# Smart parking

## Project objectives :

To Begin building our project by deploying IoT devices and then Developing a Python script on the IoT devices as per our project requirement.

What components is am going to choose do my project?

* Raspberry Pi
* Arduino Uno
* Jumper
* Ultra Sonic sensor -3
* USB cable
* ESP8266 Wi-Fi module

Raspberry Pi:

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.

It is used as hardware for various devices like iot which act as minicomputer.

For building the model for my project i use raspberry Pi as hardware device and this helps to give better support for my project as hardware.

Ultra Sonic sensor:

For better and efficient data collection about parking slot we use ultrasonic sensor in our project.

IR sensor can be used but it do not works as efficient in sun light as ultra Sonic sensor.

Ultra Sonic sensor gives as best result about empty spaces to park .so we use this sensor to identify the space for parking.

For small implementation of our project we use only 3 ultra Sonic sensor.

Arduino Uno:

Arduino Uno is a microcontroller which is act as a brain of the project. That having pins that are used to connect devices to it.

For our project we bonds the raspberry Pi to arduino Uno by using bread board and connect with careful things like pin numbers,ground, VCC etc.

Following are the steps for a smart parking system using Raspberry Pi and an ultrasonic sensor:

1. Connect the VCC pin of the ultrasonic sensor to the 5V pin of the Raspberry Pi.

2. Connect the GND pin of the ultrasonic sensor to the GND pin of the Raspberry Pi.

3. Connect the TRIG pin of the ultrasonic sensor to GPIO pin 17 (BCM) of the Raspberry Pi.

4. Connect the ECHO pin of the ultrasonic sensor to GPIO pin 27 (BCM) of the Raspberry Pi.

Above steps can help as to connect the ultra Sonic sensor and raspberry Pi.

ESP8266 Wi-Fi module:

The ESP8266 Wi-Fi module is used to send the data that collected by ultra Sonic sensor.

In this process we first collect the data from ultra Sonic sensor and this to mobile using the remote server ie. Cloud which can helps to get the data from the sensors.

For sending the data we use wifi connection to give the data to user via cloud server .

We must give the name of the channel of cloud server to send the information about empty spaces.

The code can be in any language like C, Python, Java, PHP etc

For my project i choose the code below :

#include “ThingSpeak.h”

#include <ESP8266WiFi.h>

Char ssid[] = “SSID”; //SSID here

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(115200);

WiFi.mode(WIFI\_STA);

ThingSpeak.begin(client);

Internet();

}

Void loop()

{

Internet();

If (Serial.available() > 0)

{

Delay(100);

While (Serial.available() > 0)

{

Value = Serial.readString();

If (value[0] == ‘\*’)

{

If (value[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 = “”;

}

Cloud server:

For sending the parking place status data to mobile app we use cloud server.

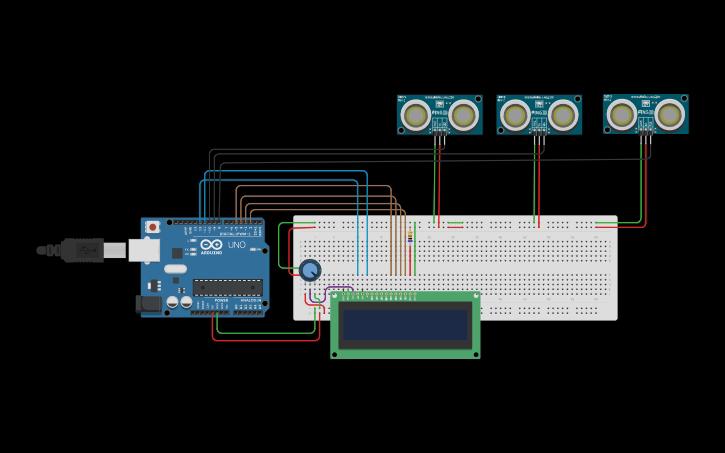
For our project we choose Thinkspeak cloud to send the information which is free for use.

For using Thinkspeak we must first have a account in the server to have the cloud server for use.

We can setup our channel in cloud as public to update the details about the parking spaces .

Now people can use the data about parking slot in the public tab using Thinkspeak cloud server.

Circuit diagram:



Code:

Python

# Import necessary libraries

Import time

Import RPi.GPIO as GPIO

Import requests

# Set GPIO mode and pin numbers

GPIO.setmode(GPIO.BCM)

Sensor\_pin = 18

# Set API endpoint for sending data

Api\_endpoint = <https://your-api-endpoint.com/parkingsensor>

# Initialize GPIO pin as input

GPIO.setup(sensor\_pin, GPIO.IN)

# Function to send sensor data to the API

Def send\_data(status):

Data = {

“status”: status,

“timestamp”: int(time.time())

}

Response = requests.post(api\_endpoint, json=data)

If response.status\_code == 200:

Print(“Data sent successfully!”)

Else:

Print(“Failed to send data.”)

# Main loop

While True:

Try:

# Read sensor status (0 or 1)

Status = GPIO.input(sensor\_pin)

# If parking spot is occupied, send status as 1

If status == 1:

Send\_data(1)

# If parking spot is vacant, send status as 0

Else:

Send\_data(0)

# Wait for 1 second before reading again

Time.sleep(1)

# Exit the program on keyboard interrupt (Ctrl+C)

Except KeyboardInterrupt:

Print(“Program terminated.”)

GPIO.cleanup()