# Smart Parking Solution using Internet of Things, Cloud Services and a Mobile Application

Shyam Ravishankar
B. E. Electronics and Communication Engineering
Sri Venkateswara College of Engineering
Sriperumbudur, Tamil Nadu, India

**Abstract-The total number of cars on the road is increasing exponentially. The lack of organised parking zones compounds the problem of fuel wastage and traffic congestion. There have been papers on using Internet of Things to build a Smart Car Parking Solution but most of them work only in certain cases and are not scalable. Several papers suggest a reservation model but this may not always be practically implementable. This paper proposes a solution which is robust, scalable and economically feasible to indicate number of free parking bays in any parking area. This is done using an Infrared in every bay which is connected to a Raspberry Pi. The raspberry Pi transfers all the data to a cloud server, which is accessible to users using a mobile application.**

***Keywords-Internet of Things, Raspberry Pi, Cloud Services, Mobile Application***

## I. INTRODUCTION

There were over two crore vehicles sold in India during the financial year 2014 to 2015, representing a close to 9% increase from the previous year's sales. All these extra cars on the road are going to need additional parking space. In an office building with a multi storey car park, one must drive all over the lower floors, searching for empty bay to reach the higher ones. These lead to a massive waste of both fuel and time.

There are other smart parking solutions on the market, such as, [1]. Existing solutions use Short Messaging Services, USSD which is dependent on signal availability. Inside several multi level car parks, mobile network is not available. [2] uses a reservation system, which can work as a double edged sword. While it can be utilised to enhance productivity, it also gives people the option of reserving slots which are needed hours later.

The solution proposed in this paper is one in which the architecture of the cloud server is in such a way that, any number of new parking areas can be added at any time without any change to the code. The mobile application developed is a Universal app which can run on Windows, Android and iOS. The same device code can be used for multiple Raspberry Pi without any modifications to it. Hence it is evident that the solution proposed is robust, scalable and economically feasible.

## II. ARCHITECTURE

The architecture of the entire solution is elucidated in the block diagram below. This consists of the following parts

### *A. Sensors*

The Ultrasonic sensor has a range of 0.02 to 0.4 metres. The sensor used was the HCSR04. It was powered up by the Raspberry Pi. It requires 5 Volts and less than 2mA of current. The output given by the Echo pin of the Ultrasonic sensor is a 5V output.

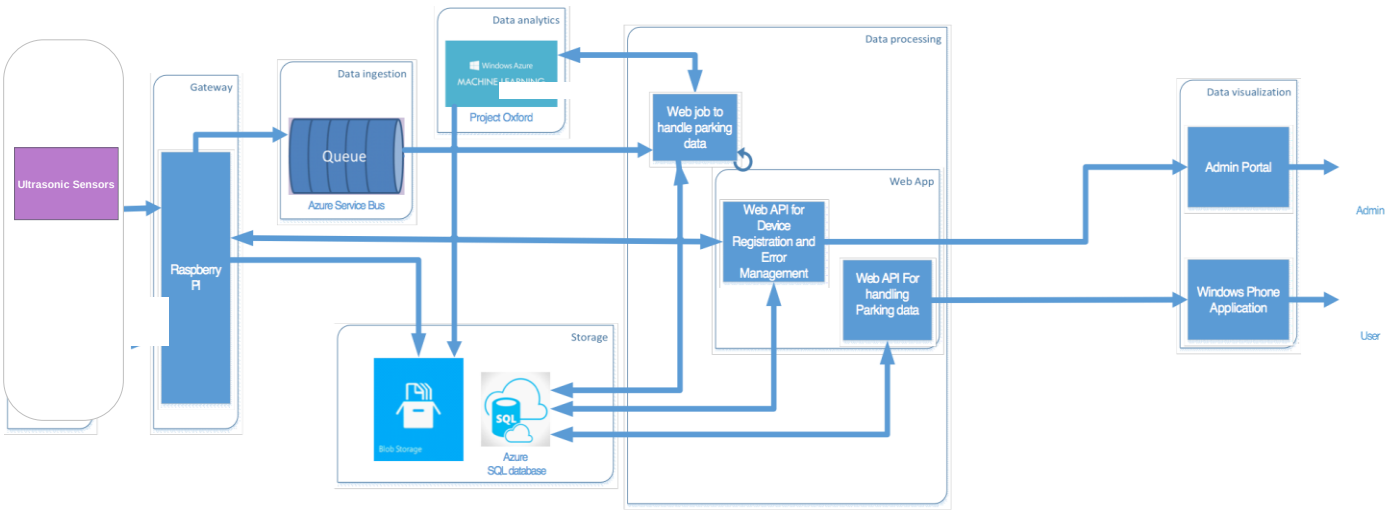
### *B. Raspberry Pi*

The Raspberry Pi is connected to a power source, an ethernet cable and to 12 sensors. The sensors give an output of 5 Volts but the GPIO pins of the Raspberry Pi can only take an input of 3.3 Volts. Thus, a voltage divider network is used to connect


the sensors to the GPIO pins of the Raspberry Pi. The device needs to be registered to be recognised as a part of the network of devices. This is done using a WebAPI. It takes the sensor number, the pins for trig and echo and slot and level of the placed sensor. It checks if there is a car or not and calls the service bus function to update the details. This is done by checking the distance of the object in the bay from the sensor to determine whether the object is a car or not.

### *C. Data Storage*

The data from the Raspberry Pi is sent to a Queue on an Azure Service Bus. This is a temporary queue for data ingestion. The data is sent for storage to a Blob on Azure. The Blob holds unprocessed, raw data. The Azure SQL Database stores the processed data that comes in from the Web jobs and Web APIs.




*D. Data Processing*

The Web job takes the Queued data from the Azure Service Bus and sends it to the SQL Server. There are two Web APIs used in this application. There is one for Device Registration and Error Management which synchronises with the Raspberry Pi to register the Pi and the sensors in the correct level or zone of the Parking Area. The other Web API is for handling parking data and it acts as a medium between the SQL Server and the Mobile Application.

*E. Admin Portal*

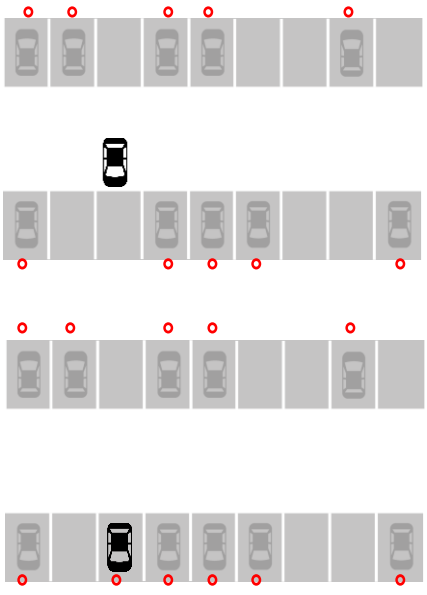
The admin portal is where the company or the parking area owner has access to all the data. This portal shows faulty sensors or hardware along with data analytics which can help the owners plan the parking zones better.

*F. Mobile Application*

The users access this solution through a mobile application developed using Apache Cordova Tools on Microsoft Visual Studio 2015. Angular javascript was used to develop a portable application which can be ported across Android, Windows and iOS platforms. The application also features a Text to Speech module which reads out the number of empty parking bays available in the user's preselected preferred parking level or zone. This was done to ensure that the users do not have to look at their phones while driving. The application also features an error reporting facility which enables users to indicate incorrect data which may have been displayed due to faulty sensors or disconnected cables.

*G. Data Analytics*

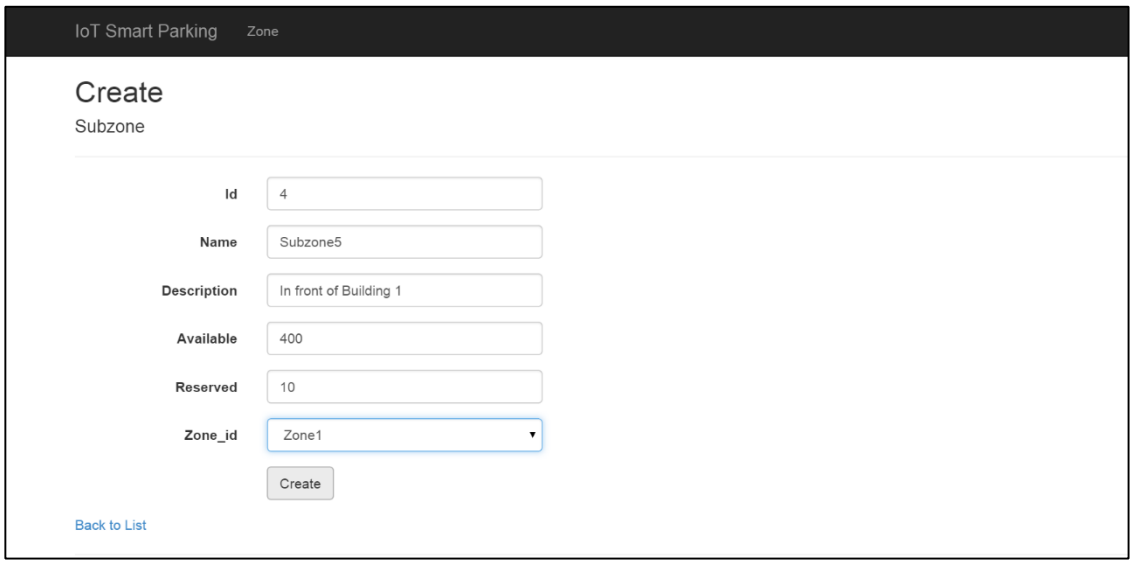
Application insights are used to provide the administrators an idea of preferred zones and levels. This is also useful to identify the pattern in which zones and levels get filled up with respect to time. This may help with renovation plans and to organize the necessary man power needed in frequently used areas.

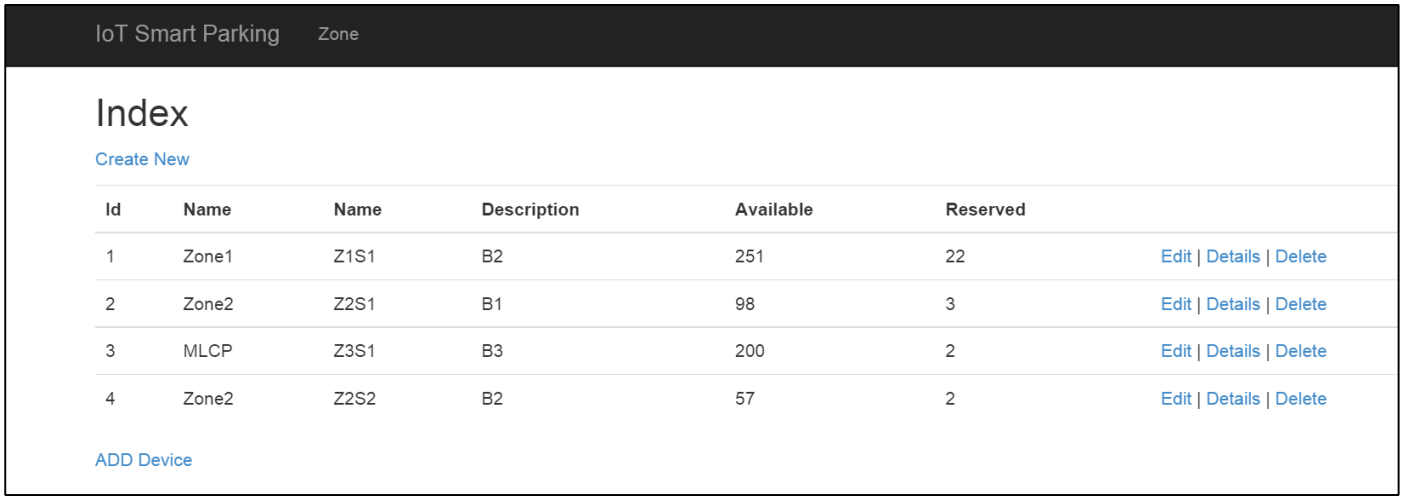


The car is not parked. The sensor is not activated

The Car is parked.The sensor activated.

Admin Portal




## ACKNOWLEDGMENT

The author would like to thank Dr. S. Ganesh Vaidyanathan, Principal, Sri Venkateswara College of Engineering and Dr. S. Muthukumar, Head of the Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering for their support and encouragement.

## REFERENCES

1. Thanh Nam Pham, Ming-Fong Tsai, Duc Binh Nguyen , Chyi-Ren Dow1 , And Der-Jiunn Deng, "A Cloud-Based Smart-arking System Based on Internet-of-Things Technologies," Publication, IEEE Access, September 2015
2. V. Hans, P. S. Sethi,J. Kinra, "An Approach to IoT Based car parking and Reservation system on Cloud", Proceedings, International Conference on Green Computing and Internet of Things, IEEE, 2015.
3. N. M. Hui, L. B. Chieng ; W. Y. Ting ; H. H. Mohamed ; M. R. Hj Mohd Arshad, "Cross-platform mobile applications for android and iOS", Wireless and Mobile Networking Conference (WMNC), IEEE, 2013

Device Code

```
import RPi.GPIO as GPIO
import time
from lxml import etree
from lxml import builder
import requests
import os
import threading
import datetime
import urllib2
import urllib
import json
from azure.servicebus import ServiceBusService, Message, Queue
GPIO.setmode(GPIO.BCM)
#Sends data to service bus using shared access key name and value, #this keeps running
till it successfully sends the data to service #bus
def sendToBus(slot, level, isCar):
      while True:
            try:
                  bus_service = ServiceBusService(
                  service_namespace = 'iotsmart',
                  shared_access_key_name = 'RootManageSharedAccessKey',
                  shared_access_key_value =
'g5WH+mVRlTxS1f5sg9LOmS9OLCRA4tNRn6Cps/o98Ho=')
                  query_args = b"{'slot':'"+slot+"', 'level' : '"+level+"' , 'isCar' :
'"+isCar+"'}"
                  msg = Message(query_args)
                  bus_service.send_queue_message('testqueue', msg)
                  print "w"
                  break
            except:
                  print "Service Bus Error"

#It takes sensor no, the pins for trig and echo and slot and level #of the placed
sensor
#It checks if there is car or not and calls the service bus #funtion to update the
details
def checkCar(sensor, trig, echo, slot, level):
      print sensor
      print " Distance Measurement In Progress"
      GPIO.setup(trig,GPIO.OUT)
        GPIO.output(trig, False)
      GPIO.output(trig, True)
      time.sleep(0.00001)
      GPIO.setup(trig,GPIO.OUT)
      GPIO.output(trig, False)
      GPIO.setup(echo,GPIO.IN)
      while GPIO.input(echo)==0:
        pulse_start = time.time()
      while GPIO.input(echo)==1:
        pulse_end = time.time()
      pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
      distance = round(distance, 2)
      print "Distance:",distance,"cm"
      if(distance <200):
            isCar = '1'
      else:
            isCar = '0'
      sendToBus(slot, level, isCar)

#The program starts over here, Here we call the check Car function #for all the placed
sensor
while True:
      #18 is trig and echo pin, 1 is the sensor no, and level = 3       #and slot = 2,
the slot and level have to passed as string
```