DAYANANDA SAGAR COLLEGE OF ENGINEERING
**Department of Electronics and Communication Engineering**
**Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru – 560 078.**
**An Autonomous Institute affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)**
***Accredited by National Assessment and Accreditation Council (NAAC) with 'A' grade***

## Open Ended Project

| | |
|---|---|
| Program: B.E. | Branch: ECE |
| Course: CCN & Embedded Lab | Semester: 6 Section: C |
| Course Code: 19EC6DLCCN | Date: 07-06-2023 |

## A Report on:

# IOT-INTEGRATED SMART PARKING SYSTEM

## Submitted by:



| USN: | NAME: | MARKS: |
|---|---|---|
| 1DS20EC127<br>1DS20EC128<br>1DS20EC140 | POOJA BG<br>POOJA S KUNDARGI<br>PRATIKSHA SAHOO | |



## Faculties In-charge:

**Dr. Shashi Raj K.**
**Prof. Manasa R. K.**

**Signatures of Faculties In-charge**

## OVERVIEW:

An IoT-based smart parking system using IR sensors, Arduino Uno, and NodeMCU ESP8266 is a solution that leverages the Internet of Things (IoT) technology to efficiently manage and monitor parking spaces. This system utilizes IR (Infrared) sensors to detect the presence of vehicles in parking spots, while Arduino Uno and NodeMCU ESP8266 are microcontrollers that facilitate data processing and communication with the IoT network.

**Hardware Setup:**

- IR Sensors: Install IR sensors in each parking spot to detect the presence of vehicles. These sensors emit and receive infrared light, and when a vehicle occupies a spot, it interrupts the light beam, indicating that the spot is occupied.
- Arduino Uno: Connect the IR sensors to the Arduino Uno microcontroller. Arduino Uno will collect data from the sensors and send it to the NodeMCU ESP8266.
- NodeMCU ESP8266: This microcontroller acts as a bridge between the Arduino Uno and the Internet. It receives data from Arduino Uno and sends it to the cloud or a local server.

**Data Processing and Communication:**

- Arduino Uno: When an IR sensor detects a change in the light beam, it sends a signal to the Arduino Uno. The Arduino processes this data and determines whether a parking spot is occupied or vacant.
- NodeMCU ESP8266: Arduino Uno communicates with NodeMCU ESP8266 via serial communication (UART). NodeMCU ESP8266 receives the data from Arduino Uno and establishes an internet connection using Wi-Fi.
- Cloud/Server: NodeMCU ESP8266 sends the parking spot status data (occupied or vacant) to a cloud platform or a local server via Wi-Fi. This data can be stored, analyzed, and accessed remotely through the server.

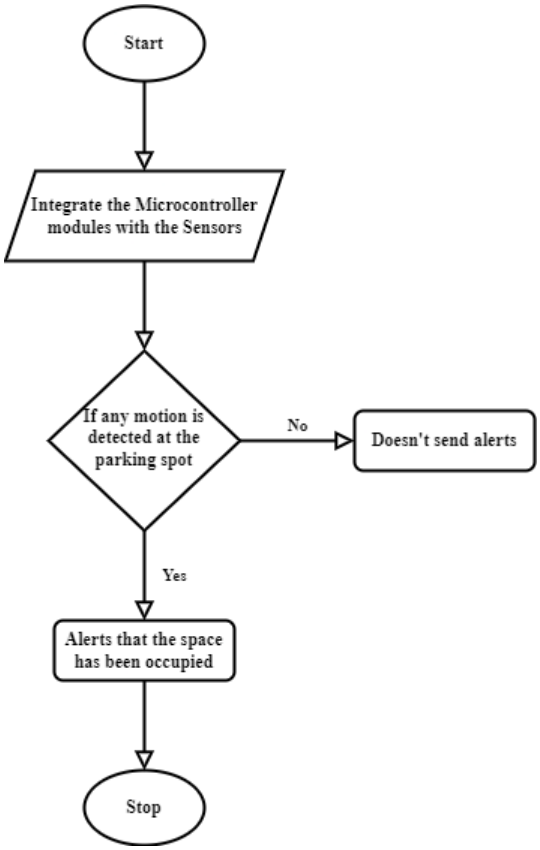
**User Interface and Mobile Application/Web Dashboard:**

- Cloud/Server: The cloud or local server processes and stores the received data. It can provide a web-based dashboard or API for users to access the parking spot status in real-time.
- Mobile Application/Web dashboard: A dedicated mobile application/web dashboard can be developed to allow users to check the availability of parking spots on their smartphones. The app can display a map showing the occupied and vacant parking spots, helping users find an available spot easily.

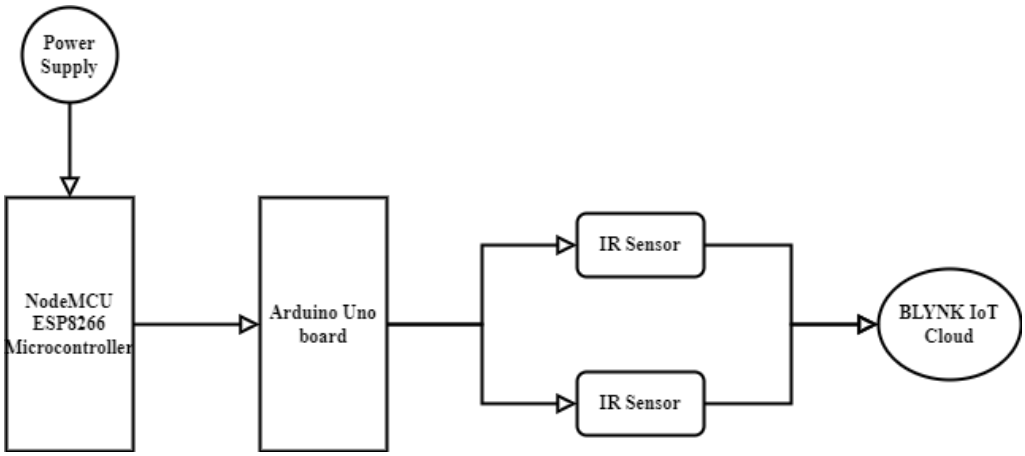
**Advantages:**

- Efficient Parking Management: The system optimizes parking space utilization, reducing the time and effort required to find an available spot.
- Real-time Information: Users can access real-time parking spot availability information through a mobile app or web interface.
- Improved Traffic Flow: With better parking management, traffic congestion caused by drivers searching for parking spaces can be minimized.
- Data Analysis: The collected data can be analyzed to gain insights into parking patterns, occupancy rates, and demand forecasting, helping in urban planning and resource allocation.

## FLOWCHART:




## BLOCK DIAGRAM:



## PROGRAM WITH COMMENTS:

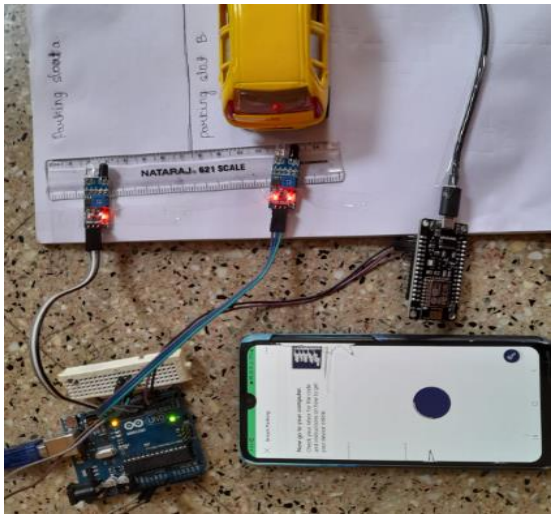
```
// Include necessary libraries
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <SoftwareSerial.h>
#include <SimpleTimer.h>

// Blynk authentication token
char auth[] = "KbiYL4u8UfxNGy3BFtfFKA3GrxYohEc2";

// WiFi credentials
char ssid[] = "Pooja";
char pass[] = "pooja9880";

SimpleTimer timer; // Timer object for scheduling events

String myString; // Complete message from Arduino, consisting of sensor data
char rdata; // Received character

int firstVal, secondVal, thirdVal; // Sensor values
int led1, led2; // LED states

// Timer event function to send Arduino's uptime to Virtual Pin (1)
void myTimerEvent()
{
  // Send uptime in seconds
  Blynk.virtualWrite(V1, millis() / 1000);
}

void setup()
{
  Serial.begin(9600); // Initialize serial communication

  Blynk.begin(auth, ssid, pass); // Connect to Blynk using authentication token and WiFi credentials

  // Set timer intervals for sensor value functions
  timer.setInterval(1000L, sensorvalue1);
  timer.setInterval(1000L, sensorvalue2);
}

void loop()
{
  // If no data available from serial communication
  if (Serial.available() == 0)
  {
    Blynk.run(); // Run Blynk
    timer.run(); // Run timer events
```

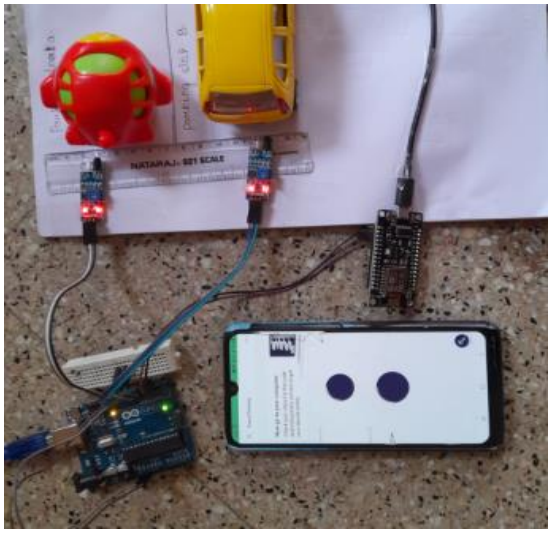
```
  }

  // If data available from serial communication
  if (Serial.available() > 0)
  {
    rdata = Serial.read(); // Read character from serial

    myString = myString + rdata; // Append character to complete message string

    // If end of message reached
    if (rdata == '\n')
    {
      // Process the received message
      String l = getValue(myString, ',', 0); // Extract first value from message
      String m = getValue(myString, ',', 1); // Extract second value from message

      led1 = l.toInt(); // Convert first value to integer and store in led1 variable
      led2 = m.toInt(); // Convert second value to integer and store in led2 variable

      myString = ""; // Clear the message string for the next message
    }
  }
}

// Function to send sensor value 1 to Blynk
void sensorvalue1()
{
  int sdata = led1;
  Blynk.virtualWrite(V9, sdata);
}

// Function to send sensor value 2 to Blynk
void sensorvalue2()
{
  int sdata = led2;
  Blynk.virtualWrite(V11, sdata);
}

// Function to extract a specific value from a string
String getValue(String data, char separator, int index)
{
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = data.length() - 1;

  // Iterate over the characters in the string
  for (int i = 0; i <= maxIndex && found <= index; i++)
  {
    // If the separator character is found or end of string reached
    if (data.charAt(i) == separator || i == maxIndex)
```

```
    {
      found++; // Increment the found counter
      strIndex[0] = strIndex[1] + 1; // Set the start index for the next value
      strIndex[1] = (i == maxIndex) ? i + 1 : i; // Set the end index for the current value
    }
  }

  // Return the extracted value as a substring
  return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
#include <SoftwareSerial.h>

SoftwareSerial nodemcu(2, 3); // SoftwareSerial object for communication with NodeMCU

int parking1_slot1_ir_s = 4; // Pin number for parking slot 1 infrared sensor
int parking1_slot2_ir_s = 5; // Pin number for parking slot 2 infrared sensor

String sensor1; // String to store sensor 1 value
String sensor2; // String to store sensor 2 value

String cdata = ""; // Complete data consisting of sensor values

void setup()
{
  Serial.begin(9600); // Initialize serial communication with Arduino
  nodemcu.begin(9600); // Initialize serial communication with NodeMCU

  pinMode(parking1_slot1_ir_s, INPUT); // Set parking slot 1 infrared sensor pin as input
  pinMode(parking1_slot2_ir_s, INPUT); // Set parking slot 2 infrared sensor pin as input
}

void loop()
{
  p1slot1(); // Check status of parking slot 1
  p1slot2(); // Check status of parking slot 2

  cdata = cdata + sensor1 + "," + sensor2 + ","; // Concatenate sensor values with comma
delimiter
  Serial.println(cdata); // Print the complete data to Arduino serial monitor
  nodemcu.println(cdata); // Send the complete data to NodeMCU

  delay(6000); // Delay for 6 seconds
  cdata = ""; // Clear the complete data string

  digitalWrite(parking1_slot1_ir_s, HIGH); // Set parking slot 1 infrared sensor pin to HIGH
  digitalWrite(parking1_slot2_ir_s, HIGH); // Set parking slot 2 infrared sensor pin to HIGH
}

void p1slot1() // Function to check status of parking slot 1
{
```

```
  if (digitalRead(parking1_slot1_ir_s) == LOW)
  {
    sensor1 = "255"; // If infrared sensor is triggered, set sensor1 value to "255"
    delay(200); // Delay to debounce the sensor
  }
  if (digitalRead(parking1_slot1_ir_s) == HIGH)
  {
    sensor1 = "0"; // If infrared sensor is not triggered, set sensor1 value to "0"
    delay(200); // Delay to debounce the sensor
  }
}

void p1slot2() // Function to check status of parking slot 2
{
  if (digitalRead(parking1_slot2_ir_s) == LOW)
  {
    sensor2 = "255"; // If infrared sensor is triggered, set sensor2 value to "255"
    delay(200); // Delay to debounce the sensor
  }
  if (digitalRead(parking1_slot2_ir_s) == HIGH)
  {
    sensor2 = "0"; // If infrared sensor is not triggered, set sensor2 value to "0"
    delay(200); // Delay to debounce the sensor
  }
}
```

## RESULTS:


## REFERENCES:

[1] Rani Astya, Soni Jain, Sukriti Sachan and Aish Aggarwal, "Smart Car Parking System", International Journal of Mechanical Engineering (IJME), India, ISSN: 0974-5823, Volume-7, Issue-6, June, 2022.
[2] Vimala Devi K, Aabha Bhatta and Turusha Ghimire, "IoT-Based Smart Parking System", International Journal of Engineering and Advanced Technology (IJEAT), India, ISSN: 2249-8958, Volume-11, Issue-5, June 2022.
[3] Essa Abdullah Hezzam, "A New Architecture: Informative Smart Parking System for Smart Cities", International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), Saudi Arabia, ISSN: 2278-3091, Volume-9, Issue-4, July–August 2020.
[4] https://github.com/guneet2919/Smart-Car-parking-IOT-Blynk-
[5] https://www.youtube.com/watch?v=6C9M12ItTr0