

# SMART PARKING SYSTEM PROJECT

Objective : To create a smart parking system that sends and alert when a parking lot gets occupied and also shows peak time during which parking lots are occupied

Components Required:

- 1.Node Mcu
- 2.Ultrasonic Sensor
3. Jumper Wires

Concept :

1. Ultrasonic sensor is used to detect the presence of car in front of it using the node mcu.
2. Data from the ultrasonic sensor is sent to the node mcu.
3. Node mcu then sends this data to thingspeak , where a graph of the data sent gets plotted. This helps us know peak time at which cars are parked .
4. Whenever a car is detected in front of the sensor , a condition is triggered that sends a message saying 'Parking lots are occupied !'.

Here instead of message , we'll send a email to the user just to display different functionalities of iot.

Procedure :

1. Connect the ultrasonic sensor the node mcu in the following configuration :

Ultrasonic

Node Mcu

Vcc	-	Vin
Gnd	-	Gnd
Echo	-	D0
Trig	-	D1

2. Now to upload data onto thingspeak from your sensor , create an account on thingspeak and create a channel where you will upload data as shown.

The screenshot shows the 'Ultrasonic Sensor' channel page on ThingSpeak. The channel ID is 642236, created by 'nachihebbbar', and is set to private access. The 'Channel Settings' tab is active, showing a progress bar at 30% completion. The settings form includes fields for Name (Ultrasonic Sensor), Description, and four data fields (Field 1 to Field 4). Fields 1 and 2 are enabled with checkboxes and have labels 'Field Label 1' and 'Field Label 2' respectively. Fields 3 and 4 are disabled. A 'Help' section on the right explains channel data storage and provides instructions for various settings like Channel Name, Description, Field#, Metadata, Tags, Link to External Site, and Show Channel Location. A note at the bottom of the help section mentions that latitude should be specified in decimal degrees.

ThingSpeak™ Channels Apps Community Support Commercial Use How to Buy Account Sign Out

### Ultrasonic Sensor

Channel ID: 642236  
Author: nachihebbbar  
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

#### Channel Settings

Percentage complete 30%

Channel ID 642236

Name Ultrasonic Sensor

Description

Field 1 Field Label 1 ☒

Field 2 Field Label 2 ☒

Field 3 ☐

Field 4 ☐

#### Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

#### Channel Settings

- Channel Name:** Enter a unique name for the ThingSpeak channel.
- Description:** Enter a description of the ThingSpeak channel.
- Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- Tags:** Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- Show Channel Location:**
  - Latitude:** Specify the latitude position in decimal degrees. For example, the

You can create as many fields as you want , but we require only 1 as only one graph will be plotted.

3. Save the channel and note down the channel id and write api key given.

5. After completing the above steps, upload the code onto the node mcu using Arduino IDE.

## Code :

thingspeak | Arduino 1.8.7

File Edit Sketch Tools Help



thingspeak \$

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>

//----- Fill in your credentials -----
char ssid[] = "wifi_name";           // your network SSID (name)
char pass[] = "1234567890";          // your network password
unsigned long myChannelNumber = 642236; // Replace the 0 with your channel number
const char * myWriteAPIKey = "PBW1CN17C2C25QM4"; // Paste your ThingSpeak Write API Key between the quotes
//-----

WiFiClient client;

int echo=D0;
int trigger=D1;

void setup() {
    //Initialize serial and wait for port to open:

    Serial.begin(115200);
    while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
    }

    WiFi.mode(WIFI_STA);
    ThingSpeak.begin(client);
    pinMode(trig, OUTPUT);
    pinMode(echo, INPUT);
}

void loop() {

    // Connect or reconnect to WiFi
    if(WiFi.status() != WL_CONNECTED){
        Serial.print("Attempting to connect to SSID: ");
        Serial.print(ssid);
    }
}
```



```

thingspeak $
void loop() {

    // Connect or reconnect to WiFi
    if(WiFi.status() != WL_CONNECTED){
        Serial.print("Attempting to connect to SSID: ");
        Serial.println(ssid);
        while(WiFi.status() != WL_CONNECTED){
            WiFi.begin(ssid, pass);
            Serial.print(".");
            delay(5000);
        }
        Serial.println("\nConnected.");
    }

    digitalWrite(trigger,HIGH);
    delay(2000);
    digitalWrite(trig,LOW);
    int duration=pulseIn(echo,HIGH);
    int meters=(duration)/85.84

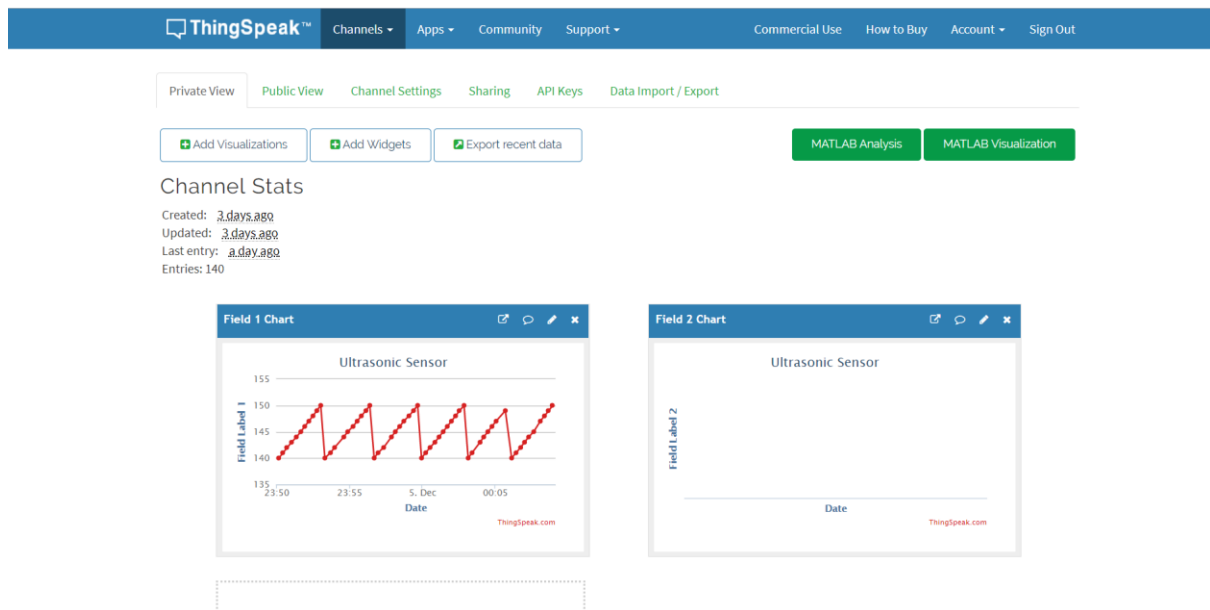
    // Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to store up to 8 different
    // pieces of information in a channel. Here, we write to field 1.
    int x = ThingSpeak.writeField(myChannelNumber, 1,meters, myWriteAPIKey);

    // Check the return code
    if(x == 200){
        Serial.println("Channel update successful.");
    }
    else{
        Serial.println("Problem updating channel. HTTP error code " + String(x));
    }

    delay(10000); // Wait 20 seconds before sending a new value
}

```

6. After uploading the code onto your node mcu you can see check your channel feed to get a graph plotted of the ultrasonic sensor's data.



From the graph you can observe peak times at which a car is parked

6. Now to set a trigger when a particular condition is met, do the following steps on thingspeak.

i. Go to Apps->React

ii. We have to create a react such that every time our sensor's data falls below a certain level (say 145), a matlab code gets executed that will send us an email.

iii. Create the react as shown

[Channels](#)
[Apps](#)
[Community](#)
[Support](#)
[Commercial Use](#)
[How to Buy](#)
[Account](#)
[Sign Out](#)

React Name

React 2

Condition Type

Numeric

Test Frequency

On Data Insertion

Condition

If channel

Ultrasonic Sensor (642236)

field

1 (Field Label 1)

is less than

145

Action

MATLAB Analysis

Code to execute

email for car parking

Options

☐ Run action only the first time the condition is met
 ☒ Run action each time condition is met

Save React

React Settings

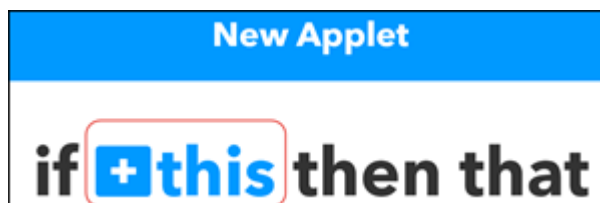
- React Name:** Enter a unique name for your React.
- Condition Type:** Select a condition type corresponding with your data. A channel can hold numeric sensor data, text, strings, status updates, or geographic location information.
- Test Frequency:** Choose whether to test your condition every time data enters the channel or on a periodic basis.
- Condition:** Select a channel, a field and the condition for your React.
- Action:** Select ThingTweet, ThingHTTP, or MATLAB Analysis to run when the condition is met.
- Options:** Select when the React runs.

[Learn More](#)

7. Now all that is left is to create an account on IFTTT, and create service that helps us send us email, whenever our matlab code calls it .

### Creating an IFTTT service :

1. Create an [IFTTT](#) account, or log into your existing account.
2. Create an Applet. Select **My Applets**, and then click the **New Applet** button.
3. Select input action. Click the word **this**.



4. Select the Webhooks service. Enter Webhooks in the search field. Select the **Webhooks** card.
5. Complete the trigger fields. After you select Webhooks as the trigger, click the **Receive a web request** box to continue. Enter

an event name. This example uses PlantData as the event name. Click **Create Trigger**.

## Complete trigger fields

Step 2 of 6

### Receive a web request

This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Event Name

PlantData

The name of the event, like "button\_pressed" or "front\_door\_opened"

Create trigger


Now the trigger word **this** is the Webhooks icon.


6. Select the resulting action. Click the word **that**. Enter email in the search bar, and select the **Email** box.

## Choose action service

Step 3 of 6

Q email

  
Email

  
Email Digest

7. Now enter the information that you want to be sent to you whenever parking lot gets occupied in the given fields.



## Complete action fields

Step 5 of 6

### Send me an email

This Action will send you an HTML based email. Images and links are supported.

**Subject**

EventName from Maker Webhooks service

Add ingredient

**Body**

What: EventName <br>  
When: OccurredAt <br>  
Extra Data: Value1

Add ingredient

Create action

8. Retrieve your Webhooks trigger information. Select **My Applets > Services**, and search for Webhooks. Select **Webhooks** and then the **Documentation** button. You see your key and the format for sending a request. Enter the event name. The event name for this example is Car\_Parking
9. `https://maker.ifttt.com/trigger/{event}/with/key/XXXXXXXXXXXXX  
XXXXXXXXXXXXX`

Now your email service is setup and all you have to do is write the matlab code that connects to IFTTT and calls this service.

10. Go to Apps->Matlab Analysis and write the following code with your credentials.



Apps / MATLAB Analysis / email for car parking / Edit

#### Name

email for car parking

#### MATLAB Code

```
1 channelID=642236;  
2 iftttURL="https://maker.ifttt.com/trigger/car_parked/with/key/c5JgAadbc9YM13xvOnXv57";  
3  
4 message_sent="Parking lot is occupied";  
5  
6 webwrite(iftttURL,'value1',message_sent);
```

Save and Run

Save

11. Save and Run ! Everytime the parking slots get occupied you get a notification , using which you can perform various actions like incrementing a number on a led display, etc . Also from the graph you can understand what is the peak time for car parking and park accordingly .

