# INTELLIGENT PARKING MANAGEMENT SYSTEM

*by*

**R. RAJ MADHAN 2013103021**
**P. ASWIN 2013103535**
**R. BHARADWAJ 2013103536**

*A project report submitted to the*

**FACULTY OF INFORMATION SCIENCE AND COMMUNICATION ENGINEERING**

*in partial fulfillment for the completion of assignment of*

**CREATIVE AND INNOVATIVE PROJECT LAB**


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ANNA UNIVERSITY, CHENNAI – 25**

**MAY 2016**

# ABSTRACT

The Intelligent Parking Management System is a system that is highly automated and reliable without the necessity of large man-power to control parking. We look forward to acquiring a secured parking spot in a very short time before heading out towards our destination in order to reduce the hassle of driving around looking for a parking spot by implementing a sensor-based automated system and displaying the parking status information to help in efficient parking.

Essentially, this project is aimed at setting up an arrangement that helps the user to know about the parking space status in a parking layout. Instead of using manpower to control the parking, an automated sensor-based system is suggested to make the experience more efficient and user friendly. The basic framework includes an input device capable of recognizing and processing the sensor input.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **IPMS** | Intelligent Parking Management System |
| **LCD** | Liquid Crystal Display |
| **RPI** | Raspberry Pi |
| **IR** | Infrared |
| **OS** | Operating System |

# CHAPTER 1

# INTRODUCTION

## 1.1 GENERAL OVERVIEW

A parking space is a location that is designated for parking, either paved or unpaved. It is usually a space delineated by road surface markings. The automobile fits inside the space, either by parallel parking, perpendicular parking or angled parking. Depending on the location of the parking space, there can be regulations regarding the time allowed to park and a fee paid to use the parking space. When the demand for spaces exceeds supply, vehicles may park onto the sidewalk, grass verges and other places which were not designed for the purpose. In a Multi-storeyed parking, the driver needs to know about the availability of parking and has to be guided to the particular parking space.

The focus is mainly on reducing the manpower required to monitor the parking in. Normally we can see in the multiplexes, theatres, marriage halls, etc, a considerable amount of manpower is required to coordinate the parking process. In cities the majority of parking spaces is controlled by manpower. Surprisingly hardly any data on usage of these spaces is available. This is changing rapidly. Parking data from sensors are considered a prominent part of that. With information from parking sensors, it is easy to guide visitors quickly to available parking spaces.

## 1.2 ABOUT THIS PROJECT

The Intelligent Parking Management System (IPMS) addresses these problems. IPMS is an Automated Sensor-based system which gives details regarding availability of parking lots for the given parking layout. Sensor networks are a natural candidate for car park management systems, because they allow status to be monitored very accurately for each parking space, if desired. Wireless sensor networks have the advantage that they can be deployed in existing car-parks without having to install new cabling for network and electricity to reach each sensing device.

The system setup consists of IR proximity sensors placed appropriately in the parking lots. These sensors sense the presence or absence of the vehicle. These sensors are connected to a Micro-controller Unit which transmits the information to the server via a wireless connection. The data is stored in a database and dynamically accessed to map the parking availability on to a Web Interface which is displayed on LCD screens. These LCD screens will be placed at strategic locations in the parking layout for the people to park their vehicles accordingly.

This will help reduce the manpower to a certain extent and make it easy for drivers to park their cars on their own. An increment of this system will be a Smart booking system that provides customers an easy way of reserving a parking space online. It overcomes the problem of traffic in parking.

# CHAPTER 2

# RELATED WORKS

*Carlo Blazquez, Felix Caicedo and Pablo Miranda* [2] proposed a system that allows customers to select a parking facility according to their preferences, rapidly park their vehicle without searching for a free stall, and pay their reservation in advance avoiding queues, interact with in-vehicle navigation systems and provide users with information in real time such as capacity, parking fee, and current parking utilization. These systems assist drivers with their trip planning in searching and waiting for vacant parking stalls while mitigating driver frustration, and decreasing queues at parking entrances, the amount of miles traveled per vehicle, and average trip time, traffic congestion, energy consumption, and air pollution

*Jay H. Ball* [1] suggested a system for determining and communi cation the availability of parking spaces. The system includes an Op tical adapted to Scan a plurality of the Parking spaces and to produce scan data for the parking spaces scanned. This invention relates gener ally to managing parking spaces, and more particularly, to determining and broadcasting the availability of vehicle parking spaces. Motorists often become frustrated while searching for an available parking space. The frustration increases as the searching continues, because random searching does not assure the motorist an on-street or facility parking space Within a fixed amount of time. Systems to determine and com

municate the availability of parking spaces could lead to more efficient use of existing parking facilities in crowded city centers.

*Joseph P. Quinn* [6] proposed a system for providing timely and efficient notification of vehicle parking space availability and locations of vehicle parking spaces to motorists or other network users. Local detector devices sense the presence or absence of a vehicle in a Particular parking Space and Communicate space identification and status information to a computer network. The present invention relates to a method and apparatus for communicating space availability data and more particularly to a method and apparatus for sensing the presence or absence of vehicles in particular parking spaces and communicating parking space location and status information.

*Ojas Pandya, Shiamak Mehta, Hemant Jarkad and Arti Mohanpurkar* [4] introduced a system thus takes care to relieve traffic congestion by making parking easier and efficient. Due to planned parking the entire road can be available exclusively for moving vehicles. Technology like GPS, Infra red and RFID will thus be deployed to develop an easy, quick and efficient parking system. RFID sensor detects the vehicle occupying the parking, it can log where exactly the vehicle in parked, including the floor and number of parking, for example, P1/ Lot 86. The system explained in this paper will have the ability to compute sense and interact with the physical environment in detail, leading to generation of huge amounts of data

*Yanxu Zheng, Sutharshan Rajasegarar and Christopher Leckie* [7] made a proposal which with the use of sensors and gps and previous experiences and predicting the parking congestion in future and also notifying registered users regarding the availability on and utilities, more aware, interactive and efficient. The realization of the Smart City is

now becoming possible with the emergence of the Internet of Things (IoT), which radically evolves the current Internet into a network of interconnected objects, such as sensors, parking meters, energy measuring devices and actuators. These networked devices have the ability to compute, sense and interact with their surroundings in fine spatial and temporal detail, and generate a vast amount of data

*Andreas Klappenecker, Hyunyoung Lee and Jennifer L. Welch* [3] proposed a system for predicting the number of available parking spaces in a parking lot. The parking lot is modeled by a continuous-time Markov chain. The parking lot regularly communicates the number of occupied spaces, capacity, arrival and parking rate through a vehicular network. The navigation system in the vehicle has to compute from these data the probability of an available parking space upon arrival.

*Elena Polycarpou and Lambros Lambrinos* [5] proposed a survey on the needs of drivers from parking infrastructures from a smart services perspective. As smart parking systems are becoming a necessity in todays urban areas, we discuss the latest trends in parking availability monitoring, parking reservation and dynamic pricing schemes.

The Intelligent Parking Management System is an inspiration from the ideas and propositions done by the above people. The main motive is to reduce manpower for the parking process, thus reducing cost.

# CHAPTER 3

# REQUIREMENTS ANALYSIS

## 3.1 HARDWARE REQUIREMENTS

1. Raspberry Pi Microcomputer
2. Infrared Proximity Sensor for Obstacle Sensing
3. USB Smartphone charger
4. Jumper Wires
5. Wifi Dongle
6. Ethernet Cable
7. Mouse
8. Keyboard

## 3.2 SOFTWARE REQUIREMENTS

1. Raspbian OS
2. Python IDE
3. Web browser (preferrably Google Chrome)

## 3.3 FUNCTIONAL REQUIREMENTS

### 3.3.1 Parking Sensing and Processing Module

1. The IR proximity Sensor should be connected to the RPI
2. The Python libraries and IDE should be installed in the RPI microcomputer
3. The RPI should be connected to the internet either wirlessly or through the ethernet cable.

### 3.3.2 Parking Data Manipulation and Mapping Module

1. The received parking data should be stored in a server's database.
2. The status on the web interface should be updated whenever the database is updated

## 3.4 ANALYSIS

The sensor module is built using the RPI, the reason for choosing which over others is made clear in the subsequent chapters. A python program is run by the RPI which processes the sensor input. Then using a PHP script, the data processed by the python script is sent to a server and stored in a database. Then another PHP script is used to retrieve from the database and map it onto a web interface.

# CHAPTER 4

# DETAILED DESIGN

## 4.1 TECHNOLOGY USED

### 4.1.1 Raspberry Pi

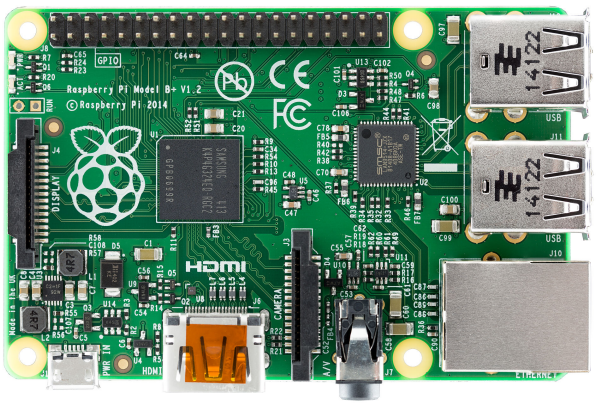

**Figure 4.1** RPI microcomputer

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 a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. Its capable of doing everything a desktop computer is expected to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

### 4.1.2 Ultrasonic Distance Sensor


**Figure 4.2** Ultrasonic Distance Sensor

Ultrasonic sensor provides an easy method of distance measurement. This sensor is perfect for any number of applications that require you to perform measurements between moving or stationary objects. Interfacing to a microcontroller is a snap. A single I/O pin is used to trigger an ultrasonic burst (well above human hearing) and then ”listen” for the echo return pulse. The sensor measures the time required for the echo return, and returns this value to the microcontroller as a variable-width pulse via the same I/O pin. It provides precise, non-contact distance measurements within a 2 cm to 3 m range. Ultrasonic measurements work in any lighting condition, making this a good choice to supplement infrared object detectors. Simple pulse in/pulse out communication requires just one I/O pin. Burst indicator LED shows measurement in progress 3-pin header makes it easy to connect to a development board, directly or with an extension cable, no soldering required.

## 4.2 SYSTEM DESIGN

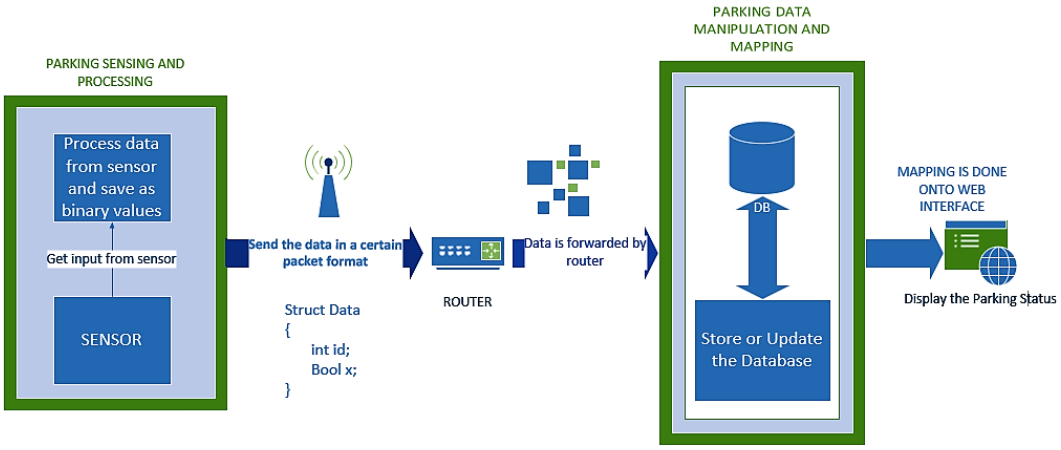


**Figure 4.3** System Architecture

The above figure 4.3 is a depiction of the architecture of the system. The blocks represent the various modules and the arrows represent the flow of data across the various modules.

The sensor is connected to the Microcontroller Unit. The MCU processes the input from the sensor with respect to the availability of the car in the parking lot. Then this data is packeted and transmitted wirelessly to the server where it stores the obtained data into the database. This wireless transmission occurs with the help of a router which forwards the packet to the server. The server retrieves the data dynamically at regular intervals and the availability of the parking lot is mapped onto a Web GUI which displays the availability status for the currently monitored parking layout.

The system has two main modules. One is the **Parking Sensing Subsystem** and the other one is the **Data Manipulating and Mapping Subsystem**.

The following contains the state transition model for the hardware, followed by the description of the two modules.

## 4.3 STATE MODEL OF THE HARDWARE

### 4.3.1 Sensor

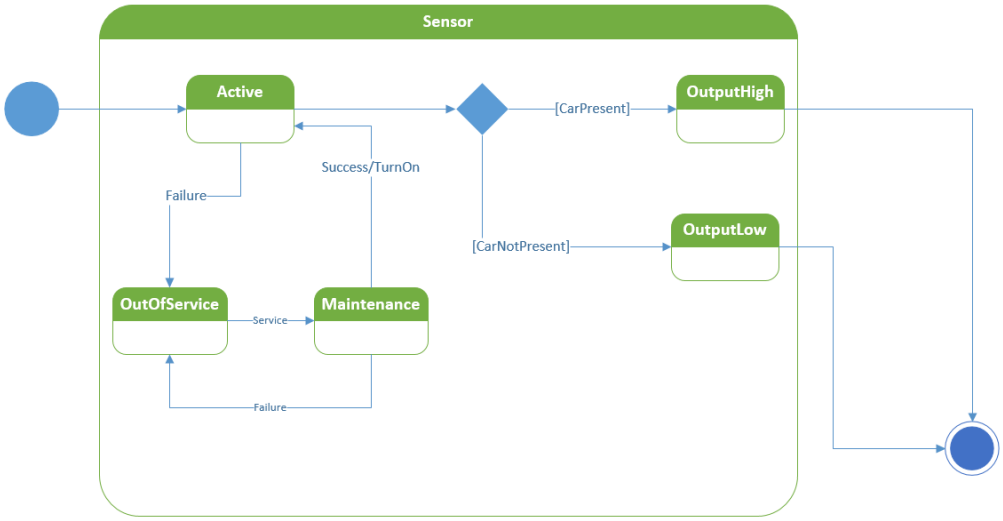


**Figure 4.4** State transition diagram for Sensor

Figure 4.4 represents the state transition diagram for the sensor. When the sensor is active its job is the sense the presence of a car in the parking lot. If the car is present, the sensor goes to ActiveHigh State, else, it goes to ActiveLow State. If the sensor goes out of order it is repaired back to active state.

### 4.3.2 Micro-Controller Unit

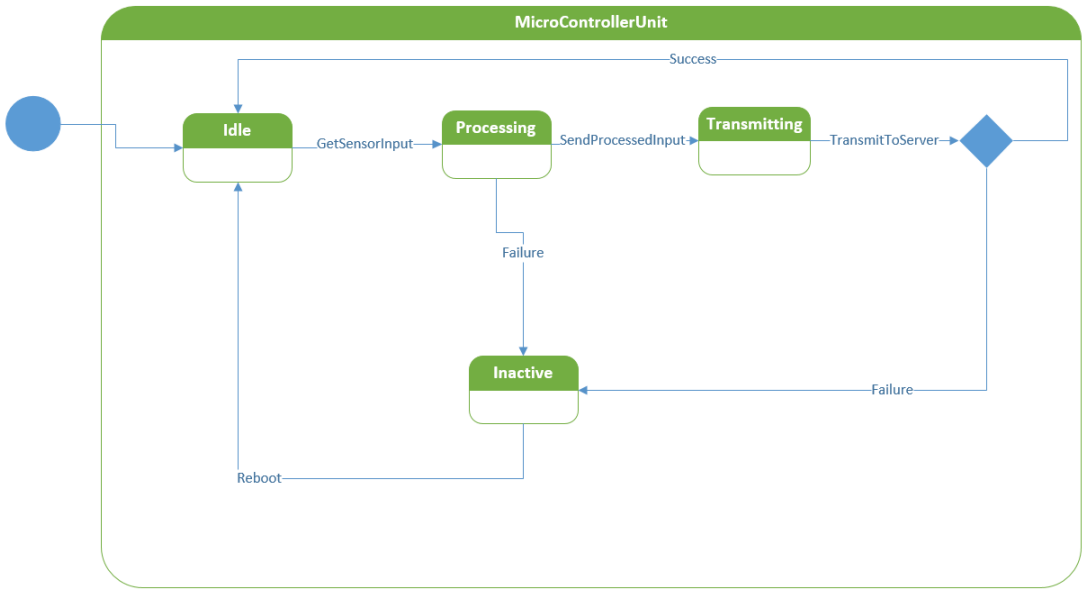


**Figure 4.5** State transition diagram for MCU

Figure 4.5 represents the state transition diagram for Micro-Controller Unit. The MCU goes from Idle state to Processing state if it receives an input from the sensor. After processing is done, it transmits the data to the server. If there is a problem in processing or transmitting, the MCU goes to inactive state. It is then rebooted to its idle state to resume processing.

## 4.4 PARKING SENSING AND PROCESSING

The role of the parking sensing module is to detect the presence of a car in the parking lot where the sensor is installed. The type of sensor used is an **Infrared (IR) Line sensor**. The sensor is connected to the Micro-Controller Unit. If there is no car present in the parking lot, then the IR sensor is in **LOW state** (Binary 0). If a car is present in the parking lot, then the sensor is in **HIGH state**(Binary 1). This data is stored in the form of binary values.

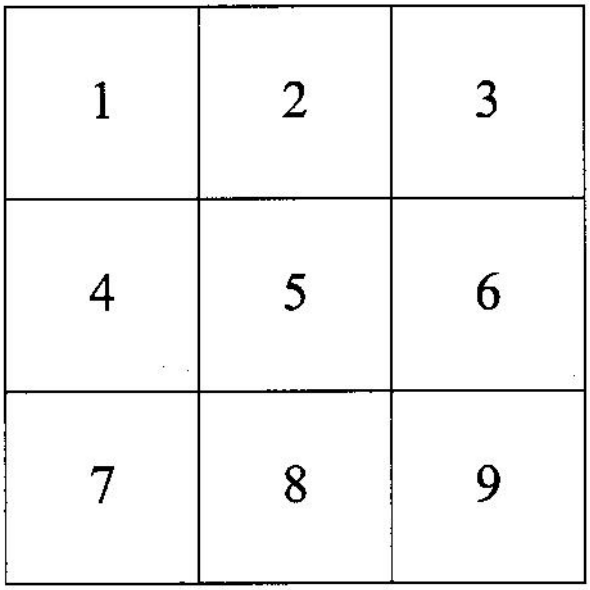


**Figure 4.6** An arbitrary example of a parking lot having id 1 to 9

Let us consider Figure [4.6] which shows a parking layout in the form of a 3x3 board (9 parking lots). This layout helps to explain the working of the system in a simple and understandable manner. The data obtained from the sensor is processed by the MCU.

The structure for receiving the data is

```
struct sensorData
{
   int id;
   bool s;
};
```

At each parking lot 2 sensors are strategically placed so that the car is in the sensor's range even if the alignment of the car is not perfect. Let these sensors be s1 and s2. Each sensor has an unique id. The following table represents the logic of the sensor I/O.

| **S1** | **S2** | **Action** |
|---|---|---|
| 0 | 0 | Car not present |
| 0 | 1 | Error Condition |
| 1 | 0 | Error Condition |
| 1 | 1 | Car is present |

**Table 4.1** Sensor I/O

Table 1 shows the sensor input logic and the correspondingly performed action. When s1=0, s2=1 or s1=1, s2=0, it means that an error condition has been encountered. The error can be due to improper parking of the car or due to obstruction by some other object.

Following is the algorithm used for handling errors due to other objects or improper parking of the vehicle.

```
while(true)
{
   value=ambientIR-objectIR;
   distance+=value;
}
If(distance< d)
   then invalid object
```

Here the **ambientIR** contains the distance value when no object is present. **objectIR** is the distance calculation if an object is present. If the difference between. If this **distance** is less than a specified value, then the object is not a car. This distance can be measured using the Ultrasonic Distance Sensor. The Ultrasonic Sensor provides an easy way for distance measurement.

Now an AND operation is performed between **SensorData** of two sensors of a particular parking lot. The resulting boolean value is stored in a separate structure.

```
struct sensorData
{
   int parking_lot_id;
   bool x;
};
```

Here **x** is the variable that stores the value after the AND operation. The structure also contains a variable that holds the parking lot id. After packeting the data, a TCP connection is established with the server. Then the data is transmitted wirelessly to the server.

## 4.5 PARKING DATA MANIPULATION AND MAPPING

This module, on receiving data from the MCU, stores it in the database. Then a web GUI is designed and the data retrieved from the database and is mapped onto the web GUI accordingly. This is web GUI is displayed in LCD screens for the user's accessibility. If the data retrieved is 0, then parking space is occupied. If it is 1, the parking space is available.

# CHAPTER 5

# EXPERIMENTAL SETUP

## 5.1 SETTING UP THE RPI MICROCOMPUTER

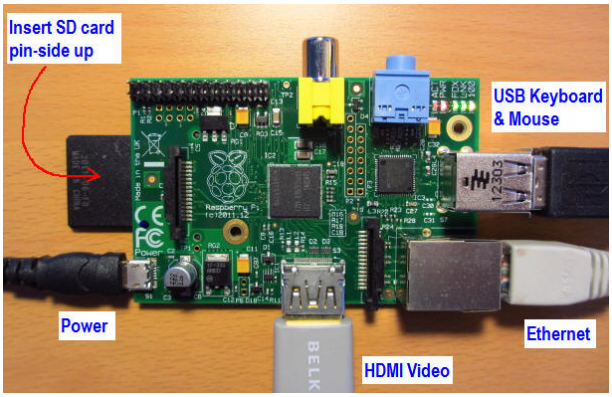


**Figure 5.1** Main Setup

Figure 5.1 shows the main setup to power up the RPI microcomputer. The SD card contains the Raspbian OS which is required to boot the RPI and also acts as the storage media for storing files. The RPI is powered up using any normal smartphone USB charger. HDMI port is used to connect to a display to run and check the python and PHP scripts. LAN cable is connected to the Ethernet port from an internet source. Alternatively, a WiFi dongle can be used for wireless internet. The Mouse and keyboard, that is, the peripherals are connected to the USB port.

## 5.2 USING OBSTACLE DETECTION SENSORS ONLY

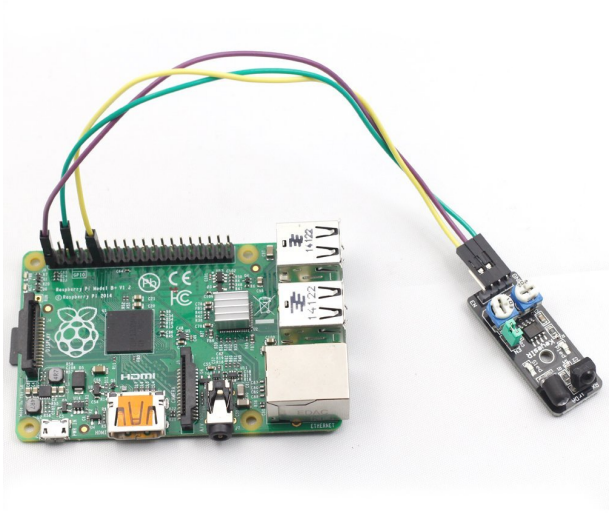

**Figure 5.2** Obstacle detection sensor only

Figure 5.2 shows the infrared obstacle sensor connected to the RPI Microcomputer using Jumper Wires. It is connected to the GPIO pins on the RPI.

## 5.3 USING ULTRASONIC DISTANCE DETECTION SENSOR

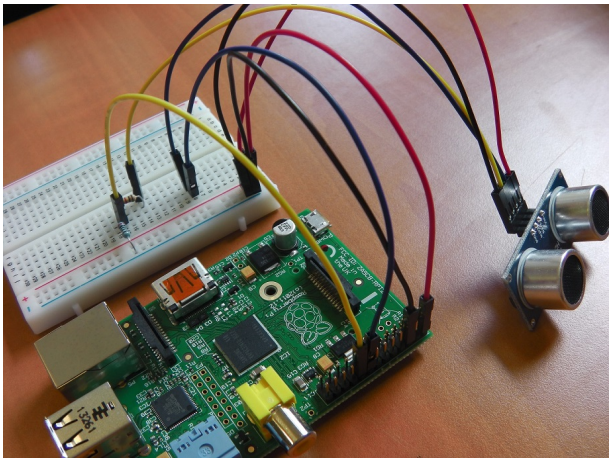

**Figure 5.3** Distance detection sensor only

Figure 5.3 gives the setup with the ultrasonic sensor connected the raspberry pi microcomputer. The ultrasonic sensor is used to measure distance of the obstacle.

## 5.4 A COMBINATION OF THE OBSTACLE AND ULTRASONIC SENSORS

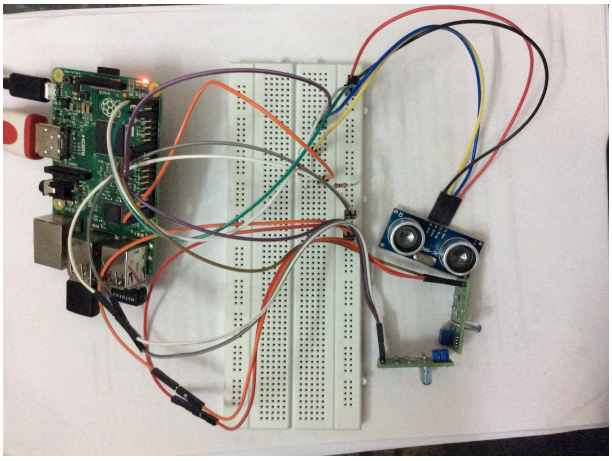

**Figure 5.4** Complete setup using both sensors

Figure 5.4 shows the complete sensor system required for the parking management system. this system uses the ultrasonic distance detection sensor optionally in case we can't confirm if there is car is present or not.

# CHAPTER 6

# SYSTEM DEVELOPMENT

## 6.1 INPUT AND OUTPUT TO SYSTEM

### 6.1.1 Input

The input sensor senses from real world entity(parking lot). For each lot two sensors have been fitted.
The following cases are possible here:

1. Both sensor detects there is car.
2. One detects there is no car and another detects there is a car.
3. Both detects there is no car.

### 6.1.2 Output

The output of the system is to show whether there is a car parked in a specified lot or not.
The following cases are possible here:

1. If both sensors are ACTIVE HIGH, then definitely there is a car parked
2. If one is ACTIVE LOW and another detects an ACTIVE HIGH, this case has to be handled by additional setup like visual conformation.
3. If both sensors are ACTIVE LOW, then there is no car.

## 6.2 INPUT AND OUTPUT FOR EACH MODULE

### 6.2.1 Parking Data Sensing and Processing

**Input**

Input to sensing module is from sensor, which gets the real world data as input (from parking lot).

**Output**

Output of sensing module is the processed binary data from raspberry pi, which is sent to server.

### 6.2.2 Parking Data Manipulation and Mapping

**Input**

The input to this section is the data from sensing module.

**Output**

The output of this module is binary data mapped to a web interface.

### 6.2.3 Constraints

1. This version of the sensing module cannot differentiate between a rogue obstacle and a car.
2. If both sensors are HIGH, visual confirmation is required to check whether the obstacle is a car or any other object.
3. This version of the sensing module cannot detect the presence of a car if it is parked in an improper manner in the parking space.

# CHAPTER 7

# RESULTS AND DISCUSSION

## 7.1 RESULTS

### 7.1.1 Assessment

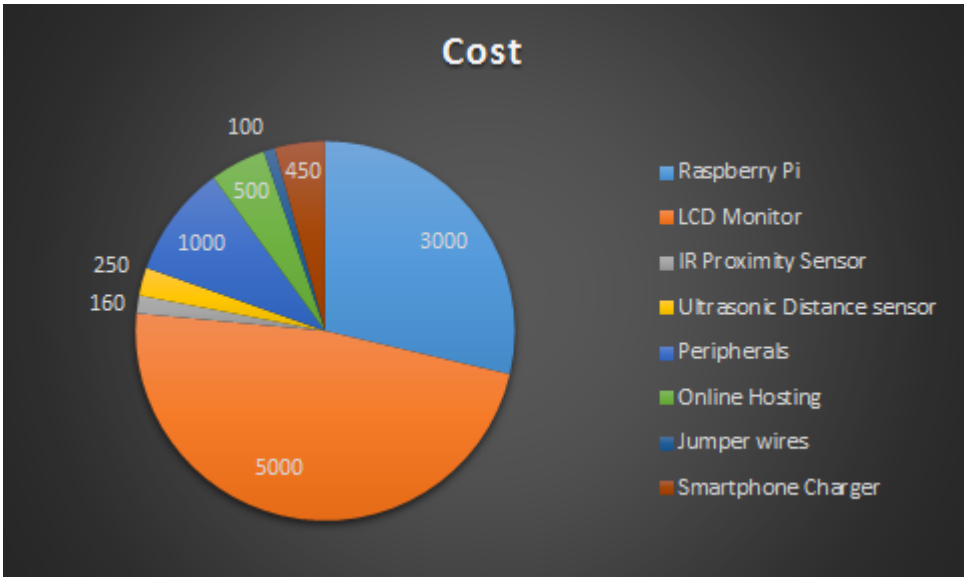


**Figure 7.1** Project Cost

Figure 7.1 shows the cost estimate for our project. The RPI microcomputer and the LCD monitor forms 75% of the total project budget.

### 7.1.2 Evaluation

The most important parameter used to evaluate the Intelligent Parking Management System is **Accuracy**. The success of the system is mainly based on how accurate the IR proximity sensor is. The sensor must be able to accurately detect the presence of an obstacle. There should

be a change in state of the sensor during the presence and absence of an obstacle respectively. Next parameter to be evaluated is **Authenticity**. The sensor must be able to differentiate between a car parked and a rogue obstacle. **Robustness** is very important for any system. The build of the system should be strong and the system should not be easily damaged by physical force.

### Measuring Accuracy

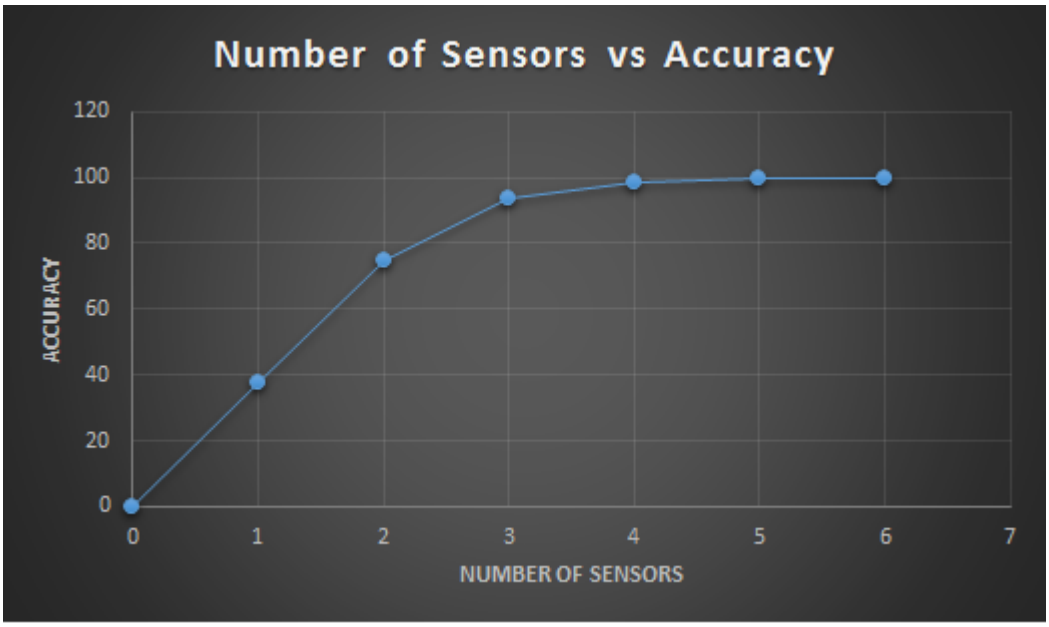


**Figure 7.2** Accuracy vs No. of sensors

The graph in Figure 7.2 evaluates our most important parameter, that is, **Accuracy**. This graph depicts the accuracy in terms of percentage with respect to the number of sensors used. Lesser number of IR sensors indicate that the accuracy is much lesser. We cannot afford to have less accuracy or else the whole system fails.

**Measuring Robustness**

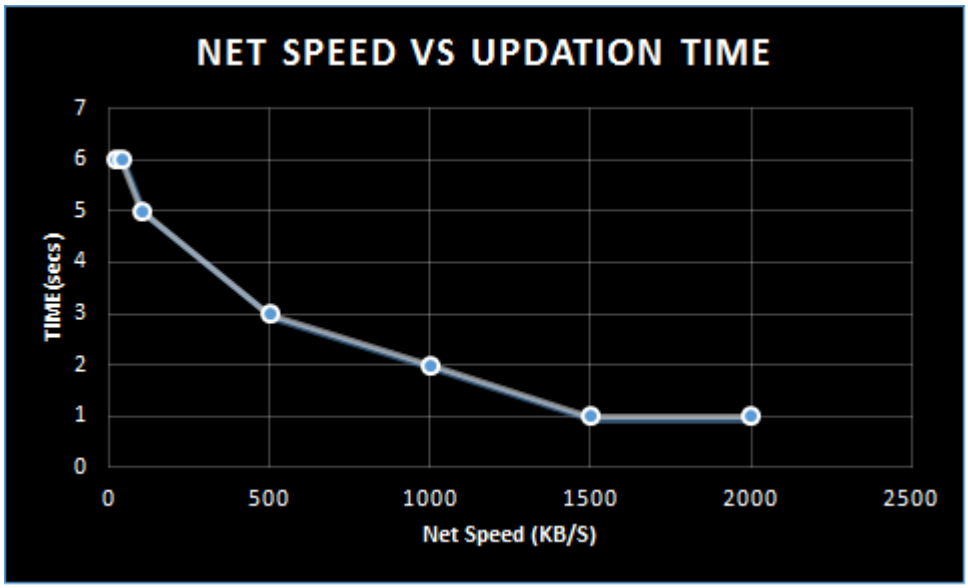


**Figure 7.3** Net Speed vs Updation time

The graph in Figure 7.3 evaluates the next most important parameter, that is, **Robustness**. The graph depicts how the system is stable in responding to various Internet Speeds. An internet with a moderate speed is required to change the state of the parking layout without any latency.

# CHAPTER 8

# CONCLUSIONS

## 8.1 CONTRIBUTIONS

This project establishes a method to integrate the various technologies that can be used for a host of applications in various fields. The interfacing of an array of infrared proximity sensors with a low cost microcomputer like the RPI and mapping the output to a web page over the internet using Python and PHP is a relatively new combination. This particular arrangement of connecting the raspberry-pi over the internet and making it interact with the real world is an example of the internet of things and has not been exploited to its absolute potential. Interaction between the Adurino and the combination of proximity sensors has been carried out through various platforms like the internet and Bluetooth, but this has not been carried forward to the raspberry-pi. Due to various power requirements and other factors most attempts at such an integration had been unsuccessful in the past. The implementation of the above arrangement using latest equipment is a first of its kind.

## 8.2 FUTURE WORKS

The basic arrangement can be extended to newer areas of application. In addition to being the system for Parking Management, this can be extended to perform multiple functions when it comes to the case of malls and theaters. Future works will involve user friendly Cross platform mobile applications that can be used to log on to the mall or theater's portal in order to access booking, payment of parking charges and other transactions related to the corresponding building. In future, the target is to achieve a fully automated system for managing the needs of the people when they visit a mall, theater, or other famous public places involving an intelligent parking management system. State of the art tech will be involved like NFC's to tag the cars when they are parked instead of using sensors.

# APPENDIX A

# APPENDIX

## PHP CODE TRANSFERRING AND STORING DATA ONTO THE SERVER

```
<?php
$status1 = ($_GET["status1"]);
$status2 = ($_GET["status2"]);
$time   = ($_GET["time"]);
file_put_contents('output.txt', $status1.$status2,$time);
if ( ! empty( $status1 ) ) {

  file_put_contents('output.txt', "poda".$status1.$status2);
  // Connect to MySQL
  $mysqli = new mysqli( 'server9.000webhost.com',
     'a5926846_aswin', '11923914', 'a5926846_aswin' );

  // Check our connection
  if ( $mysqli->connect_error ) {
    die( 'Connect Error: ' . $mysqli->connect_errno . ': ' .
       $mysqli->connect_error );
  }

  // Insert our data
  $sql = "DELETE FROM `final`";
  $mysqli->query($sql);
  $sql1 = "INSERT INTO final ( id, status, time ) VALUES ( '1',
     '$status1' ,'$time')";
```

```
  $sql2 = "INSERT INTO final ( id, status, time ) VALUES ( '2',
     '$status2' ,'$time')";
  $insert = $mysqli->query($sql1);
  $insert = $mysqli->query($sql2);

  // Print response from MySQL
  if ( $insert ) {
    echo "Success! Row ID: {$mysqli->insert_id}";
  } else {
    die("Error: {$mysqli->errno} : {$mysqli->error}");
  }

  // Close our connection
  $mysqli->close();
}
?>
```

## PHP CODE FOR DISPLAYING THE PARKING STATUS

```
<?php
// Create connection
$url1=$_SERVER['REQUEST_URI'];
   header("Refresh: 2; URL=$url1");
$conn = new mysqli( 'mysql1.000webhost.com', 'a5926846_aswin',
   '11923914', 'a5926846_aswin' );
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM final";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
```

```
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["status"]. "
            " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

---

## PYTHON CODE

---

```
import RPi.GPIO as GPIO
import time
import requests
from random import randint
from datetime import datetime

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)
GPIO.setup(23,GPIO.IN)

while True:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(24, GPIO.IN)
    GPIO.setup(23,GPIO.IN)

    input_state1 = GPIO.input(23)
    input_state2 = GPIO.input(24)
    print (int(input_state1),int(input_state2))
    if input_state1 == True and input_state2 == True :
        state1 = 'present'
#       print (int(input_state1))
    if input_state1 == False and input_state2 == False:
```

```
        state1 = 'not present'
#        print (int(input_state1))
    if input_state1 == False and input_state2 == True:
        TRIG = 20
        ECHO = 21

        print "Doing further tests to determine availability"

        GPIO.setup(TRIG,GPIO.OUT)
        GPIO.setup(ECHO,GPIO.IN)

        GPIO.output(TRIG, False)
#        print "Waiting For Sensor To Settle"
        time.sleep(2)

        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        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)
        if distance <= 200:
            print "Distance:",distance,"cm","present"
            state1 = 'present'
        else:
            print "Distance:",distance,"cm","not present"
```

```
            state1 = 'not present'
        GPIO.cleanup()


    if input_state1 == True and input_state2 == False:
        TRIG = 20
        ECHO = 21

        print "Doing further tests to determine availability"

        GPIO.setup(TRIG,GPIO.OUT)
        GPIO.setup(ECHO,GPIO.IN)

        GPIO.output(TRIG, False)
#        print "Waiting For Sensor To Settle"
        time.sleep(2)

        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        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)
        if distance <= 200:
            print "Distance:",distance,"cm","present"
            state1 = 'present'
```

```
    else:
        print "Distance:",distance,"cm","not present"
        state1 = 'not preset'
    GPIO.cleanup()


print (state1)
userdata = {"status1": state1}
#userdata = {"status1": randint(0,9),"status2": randint(0,9)}
resp = requests.post('http://aswinp.in/test1.php',
   params=userdata)
print (resp.status_code)

print
print
print
time.sleep(5)
```

## SAMPLE OUTPUT

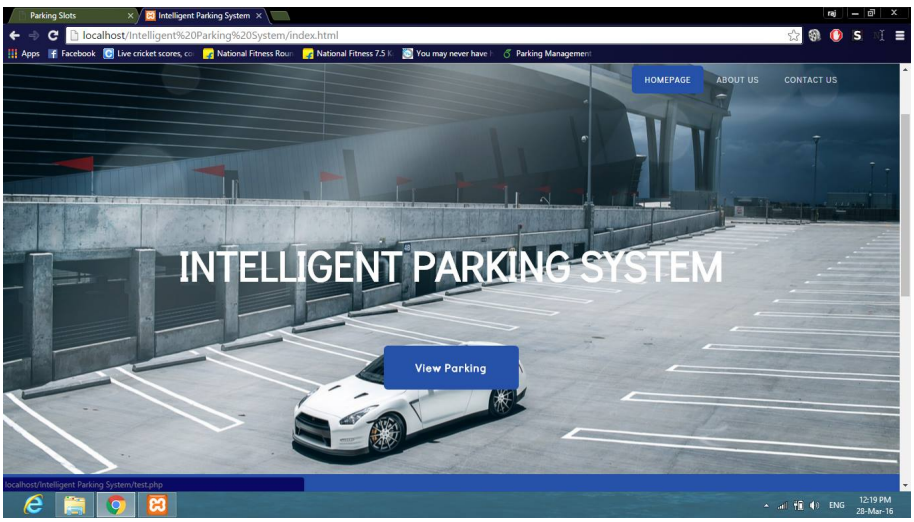


**Figure A.1** Web interface

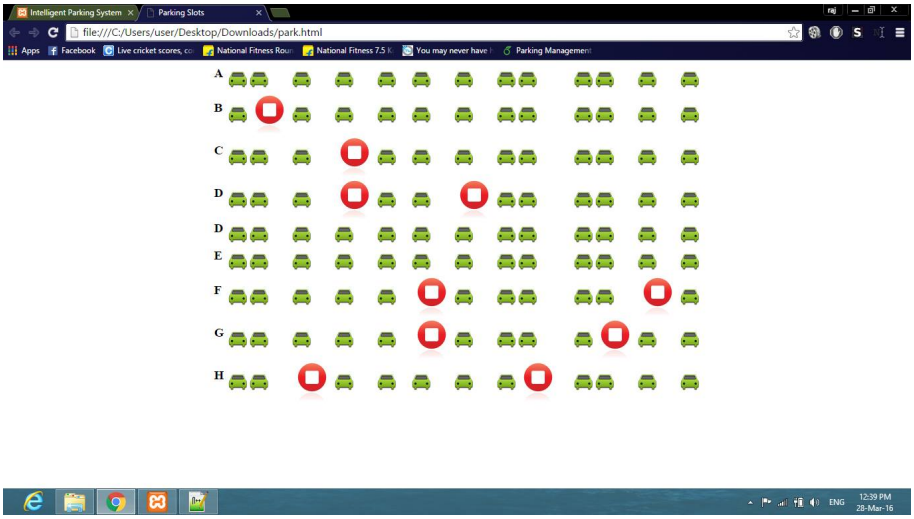


**Figure A.2** Web display to show the availability

# REFERENCES


[1] Jay H. Ball, "Determining the availability of parking spaces", *United States Patent*, 2001.

[2] Carlo Blazquez, Felix Caicedo, and Pablo Miranda, "Prediction of parking space availability in real time", *Elsevier*, 2012.

[3] Andreas Klappenecker, Hyunyoung Lee, and Jennifer L. Welch, "Finding available parking spaces made easy", *Elsevier*, 2012.

[4] Ojas Pandya, Shiamak Mehta, Hemant Jarkad, and Arti Mohanpurkar, "Predictive analytics: Parking prediction using sensors and gps", *International Conference on Advances in Engineering Science and Management*, 2015.

[5] Elena Polycarpou and Lambros Lambrinos, "Smart parking solutions for urban areas", *IEEE*, 2013.

[6] Joseph P. Quinn, "Detection and remote notification of vehicle parking space", *United States Patent*, 2005.

[7] Yanxu Zheng, Sutharshan Rajasegarar, and Christopher Leckie, "Parking availability prediction for sensor-enabled car parks in smart cities", *IEEE*, 2015.