|  |  |  |
| --- | --- | --- |
| **smart car parking System** | | |

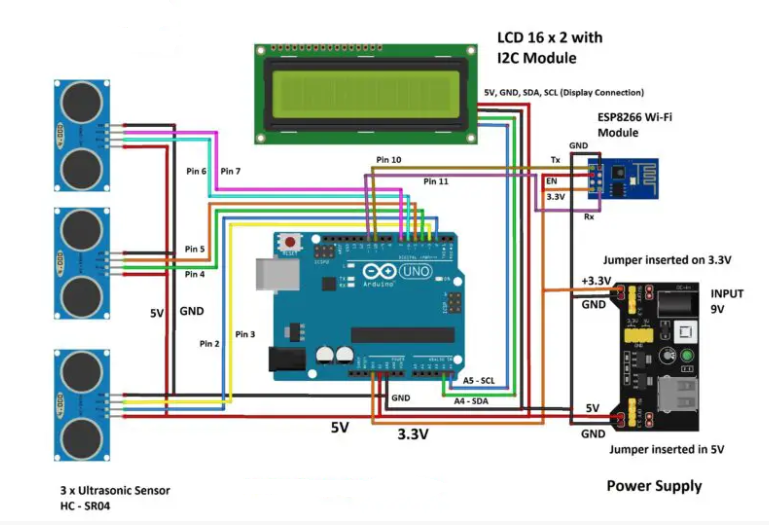
**PROBLEM DEFINITION :**

The project involves integrating IoT sensors into public transportation vehicles to monitor ridership, track locations, and predict arrival times. The goal is to provide real-time transit information to the public through a public platform, enhancing the efficiency and quality of public transportation services. This project includes defining objectives, designing the IoT sensor system, developing the real-time transit information platform, and integrating them using IoT technology and Python.

**Circuit diagram for IoT based car park monitoring system:**

The proposed smart parking lot circuit will be equipped with several sensors, inexpensive microcontrollers and Wi-Fi module using which a car / any vehicle owner can check if there is a vacant space in a parking lot using his / her phone or tablet or even on computer.

The number of vacant spaces in the smart parking lot can be viewed from anywhere in the world using a URL link or the user can scan a QR code. The scanned / shared URL can be browsed on any web browser to know how many empty parking spot exist in real time.

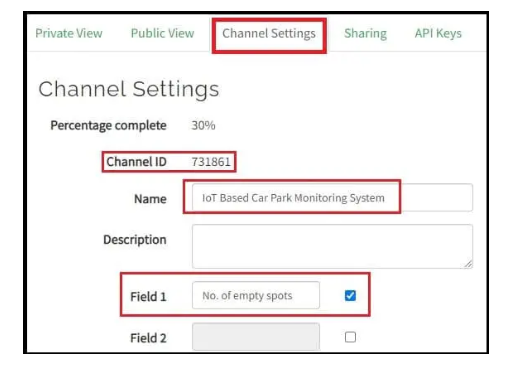


The internet cloud service we are going to use is called “Thingspeak” where the parking lot’s data to be sent, stored and displayed in real time. This concludes the block diagram.

**How to setup your Thingspeak account?**

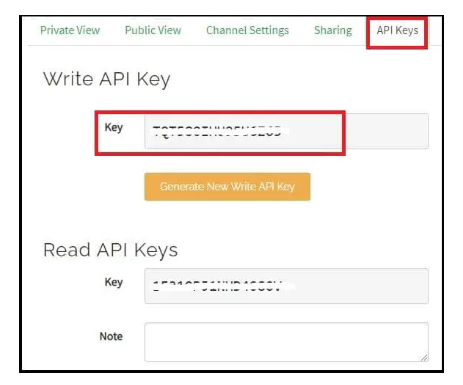
We are using a (free) cloud service called Thingspeak where we will send parking lot’s data to share it with public.

* First you need to sign up for Thingspeak:
* Enter the credentials it asks for and create a new channel and do the following to your new channel:

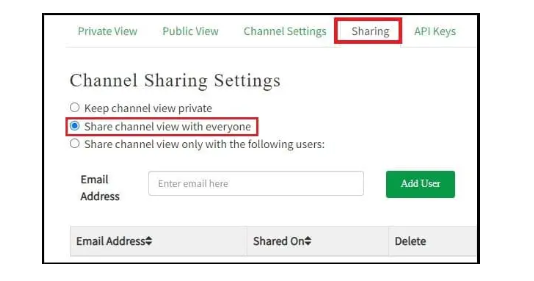


* Go to **channel settings** and enter the things as shown above and take note of your channel ID which we need to enter it in the program code.
* Scroll down and **click save** to save the changes.
* Now click on **API keys tab** and you will see your keys as illustrated below. API keys are responsible for writing and reading the data to your Thingspeak account.
* Go to **channel settings** and enter the things as shown above and take note of your channel ID which we need to enter it in the program code.
* Scroll down and **click save** to save the changes.

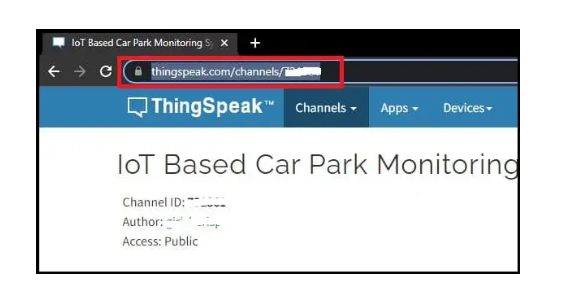
Now click on **API keys tab** and you will see your keys as illustrated below. API keys are responsible for writing and reading the data to your Thingspeak account.



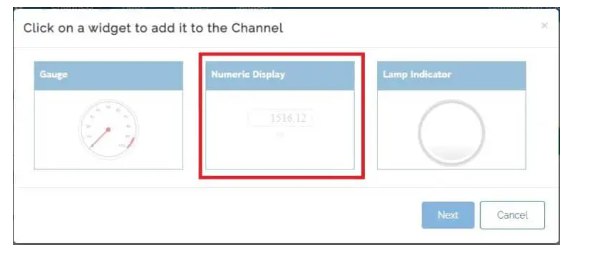
* Take note of your “write API key” which needs to be entered in the program code and read API key is not used in this project.
* Now go to sharing tab and click on “share channel view with everyone”, this makes your channel visible to those who have the URL of “public view” page.

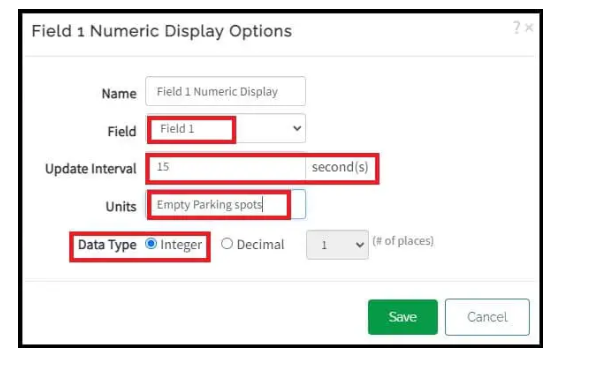


* Click on **public view tab** and you will see an empty graph field and the **URL of this page can be shared to the public**.

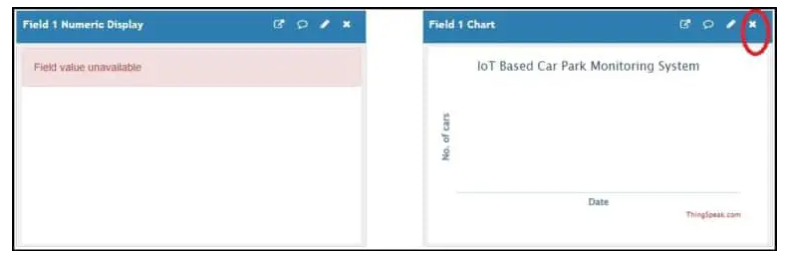


* Now on the public view tab we are going to setup a number widget where public can view the number of vacant spots on the parking lot, click on “add widget”.



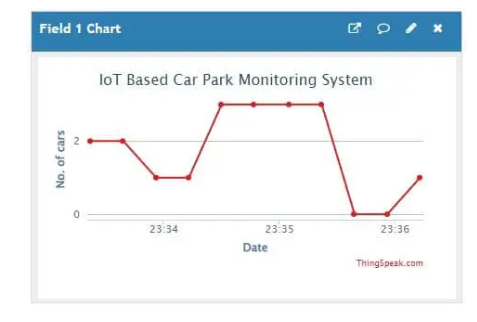


* Now you will see a new widget where number will be displayed once we send data. Close the “field chart” by clicking on ‘X’, the public just need to know the number of vacant parking spots and not the parking history.

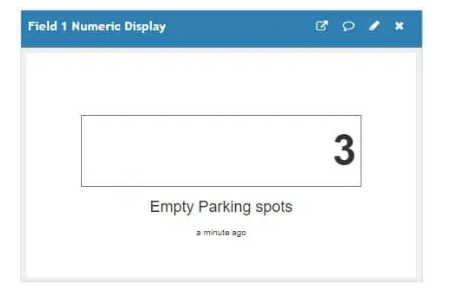


* Click on **private view tab**, you see an empty field chart; this is where you can view the history car parking history and this is not visible to public. This concludes on Thingspeak account setup.

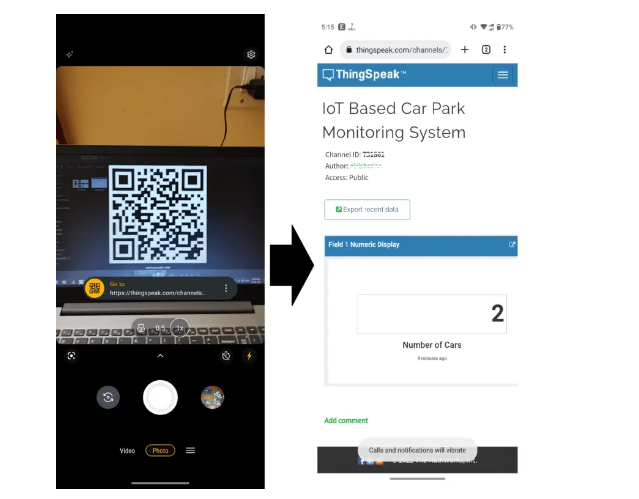
Thingspeak Private View:



**Thingspeak public view:**



**Thingspeak public view on smart phone:**



The URL of the public view page is converted in to a QR and this can be done using any online URL to QR code converter tool. This QR code can be placed at the parking spot or anywhere else, so that the users can scan and bookmark the URL and when they are visiting the parking spot, users can open the link and see how many vacant spaces exist and if this parking lot is full they can move to another.