SMART PARKING SYSTEM WITH IoT

A MINI PROJECT REPORT

Submitted by

RAMESH ADHITHYA S
MONISH KUMAR S
PRAVEEN MU
SANJAY KARTHICK R V B

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2021

ABSTRACT

These days finding a parking space are becoming more and more difficult. By statistics, an average person in a metropolitan city wastes about 17 hours of driving around to find a parking space. Implementing new technologies to reduce the time wasted and optimizing this process efficiently is an advantageous way to tackle this problem. This project demonstrates the implementation of a smart parking system that functions concurrently with the internet to optimize the need to find a parking space quicker using a prototypical model. The world is increasingly connecting itself with the internet giving rise to tremendous innovation in technologies that make use of the internet further resourceful this project exactly does that. As the world's population is socially distancing itself due to the COVID-19 pandemic people are highly reluctant to unwanted social interactions. This project removes the need for human labor so all the processes involved will be contactless. It also helps a corporation/organization to reduce operating costs as the whole system can be monitored, controlled, and governed via the internet. The prototype uses a low-cost developer-friendly microcontroller module to receive data from sensors and carry out operations to make the system function. It can also be scaled to use in major corporations and organizations with a few enhancements.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF TABLE	vii
	LIST OF FIGURE	viii
	LIST OF ABBREVIATIONS	ix
1.	**INTRODUCTION**	**1**
	1.1 Introduction	1
	1.2 Concept of smart parking	2
	1.3 Objective	3
2.	**LITERATURE REVIEW**	**4**
3.	**SPECIFICATION OF HARDWARE**	**6**
	3.1 NodeMCU ESP8266	6
	3.1.1 General Description	6
	3.1.2 Product Description	7
	3.1.3 Features	8
	3.1.4 Pin Diagram of NodeMCU 1.0	9
	3.1.5 Applications	9
	3.2 IR Obstacle Sensor	10
	3.2.1 General Description	10
	3.2.2 Product Description	10
	3.2.3 Features	11
	3.2.4 Pin Diagram of IR Sensor	12

3.2.5 Applications 12

3.3 9G Micro Servo 12

3.3.1 General Description 12

3.3.2 Product Description 13

3.3.3 Features 14

3.3.4 Pin Diagram of Micro Servo 15

3.3.5 Applications 15

4. SPECFICATION OF SOFTWARE 16

4.1 NodeMCU ESP8266 16

4.1.1 General Description 16

4.1.2 Working on Arduino IDE 17

4.2 Adafruit IO 18

4.2.1 General Description 18

4.2.2 Setting up Adafruit IO 18

4.3 Proteus Software 19

4.3.1 Simulation Design 19

4.3.2 Connections and Working 20

5. BLOCK DIAGRAM AND CIRCUIT DIAGRAM 21

5.1 Block Diagram 21

5.2 Circuit Diagram 22

6. FINAL INTEGRATION 23

6.1 Integrating nodemcu with IR sensors and Servo Motor 23

6.2 Working 24

6.3 Prototype 26

6.4 Results 26

6.5 Advantages 27

6.6 Disadvantages 27

7. CONCLUSION 28

8. REFERENCES 29

9. APPENDIX 31

I. Code for the Prototype 31

TABLE NO.	NAME OF THE TABLE	PAGE NO
3.1	Specification of NodeMCU	8
3.2	Specification of IR Obstacle Sensor	11
3.3	Specification of 9G Micro Servo	14
6.1	Integrating IR Sensor with NodeMCU	23
6.2	Integrating Servo Motor with NodeMCU	23

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO
3.1	NodeMCU	7
3.2	Pin Description of NodeMCU	9
3.3	IR Obstacle Sensor	11
3.4	Pin Description of IR Obstacle Sensor	12
3.5	9G Micro Servo	13
3.6	Pin Description of Micro Servo	15
4.1	Simulation cirucuit of the Prototype	19
5.1	Block Diagram	21
5.2	Circuit Diagram	22
6.1	Dashboard of the Prototype	24
6.2	Maximum value displayed on the dashboard	25
6.3	Prototype Model	26

LIST OF ABBREVIATION

AC	Alternating Current
ARM	Advanced RISC Machines
AIO	Adafruit Input/Output
CPU	Central Processing Unit
DC	Direct Current
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPIO	General Purpose Input Output
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IoT	Internet of Things
LAN	Local Area Network
MCU	Microcontroller Unit
MQTT	Message Queuing Telemetry Transport
PC	Personal Computer
PWM	Pulse Width Modulation
RISC	Reduced Instructions Set Computing
TCP/IP	Transmission Control Protocol/ Internet Protocol
UART	Universal Asynchronous Receiver Transmitter
USART	Universal synchronous and asynchronous receiver-transmitter

CHAPTER 1

INTRODUCTION

1.1 Introduction

The Present-day world is increasingly connected to the internet and the usage of smart devices gained popularity among people. These modern devices now can be controlled or monitored remotely via the internet. IoT extends the use of the Internet to provide communication, and thus to interconnect physical devices and objects, or “things”. The two most important words in IoT are “internet” and “things”. Internet a large collection of interconnected computers, servers, and mobiles it enables communicating of information by the use of sending/receiving protocols built into it. The “Things” mentioned in IoT can mean a lot of entities in traditional English but in this, it refers to Physical computational devices. IoT, in general consists of an inter-network of the devices and physical objects, number of objects can gather the data at remote locations and communicate to units managing, acquiring, organizing, and analyzing the data in the processes and services. It transforms conventional devices like watches, clocks, refrigerators, lights, and air conditioning devices into smart behaving advanced technologies that can communicate with other embedded remote devices and compute at a high degree of precision. The scalable nature of cloud computing is a perfect partner to use concurrently with the IoT. As applications can be built on the cloud and all the sensor data can be stored and monitored via an application or through a webpage designed for it. These modern technologies can be implemented to further boost the modernization of the world. Thus for the IoT, it can be said as the “Internet of Things” = Embedded devices + Sensors/Actuators+ Internet.

The increase in demand for smart cities has paved a way for use of more new technologies to advance society. The main issue this project address is that of the unwanted time spent driving around to find a parking space which indirectly contributes to traffic congestion. As the utilization of personal vehicles is becoming a common thing for people around and the COVID-19 pandemic boosted this demand even further than it already had.

So we came up with a simple solution to eliminate the need for manual operating of parking lots and digitized the whole process. It is highly possible now because the low-cost, low-power embedded system helps developers to build new solutions to modern world problems.

1.2 Concept Of Smart Parking

IoT (Internet of Things) is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system. IoT systems allow users to achieve deeper automation, analysis, and integration within a system. They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technologies for sensing, monitoring, and networking.

The prototype system will extensively use IoT to optimize the process to locate a parking space quickly. The time wasted in searching for a parking space indirectly affects the driver's mental health causing mild depression and anxiety and also it contributes a little to traffic congestion which leads to an increase in road accidents. The system uses a NodeMCU ESP8266 a wifi module builtin microcontroller as its CPU and data are gathered by the 5 IR sensors, two of them are placed in entrance and exit gates of the parking lot and the other three are placed in the respective parking slot to detect whether a slot is occupied or not. The IR sensor placed at the entry/exit locations sends data to the Microcontroller if it detects a car at these gates and the Microcontroller unit is also connected with a servo motor which acts as an opening/closing gate. When the signal from the Entry IR sensor is HIGH the microcontroller sends the signal to open the gate/servo motor and the car can be parked in any 3 of the available slots. The gate will stop opening if all the slots are filled. The feed data can be viewed on a webpage and can also manually override the system via the webpage.

This project can be deployed onsite promptly thus can be scaled to function in megacities within days to full operation productivity. The prototype provides real-time data which can be used to map the traffic statistics and can help in solving traffic congestion thus reducing commute duration for an average driver.

1.3 Objectives

- Our project aims to demonstrate a working prototype for a smart parking system that works adjacently with the Internet and cloud.
- To optimize the whole parking process to be quick and easy thereby helping reduce the traffic congestion.
- Reduce Labor costs by making the parking system function smartly and automatedly.
- To manually control the system using the internet.
- Eliminate the unnecessary time misused on searching for a parking space around the city.
- Remove the need for social interaction involved in the parking system.

CHAPTER 2

LITERATURE REVIEW

There are several Smart parking systems are proposed. Each one has featured some new parameters and usability. It's been researched by Ioannikis Chatzigiannakis, Andrea Vitteli, Apostolos Pyrgelis [1] in their journal " A privacy-preserving smart parking system using an IoT elliptic curve based security platform".
The following paragraphs will give a summary of the previous works in the field of IoT.

A. Khanna and R. Anand [2], in their project at the University of Petroleum and Energy Studies (UPES), Dehradun, Uttarakhand, demonstrated a parking system with RFID based sensing system. The aim behind their project is to help in the fast development of smart cities by use of new emerging technologies. In concert with them, D. Ganesh Gopal et. al. [3] from Vellore Institute of Technology, Vellore, Tamil Nadu has written a journal about Smart parking using IoT for the betterment of Smart cities.

Prabhu Ramaswamy [4] in his paper about IoT parking systems for reducing greenhouse gas emission mentioned the environmental effects of driving around due to lack of notifications and automated systems about the availability of a parking lot. Vakula and Yeshwanth Krishna Kolli [5] planned a parking system that contains an IoT module deployed on-site for managing the available parking spaces.

Jiong Shi, Liping Jin, et. al., [6] In the proposed system, the data of the sensor node is transmitted by the Narrowband Internet of Things (NB-IoT) module, which is a new cellular technology introduced for Low-Power Wide-Area (LPWA) applications. With an integrated third-party payment platform and parking guide service, the mobile application developed for drivers is easy and convenient to use.

Pranav Chippalkatti, Ganesh Kadam, and Vrushali Ichake [7] designed a system to eliminate the time wastage and irrelevant frustration faced by the drivers based on IoT for real-time monitoring of the empty slots for car parking from anywhere using a webpage or a mobile app; IoT the emerging research domain is the heart of the proposed system.

Manickam Ramasamy et. al., [8] presented an IoT-based smart parking system for a large parking lot that can be used to efficiently manage the parking system by providing information on the nearest parking slot available through the mobile application and thereby reducing the congestion of parking seekers.

Ilhan Aydin, Mehmet Karakose, and Ebru Karakose [9] proposed a method that involves the development of small devices that send data to the internet using the internet of things (IoT) technology. The free parking space closest to the current location is found by a genetic algorithm. Denis Ashok, Akshat Tiwari, and Vipul Jirge [10] The system proposes the implementation of state-of-the-art Internet of Things (IoT) technology to mold with advanced Honeywell sensors and controllers to obtain a systematic parking system for users. Unoccupied vehicle parking spaces are indicated using lamps and users are guided to an empty parking space, thus eliminating the need for searching for a parking space. The occupied parking spaces are virtually stored in the cloud to be accessed by the central system and direct the upcoming cars to empty spaces.

In this paper, we demonstrate a working prototype for a smart parking system that functions extensively with the use of the Internet. This paper is organized as follows. In Chapter 3, the system's general architecture and hardware implementations are discussed. In Chapter 4 we then describe the system's software development. Finally, we conclude our major findings and outline our future work.

CHAPTER 3
SPECIFICATION OF HARDWARE

In this chapter, we discussed the components used in this project and their specifications. The list of component used in this project are mentioned below,

- NODEMCU ESP8266
- IR OBSTACLE SENSOR
- 9G MICRO SERVO MOTOR
- JUMPER WIRES
- LAPTOP/PC

3.1 NodeMCU ESP8266

3.1.1 General Description

NodeMCU is an open-source firmware for which open-source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits. The firmware uses the Lua scripting language. The firmware is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266.

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. NodeMCU started on 13 Oct 2015. Later developers ported the MQTT client library from Contiki to the ESP8266 SoC platform and committed to the NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib (GitHub library) to the NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays. In the summer of 2015, the original creators abandoned the firmware project and a group of independent contributors took over. By the summer of 2016, the NodeMCU included more than 40 different modules.

3.1.2 Product Description

The NodeMCU is a microcontroller unit with an inbuilt wifi module. It has 4 power pins and 17 GPIO pins with I2C and UART support of 17 pins 4 pins can be used to PWM. Control Pins are used to controlling the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST), and WAKE pin. NodeMCU/ESP8266 also features SPI in slave and master modes which supports the 4 timing modes of format transfer and divided clock pulse at 80MHz. The NodeMCU is available in various package styles. Common to all the designs is the base ESP8266 core. Designs based on the architecture have maintained the standard 30-pin layout. Some designs use the more common narrow (0.9″) footprint, while others use a wide (1.1″) footprint. Figure 3.1 shows a picture of NodeMCU.

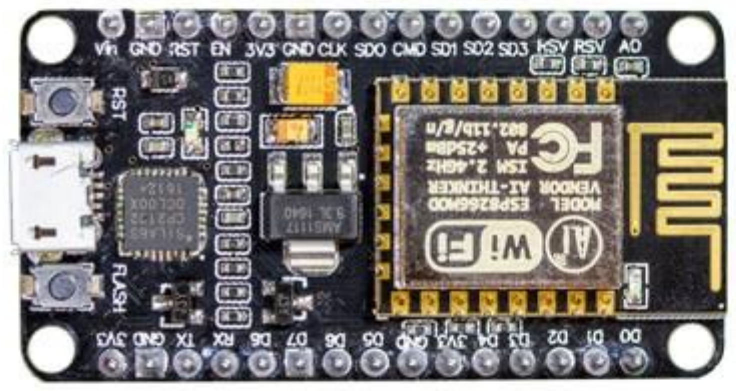


Figure 3.1 NodeMCU ESP8266

3.1.3 Features

The following Table 3.1 shows the specifications of the NodeMCU.

Table 3.1 Specifications of NodeMCU

Microcontroller	ESP-8266 32-bit
NodeMCU Size	49mm x 26mm
Operating Voltage	3.3V
Input Voltage (recommended)	4.5-10V
Pin Spacing	0.9 mm
Digital I/O Pins	11 (of which 4 provide PWM output)
Analog Input Pins	1
UART	1
I2C	1
SPI	1
ADC Range	0 - 3.3V
Wifi Built in	802.11 b/g/n
Flash Memory	4MB
SRAM	64 KB
EEPROM	512 bytes
Temperature Range	-40C - 125C
Clock Speed	80 MHz

3.1.4 Pin Diagram Of NodeMCU

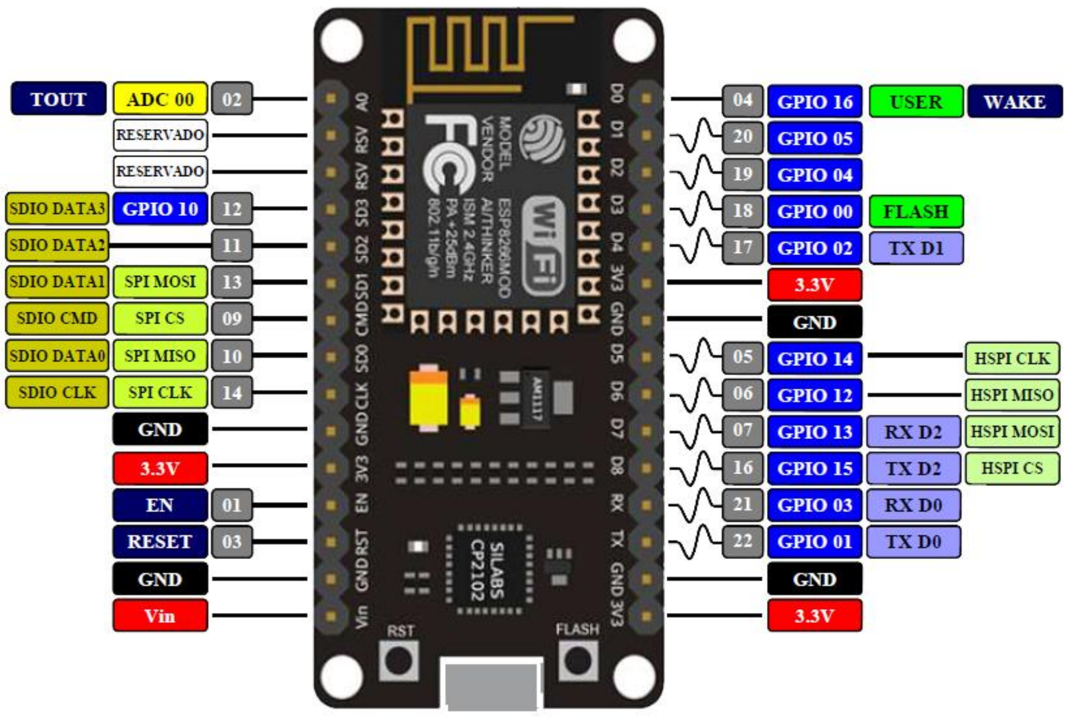


Figure 3.2 Pin description of NodeMCU

3.1.5 Applications

- Prototyping of IoT devices
- Network projects
- Projects requiring multiple I/O interfaces with Wi-Fi and Bluetooth functionalities
- Robotic applications
- Automation system development
- Sensor network

3.2 IR Obstacle Sensor

3.2.1 General Description

Infrared Sensor is the most used sensor in wireless technology where remote controlling functions and detection of surrounding objects/obstacles are involved. It is a simple electronic device that emits and detects IR radiation to find out certain objects/obstacles in its range. Some of its features are heat and motion sensing**.** Figure 3.3 shows a model of IR Obstacle Sensors.

3.2.2 Product Description

IR sensors use infrared radiation of wavelength between 0.75 to 1000μm which falls between visible and microwave regions of the electromagnetic spectrum. IR region is not visible to human eyes. The infrared spectrum is categorized into three regions based on its wavelength i.e. Near Infrared, Mid Infrared, Far Infrared**.** Infrared Sensors works on three fundamental Physics laws Planck's Radiation Law, Stephan Boltzmann Law, and Wein's Displacement Law.

The key elements in Infrared Detection System are IR Transmitter, which acts as the source for IR radiation. According to Plank's Radiation Law, every object is a source of IR radiation at temp T above 0 Kelvin. The other is Transmission Medium, It provides passage for the radiation to reach from IR Transmitter to IR Receiver. IR receivers are photodiode and phototransistors. They are capable of detecting infrared radiation. Hence IR receiver is also called an IR detector.

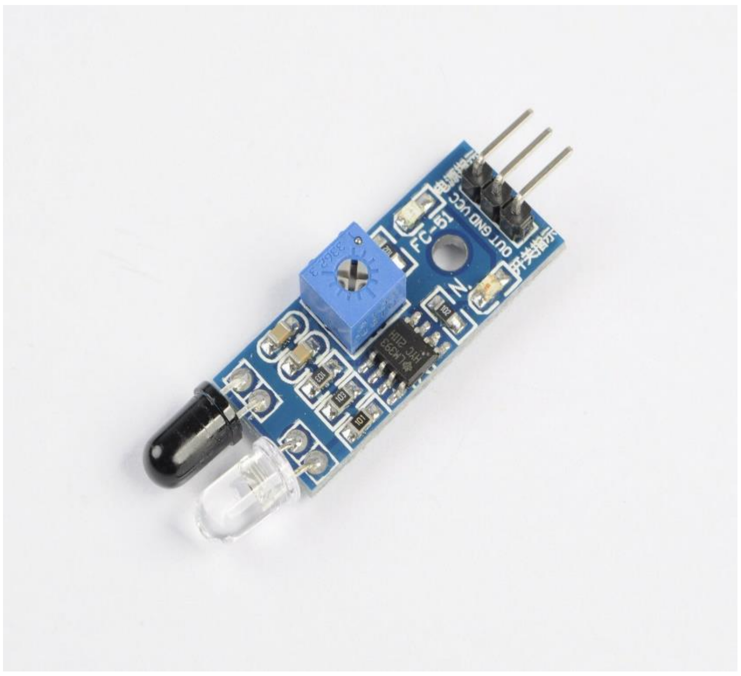


Figure 3.3 IR OBSTACLE SENSOR

3.2.3 Features Of IR Obstacle Sensor

The following table 3.2 shows the specifications of the IR Obstacle Sensor.

Dimensions	48 x 14 x 8 mm
Effective Distance Range	2cm to 80cm
Detection distance	0.5 to 5 cm
Detection angle	35 °
Comparator chip	LM393
Weight	3gm
Operating Temperature	0C to 60C

Table 3.2 Specifications of IR Obstacle Sensor.

3.2.4 Pin Diagram Of IR Obstacle Sensor

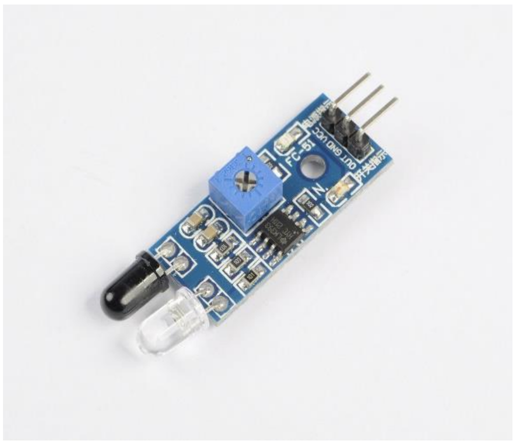


Figure 3.4 Pin description of IR OBSTACLE SENSOR

3.2.5 Applications

- IR Sensors are used in smartphones to find the distance of an object.
- Employed on Item Counters.
- Utilized in Burglar Alarm systems.
- Used in Radiation Thermometers.
- Human Body Detection.

3.3 9G Micro Servo Motor

3.3.1 General Description

A servo motor is a type of motor that can rotate with great precision. Micro Servo Motor SG90 is a tiny and lightweight server motor with high output power. A Micro Servo motor can rotate about 180 degrees (90 in each direction). It is primarily used to rotate an object at some specific angle or distance. There are fundamentally two kinds of servo motor a DC servo motor which uses a DC power supply and an AC servo motor that uses an AC power supply.

A servo motor gets its signal from the Microcontroller Unit and uses pulse width modulation provided by the control wires to rotate at greater precision. Figure 3.5 shows a 9G Micro Servo motor.

3.3.2 Product Description

The servo motor can rotate approximately 180 degrees (90 in each direction). Servo motors are rated in kg/cm (kilogram per centimeter). A Servo motor has three key elements for functioning a controlled device, output sensor, and feedback system. Servo motors have three wires out of three two of them are power supply and ground and the other one is a signal wire that communicates using signals from the MCU. The signal that comes from the MCU is a PWM signal, it manipulates the Servo hand to rotate with advanced precision. The PWM signal is determined in the MCU and transmitted via the control pin of the MCU.

Servo checks the pulse every 20 milliseconds. The pulse of 1 ms (1 millisecond) width can rotate the servo to 0 degrees, 1.5ms can rotate to 90 degrees (neutral position) and 2 ms pulse can rotate it to 180 degrees. Almost all of the servo motors operate at a range of +5V

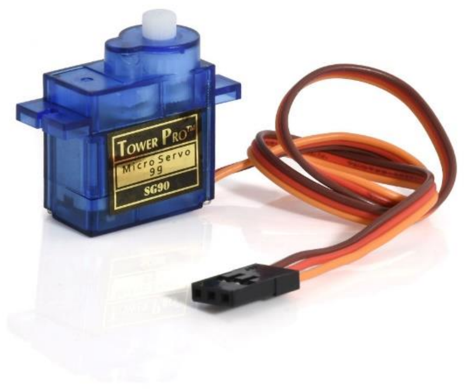


Figure 3.5 9G Micro Servo

3.3.3 Features

The following Tabulation 3.3 shows the specification of the Micro Servo.

Table 3.3 Specifications of Micro Servo

Operating voltage	5V
Operating speed	0.1s/60
Rotation	0°-180°
Weight of motor	9gm
Torque	2.5 kg/cm
Wire color – Brown	Ground wire
Wire color – Red	+5 V supply
Wire color - Orange	PWM signal

3.3.4 Pin Diagram Of 9G Micro Servo

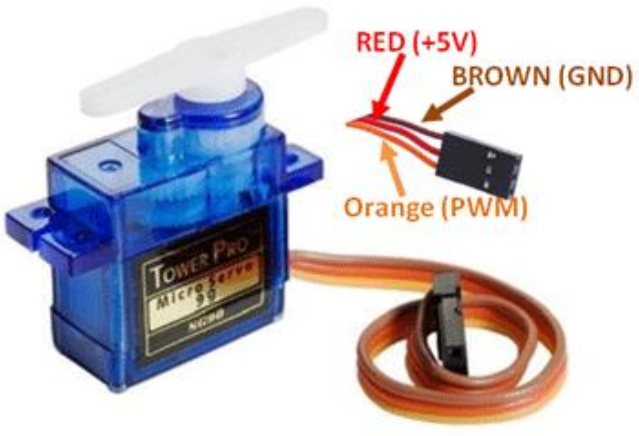


Figure 3.6 Pin description of 9G Micro Servo

3.3.5 Application Of 9G Micro Servo

- Used as actuators in many robots like Biped Robot, Hexapod, robotic arm, etc.,
- Can be used as a controllable gate for prototype projects
- Commonly used for a steering system in RC toys
- Robots where position control is required without feedback
- Less weight hence used in multi DOF robots like a humanoid robot

CHAPTER 4

SPECIFICATION OF SOFTWARE

In this chapter, we will discuss the software used in the project for coding up the Arduino, and also we will discuss the android application used for controlling the electrical appliances. The software used in the project is listed below.

- ARDUINO IDE
- ADAFRUIT IO
- PROTEUS [For Simulation]

4.1 Arduino IDE

4.1.1 General Description

The open-source Arduino environment makes it easy to write code and upload it to the I/O board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and it is based on Processing, AVR-GCC, and other open-source software. The Arduino development environment contains a text editor for writing code, message area, text console, and toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. Arduino programs are written in C or C++. Arduino IDE with features such as syntax highlighting, brace matching, and automatic indentation, and it is also capable of compiling and uploading programs to the Board with a single click. As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio, or the newer Atmel Studio, may also be used to develop software for the Arduino.

4.1.2 Working On Arduino IDE

The NodeMCU can be programmed with the Arduino software. The NodeMCU comes pre-burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header.

4.1.3 Programming Of NodeMCU ESP8266

A program written with the Arduino IDE is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde. A minimal Arduino C/C++ program consists of only two functions:

setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

loop(): After setup() has been called, function loop() is executed repeatedly in the main program. It controls the board until the board is powered off or is reset. Blink example Most Arduino boards contain a light-emitting diode (LED) and a load resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program for a beginning Arduino programmer blinks a LED repeatedly. This program uses the functions pinMode(), digitalWrite(), and delay(), which are provided by the internal libraries included in the IDE environment. This program is usually loaded into a new Arduino board by the manufacturer.

4.2 Adafruit IO

4.2.1 General Description

Adafruit IO is a system that makes data useful by allowing simple data connections with little programming required. IO is built on Ruby, and NodeJS and includes client libraries that wrap REST and MQTT APIs. It provides a place online for learning electronics and making designed products and skill levels.

Adafruit.io can handle and visualize multiple feeds of data. In this project, it helps to display data from all the 5 IR Sensors and the servo motor's current position data. Feeds are the core of Adafruit IO. They hold both the data you uploaded and meta-data about the data your sensors push to Adafruit IO and Dashboards is a feature integrated into Adafruit IO which allows you to chart, graph, gauge, log, and visualize your data. You can view your dashboards from anywhere in the world.

4.2.2 Setting Up Adafruit IO

- Set up an adafruit account to access adafruit's services to use adafruit io to monitor your feed.
- Head to Adafruit.io webpage and obtain the AIO key which will be used to authenticate into adafruit.io's sensor feeds.
- Create the necessary feeds to receive the data from your sensors to monitor and control your prototype.
- Then group your feed data and create a dashboard to view the data from the sensors.
- Add a virtual button to your dashboard to access manual control over your prototype. This will be useful in case of malfunctioning of the system.

4.3 Proteus [For Simulation]

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards. The micro-controller simulation in Proteus works by applying either a hex file or a debug file to the microcontroller part on the schematic. It is then co-simulated along with any analog and digital electronics connected to it. This enables its use in a broad spectrum of project prototyping in areas such as motor control, temperature control, and user interface design. Figure 4.3 shows the simulation design of the prototype.

4.3.1 Simulation Design

The following Figure 4.3 shows the simulation circuit of the prototype

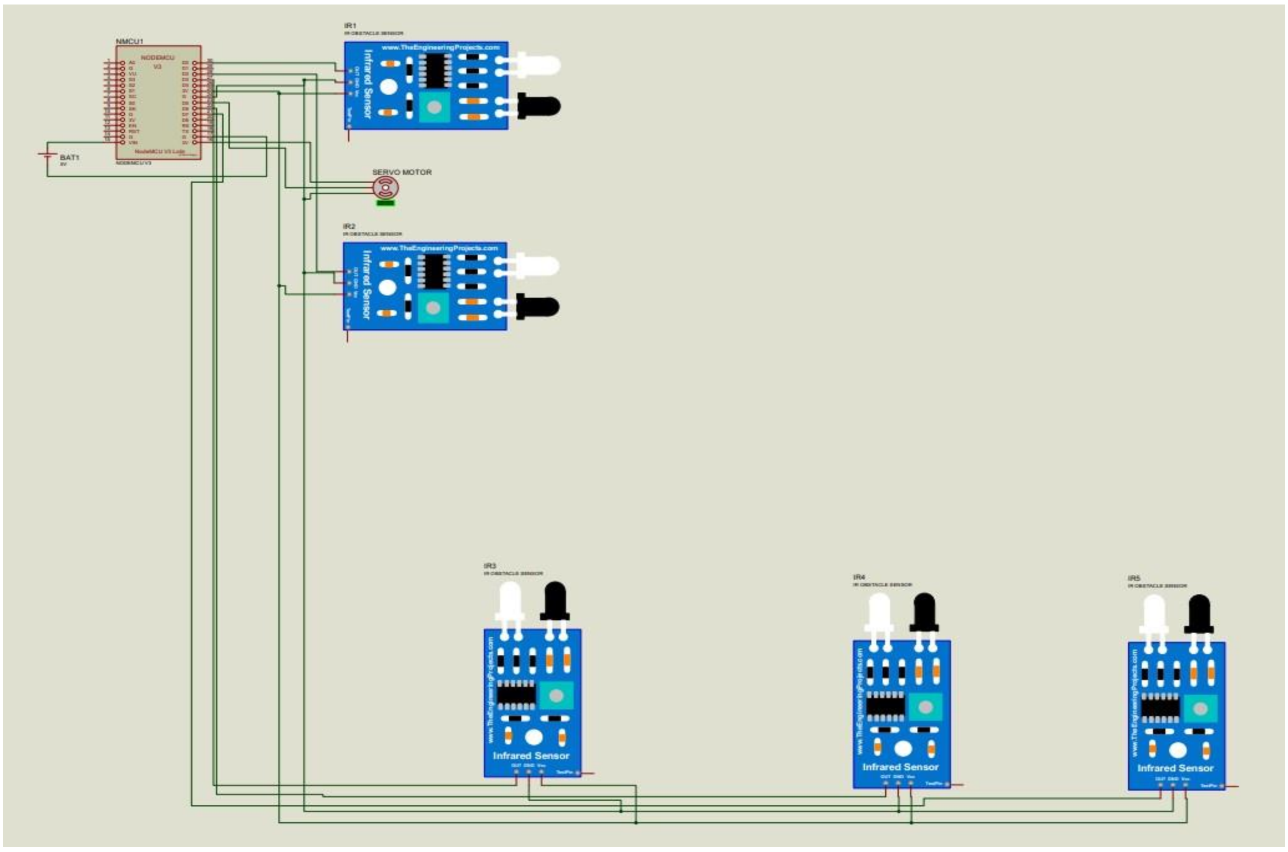


Figure 4.1 Simulation circuit of the prototype

4.3.2 Connection And Working

- Set up the libraries for the NodeMCU and add the MCU to the circuit as the CPU.
- Then include the Entry gate and Exit gate IR sensors respectively to the circuit.
- Begin adding Slot IR sensors to the circuit.
- Connect the VCC pins of the sensors i.e shown in Figure 3.4 to the VCC port of the NodeMCU shown in Figure 3.2 and the respective ground pins to a common ground.
- Add a servo motor and interface it with the NodeMCU at pin D5 after giving power supply and ground connections to it, the pin configurations are shown in Figure 3.6.
- After that, connect the IR sensor's output pins to D0, D2, D3, D6, and D7 digital ports of the NodeMCU as shown in Figure 5.2.
- Write the code for the simulation and debug the code for any errors.
- If all the connections and code were correct the LED on the IR sensors starts to glow and the servo motor will rotate at a precoded angle.
- The feed value from the sensors will be published at the Adafruit.io dashboard via the internet.
- The servo motor on the simulation circuit can be controlled manually by the blocks placed on the webpage via the internet.

CHAPTER 5

BLOCK DIAGRAM AND CIRCUIT DIAGRAM

5.1 Block Diagram

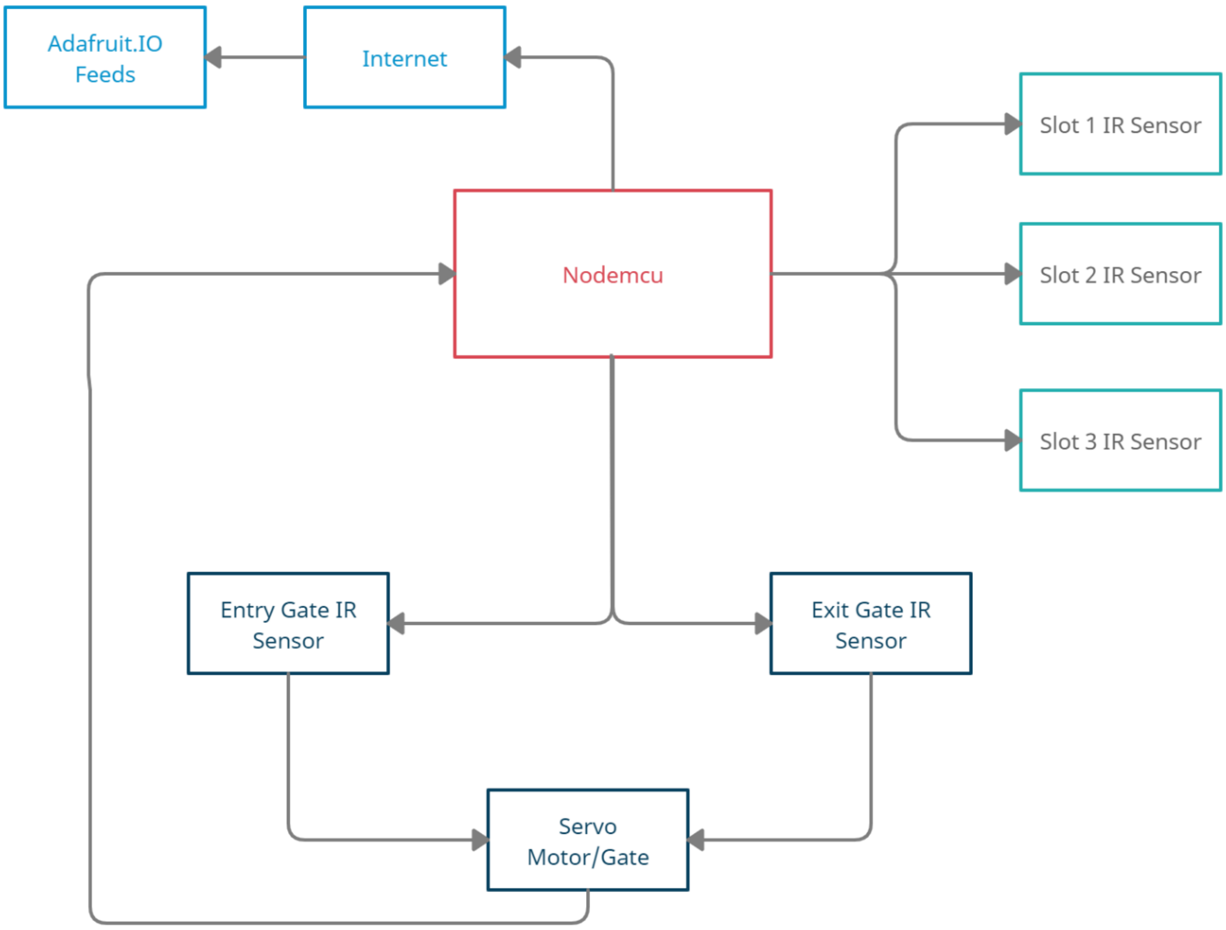


Figure 5.1 Block Diagram of Smart Parking System with IoT

5.2 Circuit Diagram

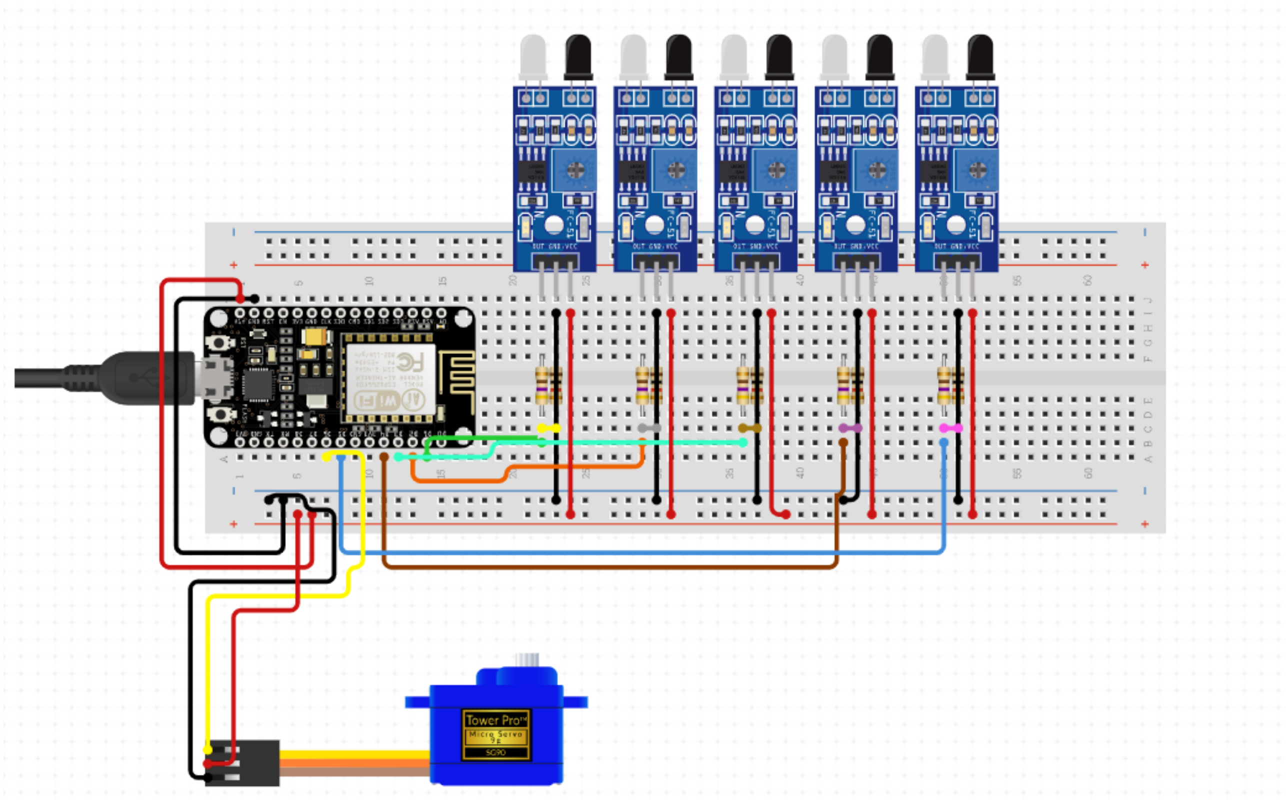


Figure 5.2 Circuit Diagram of Smart Parking System with IoT

Chapter 6

FINAL INTEGRATION

6.1 Integrating Nodemcu With IR Sensors And Servo Motor

The NodeMCU has an inbuilt wifi module so that it could send data over the internet. The 5 IR sensors have a VCC power supply, GND pin, Output signal pin through which the connections are made. The VCC and GND pins as shown in Figure 3.4 of the sensors can be concatenated and connected with the power supply and GND ports of the NodeMCU. The Output signals pins must be connected with the respective GPIO digital

NODEMCU	**IR SENSORS**
VCC PORT	VCC PIN
GND PORT	GND PIN
GPIO/DIGITAL PIN	OUTPUT SIGNAL PIN

pins of the NodeMCU. Table 6.1 shows the connection between the IR sensors and the MCU.

Table 6.1 Connection between IR and NodeMCU.

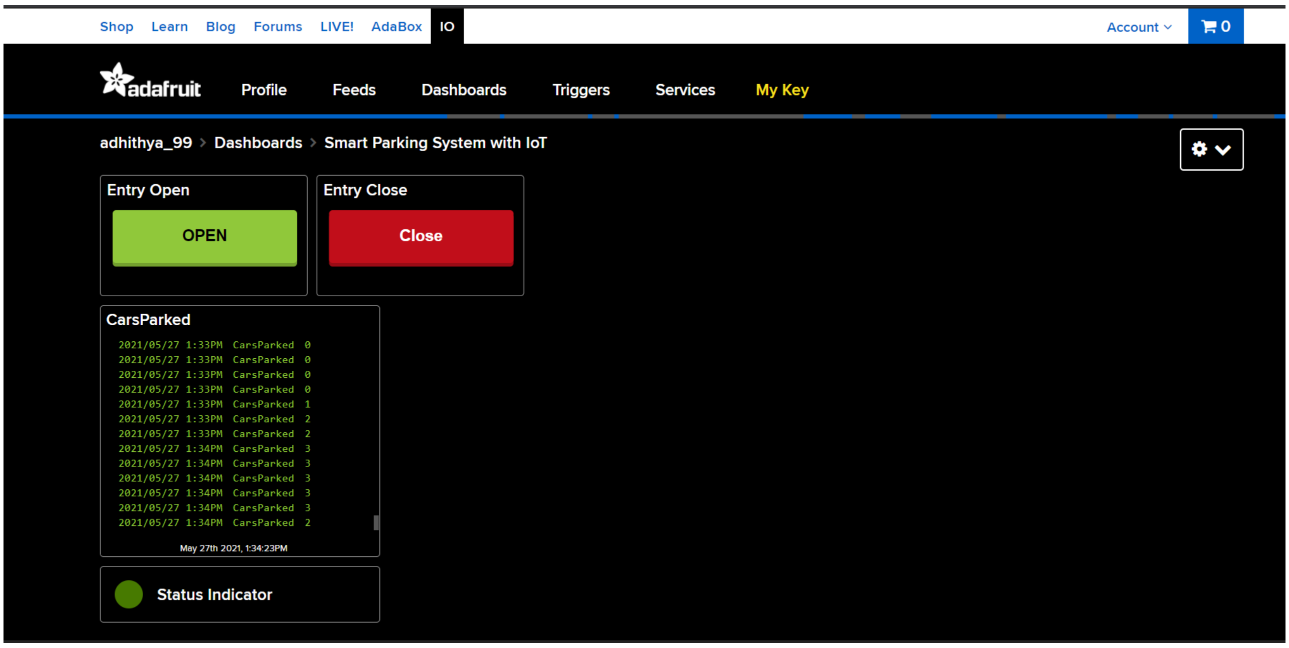
The connections for the servo motor with the NodeMCU are also similar to the connections of IR sensors. The VCC and GND pins shown in Figure 3.6 are connected to the power supply and GND ports of the NodeMCU shown in Figure 3.2 and the PWM output signal pin from the servo motor is wired with the digital pins of the MCU. Table 6.2 shows the connection between Servo motor and NodeMCU.

NODEMCU	**SERVO MOTOR**
VCC PORT	VCC PIN
GND PORT	GND PIN
GPIO/DIGITAL PIN	PWM SIGNAL PIN (D5)

Table 6.2 Connection between Servo Motor and NodeMCU.

6.2 Working

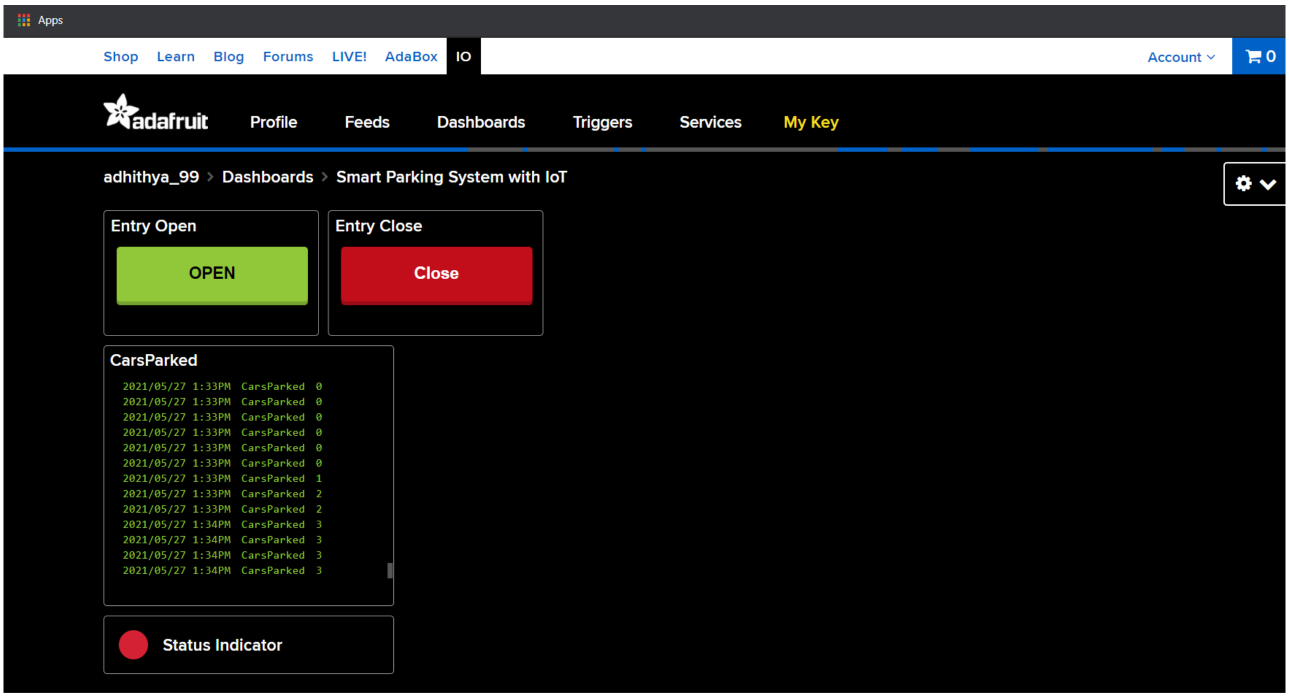
Subsequently, the NodeMCU shown in Figure 3.1 is given a power supply. The Sensors interfaced with the MCU’s VCC and GND as in Figure 3.2 are activated and tries to detect any objects within their effective range. If a car is detected at the entry gate IR sensor, it sends a signal to MCU about that object, and MCU acknowledges the signal and instructs the servo motor to rotate at a precoded angle. The car can be parked in any of the available slots. The servo comes back to its start position/closed position after a brief interval of time. This data is concurrently published to the adafruit io feed using the MQTT protocol coded to the NodeMCU via the internet. Similarly, if a car is detected at the exit gate IR sensor the MCU opens the servo/gate to facilitate the passage of the gate. If all the slots are occupied the servo motor stops functioning and the status indicator turns into red color indicating non-availability of parking spaces and turns green when a slot is available for parking. Figure 6.1 shows the working of the prototype as the status indicator on the screen is green indicates parking spaces are available and on the other dashboard displays the number of cars parked.



```
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  1
2021/05/27 1:33PM  CarsParked  2
2021/05/27 1:33PM  CarsParked  2
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  2
```

May 27th 2021, 1:34:23PM

Figure 6.1 Dashboard of the Prototype.



```
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  0
2021/05/27 1:33PM  CarsParked  1
2021/05/27 1:33PM  CarsParked  2
2021/05/27 1:33PM  CarsParked  2
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
2021/05/27 1:34PM  CarsParked  3
```

Figure 6.2 Maximum value displayed on the dashboard.

Figure 6.2 shows that the number of available spots for parking is maxed out and the dashboard displays the total number of cars parked in the prototype parking spot. “Entry Open” and “Entry Close” buttons are shown on the screen used to control them manually via the internet from anywhere around the world.

6.3 Prototype Model

Figure 6.3 shows the prototype model of this project.

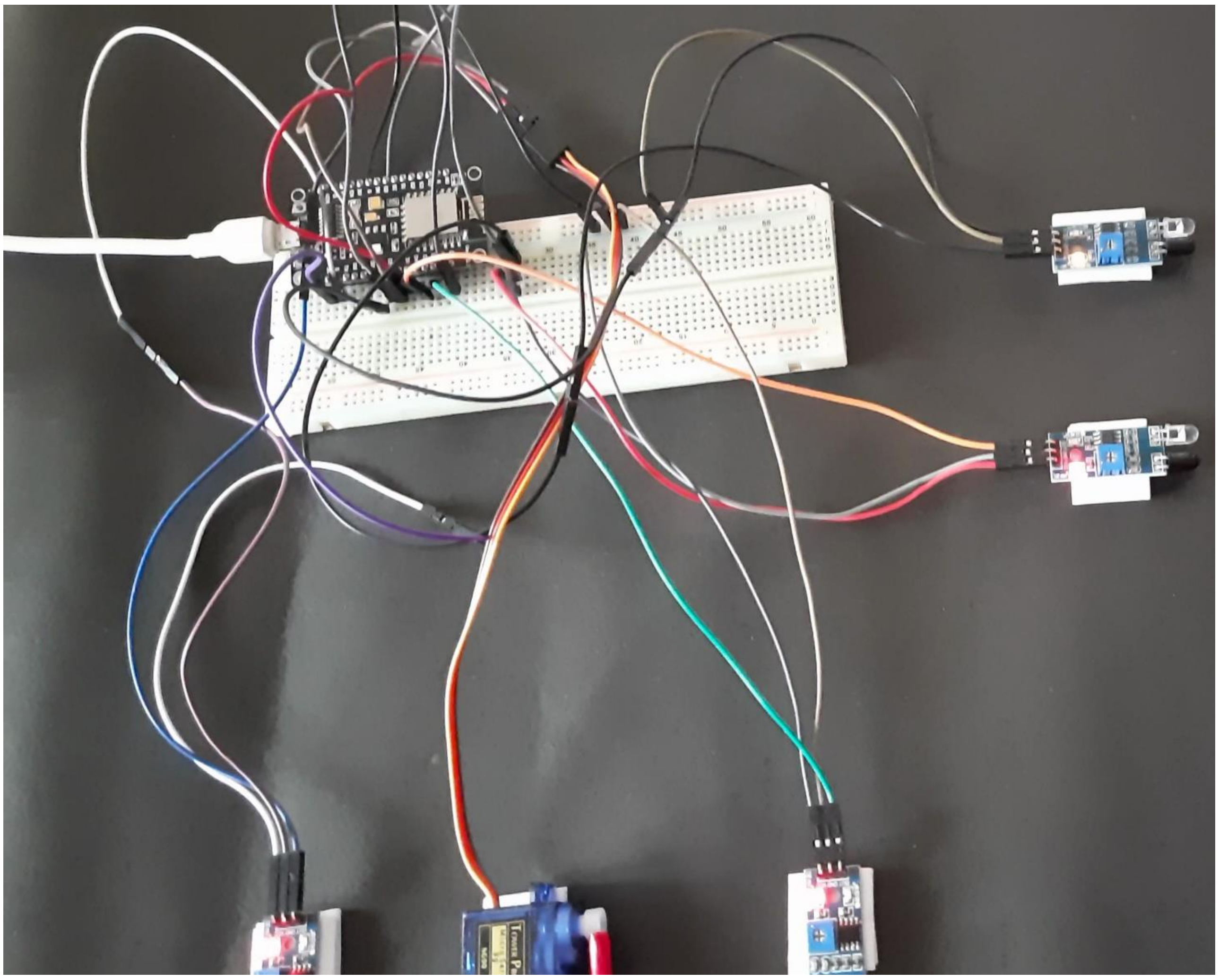


Figure 6.3 Prototype Model.

6.4 Results

After all the connections were given as shown in Figure 5.1 and Figure 5.2 and the prototype is powered on. The completed connection should look like Figure 6.3. The Serial monitor should be opened to make sure the NodeMCU is connected to your LAN. The following steps will demonstrate testing the prototype.

- Check for the blinking lights in the MCU and Power led light on the sensors
- Open the adafruit.io webpage to check the feed data with the timestamp

- If the feed data keeps updating the number of cars parked every few seconds then the prototype is working. If the feed block on the dashboard is empty then the connections need to be checked.
- Check if the number of cars parked is in a positive value, if it's in negative then the code should be revised.
- After all the slots are occupied, check the servo motor if it opens the gate one more if it does then the code should be examined again.

With the procedures mentioned, the implementation of the project “Smart Parking System with IoT” is completed successfully and implemented. The project is cost-efficient and easy to use. With the authentication key, it can be managed by organizations and corporations via the internet.

6.5 Advantages

- It has low implementation and low operating costs.
- In the long run, it saves a considerable amount of fuel for the beneficiaries.
- It is an unsophisticated system and has a trouble-free operation.
- Contactless process.
- Helps in the development of smart cities.
- Indirectly reduces traffic congestion.

6.6 Disadvantages

- Occasional system software updates and requires the system to be taken offline for a few amount of time.
- As the system can be controlled via the internet it is prone to cyberattacks.
- Sensors are sensitive to environmental changes.

Chapter 7

CONCLUSION

The smart parking system with IoT has been experimentally proven to work satisfactorily by subjecting it to the real-world-like situation and controlled it via the internet by testing with a working prototype. This work has presented the design of an efficient, cost-effective, and reliable internet/cloud-based smart parking system which can store the data of parking lot in its systems and control the system remotely via the internet from anywhere around the world. This project helps in hand with the modernization of cities even further and reduces the costs involved in managing and maintaining a parking lot.

This project is also statistically proven to be reducing the time wasted by drivers in search of a parking lot as availability of the parking slots are available which can also be changed by the organizations governing the parking lots as it is flexible to use and control. With further innovations into this idea, it can be made more user/driver-friendly to use for instance to book a parking space even the user gets into the car and reach the destination. This project is highly scalable for use in metropolises and mega-corporations and can be implemented quickly and taken to operation. Though the project is cost-effective it cannot still scale into all the public parking space due to changes in the political climate and lack of oversight from the authorities of the ministry of transportation.

Though overall the project is completed successfully, further study could be conducted to consider using other types of range effective sensors and highly responsive servos to the improvement of the system. Improvements like verification email and SMS services could be even more beneficial for the users to navigate across the system.

REFERENCES

1. Ioannikis Chatzigiannakis, Andrea Vitteli, Apostolos Pyrgelis (2016) “A privacy-preserving smart parking system using an IoT elliptic curve based security platform”, Computer Communications Vol. 89-90, pp 165-177.
2. A. Khanna, R. Anand (2016) “IoT based smart parking system”, in International Conference on Internet of Things and Applications (IOTA), Maharashtra Institute of Technology, Pune, India
3. D. Ganesh Gopal, M. Asha Jerlin and M. Abirami (2019), “A smart parking system using IoT”, World Review of Entrepreneurship, Management and Sustainable Development, , vol. 15, issue 3, pp 335-345.
4. P. Ramaswamy (2016), "IoT smart parking system for reducing greenhouse gas emission," International Conference on Recent Trends in Information Technology (ICRTIT), 2016, pp. 1-6.
5. D. Vakula and Y. K. Kolli (2017), "Low-cost smart parking system for smart cities," International Conference on Intelligent Sustainable Systems (ICISS), 2017, pp. 280-284.
6. J. Shi, L. Jin, J. Li, and Z. Fang (2017), "A smart parking system based on NB-IoT and third-party payment platform,"17th International Symposium on Communications and Information Technologies (ISCIT), pp. 1-5.
7. P. Chippalkatti, G. Kadam and V. Ichake (2018), "I-SPARK: IoT Based Smart Parking System,"International Conference On Advances in Communication and Computing Technology (ICACCT), 2018, pp. 473-477.
8. M. Ramasamy, S. G. Solanki, E. Natarajan (2018), and T. M. Keat, "IoT Based Smart Parking System for Large Parking Lot,"IEEE 4th International Symposium in Robotics and Manufacturing Automation (ROMA), 2018, pp. 1-4.
9. I. Aydin, M. Karakose and E. Karakose (2017), "A navigation and reservation-based smart parking platform using genetic optimization for

smart cities," 5th International Istanbul Smart Grid and Cities Congress and Fair (ICSG), 2017, pp. 120-124.

10. D. Ashok, A. Tiwari, and V. Jirge (2020), "Smart Parking System using IoT Technology,"International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020, pp. 1-7.

11. P. Sadhukhan, "An IoT-based E-parking system for smart cities," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017.

12. R. Kanan and H. Arbess (2020), "An IoT-Based Intelligent System for Real-Time Parking Monitoring and Automatic Billing,"IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), 2020, pp. 622-626.

13. Y. -Y. Chu and K. -H. Liu (2020), "IoT in Vehicle Presence Detection of Smart Parking System,"IEEE Eurasia Conference on IoT, Communication, and Engineering (ECICE), 2020, pp. 56-59.

14. Ioannis Agadakos; Prashant Anantharaman; Gabriela F. Ciocarlie; Bogdan Copos; Michael Emmi; Tancrède Lepoint; Ulf Lindqvist; Michael Locasto; Liwei Song (2020), "Securing Smart Cities," in Modeling and Design of Secure Internet of Things, IEEE, pp.185-215.

15. M. Meenaloshini, J. Ilakkiya, P. Sharmila, J. C. Sheffi Malar and S. Nithyasri (2019), "Smart Car Parking System in Smart Cities using IR,"3rd International Conference on Computing and Communications Technologies (ICCCT), 2019, pp. 178-182.

APPENDIX

I. Code for the Prototype

```
#include <ESP8266WiFi.h>
#include <Servo.h>
#include <stdlib.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <NTPClient.h>;
#include <WiFiUdp.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
const char *ssid =  "wifi_ssid";
const char *pass =  "password";
#define MQTT_SERV "io.adafruit.com"
#define MQTT_PORT 1883
#define MQTT_NAME "adafruit_username"
#define MQTT_PASS "aio_EgOI174ZY2GQwnF5gxxxxx"
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 19800,60000);
Servo myservo;
int carEnter = D0;
int carExited = D2
int slot1 = D3;
int slot2 = D6;
int slot3  = D7;
int count =0;
int CLOSE_ANGLE = 90;
int OPEN_ANGLE = 0;
int  hh, mm, ss;
int pos;
```

```
String h, m, EntryTimeSlot1, ExitTimeSlot1, EntryTimeSlot2, ExitTimeSlot2,
EntryTimeSlot3, ExitTimeSlot3
boolean entrysensor, exitsensor,s1,s2,s3;
boolean s1_occupied = false;
boolean s2_occupied = false;
boolean s3_occupied = false;

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, MQTT_SERV, MQTT_PORT, MQTT_NAME,
MQTT_PASS);

Adafruit_MQTT_Subscribe EntryGate = Adafruit_MQTT_Subscribe(&mqtt,
MQTT_NAME "/f/EntryGate");
Adafruit_MQTT_Subscribe ExitGate = Adafruit_MQTT_Subscribe(&mqtt,
MQTT_NAME "/f/ExitGate");

//Set up the feed you're publishing to
Adafruit_MQTT_Publish CarsParked = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/CarsParked");
Adafruit_MQTT_Publish EntrySlot1 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/EntrySlot1");
Adafruit_MQTT_Publish ExitSlot1 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/ExitSlot1");
Adafruit_MQTT_Publish EntrySlot2 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/EntrySlot2");
Adafruit_MQTT_Publish ExitSlot2 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/ExitSlot2");
Adafruit_MQTT_Publish EntrySlot3 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/EntrySlot3");
Adafruit_MQTT_Publish ExitSlot3 = Adafruit_MQTT_Publish(&mqtt,MQTT_NAME
"/f/ExitSlot3");
```

```
void setup() {
  delay(1000);
  Serial.begin (9600);
  mqtt.subscribe(&EntryGate);
  mqtt.subscribe(&ExitGate);
  timeClient.begin();
  myservo.attach(D5);
  pinMode(carExited, INPUT);
  pinMode(carEnter, INPUT);
  pinMode(slot1, INPUT);
  pinMode(slot2, INPUT);
  pinMode(slot3, INPUT);
  WiFi.begin(ssid, pass);
  Serial.print("Connecting to ");
  Serial.print(ssid);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP Address is : ");
  Serial.println(WiFi.localIP());
}

void loop() {

 MQTT_connect();
 timeClient.update();
 hh = timeClient.getHours();
 mm = timeClient.getMinutes();
```

```
ss = timeClient.getSeconds();
h= String(hh);
m= String(mm);
h +" :" + m;

entrysensor= !digitalRead(carEnter);
exitsensor = !digitalRead(carExited);
s1 = digitalRead(slot1);
s2 = digitalRead(slot2);
s3 = digitalRead(slot3);

  if (entrysensor == 1 and count<3) {
  count=  count+1;
  myservo.write(OPEN_ANGLE);
  delay(3000);
  myservo.write(CLOSE_ANGLE);
  }

  if (exitsensor == 1 and count>0) {
  count= count-1;
  myservo.write(OPEN_ANGLE);
  delay(3000);
  myservo.write(CLOSE_ANGLE);
 }
 if (! CarsParked.publish(count)) {}

 if (s1 == 1 && s1_occupied == false) {
       Serial.println("Available1 ");
       EntryTimeSlot1 =  h +" :" + m;
       //Serial.print("EntryTimeSlot1");
       //Serial.print(EntryTimeSlot1);
```

```
        s1_occupied = true;
        if (! EntrySlot1.publish((char*) EntryTimeSlot1.c_str())){}
    }
  if(s1 == 0 && s1_occupied == true) {
       Serial.println("Occupied1 ");
       ExitTimeSlot1 =  h +" :" + m;
       //Serial.print("ExitTimeSlot1");
       //Serial.print(ExitTimeSlot1);

       s1_occupied = false;
        if (! ExitSlot1.publish((char*) ExitTimeSlot1.c_str())){}
}
  if (s2 == 1&& s2_occupied == false) {
      Serial.println("Available2 ");
      EntryTimeSlot2 =  h +" :" + m;
      //Serial.print("EntryTimeSlot2");
      //Serial.print(EntryTimeSlot2);

      s2_occupied = true;
      if (! EntrySlot2.publish((char*) EntryTimeSlot2.c_str())){}
    }
  if(s2 == 0 && s2_occupied == true) {
       Serial.println("Occupied2 ");
       ExitTimeSlot2 =  h +" :" + m;
       //Serial.print("ExitTimeSlot2");
       //Serial.print(ExitTimeSlot2);

       s2_occupied = false;
        if (! ExitSlot2.publish((char*) ExitTimeSlot2.c_str())){}
  }
  if (s3 == 1&& s3_occupied == false) {
      Serial.println("Available3 ");
      EntryTimeSlot3 =  h +" :" + m;
     //Serial.print("EntryTimeSlot3: ");
      //Serial.print(EntryTimeSlot3);
      s3_occupied = true;
       if (! EntrySlot3.publish((char*) EntryTimeSlot3.c_str())){}
    }
  if(s3 == 0 && s3_occupied == true) {
       Serial.println("Occupied3 ");
       ExitTimeSlot3 =  h +" :" + m;
       //Serial.print("ExitTimeSlot3: ");
       //Serial.print(ExitTimeSlot3);
       s3_occupied = false;
        if (! ExitSlot3.publish((char*) ExitTimeSlot3.c_str())){ }
  }

  Adafruit_MQTT_Subscribe * subscription;
  while ((subscription = mqtt.readSubscription(5000)))
     {

   if (subscription == &EntryGate)
     {
      //Print the new value to the serial monitor
      Serial.println((char*) EntryGate.lastread);

   if (!strcmp((char*) EntryGate.lastread, "ON"))
      {
       myservo.write(OPEN_ANGLE);
    }
}
  if (subscription == &ExitGate)
     {
      //Print the new value to the serial monitor
      Serial.println((char*) EntryGate.lastread);

   if (!strcmp((char*) ExitGate.lastread, "ON"))
      {
       myservo.write(CLOSE_ANGLE);
    }
}
}
}
void MQTT_connect()
{
  int8_t ret;

  if (mqtt.connected())
  {
    return;
  }
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0)
  {
       mqtt.disconnect();
       delay(5000);
       retries--;
       if (retries == 0)
       {
         while (1);
       } } }
```