**ASIAN COLLEGE OF ENGINEERING AND TECHNOLOGY- COIMBATORE**

**641 110**

**TEAM MEMBERS**

**E.TAMIL NALAN(715321121053)**

**M.VAISHVARAN(715321121055)**

**R.UTHAYA(715321121054)**

**MA.AJITH KUMAR(715321121301)**

**SMART PARKING**

Today we are going to make an IOT based Smart Parking System. We will work with the entire code, connect it to the cloud as well. For User Interface, we will create a Web page and will send the data your database. Now, let us begin with the project.

Contents:

\*The traditional system of Parking

\*About the Project

\*Hardware Required

\*Arduino Code for Parking System

**The Traditional System of parking**



The old way of Parking

Poorly managed parking resources have a substantial negative impact on cities — one that has been well-documented. According to industry studies, poorly managed parking:

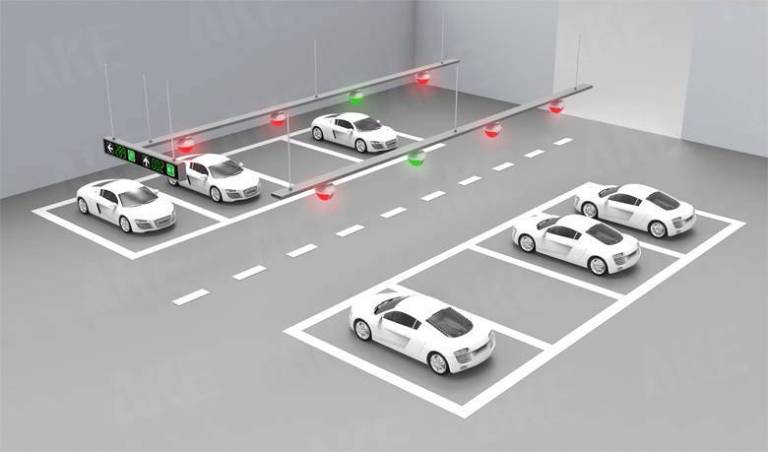
✓ Increases Traffic Congestion

✓Increases Pollution

✓Frustrates Drivers

The causes are large in number. But is there any solution for it? Definitely, yes. IOT offers a unique method of Smart parking which we have successfully implemented and that is what we are going to do today.

**About the Project**



Smart Parking System

The project is a real world’s application which can be incorporated to any parking management system. First of all, we will fix the proximity sensors as per the number of slots for parking. These sensors will tell whether any area is empty or not. This will be know to the authority in charge. It can have the report of it’s complete management on tips. On the other hand, if the user wants to avail the service at XYZ parking area. Then he will first book his slot at the parking. However, this depends on the availability as well.

Once the user books the slot with the date and time, he will be the owner of that area for the set period of time. To authenticate whether the user who is entering with the car has his pre booked slot or not, we will use RFID tags. These contain the information of the type of permit that the particular vehicle has been granted. The person enforcing it is equipped with a hand reader that captures the information of the vehicle, such as expiration time and corresponding spot location.

In this way, both the user as well as the owner will have an easy implementation without any fuss.

**Hardware Required**

\*IR sensors (3 in this model)

\*Microcontroller (2 Arduino boards)

\*Wi-Fi module (ESP8266)

\*RFID tag

\*Servo motor

\*LCD display (optional)

The hardware includes both the main system and the authentication system too. RFID, servo and an Arduino board will be required for verification at the gate. However, the rest of items are required for the main Parking management.

**Arduino Code for Parking System**

Basically, there will be two systems, one controlling the parking system and other the authentication system at the gate. Here we will discuss the prior part.

The IR sensors will be connected in different pins and will give the status of the area (1 for occupied and 0 for empty). This data will be uploaded to ‘Thing Speak‘ server. There you will see the proper visualization of the inputs given by the sensor. This will be uploaded buying the ESP8266 Wifi module. Below is the code.

**Program**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Change the HEX address

#include <Servo.h>

Servo myservo1;

Int IR1 = 2;

Int IR2 = 4;

Int SmokeDetectorPin = 6; // Digital pin for the smoke detector

Int BuzzerPin = 7; // Digital pin for the buzzer

Int Slot = 4; // Enter Total number of parking Slots

Bool flag1 = false;

Bool flag2 = false;

Unsigned long lastLcdUpdate = 0; // Variable to track the time of the last LCD update

Unsigned long lcdUpdateInterval = 1000; // Update the LCD every 1000 milliseconds (1 second)

Void setup() {

Lcd.begin(16, 2); // Initialize LCD with 16 columns and 2 rows

Lcd.backlight();

pinMode(IR1, INPUT);

pinMode(IR2, INPUT);

pinMode(SmokeDetectorPin, INPUT);

pinMode(BuzzerPin, OUTPUT);

myservo1.attach(3);

myservo1.write(100);

lcd.setCursor(0, 0);

lcd.print(“ TAMIL “);

lcd.setCursor(0, 1);

lcd.print(“ PARKING SYSTEM “);

delay(2000);

lcd.clear();

Serial.begin(9600); // Start serial communication for debugging

}

Void loop() {

If (digitalRead(IR1) == LOW && !flag1) {

If (Slot > 0) {

Flag1 = true;

If (!flag2) {

Myservo1.write(0);

Slot--;

}

} else {

displayMessage(“ SORRY ☹ “, “ Parking Full “);

}

}

If (digitalRead(IR2) == LOW && !flag2) {

Flag2 = true;

If (!flag1) {

Myservo1.write(0);

Slot++;

}

}

If (flag1 && flag2) {

Delay(1000);

Myservo1.write(100);

Serial.println(“Servo returned to initial position.”);

Flag1 = false;

Flag2 = false;

}

// Update the LCD display with a delay

If (millis() – lastLcdUpdate >= lcdUpdateInterval) {

updateLcdDisplay();

lastLcdUpdate = millis();

}

// … (Rest of your code)

}

Void updateLcdDisplay() {

If (digitalRead(SmokeDetectorPin) == HIGH) {

displayMessage(“ WARNING! “, “ Smoke Detected “);

digitalWrite(BuzzerPin, HIGH); // Turn on the buzzer

} else {

displayMessage(“ WELCOME! “, “Slot Left: “ + String(Slot));

digitalWrite(BuzzerPin, LOW); // Turn off the buzzer

}

}

Void displayMessage(const char \*line1, const String &line2) {

Lcd.clear();

Lcd.setCursor(0, 0);

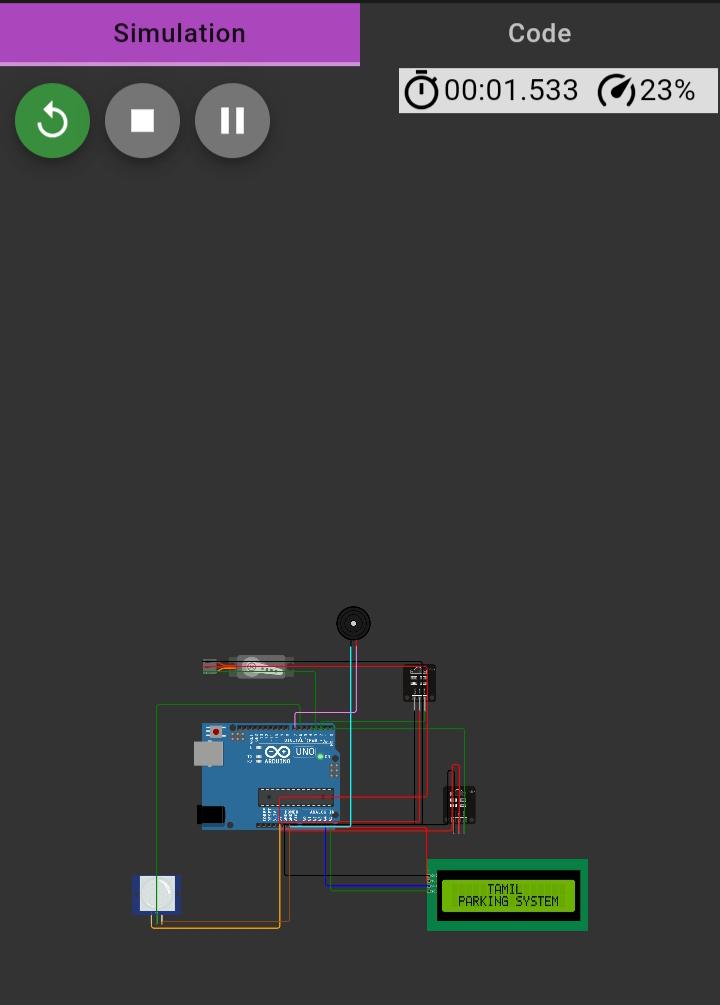
Lcd.print(line1);

Lcd.setCursor(0, 1);

Lcd.print(line2);

}

**Implementation**

****

In the code, we have just used different digital pins for different IR sensors. They will be giving the input to the controller. Subsequently, the data will be sent to the cloud and based on that we will do proper visualization of the data.