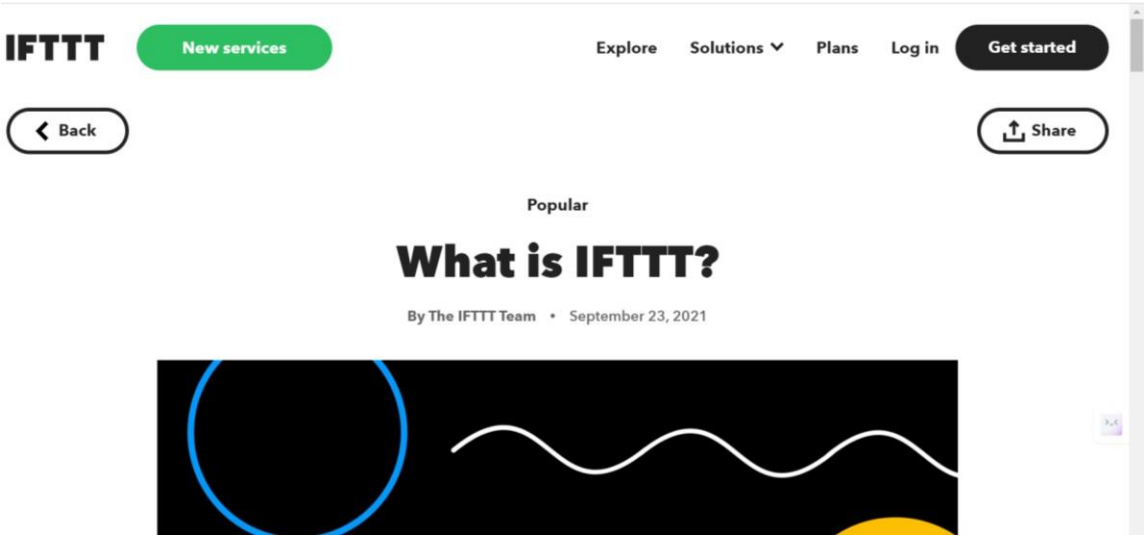


Fig 3.15 : IFTTT

3.5 Block Diagram

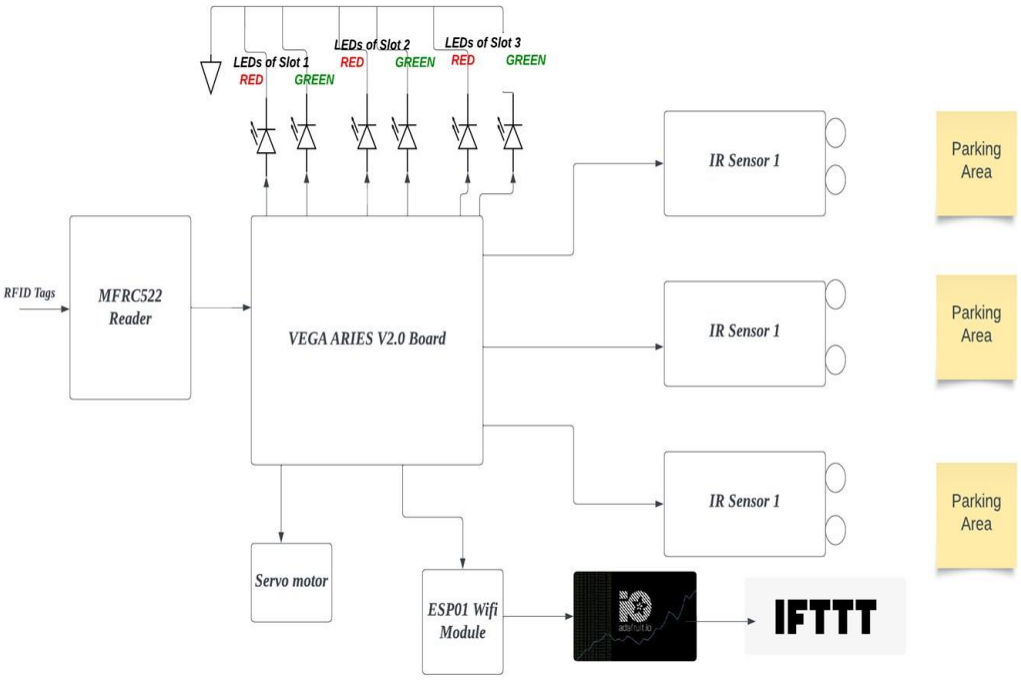
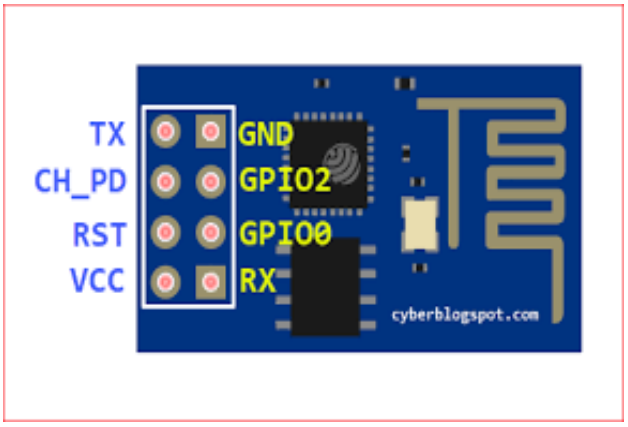


Fig 3.16 : Block Diagram of Proposed Methodology

3.6 Connections

3.6.1 Connections of ESP01 Wifi Module with VEGA Processor



ESP 01 Wifi Module Name	VEGA Aries PIN Numbers / Names
TX	RX1
CH_PD or it is called as EN (Enable) Pin	3.3 V
VCC	3.3 V
GND	GND
RX	TX1

Table 3.4 : ESP01 with VEGA V2.0

3.6.2 Connections of RFID Module with VEGA Processor

MFRC522 (RFID Receiver) Pins	VEGA Aries PIN Numbers / Names
3.3 V	3.3 V
RST	GPIO 14
GND	GND
IRQ	SS0
MISO	MISO0
MOSI	MOSI0
SCK	SCLK0
SDA	GPIO 15

Table 3.5 : MFRC522 with VEGA V2.0

3.6.3 Connections of IR Sensors with VEGA Processor

SLOT 1 (IR Sensor)	**VEGA ARIES Pin Numbers / Names**
OUT	GPIO 6
VCC	5V
GND	GND
SLOT 2 (IR Sensor)	**VEGA ARIES Pin Numbers / Names**
OUT	GPIO 7
VCC	5V
GND	GND

Table 3.6 : IR Sensors with VEGA V2.0

3.6.4 Connections of LED's with VEGA Processor

SLOT 1 (LEDs)	VEGA ARIES Pin Numbers / Names
RED: +terminal	GPIO 0
RED: -terminal	GND
GREEN: +terminal	GPIO 1
GREEN: -terminal	GND
SLOT 2 (LEDs)	**VEGA ARIES Pin Numbers / Names**
RED: +terminal	GPIO 2
RED: -terminal	GND
GREEN: +terminal	GPIO 3
GREEN: -terminal	GND

Table 3.7 : LED's with VEGA V2.0

3.6.5 Connections of Servo Motor with VEGA Processor

Servo motor pin name	VEGA Aries PIN Numbers / Names
PWM	GPIO 9
VCC	5 V
GND	GND

Table 3.8 : Servo Motor with VEGA V2.0

3.7 Flow Chart

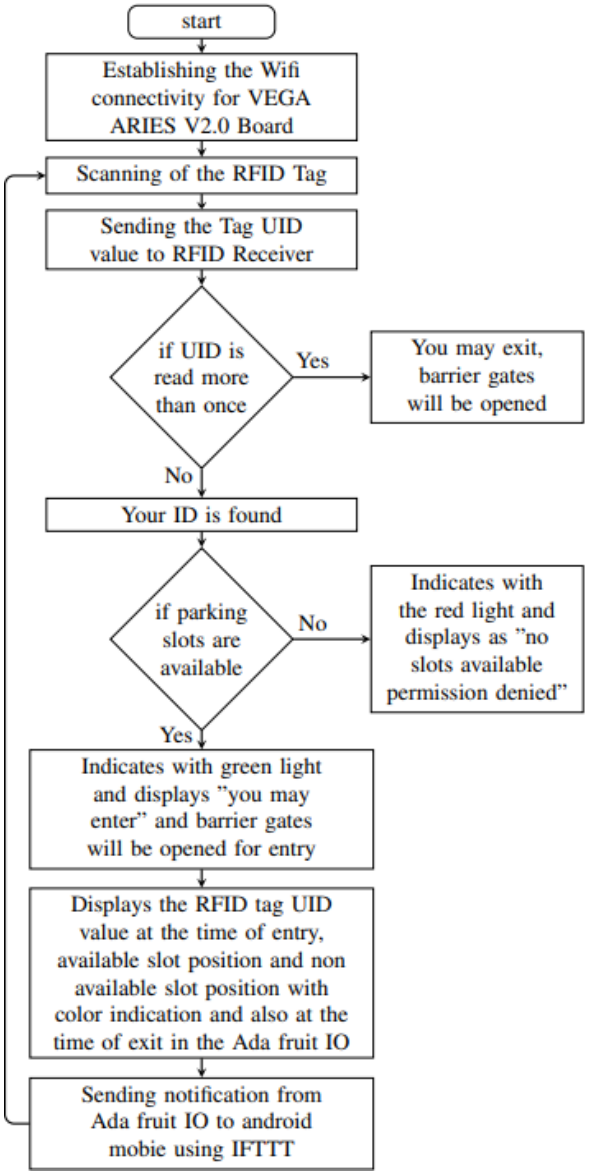


Fig 3.17 : Flowchart

CHAPTER 4 - RESULTS

4.1 Program Code

The program for implementing Car Parking System using VEGA Processor is given below which is written in **Embedded C language** in **Arduino IDE software.**

```
#include <MFRC522.h>    //RFID
#include <SPI.h>         //SPI
#include <Servo.h>      //motor
#include <esp8266.h>    //wifi module
#define RST_PIN 14    // Reset pin for the RFID
#define SS_PIN 15      // (SS) pin for the RFID
#define CH 0           // pwm0 for motor
SPIClass SPI(0);
int x;
MFRC522 rfid(SS_PIN, RST_PIN);  // Create an instance of the MFRC522 class
Servo myservo;  // create servo object to control a servo
const int MAX_CARS = 10;  // Maximum number of cars
int numCars = 0;
// wifi setting config
ESP8266Class esp8266(1);
char * AP="Shanmukh"; // Add network name here
char * PASS= "12345678"; // Add password
char * HOST="io.adafruit.com";
int PORT=80;
char * KEY="aio_Liyk496RD3s0kT9nByWBaHnQlgDH";    // Replace Key here
//all urls
char * URL0="/api/v2/Udaykiran07/feeds/rfid-input/data";    // Add URL of
your feed here
char * URL1="/api/v2/Udaykiran07/feeds/ir-sensor-1/data";   // Add URL of
your feed here
char * URL2="/api/v2/Udaykiran07/feeds/ir-sensor-2/data";   // Add URL of
your feed here
char * URL4="/api/v2/Udaykiran07/feeds/rfid-output/data";   // Add URL of
your feed here
char * URL5="/api/v2/Udaykiran07/feeds/led2motor/data/last?x-aio-
key=aio_Liyk496RD3s0kT9nByWBaHnQlgDH";         // Add URL of your feed here
char * URL6="/api/v2/Udaykiran07/feeds/sensor-status/data";
//wifi configuration
int countTrueCommand;
int countTimeCommand;
boolean found = false;
```

```
char atcommand[250]={0,};
int result1;
char data[250]={0,};
char data1[250]={0,};
char payload[250]={0,};
int result =0;                    //for final result;
boolean ledon = false;
int timeout;
int abc=0; //wifi to start
int card_data[10];  // Array to store the RFID card IDs
int card_count = 0;   // Counter for the number of cards scanned
int check_number = 0;  //to check whether no is registered or not
int check_motor = 0;  //to check whether no is registered or not
int s=0;                //for servo to work
int total_id=0;         //no of id registered
int double_tap=0;      //double tap to exit
int current_card_data = 0;     //to store current card data
int no_open;                  //not to open gates
int open_gate=0;              //to open gate to exit
int exitcard_data[10];  // Array to store the exit RFID card IDs
int l=0;                 //to store index for exit data
int xyz=0;              // for printing statements only once
//ir sensors conf
int a=digitalRead(6); //IR1
int b=digitalRead(7); //IR2
int c=digitalRead(8); //IR3
void setup()
{
  countTrueCommand=0;
  countTimeCommand=0;
  Serial.begin(115200);
  esp8266.begin(115200);
  sendCommand("AT",5,"OK");  // the basic command that tests the AT start
up. If the AT start up is successful, then the response is OK.
  sendCommand("AT+CWMODE=1",5,"OK");  // to set the WiFi Mode of operation
(1 = Station mode i.e ESP connects to the router as a client.)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CWJAP=\"%s\",\"%s\"",AP,PASS);    // This command is
to connect to an Access Point (like a router).
  sendCommand(atcommand,2,"OK");
  SPI.begin();          // Initialize SPI bus
  rfid.PCD_Init();      // Initialize MFRC522 RFID reader
  rfid.PCD_DumpVersionToSerial();  // Print the RFID reader's firmware
version to the Serial Monitor
/motor setting
  Servo myservo();
```

```
// pin setting
  pinMode(0,OUTPUT);        //RED1
  pinMode(1,OUTPUT);        //GREEN1
  pinMode(6,INPUT);         //IR1
  pinMode(2,OUTPUT);        //RED2
  pinMode(3,OUTPUT);        //GREEN2
  pinMode(7,INPUT);         //IR2

}

void sendCommand(char * command, int maxTime, char readReplay[])
{
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }
    countTimeCommand++;
  }
if(found == true)
  {
    Serial.println("-> OK");
    countTrueCommand++;
    countTimeCommand = 0;
  }
  if(found == false)
  {
    Serial.println("-> Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }
  found = false;
 }
```

```
{
    int i=500;
    while(i){
       while(esp8266.available())
       {
          Serial.print(char(esp8266.read()));
       }
       i--;
    }
 }
void command()
 {
  sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used t..o enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);    // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  sendCommand(atcommand,4,">");
  esp8266.println(payload);
  countTrueCommand++;
  delay(2000);
  sendCommand("AT+CIPCLOSE=0",5,"OK");  // Closes TCP/UDP/SSL connect
 }
void command0()
 {
  sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used t..o enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);    // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  memset(atcommand,0,250);
  memset(data,0,250);
  sprintf(data,"{\"value\": %d}",current_card_data);
  sprintf(payload,"POST %s HTTP/1.1\r\nHost: %s\r\nContent-Type:
application/json\r\nX-AIO-Key: %s\r\nContent-Length: %d\r\n\r\n%s",URL0,
HOST, KEY, strlen(data),data);
  sprintf(atcommand,"AT+CIPSEND=0,%d",strlen(payload)); //  to start
sending data in transparent transmission mode.
  sendCommand(atcommand,4,">");
esp8266.println(payload);
  countTrueCommand++;
```

```
delay(2000);
  sendCommand("AT+CIPCLOSE=0",5,"OK");  // Closes TCP/UDP/SSL connect
 }

 void command1()
 {
  sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used to enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);    // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  memset(atcommand,0,250);
  memset(data,0,250);
  sprintf(data,"{\"value\": %d}",digitalRead(6));
  sprintf(payload,"POST %s HTTP/1.1\r\nHost: %s\r\nContent-Type:
application/json\r\nX-AIO-Key: %s\r\nContent-Length: %d\r\n\r\n%s",URL1,
HOST, KEY, strlen(data),data);
  sprintf(atcommand,"AT+CIPSEND=0,%d",strlen(payload)); //  to start
sending data in transparent transmission mode.
  sendCommand(atcommand,4,">");
  esp8266.println(payload);
  countTrueCommand++;
  delay(2000);
  sendCommand("AT+CIPCLOSE=0",5,"OK");  // Closes TCP/UDP/SSL connect

 }
void command2()
 {
  sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used to enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);    // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  memset(atcommand,0,250);
  memset(data,0,250);
  sprintf(data,"{\"value\": %d}",digitalRead(7));
  sprintf(payload,"POST %s HTTP/1.1\r\nHost: %s\r\nContent-Type:
application/json\r\nX-AIO-Key: %s\r\nContent-Length: %d\r\n\r\n%s",URL2,
HOST, KEY, strlen(data),data);
  sprintf(atcommand,"AT+CIPSEND=0,%d",strlen(payload)); //  to start
sending data in transparent transmission mode.
  sendCommand(atcommand,4,">");
```

```
esp8266.println(payload);
  countTrueCommand++;
  delay(2000);
  sendCommand("AT+CIPCLOSE=0",5,"OK");  // Closes TCP/UDP/SSL connect

 }
void command4()
 {
  sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used to enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);    // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  memset(atcommand,0,250);
  memset(data,0,250);
  sprintf(data,"{\"value\": %d}",current_card_data);
  sprintf(payload,"POST %s HTTP/1.1\r\nHost: %s\r\nContent-Type:
application/json\r\nX-AIO-Key: %s\r\nContent-Length: %d\r\n\r\n%s",URL4,
HOST, KEY, strlen(data),data);
  sprintf(atcommand,"AT+CIPSEND=0,%d",strlen(payload)); //  to start
sending data in transparent transmission mode.
  sendCommand(atcommand,4,">");
  esp8266.println(payload);
  countTrueCommand++;
  delay(2000);
  sendCommand("AT+CIPCLOSE=0",5,"OK");  // Closes TCP/UDP/SSL connect

 }
void command5()
 {
  sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used to enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);   // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  memset(atcommand,0,250);
  sprintf(data,"GET %s HTTP/1.1\r\nHost: %s\r\n\r\n", URL5, HOST);
  sprintf(atcommand,"AT+CIPSEND=0,%d",strlen(data));                  //  to
start sending data in transparent transmission mode.
  sendCommand(atcommand,4,">");
esp8266.println(data);
  countTrueCommand++;
```

```
timeout=2;
  while(timeout!=0)
  {
    if(esp8266.find("\"value\":\"1\"")==0)
    {
        Serial.println("MOTOR ON");
        //Serial.println(result1);
        ledon=true;
        break;
    }
    //Serial.println(timeout);
    timeout--;
  }
  if(ledon==true)
  {
    digitalWrite(9, HIGH);
      for(s=0 ; s<=90; s++)
    {
      myservo.write(s);// sets the servo position according to the scaled
value
    }
    delay(5000);
    for(s=90 ; s>=0; s--)
    {
      myservo.write(s); // sets the servo position according to the scaled
value
    }
    delay(5000);
    digitalWrite(9, LOW);

  }
  sendCommand("AT+CIPCLOSE=0",5,"OK");              // Closes TCP/UDP/SSL
connection
  delay(500);

 }
void command6()
 {
   sendCommand("AT+CIPMUX=1",3,"OK");  // This AT Command is used to enable
or disable multiple TCP Connections. (0: Single connection, 1: Multiple
connections)
  memset(atcommand,0,250);
  sprintf(atcommand,"AT+CIPSTART=0,\"TCP\",\"%s\",%d", HOST, PORT);    // to
establish one of the three connections: TCP, UDP or SSL.
  sendCommand(atcommand,3,"OK");
  memset(atcommand,0,250);
```

```
memset(data,0,250);
  sprintf(data,"{\"value\": %d}",result1);
  sprintf(payload,"POST %s HTTP/1.1\r\nHost: %s\r\nContent-Type:
application/json\r\nX-AIO-Key: %s\r\nContent-Length: %d\r\n\r\n%s",URL6,
HOST, KEY, strlen(data),data);
  sprintf(atcommand,"AT+CIPSEND=0,%d",strlen(payload)); // to start
sending data in transparent transmission mode.
  sendCommand(atcommand,4,">");
  esp8266.println(payload);
  countTrueCommand++;
  delay(2000);
  sendCommand("AT+CIPCLOSE=0",5,"OK");  // Closes TCP/UDP/SSL connect
 }
void loop()
 {
    while(abc<3)
  {
    command();
    abc++;
  }

  if(xyz<=0)
  {
    Serial.println("WELCOME TO RFID BASED CAR PARKING SYSTEM");
    delay(1000);
    xyz=1;
  }
if(xyz==1)
  {

  Serial.println("Please scan your rfid tag");
  delay(5000);
  xyz=2;
  }
/ Check for new RFID cards
  if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
    // Store the UID as an integer

    for (byte i = 0; i < rfid.uid.size; i++) {
      current_card_data = (current_card_data << 8) | rfid.uid.uidByte[i];
      xyz=0;
    }
```

```
//Check whether card is saved or not.

    for(int i=0;i<total_id;i++)
    {
      if(card_data[i]==current_card_data)
      {
        Serial.println("Card already registered");
        delay(2000);
        command1();
        command2();
        command4();
        delay(1000);
        command5();
        delay(1000);
        check_number++;
        xyz=0;
        return;
      }
    }
 if(check_number==0)
      {
        total_id++;
        Serial.println("Card detected!");
        Serial.println(F("A new card has been detected."));
        Serial.print("Card UID: ");
        Serial.println(current_card_data);
        delay(2000);

    // Store the card ID in the array
        if (card_count < 10)
        {
         card_data[card_count] = current_card_data;
         card_count++;
         check_motor++;
         }
        xyz=0;
      }
 delay(2000);
    int a=digitalRead(6); //IR1
int b=digitalRead(7); //IR2
```

```
if(a==HIGH&&b==HIGH)
  {
    digitalWrite(0, LOW);
    digitalWrite(1, HIGH);
    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);
    // digitalWrite(4, LOW);
    // digitalWrite(5, HIGH);
    no_open=0;
    Serial.println("Slot 1, Slot 2 are open");
    result1=12;
    //delay(1000);
  }
  else if(a==HIGH && b==LOW)
  {
    digitalWrite(0, LOW);
    digitalWrite(1, HIGH);
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    // digitalWrite(4, LOW);
    // digitalWrite(5, HIGH);
    no_open=0;
    Serial.println("Slot 1 is open");
    result1=1;
    //delay(1000);
  }
   else if(a==LOW && b==HIGH)
  {
    digitalWrite(0, HIGH);
    digitalWrite(1, LOW);
    digitalWrite(2, LOW);
    digitalWrite(3, HIGH);
    // digitalWrite(4, LOW);
    // digitalWrite(5, HIGH);
    no_open=0;
    Serial.println("Slot 2 is open");
    result1=13;
    //delay(1000);
  }
else
  {
    Serial.println("all slots are booked, wait for some time");
    no_open++;
    //delay(1000);
    result1=0;
  }
```

```
command0();
  command1();
  command2();
  command6();
  }
rfid.PICC_HaltA();  // Halt the card
  rfid.PCD_StopCrypto1();  // Stop encryption on the card

  //motor starts working if rfid matched
if(check_motor!=0)
  {
    if(no_open!=0)
    {
      check_motor=0;
      Serial.println("Permission Denied");
      delay(2000);
      return;
    }
    else
    {
    Serial.println("Permission Granted");
    check_motor=0;
    delay(2000);

  command5();
  }
      }
 }
```

4.2 Outputs

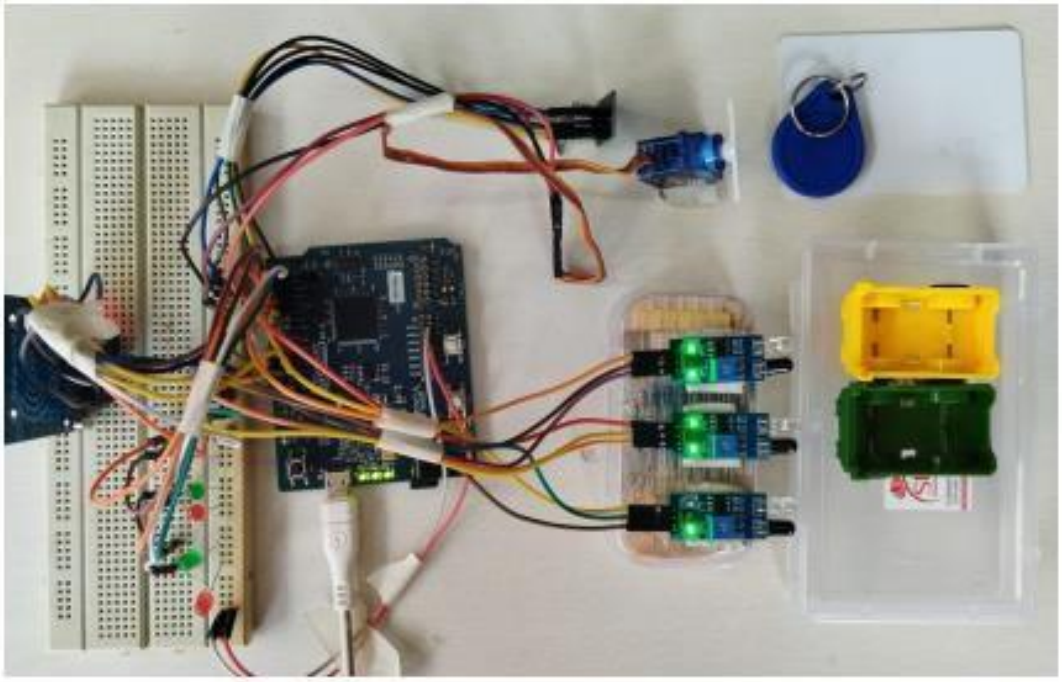


Fig 4.1 : Proposed IoT Model

The proposed IoT model consists of three primary components.

- Firstly, the Parking element comprises various sensors and modules integrated with a VEGA ARIES V2.0 board.
- Secondly, the cloud server element describes the process of data presentation on the Ada fruit IO platform from the processor.
- Lastly, the IFTTT element elucidates how notifications are transmitted from the cloud server to an Android mobile application.

When the user initiates scanning of the RFID tag, it takes some time to read the tag using the MFRC522 Reader module. The system also verifies whether the tag has been scanned previously or not. If it is the first-time scan, a command is sent to the IR sensor to display the parking slot status using LED light indicators[1]. The LED turns Green if parking slots are available, and Red if they are occupied. Subsequently, the barrier gates are opened with the assistance of Servo motors, depending on the availability of parking slots. If the RFID tag has been read before, the barrier gates

open to allow for an exit[2]. **Fig 4.1 represents the proposed IoT model which describes the arrangement of all sensors and modules with the VEGA ARIES Board.**

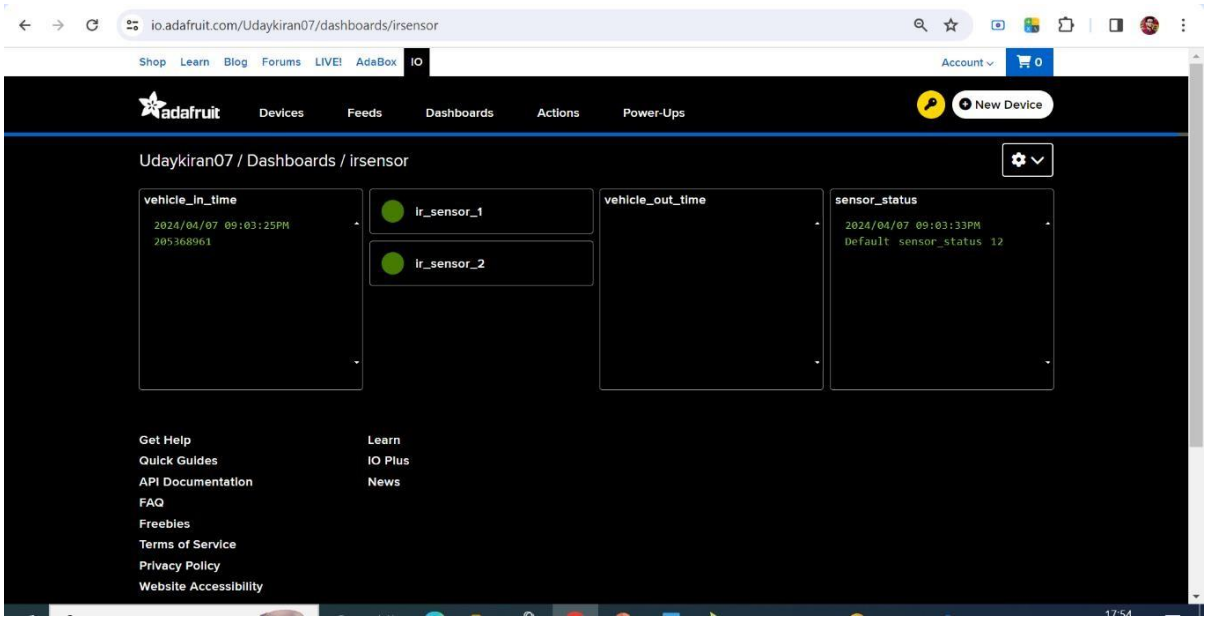


Fig 4.2 : Display of data in Adafruit IO Website and Mobile Notifications

We will connect the ESP8266 WiFi module to VEGA ARIES V2.0 using AT commands. Once the WiFi connection is established, the system will read RFID tags, and upon the first read, it will instantly add the timestamp to the Stream block in the dashboard's designated URL route on Ada fruit IO. The administrator will be able to see available parking spaces, indicated by different colors in the feed. Additionally, the exit time will be displayed alongside the parking status." **Fig 4.2 depicts the data representation and signaling process at different time points.** At the entry on the left side, the RFID UID value is displayed, and an LED light indicates the availability of the corresponding slot. Adjacent to the color indicator, the exit time is also shown. Below, the servo motor toggle is located. Only when the motor is activated will the servo motors and barrier gates operate.

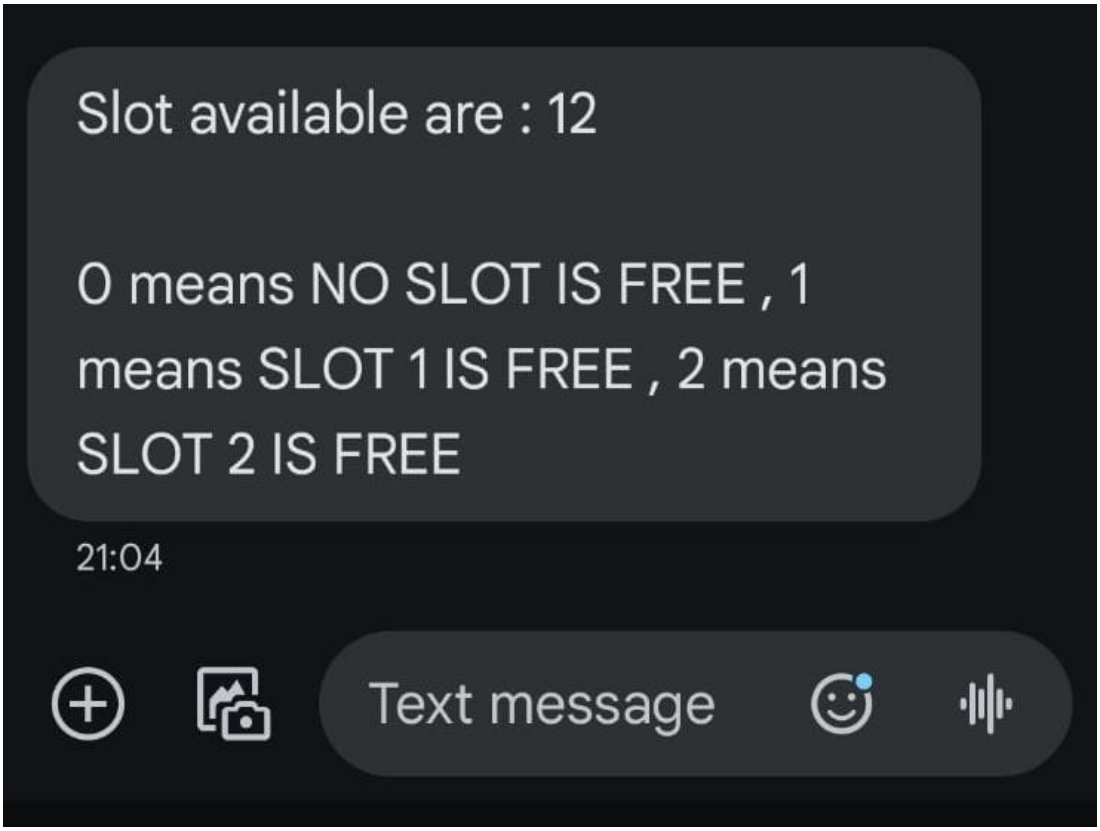


Fig 4.3 : Mobile Notifications

- Data displayed in the Ada fruit IO website as shown in Fig. 4.2 can be sent to the driver's registered mobile number in form of sms notification using IFFT applet. SMS sent to one of the driver is shown in Fig 4.3.

CHAPTER 5 – FUTURE SCOPE

Creating a mobile app using Firebase for seamless data storage and synchronization across user devices is a valuable addition to the Smart parking system. By leveraging Firebase's real-time database capabilities, the app can ensure that all users have up-to-date information on parking availability, making their parking experience efficient and convenient. The integration of payment services within the app further enhances its advantages. Users can easily identify and detect available parking spaces without the need to stop at a booth or interact with a cashier, significantly reducing the time spent in the parking process. This streamlined approach not only improves user satisfaction but also promotes efficient space utilization and traffic flow within the parking lot, contributing to a more sustainable and intelligent parking solution. With Firebase's robust infrastructure and payment services, the app empowers users to make informed decisions and enjoy a hassle-free parking experience.

CHAPTER 6 – CONCLUSION

The Smart parking system using vega processor described in the paper aims to address this challenges associated with traditional parking management, such as congestion and inefficient space utilization. By employing RFID technology, the system enables seamless and automated vehicle identification, reducing the need for manual intervention and enhancing the overall parking experience for users.By integrating with Vega V2 processor, its capabilities have been increased drastically.This processor based on RISC V pipelining can reducing the waiting time drastically which leads to efficient working of this IoT module. The integration of a cloud server, Ada fruit IO stream, not only ensures real-time tracking of vehicle movements but also allows for data storage and analysis, leading to valuable insights for future optimization. Moreover, the utilization of IFTTT for notification purposes adds an extra layer of convenience for users, enabling them to receive timely updates on parking lot availability and making their parking decision-making process more efficient and hassle-free. This innovative solution holds the potential to transform the way parking management is conducted, offering a smarter and more sustainable approach to urban mobility.

REFERENCES

[1] RVS Technical Campus, IEEE Electron Devices Society, Institute of Electrical and Electronics Engineers. (n.d.). Proceedings of the Second International Conference on Electronics, Communication and Aerospace Technology (ICECA 2018): 29-31, May 2018.

[2] Vidyavardhaka College of Engineering, Institute of Electrical and Electronics Engineers. Banglore Section., Institute of Electrical and Electronics

Engineers. (n.d.). International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC) - 2017: 8-9, September 2017.

[3] Saudi Computer Society., Institute of Electrical and Electronics Engineers.

Saudi Arabia Section, Institute of Electrical and Electronics Engineers. Region 8, Institute of Electrical and Electronics Engineers. (n.d.). 1st International Conference on Computer Applications Information Security: ICCAIS'2018: Riyadh, Kingdom of Saudi Arabia, 04-06 April, 2018.

[4] https://vegaprocessors.in/devboards/ariesv2.html

[5] https://vegaprocessors.in/blog/

[6] Institute of Engineering Management, University of Engineering Management, Institute of Electrical and Electronics Engineers. Kolkata Section, Institute of Electrical and Electronics Engineers. (n.d.). Optronix 2019: 2019 International Conference on Opto-Electronics and Applied Optics (Optronix): 18th-20th March, 2019, University of Engineering and Management, Kolkata.

[7] Kumar, L., Hasan Khan, M., Sarosh Umar, M. (n.d.). Smart Parking System using RFID and GSM Technology.

[8] B. Salah, Design, simulation, and performanceevaluation-based validation of a novel RFID-based automatic parking system, Journal of SIMULATION, 2019.

[9] Castro, L., & Fosso Wamba, S. (2007), An Inside Look at RFID

Technology, Journal of Technology Management & Innovation, 2(1),128-141.

[10] A. Mbacke, N. Mitton and H. Rivano, ´ RFID reader anticollision protocols for dense and mobile deployments, dans MDPI Electronics Special Issue ”RFID Systems and Applications”,pp 22, 2016

[11] Zhou, F., & Li, Q. (2014, November). Parking Guidance System Based on ZigBee and Geomagnetic Sensor Technology. In Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on (pp. 268-271).IEEE.

[12] International Parking Institute, “2012 Emerging Trends in Parking”.

[13] FastPark System website, http://www.fastprk.com.

[14] R. L. Kesten, "The smart parking evolution: From evolution to revolution," IEEE Pervasive Computing, vol. 14, no. 2, pp. 84-88, Apr-Jun. 2015.

[15] N. A. G. et al., "A survey of smart parking solutions," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 9, pp. 3409-3423, Sep. 2019.

[16] S. Shaheen et al., "Smart parking: An overview and synthesis of technologies and their impact on parking and transportation," Transportation Research Record: Journal of the Transportation Research Board, vol. 2147, no. 1, pp. 46-55, Dec. 2010.

[17] S. Jin et al., "Smart parking systems: A survey," IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 5, pp. 1436-1453, May. 2018.

[18] A. A. H. Kadhim et al., "A review of smart parking solutions: Technologies and challenges," IEEE Access, vol. 8, pp. 185325-185342, 2020.

[19] S. R. et al., "Smart parking systems using internet of things (IoT): A comprehensive review," Sensors, vol. 20, no. 12, 3542, Jun. 2020.

[20] H. B. et al., "Smart parking system: A systematic literature review," Journal of Computer Networks and Communications, vol. 2018, Article ID 5151017, 2018.

[21] C. Chen et al., "Smart parking solutions: A survey and comparative study," Journal of Sensors, vol. 2017, Article ID 4656183, 2017.

[22] Y. Huang et al., "A comprehensive survey of smart parking technologies," IEEE Transactions on Intelligent Transportation Systems, vol. 45, no. 6, pp. 1059-1082, Jun. 2022.

[23] C. M. Chiorean et al., "Smart parking systems: A review of efficient parking solutions," Sustainable Cities and Society, vol. 70, 102896, Apr. 2022.