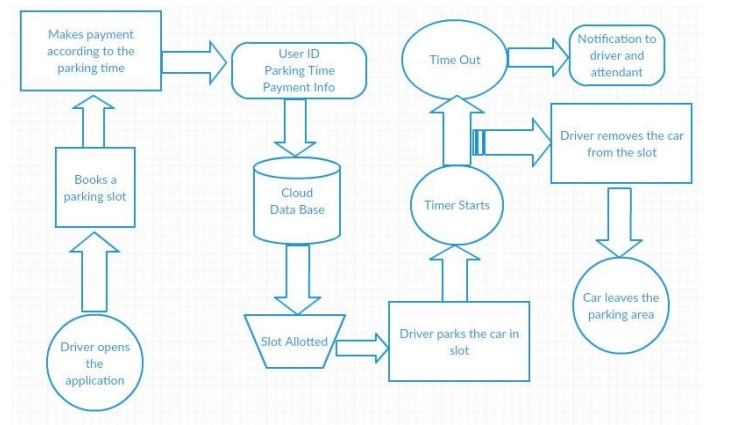
**SMART PARKING USING IOT**

**PROJECT DESCRIPTION:**

The project entitled smart parking system is to manage all the parking facilities to an user. The recent growth in economy and due to the availability of low price cars in the market, an every average middle-class individual can afford a car, which is good thing, however the consequences of heavy traffic jams, pollution, less availability of roads and spot to drive the motor car. One of the important concerns, which is to be taken in accounting, is the problem of parking those vehicles .Though, if there is space for parking the vehicle but so much time is squandered in finding that exact parking slot resulting in more fuel intake and not also environment friendly. Initially when the user is about to enter the location the LCD displays the number of empty and filled spots and when the user is with its vehicle near to the parking detect sensor ,he/she would be thrown with a notification on their mobile app of the parking slot number ,where they should park there vehicle. The main important benefit of a smart parking system is its advanced technology. It follows the latest technologies and concepts to assure profitable outcomes . The design and implementation of smart parking is very easy to supervise and manage. This system can be easily handled by the staff members because of its well organized structure.

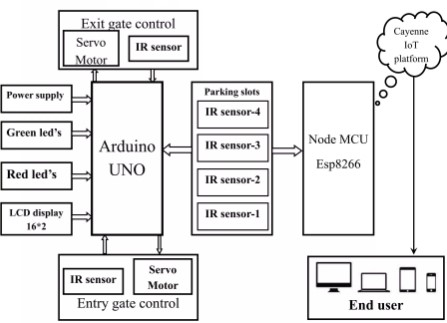
**IOT IMPLEMENTATION:** 

It consists of one node mcu , one dc motor , one 16\*2 LCD display and three IR sensors .The node mcu is the brain of our system which powers all the other devices .The 16\*2 LCD display is powered by node mcu by connecting jumper wires from the display to node mcu . The DC motor is also powered by node mcu with connecting its pins to node mcu. The IR sensor consists of three pins, where two pins refer to the power supply and ground and the other pins refer to the pin which is going to be connected in the Node mcu. On successfully connecting all the components in the given figure now we have to connect the blynk app. While using the blynk app we have to specify the widgets used in our android app and the pin number to which they are connected to node mcu in the actual model so that the mobile app will react exactly to the inputs provided in the model.

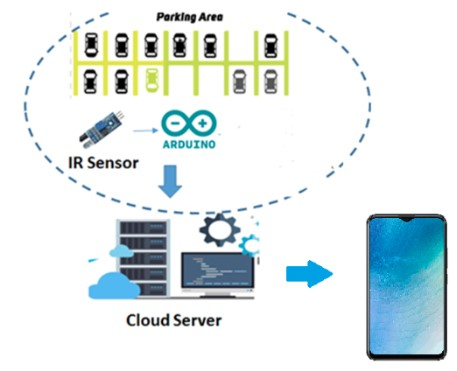
**METHODOLOGY:**

In this project we are using NodeMCU, IR sensors, and servo motors. One IR sensor is used at entry and exit gate to detect the car while two IR sensors are used to detect the parking slot availability. Servo motors are used to open and close the gates according to the sensor value. NodeMCU is an open source IoT platform .It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware, which is based on the ESP-12 module. The term “NodeMCU” by default refers to the firmware rather than the dev kits. The firmware uses the Lua scripting language. The ESP8266 is a low-cost Wi-Fi enabled microchip with full TCP/IP stack and microcontroller capabilities. The idea behind our methodology is very simple , usually users spend most of their time in looking for an empty slot where they can park their vehicle which increases fuel consumption and time wastage. We came-up with a new method where we provide the user an empty slot number where he can park his vehicle without wasting his time for finding one . Similarly we try to display the start time and end time so that the user can know for what amount of time he has parked his vehicles.

**BLOCK DIAGRAM:**



**PROTOTYPE:**



**APP DEVELOPMENT:**

There are two sensors at the entrance to detect the presence of a car before going inside or outside of the parking lot. The other two sensors are plotted inside the parking lot to detect the car individually for each parking slot. A DC Servo motor has been used at the entrance to open and close the gate according to the signals sent by the sensors through Arduino. We have used two IR sensors which when vehicle parked will show appropriate message to the user and when all the parking slots the dc motor would not open gate for the vehicle to be parked.

**CODING PROGRAM FOR ARDUINO:**

#ifndef LiquidCrystal\_I2C\_h

#define LiquidCrystal\_I2C\_h

#include <inttypes.h>

#include "Print.h"

#include <Wire.h>

#define LCD\_CLEARDISPLAY 0x01

#define LCD\_RETURNHOME 0x02

#define LCD\_ENTRYMODESET 0x04

#define LCD\_DISPLAYCONTROL 0x08

#define LCD\_CURSORSHIFT 0x10

#define LCD\_FUNCTIONSET 0x20

#define LCD\_SETCGRAMADDR 0x40

#define LCD\_SETDDRAMADDR 0x80

#define LCD\_ENTRYRIGHT 0x00

#define LCD\_ENTRYLEFT 0x02

#define LCD\_ENTRYSHIFTINCREMENT 0x01

#define LCD\_ENTRYSHIFTDECREMENT 0x00

#define LCD\_DISPLAYON 0x04

#define LCD\_DISPLAYOFF 0x00

#define LCD\_CURSORON 0x02

#define LCD\_CURSOROFF 0x00

#define LCD\_BLINKON 0x01

#define LCD\_BLINKOFF 0x00

#define LCD\_DISPLAYMOVE 0x08

#define LCD\_CURSORMOVE 0x00

#define LCD\_MOVERIGHT 0x04

#define LCD\_MOVELEFT 0x00

#define LCD\_8BITMODE 0x10

#define LCD\_4BITMODE 0x00

#define LCD\_2LINE 0x08

#define LCD\_1LINE 0x00

#define LCD\_5x10DOTS 0x04

#define LCD\_5x8DOTS 0x00

#define LCD\_BACKLIGHT 0x08

#define LCD\_NOBACKLIGHT 0x00

#define En B00000100 // Enable bit

#define Rw B00000010 // Read/Write bit

#define Rs B00000001 // Register select bit

class LiquidCrystal\_I2C : public Print {

public:

LiquidCrystal\_I2C(uint8\_t lcd\_Addr,uint8\_t lcd\_cols,uint8\_t lcd\_rows);

void begin(uint8\_t cols, uint8\_t rows, uint8\_t charsize = LCD\_5x8DOTS );

void clear();

void home();

void noDisplay();

void display();

void noBlink();

void blink();

void noCursor();

void cursor();

void scrollDisplayLeft();

void scrollDisplayRight();

void printLeft();

void printRight();

void leftToRight();

void rightToLeft();

void shiftIncrement();

void shiftDecrement();

void noBacklight();

void backlight();

void autoscroll();

void noAutoscroll();

void createChar(uint8\_t, uint8\_t[]);

void createChar(uint8\_t location, const char \*charmap);

const char bell[8] PROGMEM = {B00100,B01110,B01110,B01110,B11111,B00000,B00100,B00000};

void setCursor(uint8\_t, uint8\_t);

#if defined(ARDUINO) && ARDUINO >= 100

virtual size\_t write(uint8\_t);

#else

virtual void write(uint8\_t);

#endif

void command(uint8\_t);

void init();

void oled\_init();

void blink\_on();

void blink\_off();

void cursor\_on();

void cursor\_off();

void setBacklight(uint8\_t new\_val);

nobacklight()

void load\_custom\_character(uint8\_t char\_num, uint8\_t \*rows);

void printstr(const char[]);

uint8\_t status();

void setContrast(uint8\_t new\_val);

uint8\_t keypad();

void setDelay(int,int);

void on();

void off();

uint8\_t init\_bargraph(uint8\_t graphtype);

void draw\_horizontal\_graph(uint8\_t row, uint8\_t column, uint8\_t len, uint8\_t pixel\_col\_end);

void draw\_vertical\_graph(uint8\_t row, uint8\_t column, uint8\_t len, uint8\_t pixel\_col\_end);

private:

void init\_priv();

void send(uint8\_t, uint8\_t);

void write4bits(uint8\_t);

void expanderWrite(uint8\_t);

void pulseEnable(uint8\_t);

uint8\_t \_Addr;

uint8\_t \_displayfunction;

uint8\_t \_displaycontrol;

uint8\_t \_displaymode;

uint8\_t \_numlines;

bool \_oled = false;

uint8\_t \_cols;

uint8\_t \_rows;

uint8\_t \_backlightval;

};

#endif

**SIMULATION:**

#include “Thingspeak.h”

#include<ESP8266WiFi.h>

char ssid[]=”SSID”;

char pass[]=”PASSWORD”;

unsigned long channel\_ID=123456;

const char \*myWriteAPIKey=”ACBDE12345”;

const int Field\_Number\_1=1;

string value=” “:

int value\_1=0;

WiFiclient client;

Void setup()

{

Serial.begin(1152100);

WiFi.mode(WIFI\_STA);

Thingspeak.begin(client);

Internet();

}

Void loop()

{

Intrenet();

If(serial.available()>0)

{

Delay(100);

While(serial.available()>0)

{

Valur=serial.readstring();

If(value[0]==’\*’)

{

If(valur[2]==’#’)

{

Value\_1=value[1]-0x30;

}|

}

}

}

Upload();

}

Void internet()

{

If(WiFi.status()!=WL\_CONNECTED)

{

While(WiFi.status()!=WL\_CONNECTED)

{

WiFi.begin(ssid,pass);

Delay(5000);

}

}

]

Void upload()

{  
Thingspeak.writeField(channel\_ID,Field\_Number\_1,value\_1,myWriteAPIKey);

delay(15000);

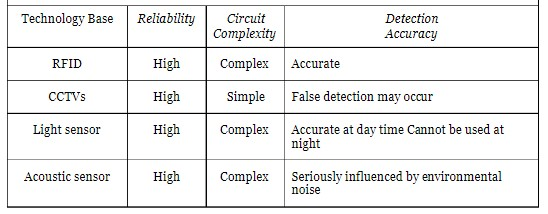
value=” “;

}

**WORKING PROTOTYPE OF THE PROTOTYPE:**

The parking lot information is storage in a database. Once defined the parking lot places, using a camera, the system will get an image named ”Base Image” of each parking lot space taking into account the X and Y coordinates and the width and height values. The”Base Image” will be storage into the database and will be used to verify whether the parking lot place is available or busy. Initially, all parking lot places will be considerate available.

**DATA ANALYSIS FROM THE PROTOTYPE:**



**CONCLUSION:**

Smart parking facilities and traffic management systems have always been at the core of constructing smart cities. In this project, we address the issue of parking and present an IoT based Cloud integrated smart parking system. The system that we propose provides real time information regarding availability of parking slots in a parking area. Users from remote locations could book a parking slot for them by the use of our mobile application. The efforts made in this project are intended to improve the parking facilities of a city and thereby aiming to enhance the quality of life of its people .