./

IOT Based Smart Parking System

**Table of Contents**

[Arduino Flow chart 4](#_Toc58490683)

[Parking Slot Module 4](#_Toc58490684)

[Exit Module 6](#_Toc58490685)

[State Transition Diagram of Smart Parking System 7](#_Toc58490686)

[MQTT Cloud Architecture 10](#_Toc58490687)

[Wi-Fi Architecture 11](#_Toc58490688)

[Registration 14](#_Toc58490689)

[Validation of details 15](#_Toc58490690)

[Menu 15](#_Toc58490691)

[Book Slot 16](#_Toc58490692)

[QR Code Generation 16](#_Toc58490693)

[Bill Generation 17](#_Toc58490694)

**Table of Figures**

[Figure 1: Entrance Module 4](#_Toc58490683)

[Figure 2: Parking Slot Module 4](#_Toc58490684)

[Figure 3: Exit Module 6](#_Toc58490685)

[Figure 4: State Transition Diagram of Smart Parking System 7](#_Toc58490686)

[Figure 5: MQTT Cloud Architecture 10](#_Toc58490687)

[Figure 6: Wi-Fi Architecture 11](#_Toc58490688)

[Figure 7: Registration 14](#_Toc58490689)

[Figure 8: Validation of details 15](#_Toc58490690)

[Figure 9:Menu 15](#_Toc58490691)

[Figure 10: Book Slot 16](#_Toc58490692)

[Figure 11: QR Code Generation 16](#_Toc58490693)

[Figure 12: Bill Generation 17](#_Toc58490694)

**ARDUINO FLOW CHART**

1. **ENTRANCE: -**

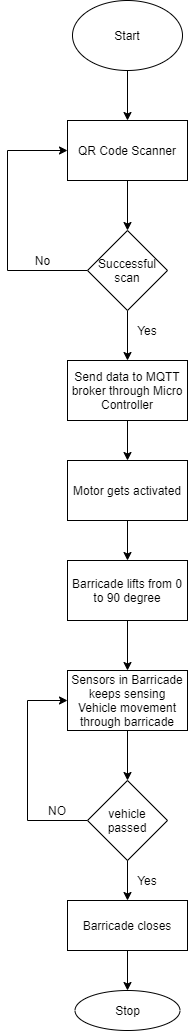


Figure 1: Entrance Module

The user can book the slot by registering through the android app, after confirmation of slot a unique QR code is generated to every user, QR code authentication is done in cloud which has the details of the user stored in MQTT broker, after successful validation of QR code the motor gets activated, barricade lifts from o degree to 90 degrees.

The sensors in the barricade keeps sensing the movement of vehicle through barricade, if vehicle passes the barricade the barricade senses the movement of vehicle and then reverts to 0 degree.

1. **PARKING SLOT: -**

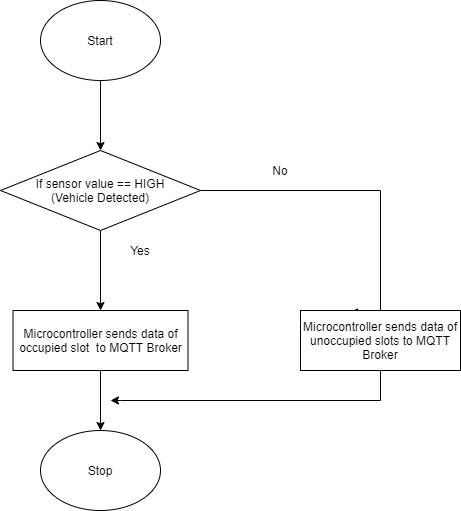


Figure 2: Parking Slot Module

The sensors placed at each parking slot senses the presence of vehicle, as each user will be provided with a slot number after registration process. As the sensors value goes high which indicates the presence of vehicle, microcontroller sends data of both occupancy and inoccupancy of slot to MQTT Broker, and the data gets stored in the MQTT Broker and is retrieved from broker to Android App.

1. **EXIT: -**

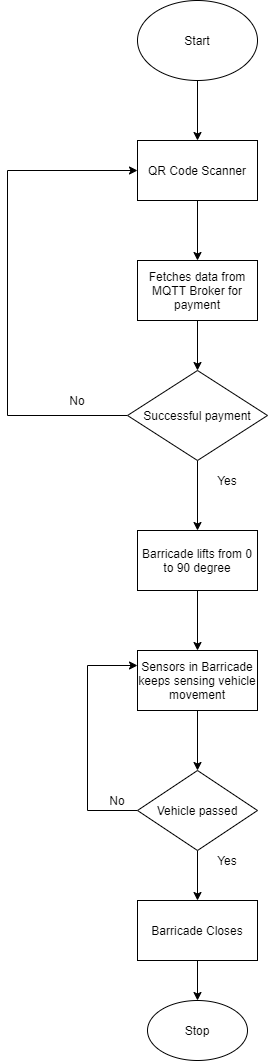


Figure 3: Exit Module

During exit, user should scan the QR code which fetches the amount of time the vehicle was parked in the respective slot. Based on the time calculated Bill is generated through the app and different online payment modes are provided.

On successful Bill payment motor gets activated and the barricade lifts from 0 degree to 90 degrees. Sensors placed in barricade senses the movement of vehicle, as vehicle passes through barricade it reverts to 0 degree.

**DESCRIPTION OF STATE TRANSITION DIAGRAM: -**

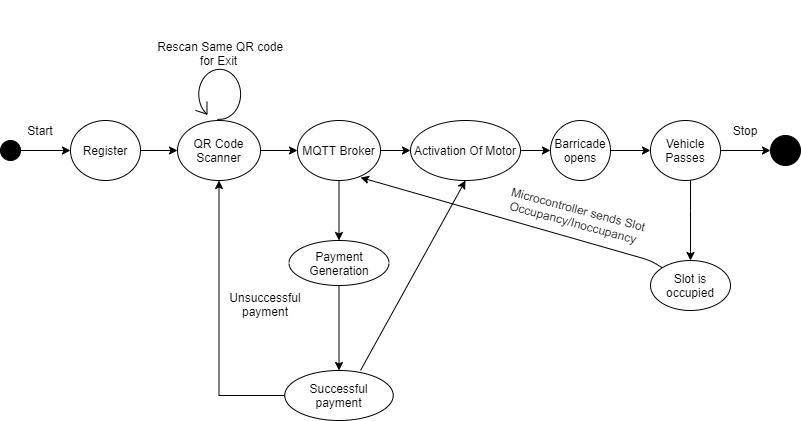


Figure 4: State Transition Diagram of Smart Parking System

**Registration**

To enter the parking system the user should first register through the app, which needs details like name, phone number and vehicle number of the user.

**QR Code Scanning**

A unique QR code will be generated for each user after registration, on successful scan of QR code the user can park the vehicle in the respective slot number.

**MQTT Broker**

MQTT Broker serves as a cloud, which stores all the data of the user, authenticates the QR code and fetches the data from sensors through microcontroller and provides android app in successful retrieval of data from cloud.

**Activation of Motor and Barricade Opens**

The DC motor placed in the barricade gets activated on successful scan of QR code, which further makes the barricade to lift from 0 degree to 90 degrees for the vehicle to pass through it.

**Vehicle Passes**

The sensors placed in the barricade keeps sensing the movement of vehicle, once the motor gets activated the barricade lifts and senses the movement of vehicle, after the vehicle crosses the barricade, it reverts from 90 degrees to 0 degree.

**Slot Occupancy**

The android app provides the slot number after confirmation of booking. The sensors placed at each slot detects the presence of vehicle and sends the data to cloud about the occupancy and inoccupancy of slot through microcontroller.

**Payment Generation**

The sensors placed at each slot helps in calculating the amount of time the vehicle was parked in the respective parking slot, by fetching the time details, bill is generated through app and different online payment modes are provided for the users to proceed with their payment.

**Successful Payment**

The payment validation is done in cloud, on successful payment the motor in the barricade gets activated and the barricade lifts from 0 degree to 90 degrees and the vehicle exits from parking system.

**SOFTWARE REQUIREMENTS: -**

* **Arduino IDE**



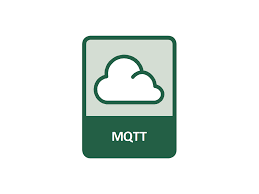
The Arduino IDE employs the program to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. It is used as the uploading tool to flash the user code onto official Arduino boards. The Arduino board is connected to a computer via USB, where it connects with the Arduino Development Environment(IDE). The user writes the Arduino code in the IDE, then uploads it to the microcontroller which executes the code, interacting with inputs and outputs such as sensors, motors and lights.

* **MIT App Inventor**



MIT App inventor for android is an open source web application maintained by Massachusetts Institute of Technology(MIT). It allows to create software applications for the android operating system. It uses graphical interface which allows users to drag and drop visual objects to create an application that can run on android device. MIT app inventor is also supported with the MQTT Broker extension, this allows to store data on MQTT Broker.

* **MQTT Broker**



MQTT is a lightweight publish/subscribe messaging protocol designed for low-bandwidth, high latency, unreliable networks. MQTT’s features make it an excellent option for sending high volumes of sensor messages to analytics platforms and cloud solutions.

**CLOUD ARCHITECTURE**

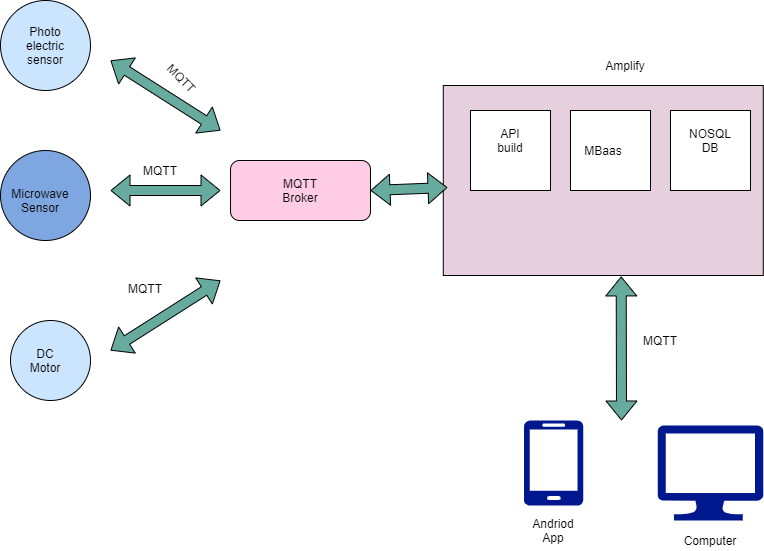


Figure 5: MQTT Cloud Architecture

The data collected from each sensor at each module is published to the MQTT Broker with assigned topic. In MQTT Broker data is efficiently stored and analysis is performed on the raw data to provide useful information to the subscribers. The Android application or any device can subscribe to the topics published by the sensors to perform necessary operations.

**WIRELESS COMMUNICATION ARCHITECTURE: -**

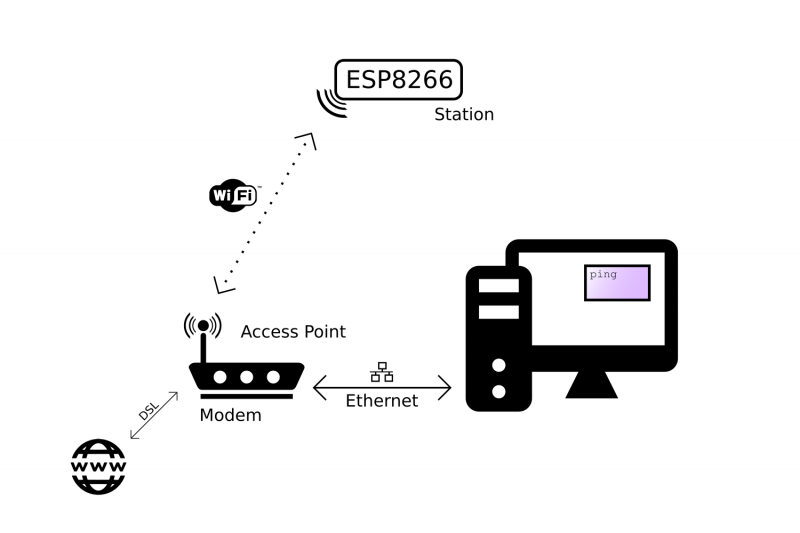
****

Figure 6: Wi-Fi Architecture

The device with the antenna serves many different purposes:

**Access point**: Other Wi-Fi devices can connect to it, to be part of the local network.

**Router:** It routes IP packets to the right sub-nets so that they will arrive at their destination. E.g. if the computer sends a message that is meant for the ESP over the Ethernet sub-net, the router will send the packet to the Wi-Fi sub-net, because it knows that's where the ESP is present.

**Modem:** if the router can't find the addressee on the local network, the packet will be passed on to the integrated modem, and it will be sent to the Internet Service Provider over a DSL line, heading for the Internet, where lots of other routers will try to get the packet to the right destination.

**CONNECTIONS: -**

VCC to +5v power

PA4 or A/D3 of uC to IN pin of photoelectric sensor

PA7 or A/D7 of uC to INA of Motor Driver (L298N)

PA6 or A/D6 of uC to INB of Motor Driver (L298N)

PA5 or A/D5 of uC to ENA of Motor Driver (L298N)

VCC of L298N connected to 5V pin of SMPS

VS of L298N connected to 12V pin of SMPS

GND of L298N connected to 99 pin of uC

PD3 or TXD1 or INT3 of uC to Rx pin of ESP8266

PD2 or RXD1 or INT2 of uC to Tx pin of ESP8266

GND of ESP8266 connected to 81 pin of uC

**Connect to WIFI and wait for some time till it gets connected.**

#include <ESP8266WiFi.h>

#define WIFI\_SSID “GENESIS”

#define WIFI\_ PASSWORD “123456”

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

while (WiFi.status()! = WL\_CONNECTED) {

delay (500);

}

**Connecting DC Motor to Arduino:**

int sensorpin = 75; //photoelectric sensor

int INA = 71;

int INB = 72;

int ENA = 73;

boolean state = LOW;

int read;

pinMode (sensorpin, INPUT);

pinMode (INA, OUTPUT);

pinMode (INB, OUTPUT);

state = digitalRead(sensorpin);

if (state == HIGH) {

digitalWrite (INA, LOW);

digitalWrite (INB, HIGH);

}

if (state == LOW) {

digitalWrite (INA, HIGH);

digitalWrite (INB, LOW);

}

**Connecting Photoelectric Sensor to Microcontroller:**

int sensorpin = 75;

boolean state = LOW;

int read;

pinMode (sensorpin, INPUT);

read = digitalRead(sensorpin);

**SCREENSHOTS OF ANDRIOD APP: -**

1. Registration

****

Figure 7: Registration

1. Validation Of details

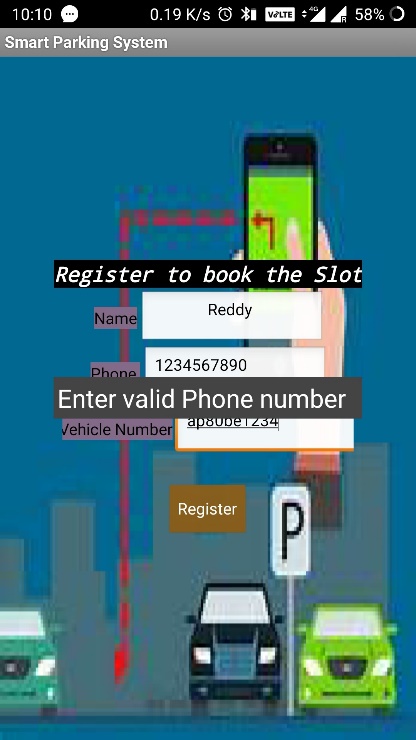


Figure 8: Validation of details

1. Menu

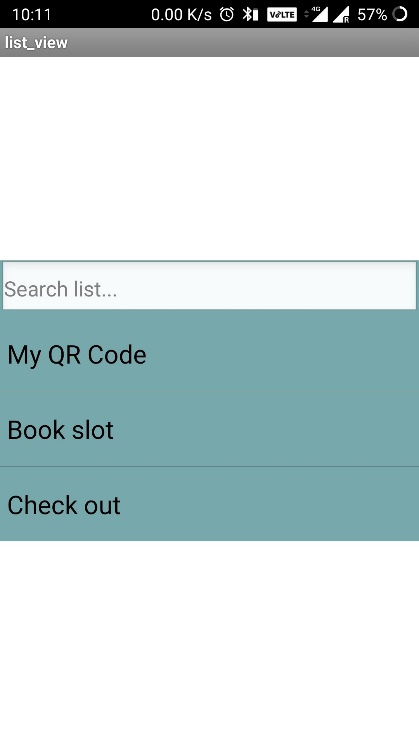


Figure 9:Menu

1. Book the Slot



Figure 10: Book Slot

1. QR Code Generation

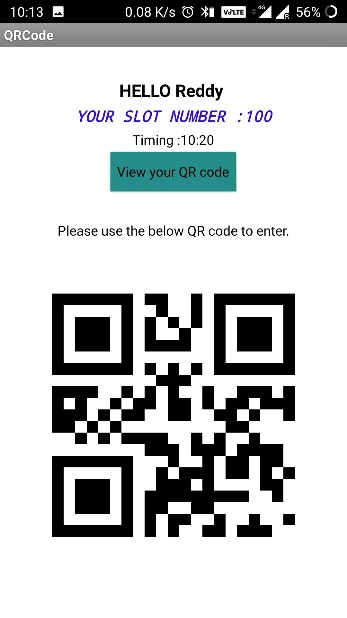


Figure 11: QR Code Generation

1. Bill Generated

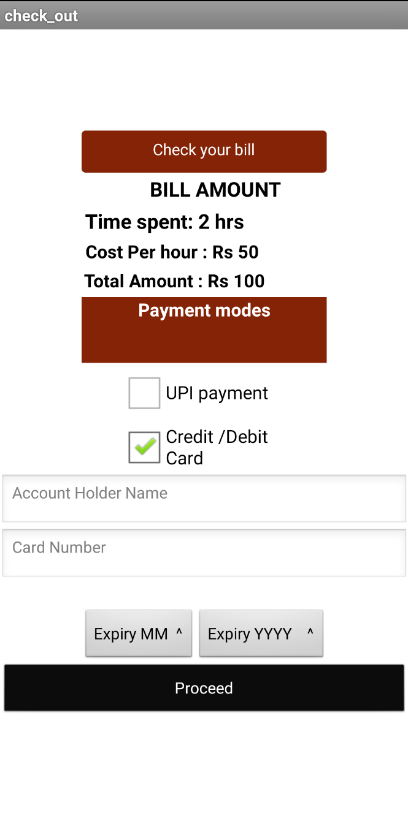


Figure 12: Bill Generation