IOT BASED EMPTY PARKING SLOT DETECTION AND VEHICLE GUIDANCE USING IMAGE PROCESSING

A PROJECT REPORT

Submitted by

C.V.AISHWARYA DEVI	**715515105001**
S.BAVATHARANI	**715515105006**
N.K.JAYA DARSHINI	**715515105019**

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2019

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"IOT BASED EMPTY PARKING SLOT DETECTION AND VEHICLE GUIDANCE USING IMAGE PROCESSING"** is the bonafide work of **"C.V. AISHWARYA DEVI(715515105001), S.BAVATHARANI (715515105006) , N. K. JAYADARSHINI(715515105019)"** who carried out the project work under my supervision.

Dr. C. L.VASU	**Mr. RAVIKRISHNA**
HEAD OF THE DEPARTMENT	**FACULTY GUIDE**
Department of Electrical and Electronics Engineering	Assistant professor
PSG Institute of Technology and Applied Research	Department of Electrical and Electronics Engineering
	PSG Institute of Technology and Applied Research

It is certified that the candidate was examined in the Viva Voce examination held on 01/04/2019.

Internal Examiner External Examiner

ABSTRACT

In recent times the concept of smart cities has gained great popularity. Problems such as traffic congestion, limited car parking facilities and road safety are being addressed by IoT. The major problem occurring in a parking area is that vehicles waste time on searching for parking slot. Sometimes it creates blockage. Condition becomes worse when there are multiple parking lanes and each lane has multiple parking slots. Use of automated system for car parking monitoring will reduce human efforts. However, in current parking system a better but not an optimal solution is being provided. It does not provide economic benefit, vehicle refusal services and there is no resource reservation mechanism leading to queuing system which is again time consuming. It also lacks to provide large scale parking system. In this model of IOT based smart parking system, empty parking slot is being detected by image processing technique. The number of empty slots are obtained from the processed image is being stored in the cloud. Once the car is being detected in the entrance, the parking space image is being captured by a camera and is detected whether there is an empty slot or not. If there is an empty slot, the gate is opened and allows the car into the nearest parking slot. Otherwise, the user will be instructed that there is no available empty slot. The user is also provided with a reservation facility. Thus the wastage of time in searching for a parking area with available empty slot is reduced.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to **Dr. P .V. Mohanram**, Principal, PSG Institute of Technology and Applied Research, for providing us the opportunity to carry out our project with the best infrastructure and facilities, without the project would have not been successful.

We extend our sincere gratitude to **Dr. C. L. Vasu,** Head of the Department, Electrical and Electronics Engineering, for his unfailing support throughout the project.

We extend our heartfelt thanks to our project guide, **Mr.S. Ravikrishna**, Assistant Professor, for his valuable advice which played a major role in completion of the project.

We express our sincere thanks to all the staff members, friends, and our parents for their constant support and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
CA	Common Anode
CC	Common Cathode
EEPROM	Electrically Erasable Programmable Read Only Memory
FTDI	Future Technology Devices International
GPS	Global Positioning System
ICSP	In Circuit Serial Programming
IFTTT	If This Then That
IoT	Internet Of Things
IPA	Intelligent Parking Assistant
IR	Infra Red
LED	Light Emitting Diode
MATLAB	Matrix Laboratory
PWM	Pulse Width Modulation
RFID	Radio Frequency Identification Number
SE	Structuring Element
SPS	Smart Parking System
SRAM	Static Random Access Memory

UART	Universal Asynchronous Receiver Transmitter
UHF	Ultra High Frequency
USB	Universal Serial Bus
VANET	Vehicular Ad Hoc Networks
VMD	Variable Message Display

CHAPTER 1
INTRODUCTION

Finding a vacant parking space is a common problem in most urban cities. This situation has become more serious especially during their peak time, be it holiday seasons, sales carnivals or any other festivals. This problem arises as most of the time, as patronscome by their own transport, resulting in abundanceor high number or transports competing for a few vacant parking spaces. The limited availability of vacant parking spaces often results in traffic congestion, as well as making a driver frustrated infinding an available parking space. In fact, it is one of the main problems which result to traffic congestion. In the existing system, there is no facility for reserving the available parking slots in advance. To tackle this problem, the smart parking system has been proposed. The smart parking system concentrates on solving the problem of proper parking management by utilizing the advancement of technologies which will definitely help in alleviating, if not solving the current traffic problem. It also provides the facility for the user to reserve the available parking space through an android application. Most of the smart parking systems (SPS) proposed in literature over the past few years provides solution to the design of parking availability information system, parking reservation system, occupancy detection and management of parking slot, real-time navigation within the parkingfacility. This report presents an internet-of things (IoT) based E-parking system that employs an integrated combination of both hardware and software.

The term smart parking refers to the technology which allows user to have a better idea about the number of vacant slots and dynamically reserve for parking if necessary.

1.1 Motivation:

Navigating the congested streets on a daily basis is a tiring experience and drains all energy from one's body and is a topic that is very common among those that fall victim to it. However, our true motivation came from a discussion where we were talking about the future prospects of IoT and cloud computing in India.

Technological advances have seen no implementation in parking management systems till date. With the vision of 'Digital India' in mind and all the resources that will be available to us, we have planned to create an effective smart parking system that significantly reduces traffic congestion.

1.2 Problem Statement:

The parking management problem can be viewed from several angles. The major problems include

- Limited number of parking slots.
- Difficulties in knowing the availability of slots.
- Unable to reserve the slots in advance.
- Tendency to park illegally on the roads.
- Inability to locate nearest parking area.

1.3 Solution:

The following solutions can be provided for an effective smart parking system.

- Monitoring of the number of free and occupied parking spaces
- Informing parking slot users about the parking slot status
- Providing reservation facility in a wider area
- Providing guidance to the user about the nearest parking slot through a mobile application. This information can also be used for further analysis.

CHAPTER 2
LITERATURE SURVEY

This chapter deals with the outline for existing methods for parking system and how smart parking system has evolved in these years. Understanding and improving the parking system can help with better urban planning. 98% of automobile trips within the Los Angeles metropolitan area start or end with free parking (California household travel survey, 2013). Most of the drivers ever abandoned their trips because of annoying and endless parking searches and some drivers park their cars on unauthorized areas (Association for European Transport 2006). Accordingly, if drivers can have real-time parking availability information, they will be able to adjust their traveling schedule without spending time cruising the city in vain. Many cities have started smart parking projects. Smart parking is a way to help drivers find more efficiently satisfying parking spaces through information and communications technology. Cities with more on-street parking spaces need comparatively to adopt smart parking to avoid drivers cruising for free parking. In addition, cities deploy smart parking services on an economic initiative basis. First, drivers can shorten their parking search time, reduce environmental pollution, reduce costs with less fuel consumption and alleviate traffic congestion through information from smart parking apps. That also increases public transportation use rate and cities revenues as well. Second, if drivers can rapidly find a parking space, the idle time for on-street parking is shorter and the parking revenues increase. Installing sensors on unauthorized areas where people frequently park their car can help detect illegal parking and can issue in a penalty charge, as with electric vehicles and disabled parking stalls. Third, once the traffic is fluent, it increases urban mobility and expands city’s capacities. It brings more population.

Idris et al did a survey on smart parking systems using different parking sensors and the affiliated functionalities, e.g., centralized or opportunistic information storage systems, park-and-ride facilities, E-parking, automated parking, reservation systems, parking assist or guidance systems and vehicle license plate recognition.

Polycarpou et al presented a survey on driver's needs for parking infrastructures, e.g., public parking, driver's behaviors, parking availability monitoring, guidance and information systems, reservation systems.

Faheem et al classified the current smart parking systems and explained their features: fuzzy-based for human-like intelligence and expertise; wireless sensor-based for the detection and monitoring of the parking facilities; GPS-based to provide real time location and guidance systems; vehicular communication for parking information distribution services among mobile vehicles; vision based for lot occupancy detection and space recognition. In some studies, the authors proposed a new algorithm for treatment planning in real-time parking. First, they used an algorithm to schedule the online problem of a parking system into an offline problem. Second, they set up a mathematical model describing the offline problem as a linear problem. Third, they designed an algorithm to solve this linear problem. Finally, they evaluated the proposed algorithm using experimental simulations of the system. The experimental results indicated timely and efficient performance. However, these papers do not mention the resource reservation mechanism (all parking requirements are derived immediately and are placed in the queue), the mechanism for assessing the resources system, the mechanism to guide vehicles to the parking space, the mechanism for handling situations when the request for service is denied and do not calculate the average waiting time and average total time that each vehicle spends on the system.

In another study Mainetti , the authors propose an SPS based on the integration of UHF frequency, RFID and IEEE 802.15.4 Wireless Sensor Network technologies. This system can collect information about the state of occupancy of the car parks, direct drivers to the nearest vacant parking spot by using a software application. However, in this work, the authors have no mathematical equations for the system architecture and do not create a large-scale parking system. The results of this paper only implement the proposed architecture; they do not mention the performance of the parking system. Other researchers have designed architecture for parking management in smart cities Barone et al. They proposed intelligent parking assistant (IPA) architecture aimed at overcoming current public parking management solutions. This architecture provides drivers with information about on-street parking stall availability and allow drivers to reserve the most convenient parking stall at their destination before their departure. They use RFID technology in this system. When a car parks or leaves the IPA parking spot, the RFID reader and the magnetic loop detect the action and send this information to the unit controller to update the information on the car park status.

In other works, authors have designed and implemented an SPS whose bottom part is composed of ZigBee network which sent pressure information to PC through a coordinator and then update database Shiyao et al. to solve the parkingproblem. A part of this system is implemented in the Zigbee network which sends urgent information to a PC through a coordinator and then updates the database.

The application layer can quickly pass the parking information over the Internet, and use the advantages of a web service to gather all the scattered parking information for the convenience of those who want to find a parking space. Bonde et al. aimed to automate the car and the car parking. The paper discusses a project which presents a miniature model of an automated car parking system that can regulate and manage the number of cars that can be parked in a given area at any given time based on the availability of parking spaces. The automated parking method allows the parking and exiting of cars using sensing devices. Entry to or exit from the car park is commanded by an Android based application. The difference between the Bonde system and the other existing systems is that the authors were aiming to make the system as little human dependent as possible by automating the cars as well as the entire car park; on the other hand, most existing systems require human intervention (the car owner or other) to park the car.

A number of parking guidance systems aiming to minimize driver inconvenience have been reported in the literature over the last decade. The aforementioned works describe architectures for real-time parking space availability which take into account information obtained from parking meters or sensors installed across the parking lot.

To avoid the cost of using parking sensors, an alternative solution by Lu et.al performs parking reservation by exploiting the capabilities of vehicular ad hoc networks (VANETS). VANETs utilize wireless communication enabling cars to communicate with each other and with the roadside infrastructure. The main drawback of such technology is that it requires special equipment to be installed in cars and the roadside; it is anticipated that such deployment will take some time and therefore aforementioned approach is not realistically implementable at present.

CHAPTER 3
EMPTY SLOT DETECTION USING IMAGE PROCESSING

3.1 General schematic block diagram

As depicted in the figure 3.1, the image is acquired using a fixed lens camera. The captured image is given to MATLAB for processing. The empty and occupied parking slots are detected using MATLAB. The output from MATLAB is forward to mobile application through cloud (Thingspeak) and to Arduino through Bluetooth. The empty slots will be displayed in a seven segment display.

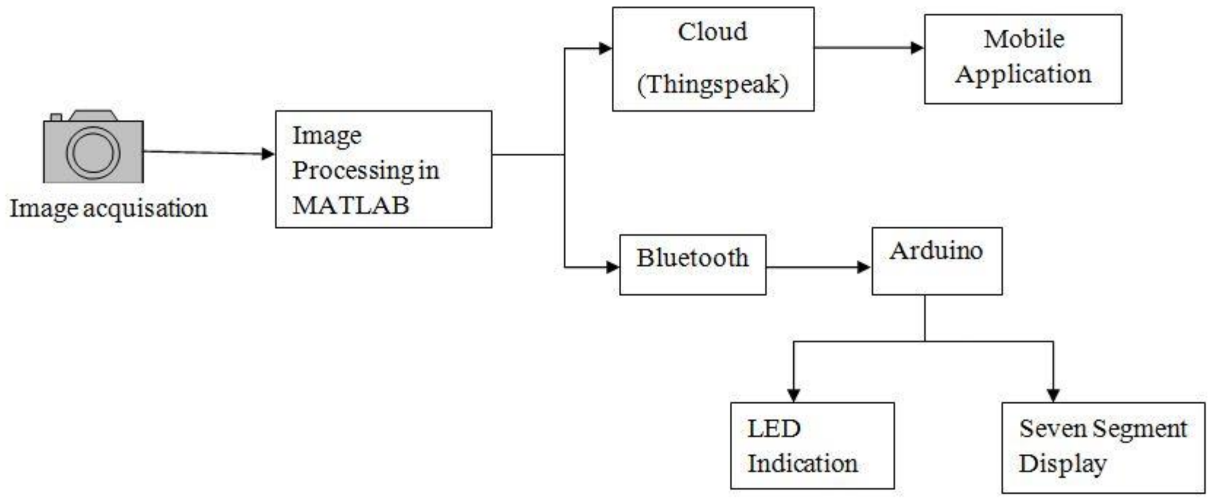


Fig. 3.1 Empty Slot Detection and Display

3.2 USB Camera

USB Cameras are imaging cameras that use USB 2.0 or USB 3.0 technology to transfer image data. A webcam is a video camera that feeds or streams its image in real time to or through a computer to a computer network. The video fed from webcam can also be used to detect empty parking slot. Webcams typically include lens, an image sensor, support electronics.

Fig. 3.2 USB Web Camera

Table 3.1 USB Camera Specifications

Name	'FRONTECH ECAM USB PC CAMERA'
Resolution	320x240
Available Resolutions	{1×9 cell}
Brightness	16
Saturation	1
Sharpness	280
Contrast	120

3.3 Arduino

Arduino is an open-source platform. It consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on the computer, used to write and upload computer code to the physical board. The Arduino board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards and other circuits.

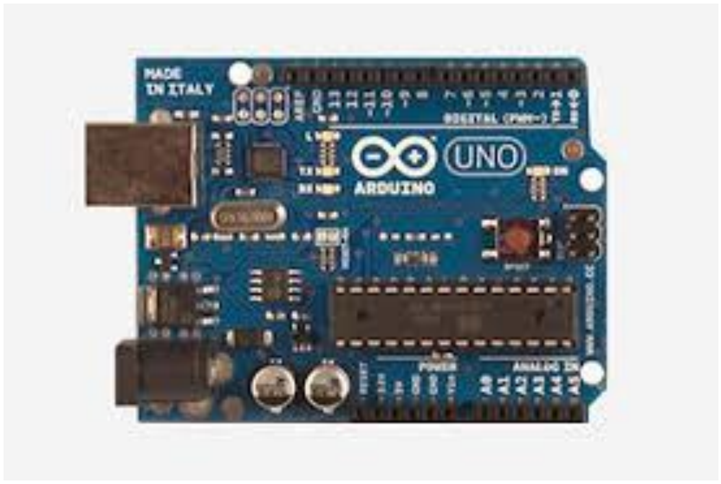


Fig. 3.3Arduino Board

3.3.1 Arduino UNO

The Arduino UNO is a microcontroller board based on the Atmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

3.3.2 Features of Arduino UNO

Table 3.2 depicts the features of Arduino UNO

Microcontroller	ATmega32
Operating Voltage	5 V
Input Voltage (recommended)	7-12V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pin	6
DC Current per I/O Pin	40 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega 328)
Clock Speed	16 MHz

3.3.3 Pin diagram

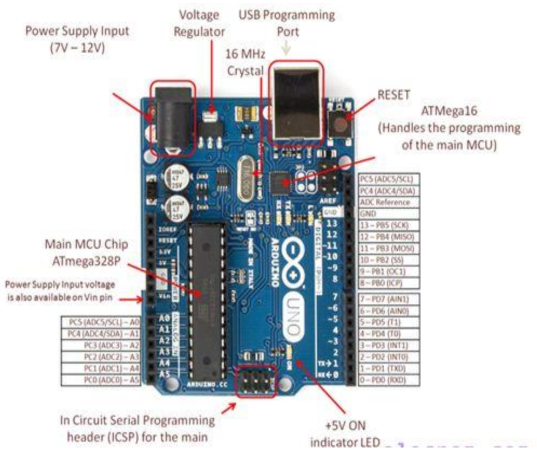


Fig. 3.4 Arduino Pin Diagram

3.3.4 Pin description

GND: Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground the circuit.

5 V & 3.3 V: the 5 V pin supplies 5 volts and 3.3 V supplies 3.3 volts.

Analog pins: The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor and convert it into a digital value that we can read.

Digital pins: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

Reset button: the Arduino has a reset button . Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino.

Power LED indicator: This LED should light up whenever we plug Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong.

TX RX LED: TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear -- once by digital pins 0 and 1, and a second time next to the TX and RX indicator LED’s **(12)**. These LED’s will give us visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

Main IC: The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company.

Voltage Regulator: The input voltage supply to Arduino is in the range of 9-12 volts. The incoming voltage is regulated to 5 volts.

3.4 Infrared sensor

An infrared sensor is an electronic device that emits infrared radiations in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. An infrared sensor circuit is one of the basic and popular sensor module in an electronic device. This sensor is analogous to human's visionary senses, which can be used to detect obstacles and it is one of the common applications in real time. Following are the principles used in IR sensor:

1. Planck's radiation law
2. Stephan Boltzmann Law
3. Wein's Displacement Law

The principle of an IR sensor working as an object detection sensor can be explained using the following figure. An IR sensor consists of an IR LED and an IR Photodiode; together they are called as Photo – Coupler or Opto – Coupler.

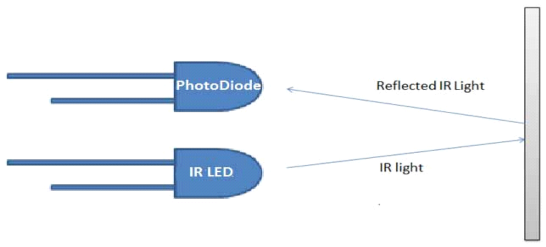


Fig. 3.5 Working of an IR Sensor

Fig. 3.5 depicts the working of an IR sensor

When the IR transmitter emits radiation, it reaches the object and some of the radiation reflects back to the IR receiver. Based on the intensity of the reception by the IR receiver, the output of the sensor is defined.

A typical system for detecting infrared radiation is given in the following block diagram

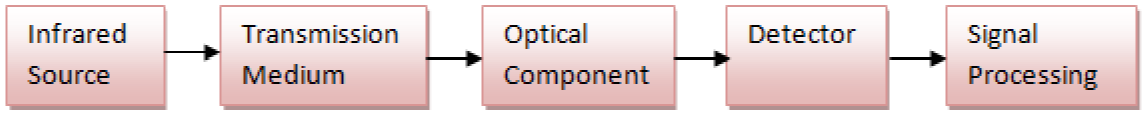


Fig 3.6 Block Diagram of a Typical System for Detecting Infrared Radiation

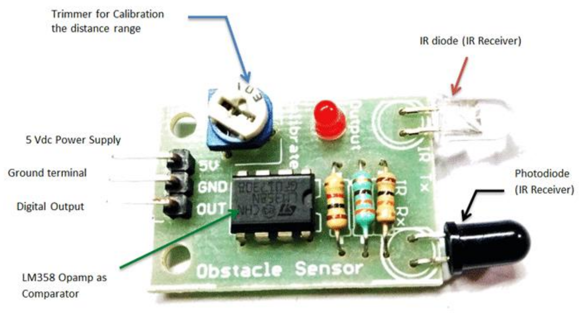


Fig. 3.7 A Typical IR Sensor Module

3.5 Bluetooth (HC-05)

A Bluetooth technology is a high speed low powered wireless technology link that is designed to connect phones or other portable equipment together. IEEE 802.15.1 is a specification for the use of low power radio communications to link phones, computers and other network devices over short distance without wires. Wireless signals transmitted with Bluetooth cover short distances, typically up to 30 feet (10 meters).

Some of the types of Bluetooth module are as follows:

- HC-05
- HC-06 RS232 TTL
- BLE Link Bee
- BLE Mini
- Blue SMiRF

Bluetooth can be connected up to “eight devices” simultaneously and each device offers a unique 48 bit address from the IEEE 802 standard with the connections being made point to point or multipoint.HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup. The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project etc.

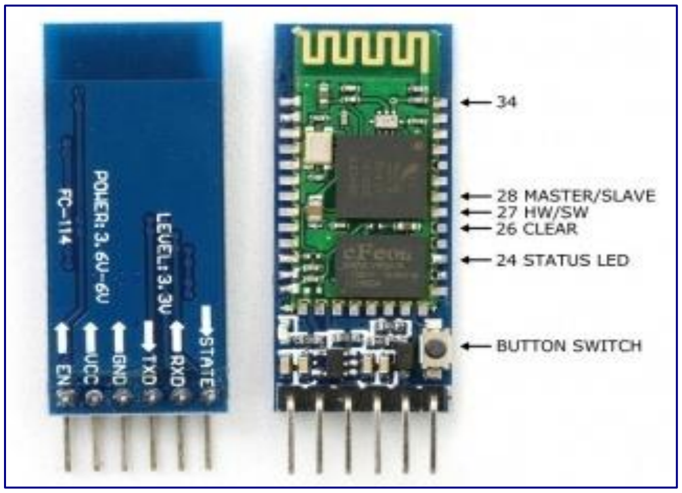


Fig. 3.8 Pin Diagram of a Bluetooth Module

3.6 Light Emitting Diode

Light emitting diodes **(**LEDs**)** are semiconductor light sources. The light emitted from LED**s** varies from visible to infrared and ultraviolet regions. They operate on low voltage and power. LEDs are one of the most common electronic components and are mostly used as indicators in circuits. They are also used for luminance and optoelectronic applications. Based on semiconductor diode, LED**s** emit photons when electrons recombine with holes on forward biasing. The two terminals of LEDs are anode and cathode and can be identified by their size. The longer leg is the positive terminal or anode and shorter one is negative terminal.

Fig. 3.9 Light Emitting Diode

3.7 Seven Segment Display

The 7-segment display, as depicted in fig. 3.10 , consists of seven LEDs (hence its name) arranged in a rectangular fashion. The seven segment display is the most common display device used in many gadgets, and electronic appliances like digital meters, digital clocks, microwave oven and electric stove, etc. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point, (DP) when two or more 7-segment displays are connected together to display numbers greater than ten. Seven segments are indicated as A-G and the eighth segment is indicated as H. These segments are arranged in the form of 8 which is shown in the seven segment display circuit diagram. As each LED has two connecting pins, one called the "Anode" and the other called the "Cathode", there are therefore two types of LED 7-segment display namely,

- Common Cathode (CC)
- Common Anode (CA)

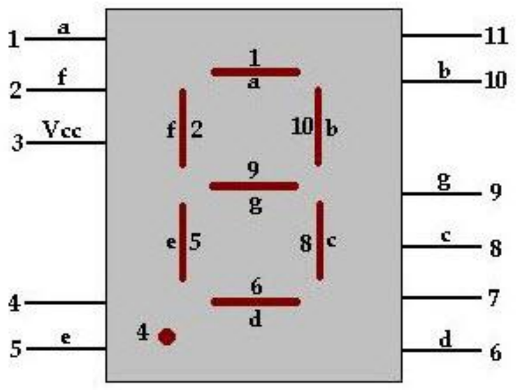


Fig. 3.10 Pin Diagram of a Seven Segment Display

Fig. 3.11 depicts the connection from MATLAB to Arduino to Bluetooth

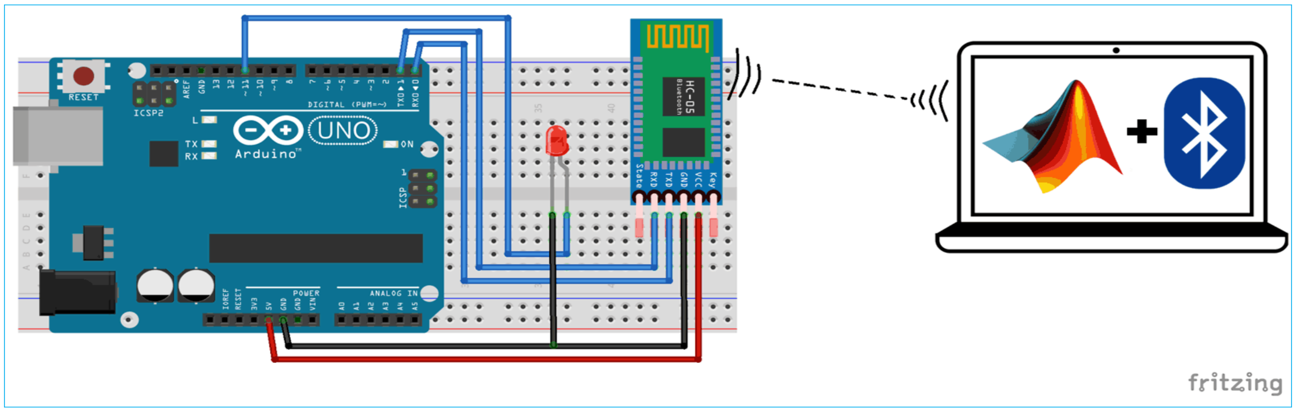


Fig. 3.11 Connection From MATLAB To Arduino Using Bluetooth

3.8 MATLAB

MATLAB (MATrix LABoratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by Math Works. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs & using other languages like c, c#, java and python. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Typical uses include:

- Math and computation
- Algorithm development
- Modelling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or FORTRAN. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation. MATLAB has evolved over a period of years with input from many users.MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. In MATLAB, the IPT is a collection of functions that extends the capability of the MATLAB numeric computing environment. It provides a comprehensive set of reference-standard algorithms and workflow applications for image processing, analysis, visualization and algorithm development. It can be used to perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration and 3D image processing operations. Many of the IPT functions support C/C++ code generation for desktop prototyping and embedded vision system deployment.

3.8.1 Image processing

Digital image processing is the use of computer algorithms to create, process, communicate, and display digital images. Digital image processing algorithms can be used to:

- Convert signals from an image sensor into digital images
- Improve clarity, and remove noise and other artifacts
- Extract the size, scale, or number of objects in a scene
- Prepare images for display or printing
- Compress images for communication across a network

Image Processing Toolbox™ in MATLAB provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development.

Following are the functions performed using image processing toolbox:

- image segmentation,
- image enhancement,
- noise reduction,
- geometric transformations,
- image registration, and
- 3D image processing.

3.8.2 Digital image

A digital image may be defined as a two-dimensional function f(x, y), where ‘x’and ‘y’ are spatial coordinates and the amplitude of ‘f’ at any pair of coordinates Is called the intensity of the image at that point. When ‘x,’ ‘y’ and amplitude values of ‘f’ are all finite discrete quantities, the image is referred to as a digital image.

Digitizing the coordinate values is referred to as ‘sampling,’ while digitizing the amplitude values is called ‘quantization.’ The result of sampling and quantization is a matrix of real numbers.

A digitized image is represented as:

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Each element in the array is referred to as a pixel or an image element.

3.8.3 Empty slot detection

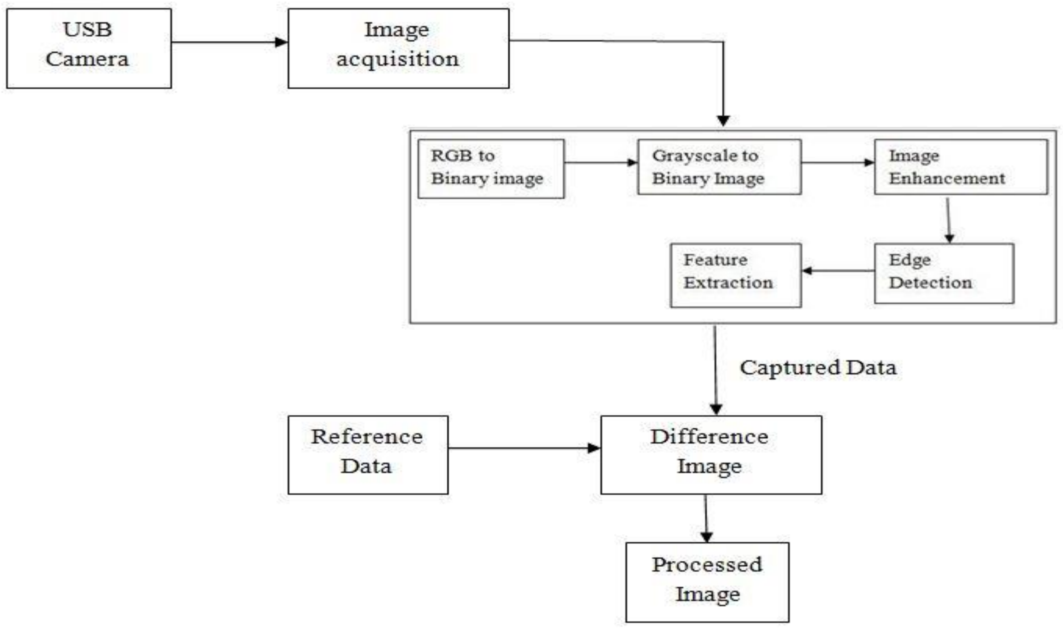


Fig. 3.12 Block Diagram Representing Empty Slot Detection

3.8.4 Image processing algorithm

Images are acquired from the parking area with the help of a fixed camera. The captured image is segmented into frames. Then from each segment a key frame is extracted and further processing is applied on this key frame, to reduce the computational complexity.

In the initial stage, a certain number of images are captured and their average is calculated to make an averaged background reference image. This reference image does not contain any cars. The main purpose is to identify the parking slots in the image. The camera which is used to take the images is fixed at a certain position and it faces a fixed direction all the time.

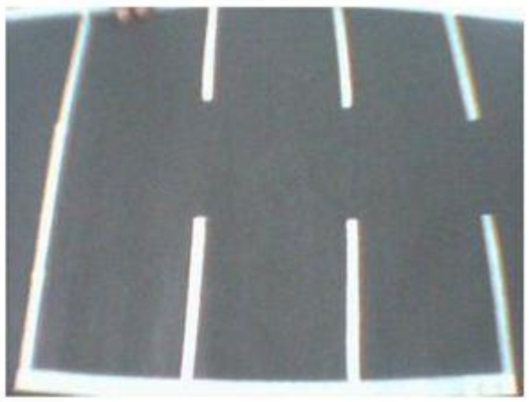


Fig. 3.13 Reference Image

Image segmentation is the process of partitioning the digital image into various segments in order to change the representation of the image into something meaningful which is easier to analyze. The RGB image acquired is then converted to gray-scale image and then binary image is created in the Image segmentation module.

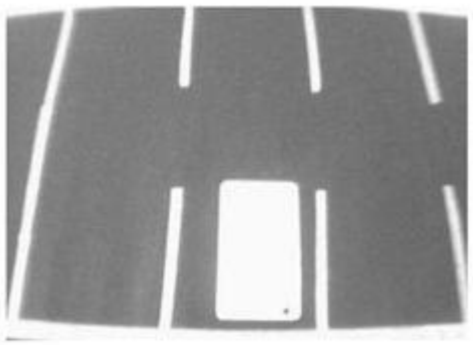


Fig. 3.14 Grayscale Image of the Captured Image

Fig. 3.15 shows the gray scale image of the parking space with cars. From the resulting gray-scale image, binary image is obtained using thresholding technique. The binary image contains all the information about the position and shape of interest. The threshold level is set in such a way that the objects of interest are made into white and the rest of the image black.

The binary image contains a lot of noise which is removed using morphological operations such as dilation, erosion etc. The objective of enhancement is to process an image so that result is more suitable than the original image for the specific application.

The edges in the captured image are detected, features are extracted and finally compared with the reference image. Fig. 3.16 depicts the processed slots and the empty slots are depicted in fig. 3.17.

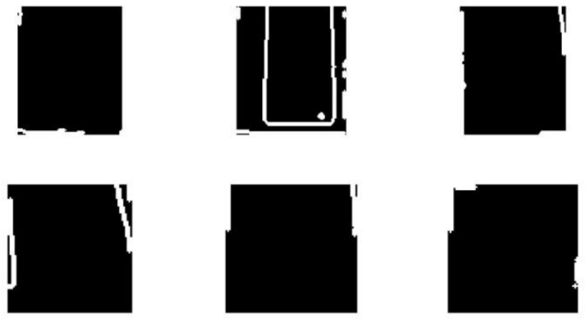
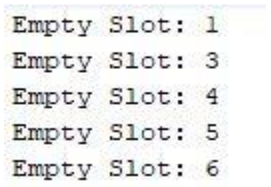


Fig. 3.15 Empty and Occupied Slots



```
Empty Slot: 1
Empty Slot: 3
Empty Slot: 4
Empty Slot: 5
Empty Slot: 6
```

Fig. 3.16 Display of Empty Slots

CHAPTER 4

CLOUD BASED PARKING SLOT MANAGEMENT

4.1 Internet of Things

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. Organizations in a variety of industries are using IoT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business. An IoT ecosystem consists of web-enabled smart devices that use embedded processors, sensors and communication hardware to collect, sent and act on data they acquire from their environments. IoT devices share the sensor data they collect by connecting to an IoT gateway where data is either sent to the cloud to be analyzed locally. The device do most of the work without human intervention, although people can interact with the devices - for instance, to set them up, give them instructions or access the data. The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

4.1.1 Thingspeak API

Thingspeak is an Internet of things platform that lets you collect and store sensor data in the cloud and develop IoT applications. The Thingspeak IoT platform provides apps that let you analyze and visualize your data in MATLAB, and the act on the data. Sensor data can be sent to Thingspeak from Arduino, RaspberryPi and other hardware.

4.1.2 Thingspeak features

- Collect data in private channels
- Share data with public channels
- MATLAB analytics
- Alerts and event scheduling
- Apps integration

4.1.3 Creating a channel in Thingspeak

- Sign in to Thingspeak using your MathWorks Account, or create a new account
- Click **Channels** > **My Channels**.

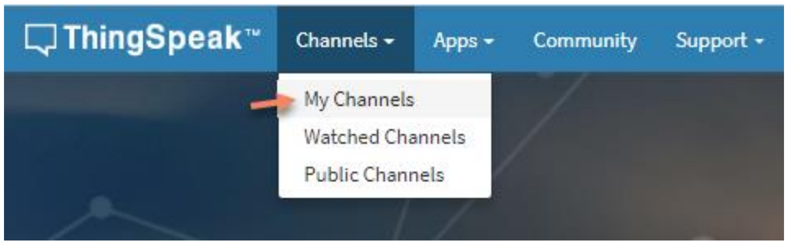
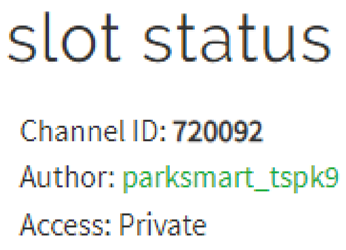


Fig. 4.1 Thingspeak Login Page

The Fig. 4.2 depicts the Thingspeak login page

- On the Channels page, click **New Channel**.
- Create a new channel with one field label



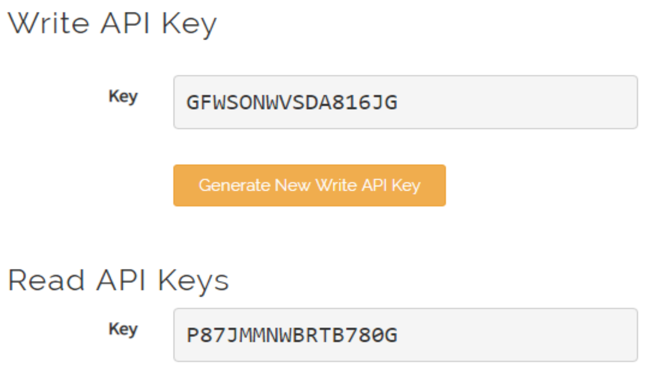
slot status

Channel ID: 720092
Author: parksmart_tspk9
Access: Private

Fig. 4.2 Status of the Channel

The status of the channel is depicted in the Fig. 4.3

- Get the API Key



Write API Key

Key GFWSONWVSDA816JG

Read API Keys

Key P87JMMNWBRTB780G

Fig. 4.3 API Keys

The Fig. 4.3 denotes the obtained API keys.

The processed parking slot status is sent to Thingspeak through MATLAB using the following code

thingspeakWrite(720092,'Fields',[1,2,3,4,5,6],'Values',a,'WriteKey','GFWSONWVSDA816JG');

x=thingspeakRead(720092,'Fields',[1,2,3,4,5,6],'ReadKey','P87JMMNWBRTB780G', 'OutputFormat','table')

The value updated in the field is given by Fig. 4.4 and Fig. 4.5.

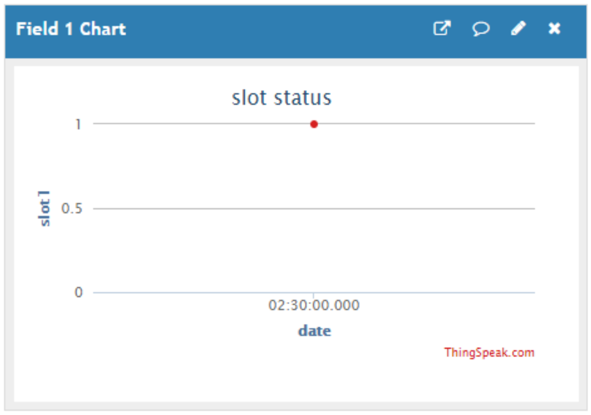


Fig. 4.4 Field 1 Data

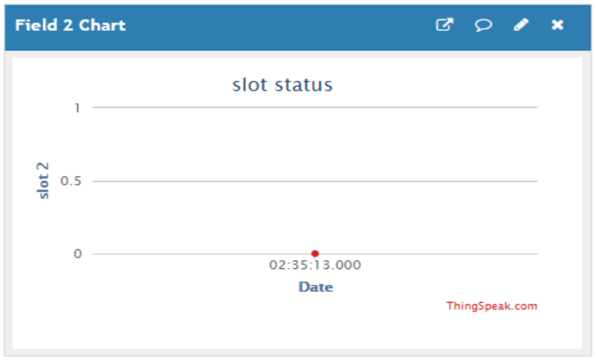


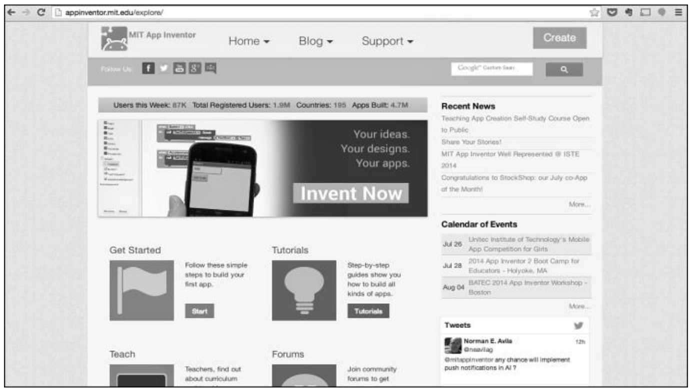
Fig. 4.5 Field 2 Data

4.2 MIT App Developer

MIT App Inventor is a web-based tool for building Android apps. This is often referred to as visual programming, which means the user is able to perform programming tasks without entering any computer code. App Inventor is actively managed and developed by MIT's Mobile Learning Lab (the project was originally built by Google). App Inventor is a free, cloud-based service that allows you to make your own mobile apps using a blocks based programming language. You access App Inventor using a web browser (Chrome, Firefox and Safari). The work takes place in two key sections of App Inventor: the Designer and the Blocks Editor. In the Designer, what actions the app will perform and how it will look will be decided. The programming takes place in the Blocks Editor. There the app is instructed what it should do and give specific instructions for making that happen. App Inventor also offers a method for using the app in real time while performing work on it: the AI Companion app. The Companion app also works wirelessly, so it is not necessary to physically connect your phone to a computer while working in App Inventor. Most IoT projects need a user interface and while IFTTT and DO Buttons work well, sometimes you want something a bit more versatile. It allows newcomers to computer programming to create software applications for the Android operating system (OS). It uses a graphical interface, very similar to Scratch and the Star Logo TNG user interface, which allows users to drag-and-drop visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments. App Inventor also supports the use of cloud data via an experimental Firebase DB component.

4.2.1 Procedure

To begin, launch your browser and go to appinventor.mit.edu. The home page includes the portal to the App Inventor tool, along with many online tutorials and other helpful materials.



Users this Week: 87K Total Registered Users: 1.9M Countries: 195 Apps Built: 4.7M

Follow these simple steps to build your first app.

Step-by-step guides show you how to build all kinds of apps.

Teachers, find out about curriculum

Join community forums to get

Teaching App Creation Self-Study Course Open to Public

Share Your Stories!

MIT App Inventor Well Represented @ ISTE 2014

Congratulations to StockShop: our July co-App of the Month!

Jul 26	Unitec Institute of Technology's Mobile App Competition for Girls
Jul 28	2014 App Inventor 2 Boot Camp for Educators - Holyoke, MA
Aug 04	BATEC 2014 App Inventor Workshop - Boston

Norman E. Avila @neavilag 12h
@mitappinventor any chance will implement push notifications in AI ?

Fig. 4.6 MIT App Inventor Sign In Window

4.2.2 Signing in

To begin a session with App Inventor, click the Create button at the top of the home page.

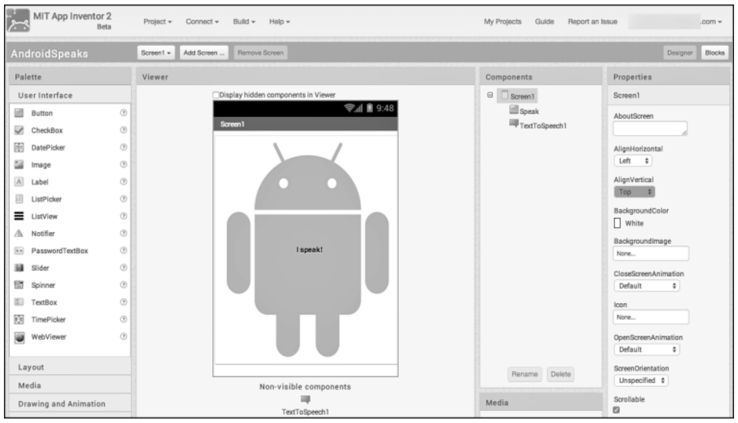


Fig. 4.7 MIT App Inventor Home Page

4.2.3 Designer

App building begins in the Designer. Here the user Interface, or the "look and feel" of the app can be created. The components needed to receive input from the user, as well as the components needed to display output or information to the user can also be added. The Designer is where in which non-visible components the app will use, such as the dialer, GPS, or SMS are specified.

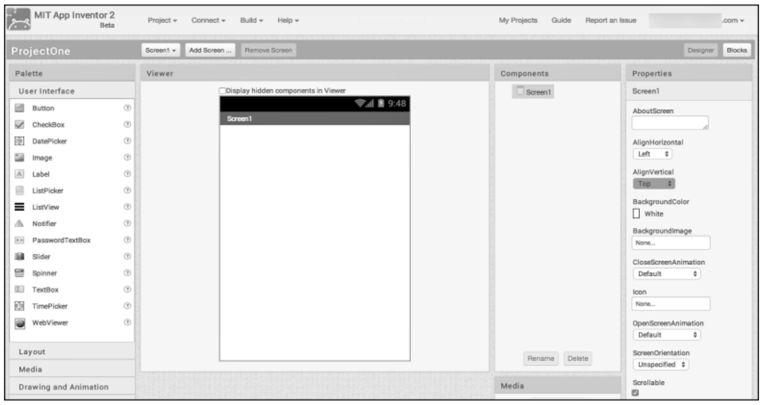
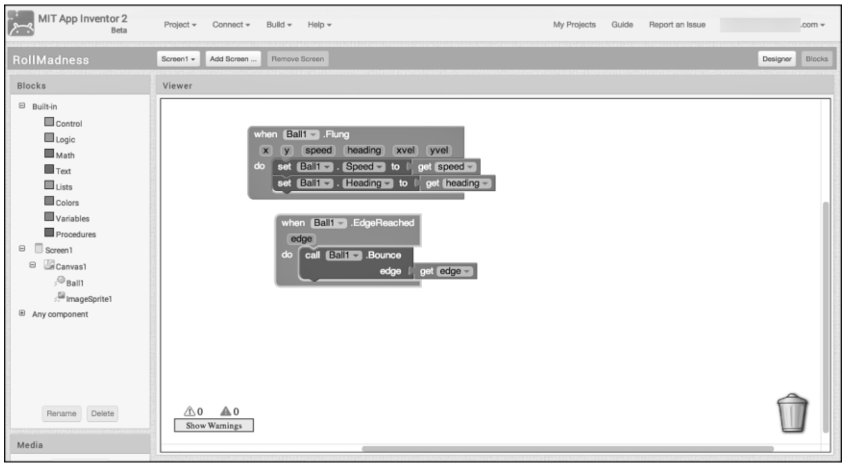


Fig. 4.8 MIT App Inventor Designer Window

4.2.3 Blocks editor

The Blocks Editor is where an app's behavior is programmed. Here, the commands that do the work of the app will be added.

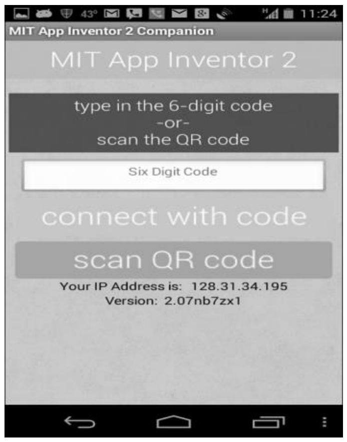


```
when Ball1 .Flung
  x  y  speed  heading  xvel  yvel
do  set Ball1 . Speed to  get speed
    set Ball1 . Heading to  get heading

when Ball1 .EdgeReached
  edge
do  call Ball1 .Bounce
              edge  get edge
```

Fig. 4.9 MIT App Inventor Block Editor Window

4.2.4 The AI2 companion app

App Inventor has a useful tool for continuously seeing the app in real time on an Android device during each step of the development process. When an app is being built, the computer and Android device must be connected to the same wireless network (the desktop machine might have a wired connection).



type in the 6-digit code
-or-
scan the QR code

Your IP Address is: 128.31.34.195
Version: 2.07nb7zx1

Fig.4.10 Android Device Homepage

One can then type in a six-digit code or scan the QR code with ones device, using the App Inventor app. Doing so brings up a live view of in the application. As elements are added to it, with the MIT App Inventor software, those changes are reflected in real time on your device.

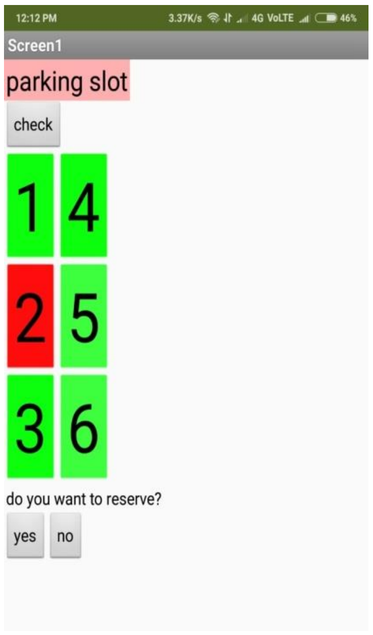


Fig. 4.11 Displaying Empty and Occupied Slots

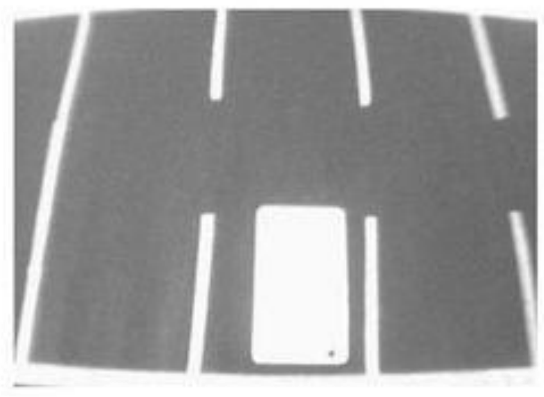
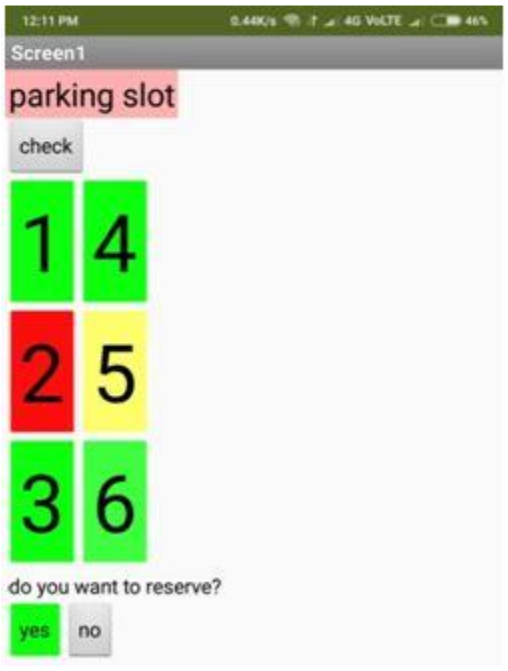


Fig. 4.12 Captured Image

When the user clicks check button , the app displays the occupied and empty slots . In the Fig. 4.11, the 2nd slot is occupied and the remaining slots are empty.



🟩 Empty slots

🟥 Occupied slots

🟨 Reserved slots

Fig. 4.13 Displays the Reserved Slots

The Fig. 4.13 depicts the reserved slots. Here the user has reserved 5th slot.

CHAPTER 5

RESULTS AND FUTURE DISCUSSION

5.1 Conclusion:

The ease of parking system is quite a challenge in modern days. Since the advent of industrialized cities, number of cars has been increasing and day by day people are facing bigger trouble while trying to manage their cars into a parking lot. This scenario of parking crisis gives rise to new solutions with the help of Internet of things (IOT) thus managing car parking systems. Our paper addresses the crisis of car parking across a remote city and comes out with an IoT based assistant mobile application system. The proposed project provides real time information of a car parking lot and is able to coordinate with the mobile application thus giving user the information of the empty parking slots.

5.2 Limitations:

- Only limited surface is considered for vacant slot parking, challenges will be greater if larger parking area is considered.
- The problem with USB camera is that the angle of capturing is difficult for accurate positioning.
- If a person or an object is placed in a slot, due to filtering issues, error may occur. So noises should be filtered perfectly.

5.3 Future Scope:

- **Navigation to the nearest parking area:** The user can be navigated to the nearest parking area from his current location.
- **Multi-Platform application:** The Android application that has been developed can re-built in a multi-platform structure so that iPhone users can also use it.
- **Cost estimation:** The cost can be estimated based on the duration for which the car is parked and the information can be given to the user.

APPENDICES

MATLAB CODE

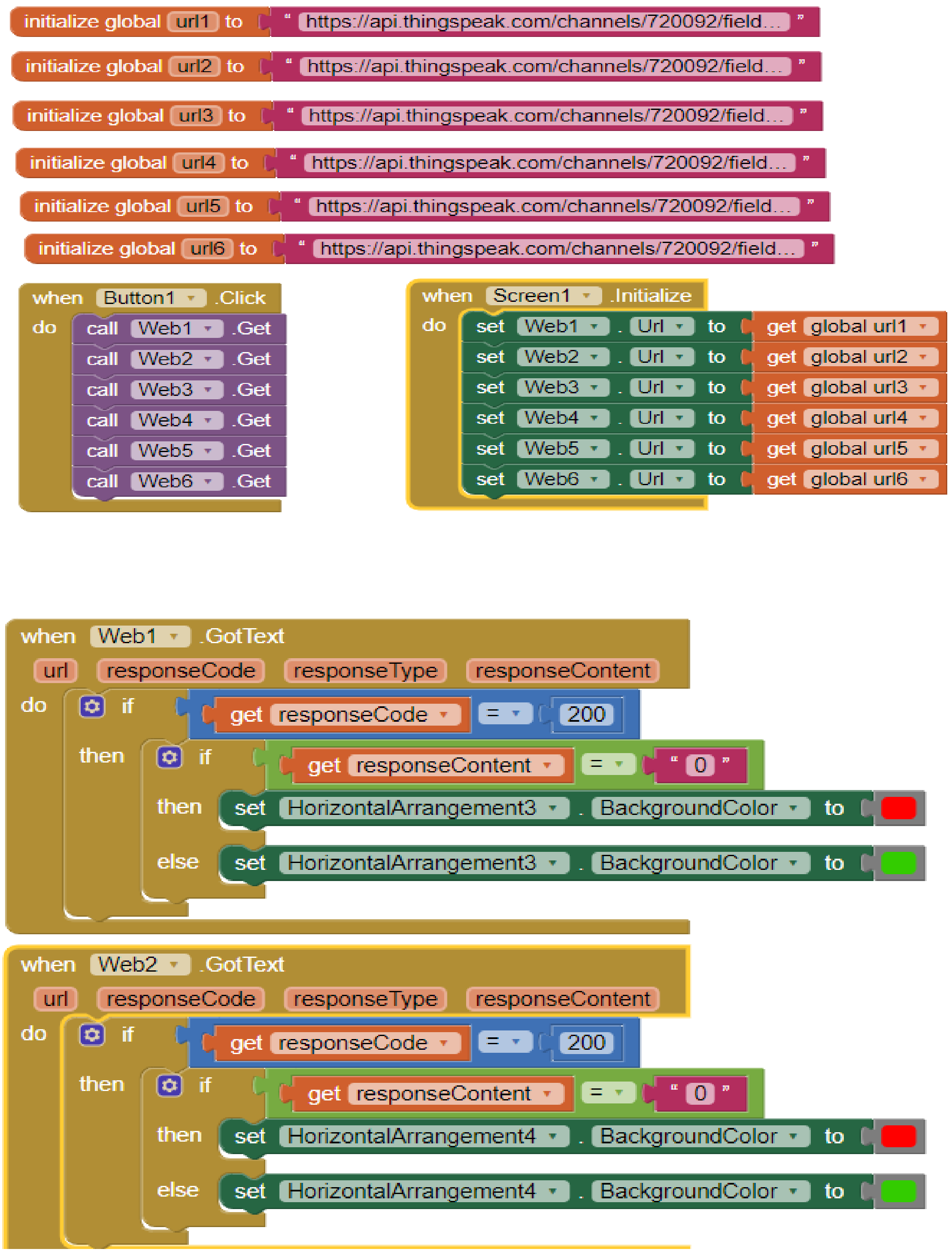
```
%%
% SLOT DETECTION PROJECT
clc;
% Insert Camera capture function
% Converting to Gray Scale
ref_img=imread('ref.jpg'); % Empty slots Image as reference
ref_img=rgb2gray(ref_img);
cap_img=imread('2.jpg'); % Captured Image
cap_img=rgb2gray(cap_img);
[rr,rc]=size(ref_img);
[cr,cc]=size(cap_img);
% Detecting Edges - im1
im1=edge(cap_img);
sel=strel('disk',1);
im1=imdilate(im1,sel);

% Cropping the image Need to tune according to the setup
c_ref1=cropimg(cap_img);
r_ref1=cropimg(ref_img);
figure(1)
imshow(c_ref1);
figure(2)
imshow(r_ref1);
```

```
%%
% Changes according to layout
% Slot Coordinates [col_minrow_min width height]
% No. of slots
nos=6;% slotco =[103.0000   14.0000  217.0000  111.0000;
%     340.0000    5.0000  208.0000  129.0000;
%     573.0000    8.0000  180.0000  130.0000;
%     55.0000  296.0000  241.0000  167.0000;
%     330.0000  281.0000  249.0000  183.0000;
%     595.0000  278.0000  235.0000  183.0000];
slotco=[44 4 69 68;
    123 2 74 72;
    203 5 68 70;
    22 130 78 96;
    116 129 87 102;
    211 129 80 97];

slotco1=[ 22 130 78 96;
    116 129 87 102;
    211 129 80 97;
    203 5 68 70;
    123 2 74 72;
    44 4 69 68];
% Creating reference data
```

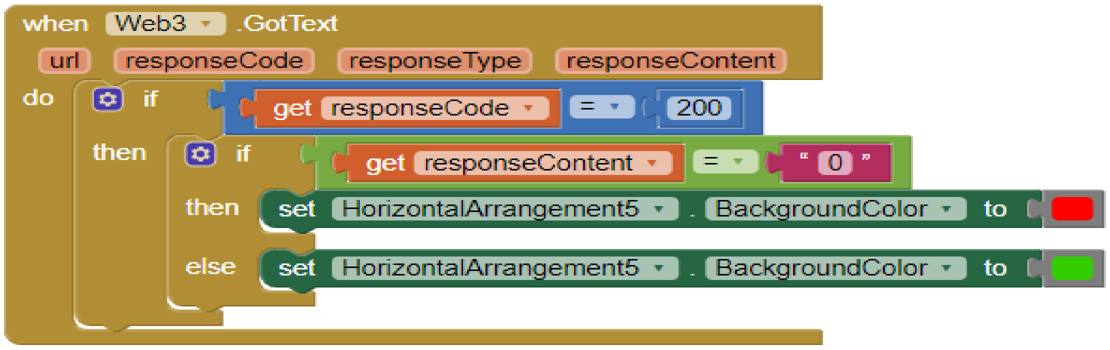
```
for i=1:1:nos
    slot=slotco1(i,:);
slot_img=slotdetect(r_ref1,slot);
   % hist_img=histogram(slot_img);
   im1=edge(slot_img);
sel=strel('disk',1);
im1=imdilate(im1,sel);
figure(4);
subplot(2,3,i);
imshow(im1);
refdata(i)=std(std(im1));
meandata(i)=mean(mean(im1));
vardata(i)=var(var(im1));
normdata(i)=sum(sum(im1));
rdata=(refdata+meandata+vardata+normdata)/2;
end

 for i=1:1:nos
    slot=slotco1(i,:);
slot_img=slotdetect(c_ref1,slot);
   % hist_img=histogram(slot_img);
   im1=edge(slot_img);
sel=strel('disk',1);
im1=imdilate(im1,sel);
figure(3);
```

```
subplot(2,3,i);
imshow(im1);
capdata(i)=std(std(im1));
cmeandata(i)=mean(mean(im1));
cvardata(i)=var(var(im1));
cnormdata(i)=sum(sum(im1));
cdata=(capdata+cmeandata+cvardata+cnormdata)/2;
 end
val=cdata-rdata;
for i=1:1:nos
    if val(i)<0
val(i)=val(i).*(-1);
    end
end
val=val;
[osr,osc]=find(val<100);
fprintf('Empty Slot: %d\n',osc);
a=val>100;
disp(a);
thingSpeakWrite(720092,'Fields',[1,2,3,4,5,6],'Values',a,'WriteKey','GFWSONWV
SDA816JG');
x =
thingSpeakRead(720092,'Fields',[1,2,3,4,5,6],'ReadKey','P87JMMNWBRTB780G'
,'OutputFormat','table');
```

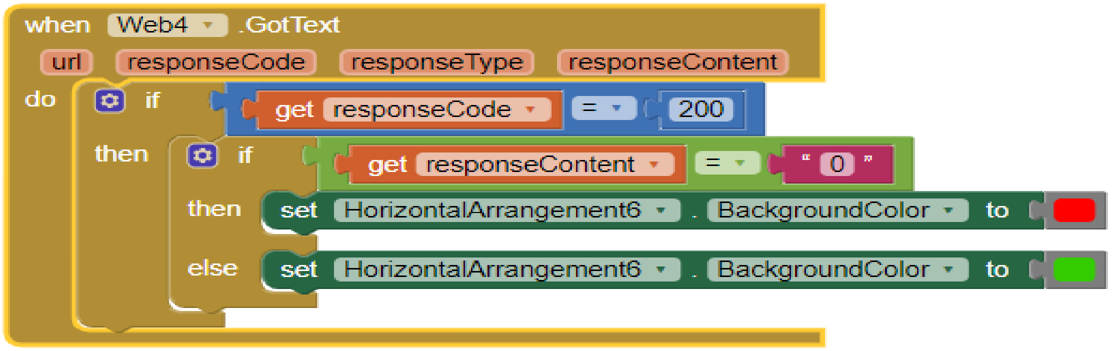
4.2.8 MIT app inventor block editor window:



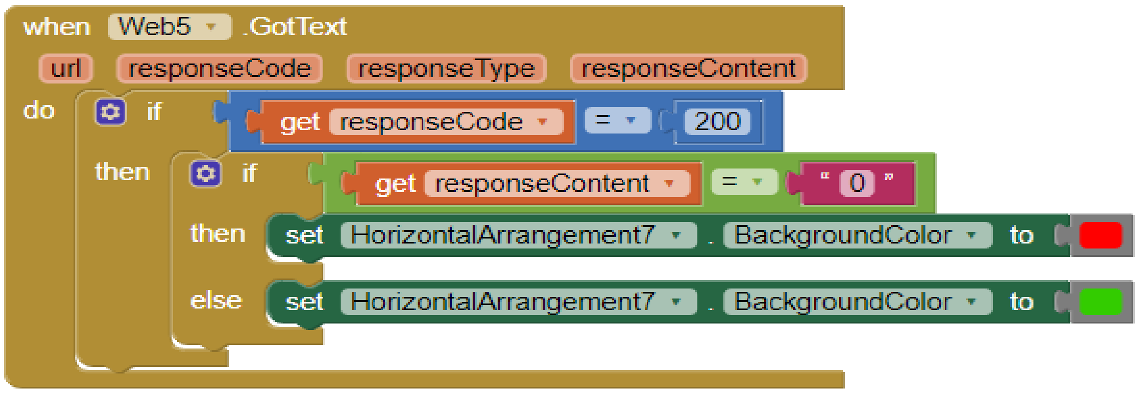
```
initialize global url1 to " https://api.thingspeak.com/channels/720092/field... "
initialize global url2 to " https://api.thingspeak.com/channels/720092/field... "
initialize global url3 to " https://api.thingspeak.com/channels/720092/field... "
initialize global url4 to " https://api.thingspeak.com/channels/720092/field... "
initialize global url5 to " https://api.thingspeak.com/channels/720092/field... "
initialize global url6 to " https://api.thingspeak.com/channels/720092/field... "

when Button1 .Click
do  call Web1 .Get
    call Web2 .Get
    call Web3 .Get
    call Web4 .Get
    call Web5 .Get
    call Web6 .Get

when Screen1 .Initialize
do  set Web1 . Url to get global url1
    set Web2 . Url to get global url2
    set Web3 . Url to get global url3
    set Web4 . Url to get global url4
    set Web5 . Url to get global url5
    set Web6 . Url to get global url6

when Web1 .GotText
  url  responseCode  responseType  responseContent
do  if get responseCode = 200
    then  if get responseContent = " 0 "
          then set HorizontalArrangement3 . BackgroundColor to 🟥
          else set HorizontalArrangement3 . BackgroundColor to 🟩

when Web2 .GotText
  url  responseCode  responseType  responseContent
do  if get responseCode = 200
    then  if get responseContent = " 0 "
          then set HorizontalArrangement4 . BackgroundColor to 🟥
          else set HorizontalArrangement4 . BackgroundColor to 🟩
```



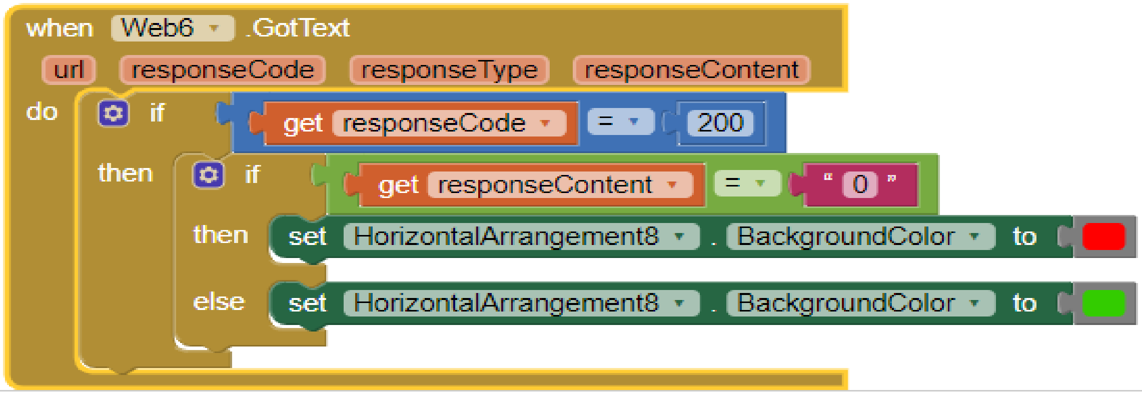
```
when Web3 .GotText
  url  responseCode  responseType  responseContent
do  if  get responseCode = 200
    then  if  get responseContent = " 0 "
          then  set HorizontalArrangement5 . BackgroundColor to 🟥
          else  set HorizontalArrangement5 . BackgroundColor to 🟩
```



```
when Web4 .GotText
  url  responseCode  responseType  responseContent
do  if  get responseCode = 200
    then  if  get responseContent = " 0 "
          then  set HorizontalArrangement6 . BackgroundColor to 🟥
          else  set HorizontalArrangement6 . BackgroundColor to 🟩
```



```
when Web5 .GotText
  url  responseCode  responseType  responseContent
do  if  get responseCode = 200
    then  if  get responseContent = " 0 "
          then  set HorizontalArrangement7 . BackgroundColor to 🟥
          else  set HorizontalArrangement7 . BackgroundColor to 🟩
```



```
when Web6 .GotText
  url  responseCode  responseType  responseContent
do  if  get responseCode = 200
    then  if  get responseContent = " 0 "
          then  set HorizontalArrangement8 . BackgroundColor to 🟥
          else  set HorizontalArrangement8 . BackgroundColor to 🟩
```

REFERENCES

1. https://ieeexplore.ieee.org/document/8391155
2. https://ieeexplore.ieee.org/document/8282631
3. https://www.researchgate.net/publication/303842610_IoT_based_Smart_Parking_System
4. https://create.arduino.cc/projecthub/102513/iot-based-parking-system-b6a947
5. Rafael C. Gonzalez, Richard E.Woods, Steven L. Eddins. (2004)'digital image processing using MATLAB'
6. https://create.arduino.cc/projecthub/KaustubhAgarwal/smart-parking-bdfa99
7. https://in.mathworks.com/discovery/digital-image-processing.html
8. https://in.mathworks.com/help/thingspeak/thingspeakwrite.html
9. http://appinventor.mit.edu/explore/ai2/beginner-videos.html
10. https://www.oreilly.com/library/view/learning-mit-app/9780133799286/ch08lev2sec3.html
11. https://in.mathworks.com/matlabcentral/answers/124487-connecting-arduino-with-matlab-through-bluetooth
12. Anil K. Jain 'Fundamentals Of Digital Image Processing' , Chapter 5.11 pp 163-174.Prentice Hall Inc., prentice hall international edition,1989