Chapter 1

Introduction

The Internet of Things(IoT), which is a popular word in computer science, was originally designed as a means of tracking, monitoring and controlling various devices through the Internet. Now-a-days IoT is the basic foundation of the so called Smart Cities, where a city acts as a network. In this futuristic design, almost every component of the city is connected with each other following some defined protocols defined to regulate the data transmission between them.

The population is heading towards the cities in a vast number in assurance of a better life, health, education as well as better opportunity for every aspect of life. According to International Parking Institute (IPI), 60 percent of the total population will live in cities by 2030, and Green Car Reports (News-feed Reportage Website) estimates that the number of vehicles on the road is about 1.2 billion now and is likely to reach 2 billion by 2035. With an increase in number of vehicles, the requirement for parking is also increasing drastically. The population is exploding but the world is not. We have a limited land resource to solve our parking issues. So the need of a balanced as well as an environment friendly parking system is of the essence.

1.1 Why a new Parking System

Traffic congestion caused by vehicle is an alarming problem at a global scale and it has been growing exponentially. Car parking problem is a major contributor and has been, still a major problem with increasing vehicle size in the luxurious segment and confined parking spaces in urban cities. Searching for a parking space is a routine (and often frustrating) activity for many people in cities around the world. This search burns about one million barrels of the world's oil every day. As the global population continues to urbanize, without a well-planned, convenience-driven retreat from the car these problems will worsen.

According to a report, Smart Parking could result in 2,20,000 gallons of fuels saving till 2030 and approx. 3,00,000 gallons of fuels saved by 2050 , if implemented successfully.[4]

Smart Parking systems typically obtains information about available parking spaces in a particular geographic area and process is real-time to place vehicles at available positions. It involves using low-cost sensors, real-time data collection, and mobile-phone-enabled automated payment systems that allow people to reserve parking in advance or very accurately predict where they will likely find a spot. When deployed as a system, smart parking thus reduces car emissions in urban centers by reducing the need for people to needlessly circle city blocks searching for parking. It also permits cities to carefully manage their parking supply.

Smart parking helps one of the biggest problems on driving in urban areas; finding empty parking spaces and controlling illegal parking.This implies M2M technologies aims rightness/safety as well as convenience.

1.2 What is IoT

The Internet of things (IoT) is the network of physical devices, vehicles, home appliances and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these objects to connect and exchange data. Each thing is uniquely identifiable through its embedded computing system but is able to inter-operate within the existing Internet infrastructure.

The figure of online capable devices increased 31% from 2016 to 8.4 billion in 2017. Experts estimate that the IoT will consist of about 30 billion objects by 2020. It is also estimated that the global market value of IoT will reach $7.1 trillion by 2020.[3]

The IoT allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit in addition to reduced human intervention. When IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, virtual power plants, smart homes, intelligent transportation and smart cities.

"Things", in the IoT sense, can refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, cameras streaming live feeds of wild animals in coastal waters, automobiles with built-in sensors, DNA analysis devices for environmental/food/pathogen monitoring, or field operation devices that assist firefighters in search and rescue operations. Legal scholars suggest regarding "things" as an "inextricable mixture of hardware, software, data and service".[3]

1.3 Challenges faced in a Smart Parking System

A smart parking system is made to assist the drivers to search for a parking place. So the challenges faced by the system are the challenges faced by the drivers everyday. In addition to that, for a system to work properly it requires some maintenance and care. Some of the problems are :

1. Parking Detection and Driver Localization

   The system should be able to detect a free space that is unoccupied and this detection should be done using real-time data. Again driver localization is also necessary so that the user can be directed to the decided parking spot. Proposed architecture relies on the knowledge of real-time parking information, based on which it makes and upgrades allocations for drivers.

   whenever the system starts to make an allocation, it requires the location information of all cars pending for allocation. Based on this information, it estimates the travelling time to the spot to be allocated, and provides driving directions after the allocation

   Vehicle tracking systems combine GPS tracking technology with flexible, advanced mapping and reporting software. A vehicle tracking device is installed on a vehicle which collects and transmits tracking data via a cellular or satellite network. The system receives real-time vehicle tracking updates, including location, direction, speed, idle time, start/stop and more.

2. V2I and I2V Communication

   This is a two-way communication including Vehicle-to-infrastructure (V2I) and Infrastructure-to-vehicle (I2V)communication. In our smart-parking system, V2I communication includes drivers sending their parking requests, providing driver information and confirming reservation to the system. I2V communication involves the system sending allocation results, driving directions, payment details and more, back to vehicles. Cellular networks (CN) are usually applied in V2I and I2V solutions, i.e., drivers interact with the system through their mobile phones.

3. Reservation Guarantee

   Parking reservations are a key feature of the smart-parking system. In order to implement this function, when a parking spot is reserved by the driver, the system must guarantee that this parking space will not be taken by other cars. For off-street parking resources, it is relatively easy to prevent drivers from taking the spots which have been reserved by others. The system can perform

ID checking (with RFID technology) at the gate of a garage or a parking lot. If the driver has a reservation, the gate opens and a spot number will be provided to him. Otherwise, he may be rejected, or allowed to park if there are vacant unreserved parking spaces.

For on-street parking resources, the scheme is more complicated because there is no ID checking capability for on-street parking spaces. Drivers may park in any spot if it is vacant. One method is through wireless technology interfacing a vehicle with hardware that makes a spot accessible only to the driver who has reserved it. Examples include gates, folding barriers and obstacles that emerge from and retract to the ground under a parking spot; these are wirelessly activated by devices on-board vehicles, similar to mechanisms for electronic toll systems.

4. Optimal Allocation

   In general, the system shouldn't decide parking lots based on biased decisions. By biased decisions, we mean that it must take into account for both the primary costs,i.e.,the individual cost and the social cost. Details about these costs will be explained later. But the point is that, while choosing a parking spot the system must consider the current distance of the user from the nearby lots along with the parked vehicle density in each of them.

   Only after that a system can decide the best suitable parking lot for the user. Again as discussed earlier, the decision of the system might be upgraded depending on the real-time data. In that case, if a more efficient parking lot is available, the user will be notified of the upgraded decision. Afterwards he/she may choose whether or not to accept the upgraded decision.

5. Maintenance of the System

   The architecture is based on microprocessors and sensors that are deployed in the paring lots. The maintenance and reliability of these electronic devices is also a challenge. The working duration of these devices will be relatively long, so proper care and maintenance should be done in regular intervals. A single failure or miscalculation in any of the components may provide a faulty data based on which the decisions made by the system might prove itself wrong.

1.4 Features of a IoT Based Parking System

Smart Parking involves the use of low cost sensors, real-time data and applications that allow users to monitor available and unavailable parking spots. The goal is to automate and decrease time spent manually searching for the optimal parking floor, spot and even lot. Some top benefits of a parking solution are -

(1) Optimized Parking: Users find the best spot available, saving time, resources and effort. The parking lot fills up efficiently and space can be utilized properly by commercial and corporate entities.

(2) Reduced Traffic: Traffic flow increases as fewer cars are required to drive around searching for available parking spots.

(3) Reduced Pollution: Searching for parking burns around 8.37 million gallons of gasoline per day. An optimal parking solution will significantly decrease driving time, thus lowering the amount of daily vehicle emissions and ultimately reducing global environmental footprints.

(4) Increased Safety: Parking lot employees and security guards contains real-time lot data that can help prevent parking violations and suspicious activity. Also decreased spot-searching traffic on the streets can reduce accidents caused by the distractions of searching for parking.

(5) Real-Time Data and Trend Insight: Over time, a smart parking solution can produce data that uncovers correlations and trends of users and lots. These trends can prove to be invaluable to lot owners as to how to make adjustment and improvements to drivers.

1.5 The Problem

As mentioned earlier, there are many wastage instances of natural resources due to the cruising vehicles. Again the carbon footprint on natural biomes is also a considerable factor. Hence, an optimal strategy to find a parking spot can relieve traffic congestion, reduce air pollution and enhance driving comfortability.

Basically the parking optimization system is based on two costs.

1. Individual Cost
   The penalty on an individual due to the parking problem. Includes Gas cost, wastage of time, exhaustion, unnecessary delay due to traffic congestion etc.

2. Social Cost
   The penalty on our natural biomes due to the parking problem. Includes air pollution, carbon footprint, noise pollution etc.

1.6 Scope of the Problem

The scope of this problem is worldwide. Because where there are vehicles, there is parking and there is congestion. But for sake of simplicity, the problem domain is divided into two categories.

(A) Requirement in Global Scenario

- There are estimated 600 million cars in the world, a figure that's rapidly growing, especially in china and other emerging market, shows us the vast population of cars and the necessity of parking systems.
- 30% of traffic congestion is caused due to vehicles in search for parking.
- A waste of 8.37 million gallons of gasoline and over 129000tons of CO_2 emission are taking place due to improper parking systems and congestion.

(B) Requirement in Indian Scenario

- 20 minutes are wasted by drivers in India in search for parking.
- 43% of total driving time is spent looking for a parking spot.
- A recent survey shows that during rush hours in most big cities, the traffic generated by cars searching for parking spaces takes up to 40% of the total traffic.

1.7 Summary

In this chapter, we have discussed about what is IoT, what this project is about, what are the challenges we have to face to solve this problem and the targeted audience of this problem.

Chapter 2

Background & Literature Survey

In this thesis, we mainly focus on a management system that assists drivers to find parking spaces in a specific parking district, and satisfies the needs of both parking providers and drivers. In addition, an important goal of the system is to reduce the traffic searching for parking, hence reduce energy consumption and air pollution. In this chapter, we review background on smart parking systems, including the performance metrics, existing solutions and challenges.

2.1 Performance Metrics

To evaluate the performance of a parking management system, we introduce the following metrics, which reflect the needs of involved parties, and our concerns on traffic congestion and environmental protection.

- Walking Distance: Walking distance is defined as the distance from a driver's selected parking space to the destination. It is an important factor for a driver to determine where to park. Usually, a driver wants to park as close to the destination as possible if his budget permits. Therefore, the walking distance indicates the satisfaction of drivers.
- Parking Revenue: Regardless of whether a parking lot is privately owned or municipally owned, parking revenue represents the benefit to the parking providers. Since multiple parties are involved in this system, our design does not aim to maximize the parking revenue for service providers only, but allows them to obtain profits at reasonable level.
- Service Differentiation: Usually a driver wishes to pay as little as possible, and has a certain budget for parking. In a crowded area, the parking resource is limited. To alleviate the contention on parking resource and maintain reasonable parking revenue for service providers, the management system should differentiate the drivers according to their budget and need.

- Traffic Searching for Parking: The traffic generated by drivers searching for parking is not negligible and reflects the social welfare. Hence, an efficient parking guidance system should efficiently reduce the traffic searching for parking. Also, reducing the amount of searching time for parking is desired by drivers.[2]

We investigate performance of the proposed smart parking system using these performance metrics.

2.2 State-of-the-art Parking Management Solutions

We now introduce several existing parking guidance approaches and show their limitations. We simulate the parking system performance under specified parking architecture and show results in Chapter 4.

1. Blind Search: Blind searching is adopted by users when no parking information is available. So drivers search parking spaces randomly within a certain distance to their destination. If a driver gets an available space, the driver will stop searching; otherwise, the driver will continuously searching in the neighbouring parking lots until he finds space. In this case, there is no control signal to guide driver's behaviour.

2. Parking Information Sharing (PIS): This mechanism represents the current state of the smart parking system design. When a driver obtains the parking availability information near his or her destination, the driver will know if the desired parking lot has available spaces. Hence, individual drivers make the decision according to the parking availability information. If a parking lot has a very few parking spaces available in busy hours, it is likely that more drivers struggle for less parking spaces. This phenomenon is called “multiple-car-chasing-single-space”, which may cause severe congestion.[2]

3. Buffed PIS (BPIS): To alleviate the “multiple-car-chase-single-slot” phenomenon, some designers of smart parking systems have devised a solution to leave a buffer when publishing the live availability information. Therefore, if a parking lot has less unoccupied spaces than a threshold, the system will show that the parking lot is fully occupied. But it is difficult to determine the threshold for the buffer. If the buffer is too small, the problem of “multiple-car- chase-single-space” will not be eliminated. If it is too large, the utilization of parking spaces will be low.

2.3 Challenges

Given the design objectives of smart parking systems that requires the coordination among multiple parties, we summarize the main design considerations as follows:

1. Performance v.s. Overhead: In order to complete parking space reservation, drivers have to communicate with the system. However, there is a trade off between the overhead to generate and convey control signals to drivers v.s. performance optimality. For example, if the system computes the fine-grained control signals for individual drivers and adopts unicast to inform the drivers about parking system status and instructions, the overhead will be very large. Hence, the parking management system generates a uniform control signal (e.g., the utilization of parking lot and parking price), and broadcasts the control signal to drivers.

2. Trade-off Between Benefits to Drivers and Service Providers: Multiple parties (drivers and service providers) are involved in the parking system operation. The state of the system depends on their interaction with each other. To balance the needs of involved parties, we use parking price as the control signal to coordinate the involved parties.

3. Differentiated Service for Large Scale Autonomous Drivers: Thousands of drivers make parking decision autonomously. They have different needs and budgets for parking and their interpretation of parking information is different. Providing differentiated service for drivers is important to satisfy individual users. In this sense, the service quality is determined by parking price.

2.4 Related Work

Here we discuss about some of the existing technologies that are being used for the partial fulfilment of realizing the term smart parking. The technologies separately serve the benefits of a smart parking system, but none of them provides the complete benefits of it. In other words, an advanced parking system must contain all the below mentioned technologies but as one.

2.4.1 Parking Guidance Information System

The implementation of parking guidance and information system (PGIS) encompasses two major categories. The PGIS can either include the entire city area or function only within the car park facility. Setting aside the differences both the PGIS implemented in many major cities in Europe, Japan, the United kingdom and the United States offer similar advantages. Both provides information which aids the decision making process of the drivers in reaching their destination location and aids them in locating a vacant parking space within the car park facility.[4] The city wide PGIS is indeed helpful in assisting drivers to car park with vacant parking spaces via the information occupancy status for various car parks around the city as well as other relevant information. On the other hand, guidance in locating the vacant parking space within the car park is ultimately provided by PGIS implemented within the car park.

PGIS can be summarized as consisting of 4 major components: namely, information disseminating mechanism, information gathering mechanism, control center and telecommunication networks. Static/dynamic Variable Message Signs (VMS) have been used in providing drivers with direction either on the road or within the car park.For guidance on the road; various implementation methods can be adopted. For example, the system in Shinjuku and Pittsburgh, Pennsylvania segregates the city area into color coded areas for in providing guidance. The PGIS in Pittsburgh, Pennsylvania also functions in directing drivers to special attraction in the area. Meanwhile, in Yokohama, Japan, the city is divided into four zones where by the information specificity increases with each zone that the driver cross to arrive at the destination location. Additional information on traffic-flow provided by the Aichi prefecture police headquarters traffic control center and Japan highway public cooperation Nagoya Department is also provide by the system implemented in Toyota, Japan.

Mobile phones can also be used for guidance based on the research which utilizes Global Positioning System (GPS) for vehicle detection. A map of the driver's current position based on the GPS data along with the status of three of the nearby car park are sent to their mobile phones based on the patron's current location. Besides that the parking guidance system developed based on web and GIS technology are able to disseminate information to the users via internet, mobile phones and/or PDA. The guidance system can be with the conventional parking management system as well. In order to guide the patrons effectively, the car park map is printed on the parking ticket equipped with Radio Frequency Identification (RFID) tax for guidance so that patrons can locate the assigned parking slot with ease. There are also no worries about forgetting the location of the assigned parking slot during exit.

Vehicle detection sensors are commonly installed at entrances, exits and /or individual parking space to detect vehicle occupancy. Indicator lights integrated with sensors are also sometimes installed at every Individual parking space within the parking facility. The occupancy status detected by the sensors can either be occupancy of each individual parking space or in terms of vehicle counts in the car park depending on the installation of the sensors. Moving on, the control center gathers and processes the traffic and occupancy information as well as controls the display of information for drivers whereas the telecommunication network facilitates the transfer of information among the other three modules. With the advent of advanced of technologies, the implementation of devices such as micro-controller and Filled Programmable Gate Array (FPGA) are incorporated for faster information processing. Not only that, the telecommunication network no longer dependent on conventional electrical wiring but wireless technologies are able to be utilized.

2.4.2 Transit Based Information System

The functionality of transit based information system implemented in countries such as France, Germany, Ireland, Japan, Switzerland, the United Kingdom and the United State is actually similar to PGIS. The difference exists in the fact the Transit Based Information System concentrates on guiding user to park-and-ride facilities. It provides real-time information on the status of each car park and public transportation such as the schedules and traffic condition to the public. The additional information provided enables the patrons to plan of transit in advance without getting into any inconvenience. Among its benefits includes increase in the utilization of public transportation as the primary means of transportation as they can leave their vehicle in the car park and switch to public transportation with ease. This will indirectly lead to an increase in the transit revenue.

No doubt, for the transit based information system to achieve success in its implementation, proper planning must be conducted. This is especially true in selecting the location for the park-and-ride car parks that maximize transit whereby the concept of catchment area/commuter sheds are often used. In the network flow-based technique introduced, it improves on the conventional spatial model used in determining the park-and-ride facility location by taking into consideration the traffic flow and works in reducing the vehicle during the beginning stage of the journey.[3]

There have been many researches centred upon using Geographic Information System (GIS). Among them are the research conducted for sighting park-and-ride car parks in Columbus, Ohio. It incorporated multi-objective spatial optimization model in locating the park-and-ride facilities while considering numerous objectives and constraints as well as taking into consideration the existing system. Since it made no assumption on user demands, Horner and Grubesic used Principal Component Analysis (PCA) in representing the index of user demands which will be converted to demand points when coupled with information obtained via Geographic Information System (GIS). Subsequently, additional calculation conducted by Horner and Groves takes into account various other factors which includes; geographical, network, travel time from demand points to the location of the park-and-ride facilities and the constraints of computer shed shapes are performed in determining the location and commuter shed area for the park-and-ride lots.

2.4.3 Smart Payment System

The smart payment system is implemented in the effort to overcome the limitation of the conventional payment methods by revamping the payment method via parking meter and introduce new technologies. This is because the conventional method causes delay and inconvenience for the patrons as they have to deal with cash. It also reduces maintenance and staffing requirement for payment handling purposes as well as traffic control. In general, the smart payment system implemented in places such as Finland, Italy, London and United States consists of contact method, contact-less method and mobile devices. While the contact method involves the use of smart cards, debit cards and credit cards, the contact-less method involves the use of contact-less cards, mobile devices as well as Automated Vehicle Identification (AVI) tag whereby RFID technologies are utilized. As contact method requires contact of the cards with parking meter or payment machines in the facility, the latter offers more convenience to the patrons.

Parking meters have now been improvised with technologies which revolutionize the payment system via implementing various improvements such as the acceptance of various types of cards such as credit card, debit cards band smart cards. It also incorporates other technologies such as having solar power source and wireless connectivity. The Photo Violation Meter (Photo Violation Technologies) which caters for various types of payment methods uses ground sensors in detecting vehicle presence. Most importantly, technologies such as Wi-Fi connectivity, together with its ability in handling payment of fines and taking photos of vehicles which violates parking regulations for evidence are also incorporated. Personal parking meters which are essentially placed in the vehicle have also been introduced in Buffalo, New York and Aspen, Colorado; after test studies have been conducted.

The incorporation of RFID technologies for making payments were implemented in commercial systems such as Mobipower Ltd., which utilized RFID-based cellular technology and EZPass Systems, have also developed payment system via RFID for car parks and toll facilities. Similarly between the two systems exists in the requirement for placement of transponder unit in the vehicle. Moving on, the implementation of mobile devices such as mobile phones and PDA are normally seen to incorporate other devices such as parking meters and cards. For some systems such as those implemented in Groningen, Netherlands and Oulu, Finland, prior registration via internet is required. As mobile phones are utilized, the system implemented in Oulu Finland also has the capability of sending Short Message Service (SMS) notification to remind the patrons that the time is almost due and allow them to settle the payment for the additional time extension required. The main concern hindering the implementation of the Smart Payment System would have to be skepticism on the privacy and security issues. This is due to the fact that confidential data of the patrons such as personal information and probably account information are being dealt with which are highly confidential. With the emergence of various threats, it is justifiable to be worried. In RFID implementation alone, exploits, malwares and worms, as well as attacks such as sniffing, spoofing, replay attack and denial of services are just a fraction of it. Of course, methods have been developed in securing the data and overcoming the threats as it discovered ranging from the cryptography, detection and evasion as well as temporary deactivation which are constantly improved from the conventional method implemented which dates back to World War II.

2.4.4 E-Parking

E-parking provides an alternative for patrons to enquire the availability and reserve a parking space at their desired parking facility to ensure the availability of vacant car park space when they arrive at the parking facility. The system can be accessed via numerous methods such as SMS or through the internet. Some of the additional benefits of using the E-parking system aside from those collectively gained by smart parking system are that it can be extended easily to incorporate the payment mechanism of smart payment system whereby payments by the patrons are made hassle free using the technologies discussed previously. Customized information can also be provided to the patrons either before or during their trip to the car park.

In a study, reservations can be made through the utilization of mobile phones or any reservation centres convenient to the patrons. On the other hand, the study by Hodel and Cong revealed options of using the internet via Wireless Application Protocol (WAP) enabled mobile phones Personal Digital Assistants (PDAs) and even conventional computer in addition to SMS service for the drivers in accessing the information as well as making reservations. Later the implementation was taken a step further by incorporating fuzzy logic in decision making where by the parking reservation request can either accepted or rejected. It also facilitates the enforcement of tariff classes to enable the maximization of revenue for car park operators. The system is one of the systems integrating PGIS with E-parking system, where the patrons are able to reserve parking slots after reviewing the status of the car park and its proximity to the patron's current location. Thus far, numerous systems have been implemented online for city wide implementation, university campuses as well as complexes Among the examples of companies involved in the development of E-parking system are companies such as Parking Carma, Click and Park and City and Suburban Parking Ltd.,. As the E-parking systems implemented online can also be considered as

E-commerce applications, it has been proposed that the Unified Modelling Language (UML) is employed in modelling the system's performance. This can be achieved by converting the class diagram and sequence diagram to an execution diagram via an intermediary actor-event graph and combining it with other necessary information pertinent to the system.

As observed, there are many different implementation methods that can be enforced by incorporating various technologies. Besides the difference in reservation method, different reservation types can also be enforced, whereby patrons can choose not to declare the exit time and park for an indefinite amount of time. To gain access into the car park, printed receipts, permits or passes are utilized by the patrons. More sophisticated implementation proposed which requires the use of smart cards or magnetic cards and Bluetooth are also implemented in granting access to the patrons. It can also be implemented together with Smart Payment System payment schemes such as cards, pre and post paid methods as well as M-Payment are utilized.

2.4.5 Automated Parking

It involves the use of computer controlled mechanism, which allows patrons to drive up to the bay, lock the cars and let the machines automatically place the vehicle in the allocated space. This type of car park involves maximum utilization of space as it is machine controlled unlike conventional car park where space is needed for navigation of vehicle with in the car park. Among its benefits is that the implementations, works great in the location where there are limited room for extension according to their structure. Besides that automated parking also offers efficiency in car storing as it allows car stacking and the patron does not even need to go into the car park which indirectly provides extra safety measure which covers both the vehicles and patrons.

Among the automated parking system reviewed in countries like Japan, Canada, USA as well as the commercial system developed by the companies like automatic parking system, robotic parking and Fata SKYPARKS, it generally utilizes computer controlled mechanism in placing the vehicles in the storage bay within the parking facility. Automated parking can also be implemented in conventional car park via additional equipments installed such as developed by Fata SKYPARKS. There are many variations whereby the automated parking system can be implemented, from the design of car park structure to the working of computer controlled docks as well as placement of vehicles where by user participation are sometime required . The safety features are geared towards the vehicle whereby it is important to ensure that the vehicle remains safe and undamaged with all the handling by the computer controlled mechanism. Research by Mathijssen and Preterious introduced a 3 level software design that includes Logical Layer (LL) , Hardwire Abstraction Layer(HAL) , Safety

Layer(SL) to ensure correct and efficient storage of vehicles in a safe manner. As the car park facility is designed with the conveyor belts, Rotatable lifts and shuttles, it has to be coordinated to ensure successful and safe placement and retrieval of the vehicle.

2.5 Summary

This chapter was all about the what has been done in our country in both state as well as national level to tackle this parking problem. We also discussed about some performance metrics and challenges faced in economical and technical designs of this system.

Chapter 3

Proposed Architecture and Solution

In this chapter, we present the proposed reservation based Smart Parking Management Solution, which implements the Iot technology combined with a reservation policy.

3.1 System Overview

The system is derived from the idea of IoT. The system uses IoT technology combined with cloud storages to monitor car parks. Using sensor networks, the percentage of free parking spaces in each car park is calculated. The use of IR sensors facilitates implementation of a large-scale system at low cost. The system provides a mechanism to prevent disputes in the car park and helps minimize wasted time in looking for a parking space.

After logging into the system, the user can choose a suitable parking space. Information on the selected parking location will be confirmed to the user via notification. Then, the system updates the status of the parking space to “pending” during which time the system will not allow other users to reserve it. If after a certain period of pending time the system determines that no car is parked in that space, then it changes the status to “available.” The system will update the status from the sensor node (the status of car park spaces) when a new car joins in the system. Therefore, the status of the overall parking system is always updated in real time. The system will help plot the parking time for each parking space in real time and can support the business with hourly parking charges.

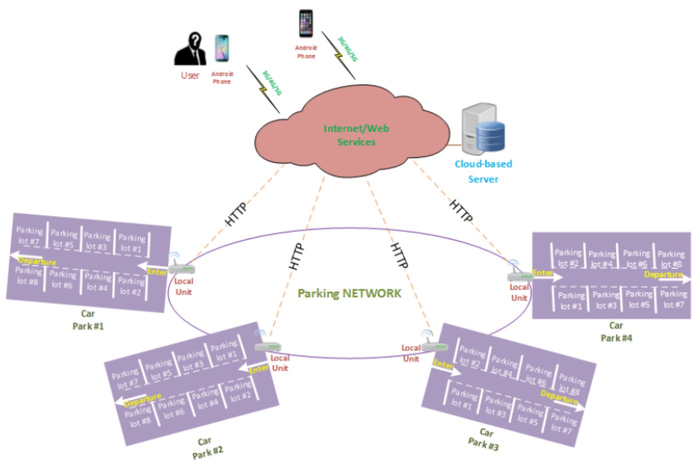


Figure 3.1: Architecture of the proposed System

3.2 System Architecture

The system consists of some elements which are explained below in detail.

3.2.1 Cloud Based Database

This is a Web entity that stores the resource information provided by local units located at each car park. The system allows a driver to search and find information on parking spaces from any place without the need to access the local server node by directly accessing the cloud-based server.

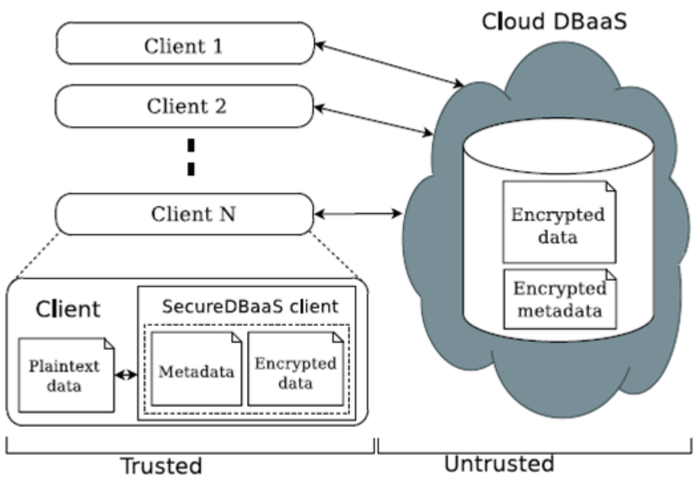


Figure 3.2: Working of a cloud database

3.2.2 Local Unit

This unit is located in each car park(s) and calculates the information about each parking space, as shown. The local unit includes the following:

1. **Control Unit:** This is an Arduino module, which is connected to the IR sensors. If the IR sensors detect the presence of a vehicle,the Arduino module will control the opening of the door for the vehicle to enter.It is also connected to each and every sensor that are deployed in each of the parking lots and it connects with the cloud server through an Internet connection to transfers data from the local car park to the cloud database.

2. **IR Sensors:** It consists of an IR LED, a photodiode, a potentiometer, an IC Operational amplifier and an LED.

   IR LED emits infrared light. The Photodiode detects the infrared light. An IC Op–Amp is used as a voltage comparator. The potentiometer is used to calibrate the output of the sensor according to the requirement.

   When the light emitted by the IR LED is incident on the photodiode after hitting an object, the resistance of the photodiode falls down from a huge value. One of the input of the Op–Amp is at threshold value set by the potentiometer. The other input to the op-amp is from the photodiode's series resistor. When the incident radiation is more on the photodiode, the voltage drop across the series resistor will be high. In the IC, both the threshold voltage and the voltage across the series resistor are compared. If the voltage across the resistor series to photodiode is greater than that of the threshold voltage, the output of the IC Op-Amp is high. As the output of the IC is connected to an LED, it lightens up. The threshold voltage can be adjusted by adjusting the potentiometer depending on the environmental conditions.

3. **Screen:** This can be used as a visual aid or as a Variable Message Sign(VMS), which displays information on the capacity of the local car park, the total current percentage of free spaces, the status of each slot, the user identification when entering, and a mini map of the local car park.

3.2.3 Software Client

This is an application software system, running on JAVA compiler and Windows Operating System. The users can access it through their smartphones and use it to reserve parking spaces. The users access the system via mobile connections or via computers.

3.3 Network Architecture

Here the design of the network is explained along with the base formula to be used in the network.

3.3.1 Parking Network

We use the car park network (CPN) architecture as the infrastructure/ backbone.This type of parking network includes IR sensors; the basic sensing elements which form the infrastructure for the proposed parking system. The CPN infrastructure/backbone can be built to allow sensor networks to connect using wireless radio technologies through routers, because in real life scenarios, the parking lots will be separated by greater distances. So wired connections will prove inefficient in those cases.

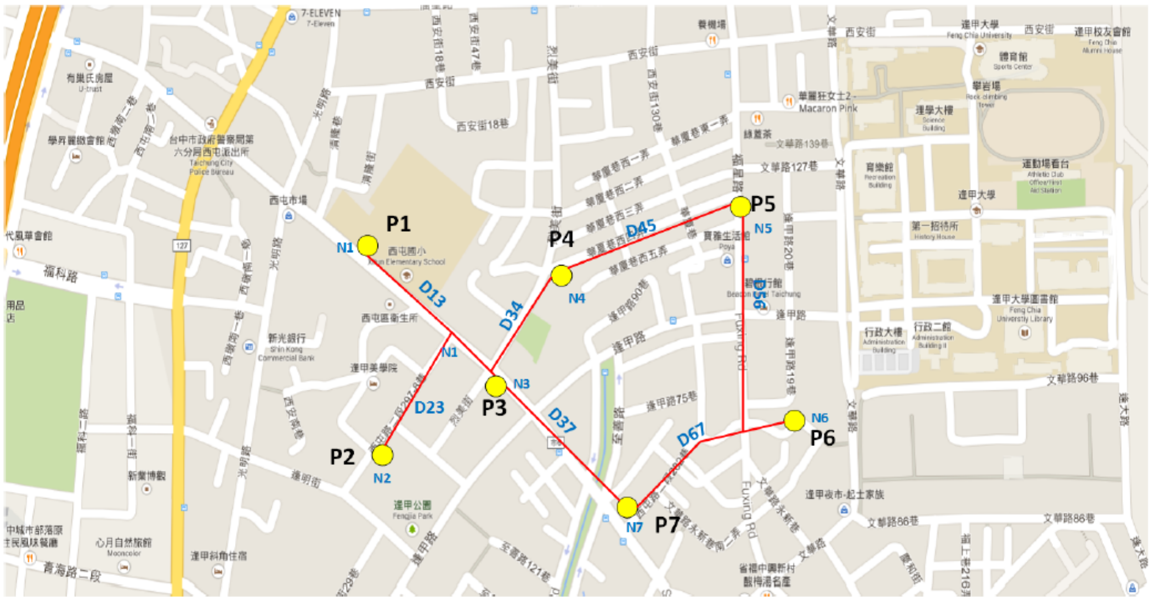


Figure 3.3: A real-life scenario

Routers can be connected to the Internet through gateway functionality. This approach, also referred to as infrastructure meshing, provides the backbone for conventional clients and enables integration of CPNs with existing WSNs through gateway/bridge functionalities in the routers. Conventional clients with the same radio technologies as the routers can directly communicate with the routers.

We have assumed that each car park is a node in a CPN. The deployment network in a simulated environment is shown in Fig. 3.4, where each car park is labelled. For the sake of simplicity, we have assumed a rather small network, where as actual scope of this network can be quite large and complex.

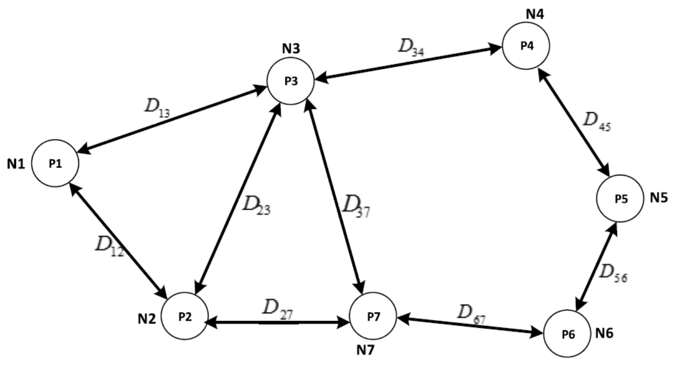


Figure 3.4: graphical representation

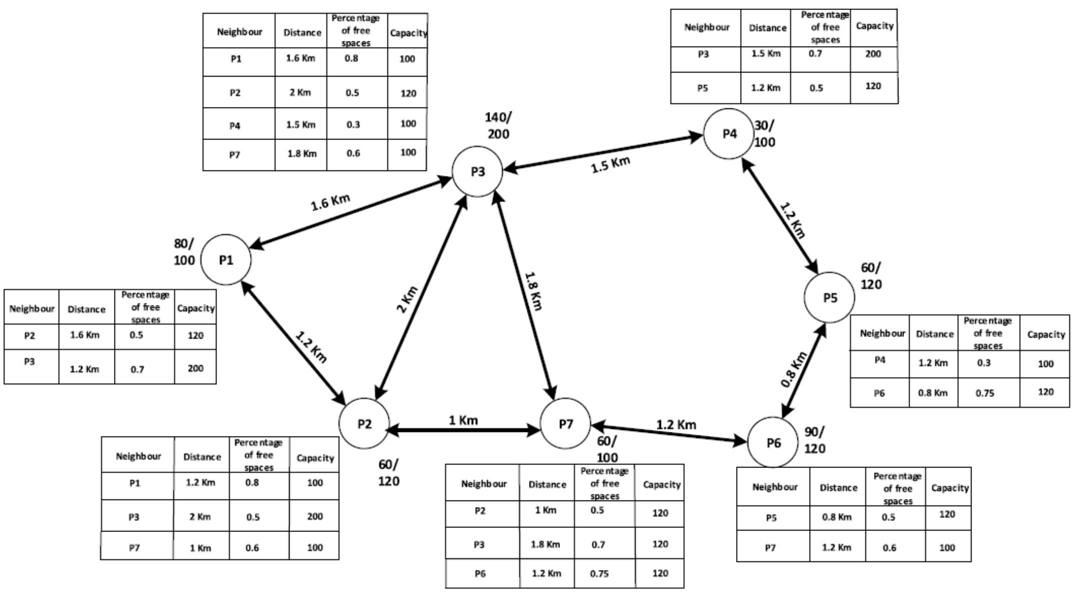
Here the assumption is :

- P_1 is car park number 1; N_1 is the total parking spaces in P_1.
- P_2 is car park number 2; N_2 is the total parking spaces in P_2.
- P_n is car park number n; N_n is the total parking spaces in P_n.

The total capacity of the system is N = N_1+N_2+N_3+N_4+.....+N_n.

D is the real distance between two nodes in the network. D_{ij} is the distance between nodes P_i and P_j.

Each node has a neighbour table to maintain information on the current status of the network and a queue with predefined length. The neighbour table for each node contains information on the neighbouring nodes directly linked to it. On the other hand, the queue is used to control the number of vehicles forwarded to the node, which aims to prevent overloading in the number of vehicles beyond the capacity of the node. In our proposed system, each node will broadcast a message to its neighbouring nodes after a new vehicle enters or leaves it. This message includes information on its total free resources. The neighbouring node that receives this message will update its neighbour tables.



Neighbour	Distance	Percentage of free spaces	Capacity
P1	1.6 Km	0.8	100
P2	2 Km	0.5	120
P4	1.5 Km	0.3	100
P7	1.8 Km	0.6	100

Neighbour	Distance	Percentage of free spaces	Capacity
P3	1.5 Km	0.7	200
P5	1.2 Km	0.5	120

Neighbour	Distance	Percentage of free spaces	Capacity
P2	1.6 Km	0.5	120
P3	1.2 Km	0.7	200

Neighbour	Distance	Percentage of free spaces	Capacity
P4	1.2 Km	0.3	100
P6	0.8 Km	0.75	120

Neighbour	Distance	Percentage of free spaces	Capacity
P1	1.2 Km	0.8	100
P3	2 Km	0.5	200
P7	1 Km	0.6	100

Neighbour	Distance	Percentage of free spaces	Capacity
P2	1 Km	0.5	120
P3	1.8 Km	0.7	120
P6	1.2 Km	0.75	120

Neighbour	Distance	Percentage of free spaces	Capacity
P5	0.8 Km	0.5	120
P7	1.2 Km	0.6	100

Figure 3.5: Neighbour Nodes in the Network

To increase the performance of finding a free parking resource, the neighbour table in each node contains information on the current number of free parking resources in the neighbouring nodes. Our idea is to use the number of total free parking resources in each node to calculate the cost for choosing a car park.

3.3.2 Selection Based on Cost

We use a function named $f(a,b)$ to calculate the cost between the nodes in the network. $f(a,b)$ is a function that depends on the distance between two nodes and the number of free parking spaces in the destination node. $f(a,b)$ is considered to be a weighted link between two nodes in the parking network. If two nodes are not directly linked, then $f(a,b) = \infty$. If the vehicle comes into a node and that node is full, the vehicle will be forwarded to the next node, which is a neighbour of this node with the smallest value of $f(a,b)$ in the neighbour table. We calculate the cost function $f(a,b)$ from node P_i to node P_j, i.e.,

$$f(a,b) = a \times \frac{d_{ij}}{D_{up}} + b \times \frac{s_j}{S_{up}}$$

where a is a coefficient that depends on the length of the path between two nodes and b is a coefficient that depends on the number of free slots in the destination node. $f(a,b)$ is inversely proportional to the distance between two nodes and directly proportional to the total free slots in the destination node. Depending on which parameter we consider to be the more important of the two parameters, i.e., the distance or the free slots, we can adjust a and b to achieve better network performance.

a and b are parameters derived from the experiment, and their value is [0, 1]. If $a = 0$, we only consider the number of free spaces to calculate the cost to the user. If $b = 0$, we only consider the distance between two nodes to calculate the cost to the user.

In above equation, we calculate the cost function based on the distance between two nodes and the percentage of free parking spaces at each node. We use the upper bound of the distance between two nodes and the upper bound of the capacity for parking in each car park. In above equation, d_{ij} is the distance between nodes P_i and P_j, D_{up} is the upper bound of the distance and is a global parameter, s_j is the number of spaces that are occupied at node P_j, and S_{up} is the upper bound of the capacity of the overall parking network and is a global parameter.

3.4 Working Algorithm

The algorithm utilized to operate the system is discussed in this section.
A master algorithm is utilized to manage the system operations and a simple calculative algorithm is used to update the current status of each parking node.
(1) System Operation Algorithm

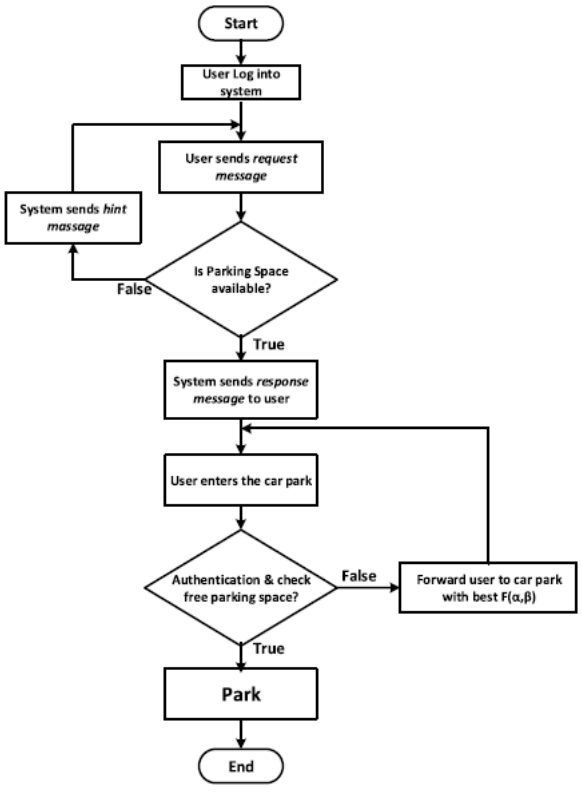

Start
User Log into system
User sends *request message*
System sends *hint massage*
Is Parking Space available?
False
True
System sends *response message* to user
User enters the car park
Authentication & check free parking space?
False
Forward user to car park with best F(α,β)
True
Park
End

Figure 3.6: System Operation Algorithm

1. Before using this online parking system, initially the user has to do his/her registration in the website.

2. Once registration is done the user has to login through the account he/she has created. After the login the user will be sending the request i.e. the location where he/she is going.

3. When the request is received, first of all it is checked that across the parking network in the desired location, any parking space is available or not.

4. If parking space in not available then apology information is sent to the user for no availability of parking space.

5. Else, if parking space is available then the system using the formula F(a,b) will be finding the best optimal parking lot among the several parking lots.

6. As per the instruction sent by the system, the user will reserve a parking space in that particular parking lot. Then a countdown timer will be set by the system. Within that particular interval of time the reserved user have to park their car. When user parks the car the sensor detects the occupied space and automatically will increment the total parking space value across the particular parking lot.

7. If within the time interval user parks the car then the system will increment the parking space value of that parking lot and finally send the bill.

8. If the user does not park his/her car within the particular time interval then fine will be introduced whose value will go on increasing till the user parks the car; or after a certain amount of time the reservation will be discarded.

9. Towards the end when user leaves the parking lot with his car, the sensor will detect the free parking space and automatically will decrement the total parking space value across the particular parking lot.

(2) Status Updater Algorithm

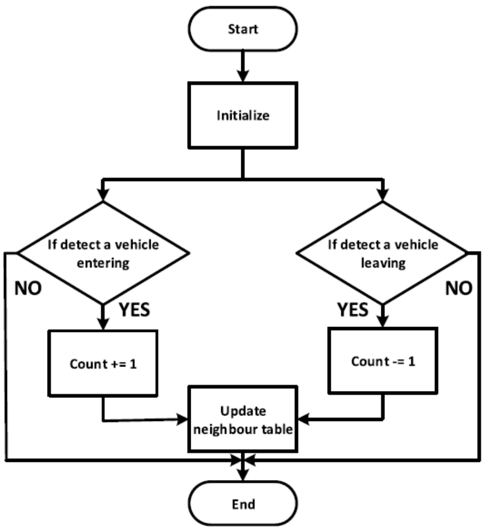


Figure 3.7: Status Updater Algorithm

1. It focuses on whether a car is parked in parking space or leaving the parking space and how the system interprets it.
2. Initially if there is no car parked across the parking lot, the value of counter (i.e. the space occupied counter) equals to zero.
3. When the car is parked then the counter value is incremented by 1 else when the car leaves the counter value decrements by a value 1. All these value gets continuously updated in $f(a,b)$ in order to find the next optimal parking space.

3.5 Mathematical Model

The mathematical representation of the system is discussed in this section.
Assume P denotes the set of all vehicles with parking queries in the queue. Let S denote the total of all available car parks.We let W denote the set of w_{ij}, where w_{ij} is the cost between vehicle $p_i(p_i \in P)$ and car park $s_j(s_j \in S)$. We can achieve W by calculating the distance from the vehicle to the car park (GPS address) and the number of free spaces in car park S_j. We let M and N be the size of P and S, respectively. Therefore, the size of W is $M \times N$. By assuming that vehicles are jobs and parking places are servers, w_{ij} is the cost for server s_j to do job p_i. We save the solution in X, where $x_{ij} \in X$, i.e.,

$$x_{ij} = \begin{cases} 1, & \text{if } p_i \text{ will park at } s_j \\ 0, & \text{if } p_i \text{ will not park at } s_j \end{cases}$$

We let C be the total cost for all vehicles in P to go to the parking places assigned to them by the SPS, i.e.,

$$C = \sum_{i=1}^{M} \sum_{j=1}^{N} w_{ij} \times x_{ij}$$

Since here we use $f(a, b)$ as the cost; thus, we have a new total cost :

$$C = \sum_{i=1}^{M} \sum_{j=1}^{N} f_{ij}(a, b) \times x_{ij}$$

To decrease the cost to the user, we will choose the minimum value of $f(a, b)$. We aim to make C minimum on the condition that each vehicle obtains exactly one parking resource and each car park space can be assigned to only one vehicle.

3.6 Summary

In this chapter, We discussed about the proposed system and it's architecture along with the networking system. Also we have gone through the working principle and mathematical modelling of the whole system to help us find a solution.

Chapter 4

Simulation of the Proposed System

4.1 Introduction

In mathematical modelling section, we described the base formula that is being used to run the whole system. The function $f(a, b)$ depends upon the values of both a and b which vary between 0 and 1.We simulated the parking scenario with the help of a discrete event simulator, ***ARENA*** and used multiple values of both a and b between the given ranges and tried to find out for which values of a and b the system shows the best and optimal performance.

4.2 Simulation Setup

In this section, we have described the setup of the proposed system explaining the different parameters and scenarios that we have assumed during the simulation.

This model has been designed as a central model rather than a distributed one. Which means, this model is under the assumption that vehicles enter the system from a single point.We have assumed a total 4 parking lots.

In our model the distance of the central entrance from each parking node and the distance between neighbouring nodes is assumed as;

1. The distance between Source and parking lot 1 = 2 kilometres
2. The distance between Source and parking lot 2 = 2.8 kilometres
3. The distance between Source and parking lot 1 = 3.2 kilometres
4. The distance between Source and parking lot 1 = 4.2 kilometres
5. The distance between parking lot 1 and parking lot 2 = 0.6 kilometres
6. The distance between parking lot 2 and parking lot 4 = 0.8 kilometres

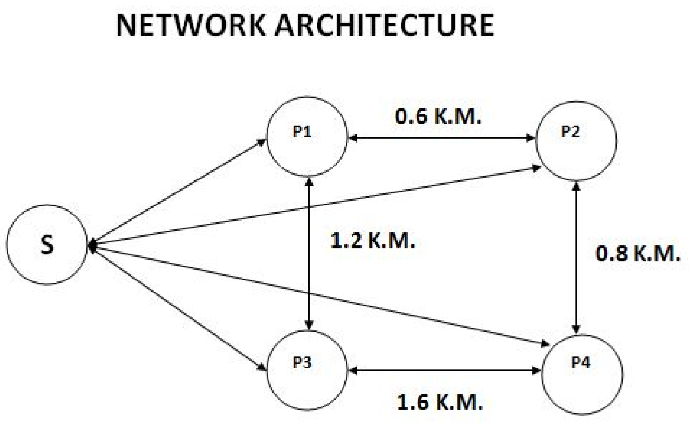


Figure 4.1: The simulated network

7. The distance between parking lot 3 and parking lot 4 = 1.6 kilometres
8. The distance between parking lot 3 and parking lot 1 = 1.2 kilometres

The total capacity of the assumed network is 9 spaces which is again distributed as;

1. Capacity of parking lot 1 = 4 vehicles
2. Capacity of parking lot 2 = 1 vehicle
3. Capacity of parking lot 3 = 2 vehicles
4. Capacity of parking lot 4 = 2 vehicles

To evaluate the performance of the processes, we simulated a network deployment, including the car park architecture mentioned above. We used the network simulation tool ***ARENA*** to simulate this network. To simulate the mathematical model, we randomly created vehicles to join the network. The arrival process of the vehicles followed the Poisson distribution in our simulation, which was denoted as $POIS(X)$, where X is the inter-arrival time between successive vehicles arriving at the car park. In this simulation X = 30 min.

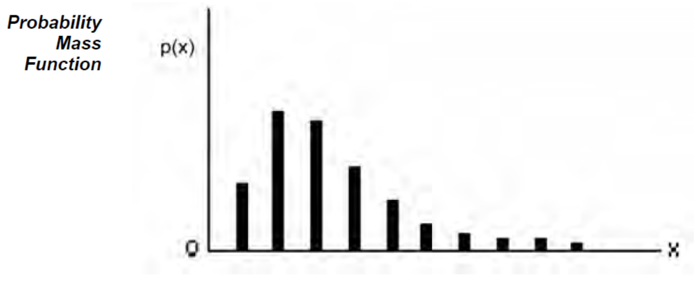


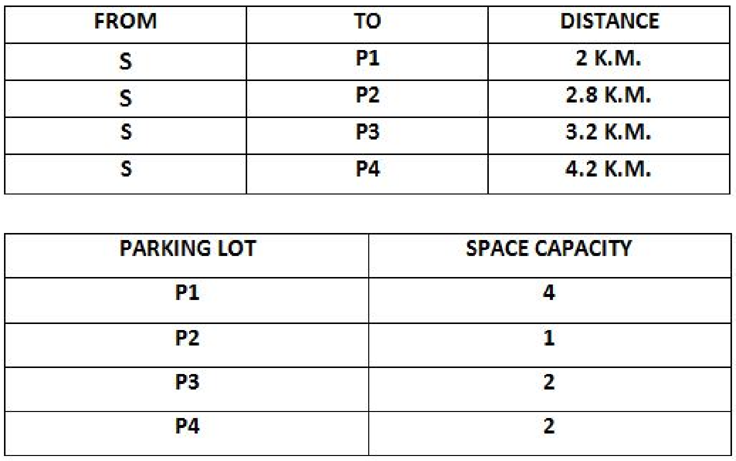
Figure 4.2: The Poisson Curve

We considered the parking process as the job and the vehicles as the entities doing the job. The time for doing the job followed an exponential distribution, which was denoted as $EXPO(Y)$ in this simulation, where Y is the average service time that a vehicle stays in the parking space. We chose Y = 120 min in this case.

Probability Density Function
f(x)
$\frac{1}{\beta}$
0
x

Figure 4.3: The Exponential Curve

We set up each car park with some capacity of parking spaces as the resources shown in the fig:



FROM	TO	DISTANCE
S	P1	2 K.M.
S	P2	2.8 K.M.
S	P3	3.2 K.M.
S	P4	4.2 K.M.

PARKING LOT	SPACE CAPACITY
P1	4
P2	1
P3	2
P4	2

Figure 4.4: Resource Allocation

We also created the same number of random vehicles to arrive at each car park. In this simulation, the number of vehicles arriving at each car park is 30, 60, 70, 80, 90, and 100. We ran the simulations until all vehicles were serviced.

To compare the network performance and provide an optimal solution for our proposed network, we set up a simulation based on various values of a and b. We simulated all possible cases in the range from zero to one. The following are some outstanding values in the range from zero to one. The value of a changed from 0, 0.2, 0.5, 0.8, 1. The value of b changed from 0, 0.2, 0.5, 0.8, 1.

4.3 ARENA Simulation Tool

4.3.1 What is Arena Software

Arena software enables us to bring the power of modelling and simulation to our business. It is designed for analyzing the impact of changes involving significant and complex redesigns associated with supply chain, manufacturing, processes, logistics, distribution and warehousing, and service systems. Arena software provides the maximum flexibility and breadth of application coverage to model any desired level of detail and complexity.

Typical scenarios include:

- Detailed analysis of any type of manufacturing system, including material-handling components
- Analysis of complex customer service and customer management systems
- Analysis of global supply chains that include warehousing, transportation, and logistics systems
- Predicting system performance based on key metrics such as costs, throughput, cycle times, and utilizations
- Identifying process bottlenecks such as queue build ups and over-utilization of resources
- Planning staff, equipment, or material requirements.

4.3.2 Working with Arena

We can simulate an application to any level of design and complexity with Arena. Basically we can:

- **Model** our processes to define, document, and communicate.
- **Simulate** the future performance of your system to understand complex relationships and identify opportunities for improvement.
- **Visualize** our operations with dynamic animation graphics.

- **Analyze** how your system will perform in its “as-is” configuration and under a myriad of possible “to-be” alternatives so that you can confidently choose the best way to run your business.

4.3.3 The Arena Modelling Environment

To model our process in Arena, we’ll work in three main regions of the application window. The Project Bar hosts panels with the primary types of objects that we will work with:

1. **Basic Process**, **Advanced Process**, and **Advanced Transfer panels**: Contain the modeling shapes, called modules, that we’ll use to define our process.
2. **Reports panel**: Contains the reports that are available for displaying results of simulation runs.
3. **Navigate panel**: Allows us to display different views of your model, including navigating through hierarchical sub models and displaying a model thumbnail view.

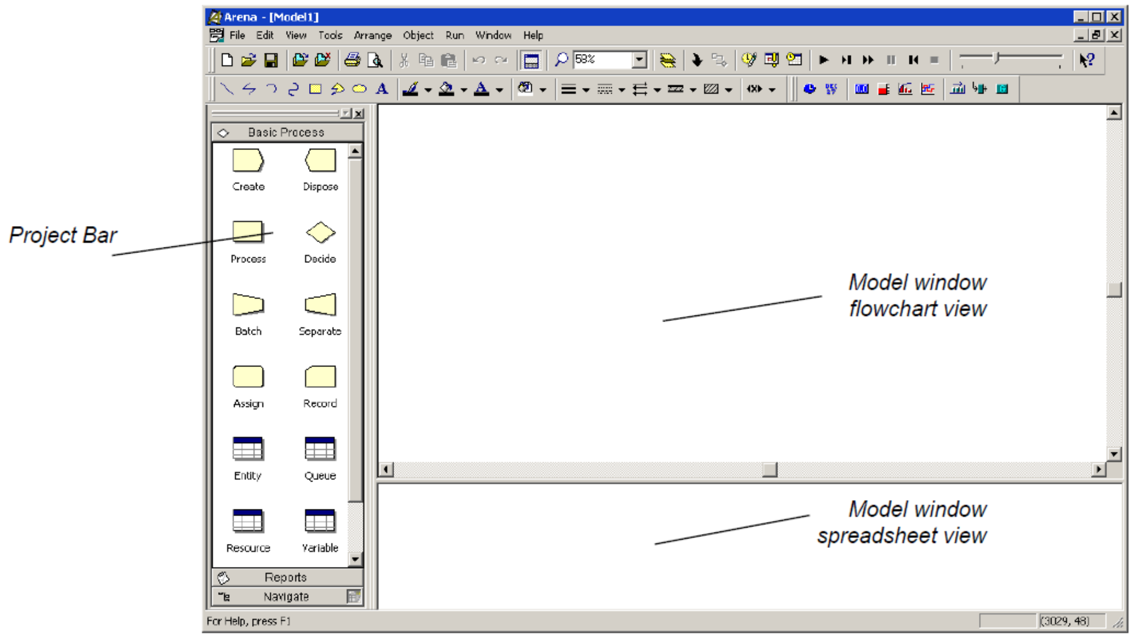


Figure 4.5: The simulator

In the model window, there are two main regions. The flowchart view will contain all of the model graphics, including the process flowchart, animation, and other drawing elements. The lower, spreadsheet view displays model data, such as times, costs, and other parameters.

4.3.4 Entities in Arena

Entities are the items—customers, documents, parts—that are being served, produced, or otherwise acted on by your process. In business processes, they often are documents or electronic records (checks, contracts, applications, purchase orders). In service systems, entities usually are people (the customers being served in a restaurant, hospital, airport, etc.). Manufacturing models typically have some kind of part running through the process, whether it's raw material, a subcomponent, or finished product. Other models might have different types of entities, such as data packets in network analysis or letters and boxes in package-handling facilities.

We may have different types of entities in the same model. For example, customers moving through a check-in counter at an airport might be separated into regular, first-class, and priority entity types. In some cases, entity types might be of an altogether different form rather than classifications of some basic type. For instance, in a pharmacy, prescriptions would be modelled as entities, running through the process of being filled. At the same time, customers might be competing for the pharmacist's attention with medical inquiries; they would also be modelled as entities.

However, in our case, since we are simulating a parking model we have assumed vehicle images as entities to represent the work-flow in our model.

4.3.5 Modules in Arena

In Arena, modules are the flowchart and data objects that define the process to be simulated.

All information required to simulate a process is stored in modules. For now, we're working with flowchart modules—those that are placed in the model window to describe the process. In the Basic Process panel, these are the first eight shapes:

- **Create**: The start of process flow. Entities enter the simulation here.
- **Dispose**: The end of process flow. Entities are removed from the simulation here.
- **Process**: An activity, usually performed by one or more resources and requiring some time to complete.
- **Decide**: A branch in process flow. Only one branch is taken.
- **Batch**: Collect a number of entities before they can continue processing.
- **Separate**: Duplicate entities for concurrent or parallel processing, or separating a previously established batch of entities.

- **Assign**: Change the value of some parameter (during the simulation), such as the entity's type or a model variable.
- **Record**: Collect a statistic, such as an entity count or cycle time.

Now we'll discuss the modules that has been used in the simulation in more detail like individual parameters and their uses.

CREATE Module

This module is intended as the starting point for entities in a simulation model. Entities are created using a schedule or based on a time between arrivals. Entities then leave the module to begin processing through the system. The entity type is specified in this module.

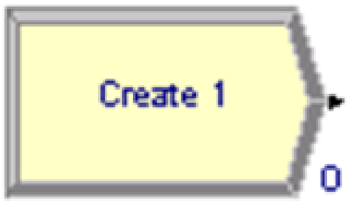


Figure 4.6: Create Module

TYPICAL USES

- The start of a part's production in a manufacturing line
- A document's arrival (e.g., order, check, application) into a business process
- A customer's arrival at a service process (e.g., retail store, restaurant, information desk)

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.
- **Entity Type**: Name of the entity type to be generated.
- **Type**: Type of arrival stream to be generated. Types include **Random** (uses an exponential distribution, user specifies mean), **Schedule** (uses an exponential distribution, mean determined from the specified Schedule module), **Constant** (user specifies constant value; e.g., 100), or **Expression** (drop-down list of various distributions).
- **Value**: Determines the mean of the exponential distribution (if Random is used) or the constant value (if Constant is used) for the time between arrivals. Applies only when Type is Random or Constant.

- **Schedule Name**: Identifies the name of the schedule to be used. The schedule defines the arrival pattern for entities arriving to the system. Applies only when Type is Schedule.

- **Expression**: Any distribution or value specifying the time between arrivals. Applies only when Type is Expression.

- **Units**: Time units used for interarrival and first creation times. Does not apply when Type is Schedule.

- **Entities per Arrival**: Number of entities that will enter the system at a given time with each arrival.

- **Max Arrivals**: Maximum number of entities that this module will generate. When this value is reached, the creation of new entities by this module ceases.

- **First Creation**: Starting time for the first entity to arrive into the system. Does not apply when Type is Schedule.

DISPOSE Module

This module is intended as the ending point for entities in a simulation model. Entity statistics may be recorded before the entity is disposed.

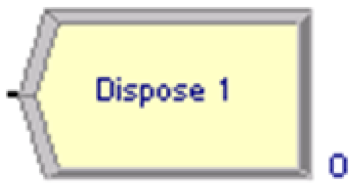


Figure 4.7: Dispose Module

TYPICAL USES

- Parts leaving the modelled facility

- The termination of a business process

- Customers departing the store

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.

- **Record Entity Statistics**: Determines whether or not the incoming entity's statistics will be recorded. Statistics include value-added time, non-value-added time, wait time, transfer time, other time, total time, value-added cost, nonvalue-added cost, wait cost, transfer cost, other cost, and total cost.

PROCESS Module

This module is intended as the main processing method in the simulation. Options for seizing and releasing resource constraints are available. Additionally, there is the option to use a "submodel" and specify hierarchical user-defined logic. The process time is allocated to the entity and may be considered to be value added, non-value added, transfer, wait, or other. The associated cost will be added to the appropriate category.

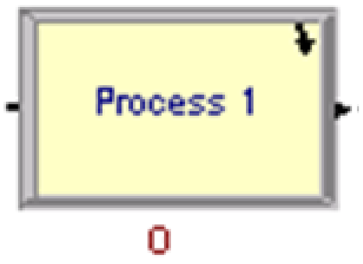


Figure 4.8: Process Module

TYPICAL USES

- Machining a part
- Reviewing a document for completeness
- Fulfilling orders
- Serving a customer

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.
- **Type**: Method of specifying logic within the module. **Standard** processing signifies that all logic will be stored within the Process module and defined by a particular Action. **Submodel** indicates that the logic will be hierarchically defined in a "submodel" that can include any number of logic modules.
- **Action**: Type of processing that will occur within the module. **Delay** simply indicates that a process delay will be incurred with no resource constraints. **Seize Delay** indicates that a resource(s) will be allocated in this module and delay will occur, but that resource release will occur at a later time. **Seize Delay Release** indicates that a resource(s) will be allocated followed by a process delay and then the allocated resource(s) will be released. **Delay Release** indicates that a resource(s) has previously been allocated and that the entity will simply delay and release the specified resource(s). Applies only when Type is Standard.

- **Priority**: Priority value of the entity waiting at this module for the resource(s) specified if one or more entities are waiting for the same resource(s) anywhere in the model. Not visible when Action is Delay or Delay Release or when Type is Submodel.
- **Resources**: Lists the resources or resource sets used for entity processing. Does not apply when Action is Delay, or when Type is Submodel.
- **Delay Type**: Type of distribution or method of specifying the delay parameters. **Constant** and **Expression** require single values, while **Normal**, **Uniform**, and **Triangular** require several parameters.
- **Units**: Time units for delay parameters.
- **Allocation**: Determines how the processing time and process costs will be allocated to the entity. The process may be considered to be **Value Added**, **Non- Value Added**, **Transfer**, **Wait**, or **Other** and the associated cost will be added to the appropriate category for the entity and process.
- **Minimum**: Parameter field for specifying the minimum value for either a uniform or triangular distribution.
- **Value**: Parameter field for specifying the **mean** for a normal distribution, the **value** for a constant time delay, or the **mode** for a triangular distribution.
- **Maximum**: Parameter field for specifying the maximum value for either a uniform or triangular distribution.
- **Std Dev**: Parameter field for specifying the standard deviation for a normal distribution.
- **Expression**: Parameter field for specifying an expression whose value is evaluated and used for the processing time delay.
- **Report Statistics**: Specifies whether or not statistics will be automatically collected and stored in the report database for this process.

PROCESS Module - Resource Dialog

- **Type**: Specification of a particular resource, or selecting from a pool of resources (i.e., a resource set).

- **Resource Name**: Name of the resource that will be seized and/or released. Applies only when Type is Resource.
- **Set Name**: Name of the resource set from which a member will be seized and/or released. Applies only when Type is Set
- **Quantity**: Number of resources of a given name or from a given set that will be seized/released. For sets, this value specifies only the number of a selected resource that will be seized/released (based on the resource's capacity), not the number of members of a set to be seized/released.
- **Selection Rule**: Method of selecting among available resources in a set. Cyclical will cycle through available members (e.g., 1st member–2nd member–3rd member–1st member–2nd member–3rd member). **Random** will randomly select a member. **Preferred Order** will always select the first available member (1st member, if available; then 2nd member, if available; then 3rd member, etc.). **Specific Member** requires an input attribute value to specify which member of the set (previously saved in the Save Attribute field). **Largest Remaining Capacity** and **Smallest Number Busy** are used for resources with multiple capacity. Applies only when Type is Set.
- **Save Attribute**: Attribute name used to save the index number into the set of the member that is selected. This attribute can later be referenced with the Specific Member selection rule. Does not apply when Selection Rule is Specific Member. If Action is specified as Delay Release, the value specified defines which member (the index number) of the set to be released. If no attribute is specified, the entity will release the member of the set that was last seized.
- **Set Index**: The index number into the set of the member requested. Applies only when Selection Rule is Specific Member. If Action is specified as Delay Release, the value specified defines which member (the index number) of the set is to be released.

DECIDE Module

This module allows for decision-making processes in the system. It includes options to make decisions based on one or more conditions (e.g., if entity type is Gold Card) or based on one or more probabilities (e.g., 75%, true; 25%, false). Conditions can be based on attribute values (e.g., Priority), variable values (e.g., Number Denied), the entity type, or an expression (e.g., NQ(ProcessA.Queue)).

There are two exit points out of the Decide module when its specified type is either 2-way by Chance or 2-way by Condition. There is one exit point for "true" entities and one for “false” entities. When the N-way by Chance or by Condition type is specified, multiple exit points are shown for each condition or probability and a single “else” exit. The number of entities that exit from each type (true/false) is displayed for 2-way by Chance or by Condition modules only.

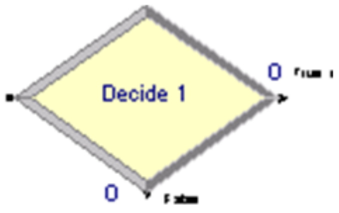


Figure 4.9: Decide Module

TYPICAL USES

- Dispatching a faulty part for rework
- Branching accepted vs. rejected checks
- Sending priority customers to a dedicated process

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.
- **Type**: Indicates whether the decision is based on a condition (if X>Y) or by chance/percentage (e.g., 60%, yes; 40%, no). The type can be specified as either 2-way or N-way. 2-way allows for one condition or probability (plus the “false” exit). N-way allows for any number of conditions or probabilities to be specified as well as an “else” exit.
- **Conditions**: Defines one or more conditions used to direct entities to different modules. Applies only when Type is N-way by Condition.
- **Percentages**: Defines one or more percentages used to direct entities to different modules. Applies only when Type is N-way by Chance.
- **Percent True**: Value that will be checked to determine the percentage of entities sent out a given True exit.
- **If**: Types of conditions that are available for evaluation: Variable, Variable Array (1D), Variable Array (2D), Attribute, Entity Type, Expression.
- **Named**: Specifies either the name of the variable, attribute, orentity type that will be evaluated when an entity enters the module. Does not apply when Type is Expression.

- **Is**: Evaluator for the condition. Applies only to Attribute and Variable conditions.
- **Row**: Specifies the row index for a variable array. Applies only when Type is N-way by Condition or 2-way by Condition and Variable is Array 1-D or Array 2-D.
- **Column**: Specifies the column index for a variable array. Applies only when Type is N-way by Condition or 2-way by Condition and Variable is Array 1-D or Array 2-D.
- **Value**: Expression that will be either compared to an attribute or variable or that will be evaluated as a single expression to determine if it is true or false. Does not apply to Entity Type condition. If Type is Expression, this value must also include the evaluator (e.g., Color<>Red).

ASSIGN Module

This module is used for assigning new values to variables, entity attributes, entity types, entity pictures, or other system variables. Multiple assignments can be made with a single Assign module.

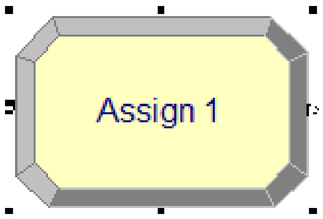


Figure 4.10: Assign Module

TYPICAL USES

- Accumulate the number of sub-assemblies added to a part
- Change an entity's type to represent the customer copy of a multi-page form
- Establish a customer's priority

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.
- **Assignments**: Specifies the one or more assignments that will be made when an entity executes the module.

- **Type**: Type of assignment to be made. Other can include system variables, such as resource capacity or simulation end time.
- **Variable Name**: Name of the variable that will be assigned a new value when an entity enters the module. Applies only when Type is Variable, Variable Array (1D), or Variable Array (2D).
- **Row**: Specifies the row index for a variable array.
- **Column**: Specifies the column index for a variable array.
- **Attribute Name**: Name of the entity attribute that will be assigned a new value when the entity enters the module. Applies only when Type is Attribute.
- **Entity Type**: New entity type that will be assigned to the entity when the entity enters the module. Applies only when Type is Entity Type.
- **Entity Picture**: New entity picture that will be assigned to the entity when the entity enters the module. Applies only when Type is Entity Picture.
- **Other**: Identifies the special system variable that will be assigned a new value when an entity enters the module. Applies only when Type is Other.

RECORD Module

This module is used to collect statistics in the simulation model. Various types of observational statistics are available, including time between exits through the module, entity statistics (time, costing, etc.), general observations, and interval statistics (from some time stamp to the current simulation time). A count type of statistic is available as well. Tally and Counter sets can also be specified.

Figure 4.11: Record Module

TYPICAL USES

- Collect the number of jobs completed each hour
- Count how many orders have been late being fulfilled

- Record the time spent by priority customers in the main check-out line

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.
- **Type**: Type of observational (tally) or count statistic to be generated. **Count** will increase or decrease the value of the named statistic by the specified value. **Entity Statistics** will generate general entity statistics, such as time and costing/duration information. **Time Interval** will calculate and record the difference between a specified attribute's value and current simulation time. **Time Between** will track and record the time between entities entering the module. **Expression** will record the value of the specified expression.
- **Attribute Name**: Name of the attribute whose value will be used for the interval statistics. Applies only when Type is Interval.
- **Value**: Value that will be recorded to the observational statistic when Type is Expression or added to the counter when Type is Count.
- **Tally Name**: This field defines the symbol name of the tally into which the observation is to be recorded. Applies only when Type is Time Interval, Time Between, or Expression.
- **Counter**: This field defines the symbol name of the counter to Name increment/decrement. Applies only when Type is Counter.
- **Record into Set**: Check box to specify whether or not a tally or counter set will be used.
- **Tally Set Name**: Name of the tally set that will be used to record the observational-type statistic. Applies only when Type is Time Interval, Time Between, or Expression.
- **Counter Set Name**: Name of the counter set that will be used to record the count-type statistic. Applies only when Type is Count.
- **Set Index**: Index into the tally or counter set.

DELAY Module

The Delay module delays an entity by a specified amount of time.

When an entity arrives at a Delay module, the time delay expression is evaluated and the entity remains in the module for the resulting time period. The time is then

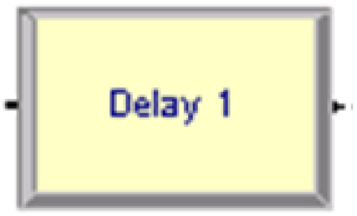


Figure 4.12: Delay Module

allocated to the entity's value-added, non-value added, transfer, wait, or other time. Associated costs are calculated and allocated as well.

TYPICAL USES

- Processing a check at a bank
- Performing a setup on a machine
- Transferring a document to another department

PROMPTS

- **Name**: Unique module identifier displayed on the module shape.
- **Allocation**: Type of category to which the entity's incurred delay time and cost will be added.
- **Delay Time**: Determines the value of the delay for the entity.
- **Units**: Time units used for the delay time.

4.4 Simulation Scenarios

While designing the proposed network using ARENA, we faced a limitation that no more than 150 entities can exist on a single simulated environment at any point of time. This fact forced us to simulate the system in two different models.

- Animation Based Model
- Entity Based Model

4.4.1 Animation Based Model

The first model uses a lot of modules and to clearly demonstrates the animated view of the parking system. In this model, the graphical representation has been prioritized rather than data based analysis of the system performance.

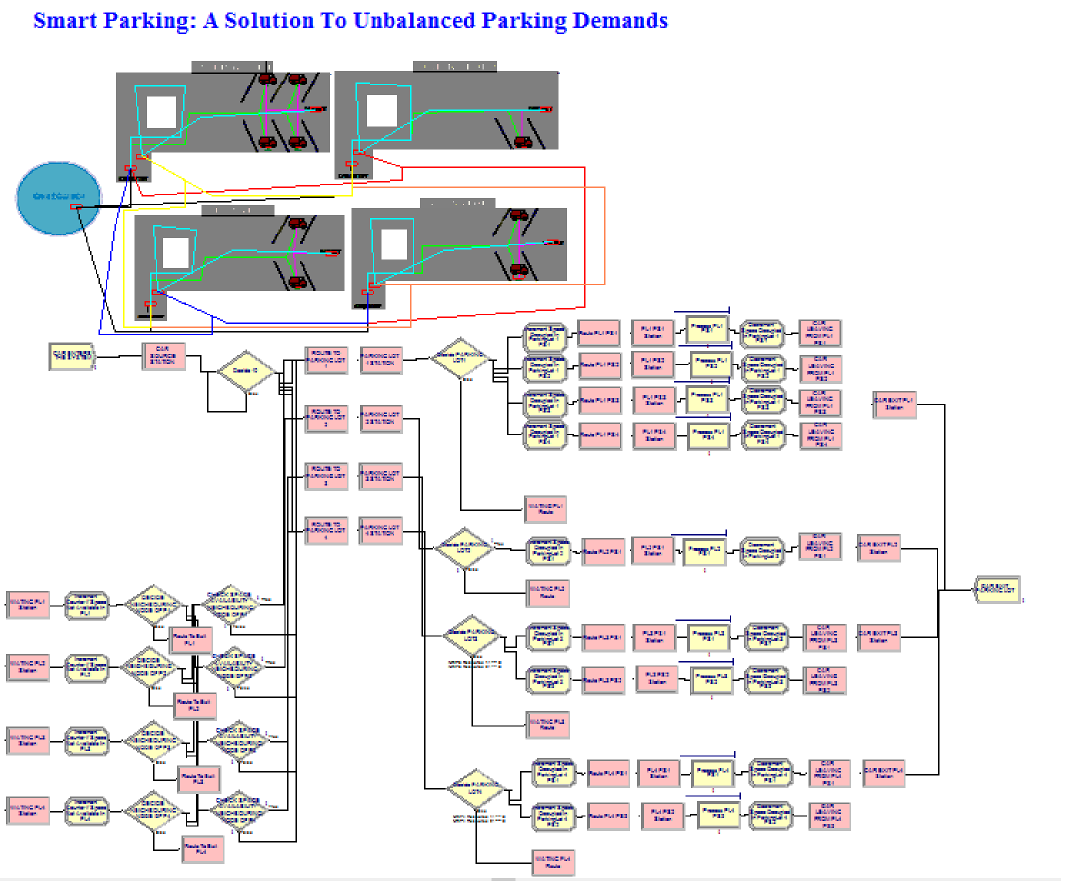


Figure 4.13: Animation Based Model

4.4.2 Entity Based Model

In this model,rather than animation, the analysis of the system using a higher number of entities has been prioritized. Less number of modules hav e been used in this model while focusing on a higher number of entities.

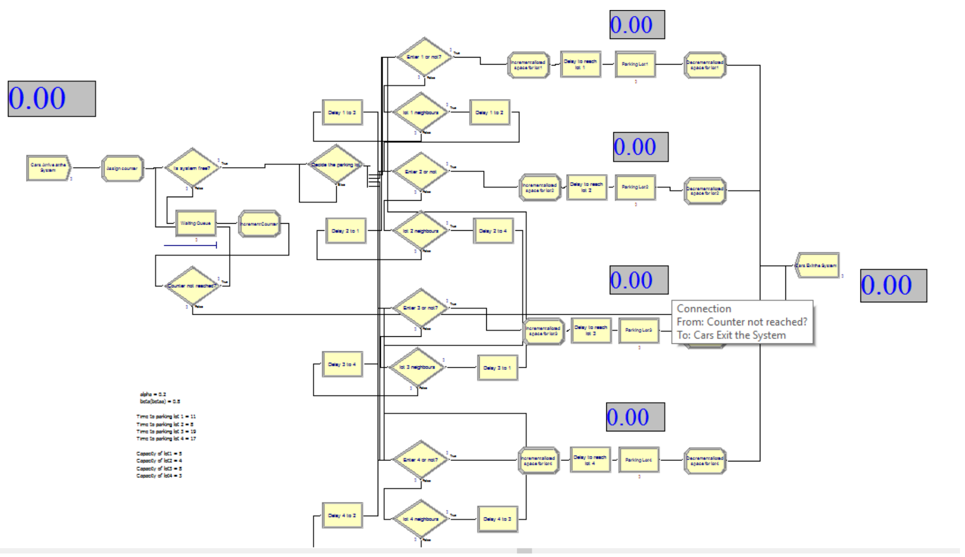


Figure 4.14: Entity Based Model

4.4.3 Working Method

The CREATE module helps in generating Entity(cars) under Poisson distribution i.e. $POIS(X)$. Here the value of X=30 min. It means within the interval of $POIS(30)$ a car will be generated. For our simulation purpose we have considered maximum 30 arrivals.

Once cars are generated then we have to decide the optimal parking station where the car needs to be parked. Therefore DECIDE module is used which determines the value of $f(a, b)$ i.e. our mathematical formula to determine the best optimal parking lot, since we have taken 4 Parking Stations into consideration.

To compute the value of $f(a, b)$ we have used several variables like Distance to parking lot 1, Distance to parking lot 2, Distance to parking lot 3, Distance to parking lot 4 for holding the distance between car source station to the several parking lot. Similarly for current free space, variables like space occupied in parking lot 1, space occupied in parking lot 2, space occupied in parking lot 3, space occupied in parking lot 4 are used. Using these values along with the value of parameters(i.e. a=0 to 1,b=0 to 1) the value of $f(a, b)$ was calculated for each parking lot.

Out of the four values of $f(a, b)$ the minimum value will be the optimal value, hence the car will be directed to that particular parking lot and the car is parked. Regarding to time taken by average cars to be parked in the parking lot we have taken exponential distribution i.e. $EXPO(Y)$ where the value of Y=120 min. So any random value generated between 0 to 120 exponentially, within that particular time interval the car will be parked an then after the car exits the system through the DISPOSE module.

Now in the directed Parking Station if parking space is not available, then as per our algorithm we will search for a free space in the neighbouring parking lots. For example if in parking lot 1, not a single space is available then we have to search for it's neighbours i.e. parking lot 2 and parking lot 4. To decide the best optimal parking lot among the neighbouring parking lot we will again use the formula $f(a, b)$ by considering the distance between the directed parking station and neighbouring parking station and the space occupied in the neighbouring parking lots.

In this manner we will check for all the neighbouring parking stations until we find a free parking space. If space is available then the car is parked and for the worst case scenario if no space is available in the whole system,then car won't get parked, hence the car simply leaves the parking system through the DISPOSE module.

4.5 Simulation Results

As we described earlier, the sole purpose of this simulation was to find out the appropriate values for a and b in the equation of $f(a,b)$.So we assumed a set of possible values for both these parameters in the range of 0 to 1 and we performed the simulation under all possible permutations of both these values.

Using either a or b as 0 or 1 actually invalidates the purpose of the base formula.Because our base formula for cost calculation states as:

$$f(a,b) = a \times \frac{d_{ij}}{D_{up}} + b \times \frac{s_j}{S_{up}}$$

Since a and b are multiplicative coefficients, setting any one of these to zero will nullify the whole parameter. For example, if we set a as zero, then we are not considering the inter-mediate distances as a parameter, rather we are only concerned about the space availability. Similarly, setting $b = 0$ also nullifies the space availability criterion. So these kinds of cases have been ignored during the simulation.

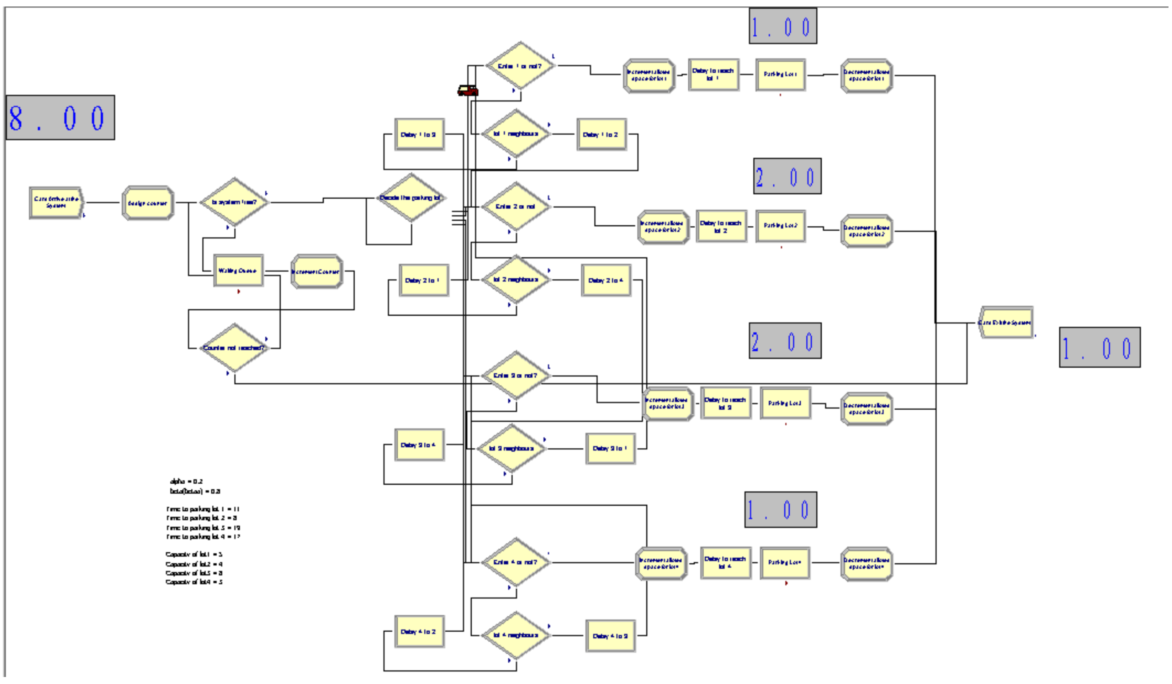


Figure 4.15: A simulation snapshot of the Entity Based Model

Putting aside the above mentioned conditions, the performance of the network and vehicular distribution was verified for all the other possible combinations and it was found that, for $\mathbf{a} = \mathbf{0.2}$ and $\mathbf{b} = \mathbf{0.8}$ our network achieved the optimum performance.

Using the vehicle distribution data from all the possible values of a and b, a plot was drawn displaying distribution concentration among all the parking nodes in the network.

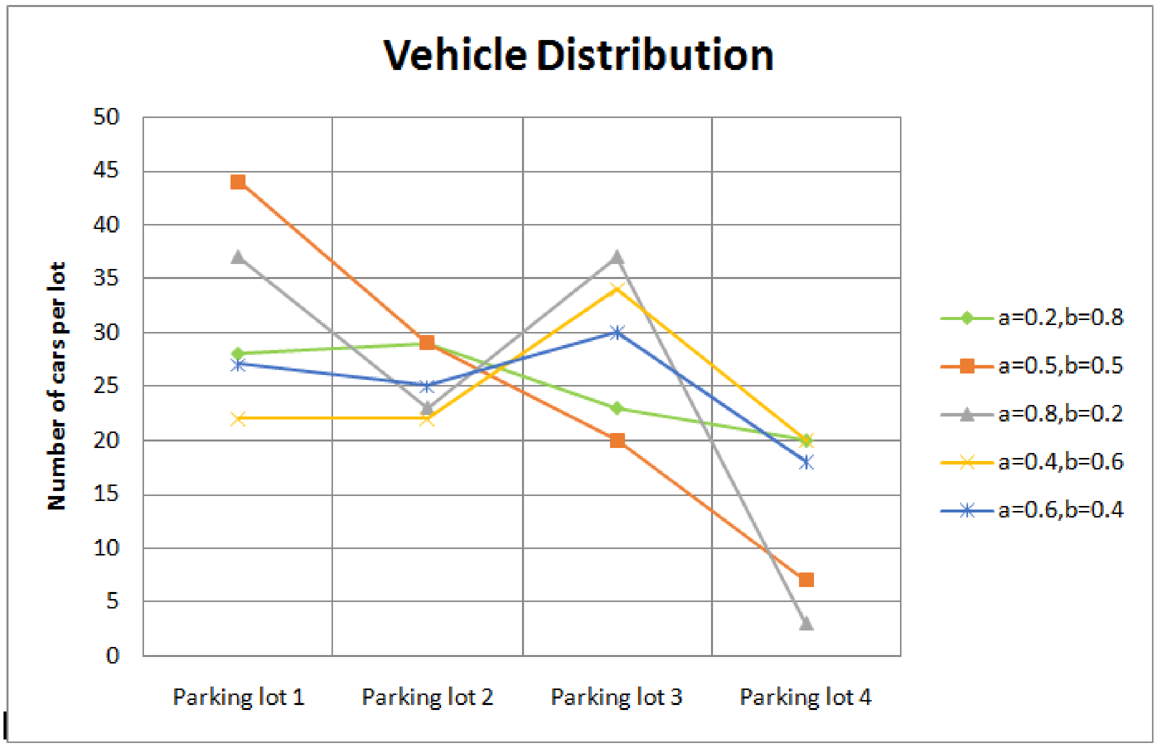


Figure 4.16: Graph showing vehicle distribution

4.6 Summary

In this chapter, the proposed system has been simulated as described previously in the work-flow model. From simulation, we have calculated the best possible values for both a and b for the function $f(a, b)$ which will be further used in the hardware implementation of this project.

Chapter 5

Implementation of the System using IoT

After a successful simulation we arrived at the optimal values of both a and b for which the system shows the best performance. But the simulation only is not enough; because after every simulation there comes an implementation. To apply the simulated environment into a real-life scenario we took the help of sensory networks, IoT(Internet-Of-Things) and cloud technology.

5.1 Hardware Requirements and Specifications

The hardware we have used to implement the system are:

1. Infra-red Sensors
2. Arduino Micro-Controller
3. Servo Motors
4. Bread Board
5. ESP8266 Module

Now we'll discuss each of these component in detail with their operations and working principles.

5.1.1 Infra-Red Sensors

Introduction

Infra-red technology addresses a wide variety of wireless applications. The main areas are sensing and remote controls. In the electromagnetic spectrum, the infra-red portion is divided into three regions: near infra-red region, mid infra-red region and far infra-red region.

The wavelengths of these regions and their applications are shown below:

- Near infrared region — 700 nm to 1400 nm — IR sensors, fiber optic
- Mid infrared region — 1400 nm to 3000 nm — Heat sensing
- Far infrared region — 3000 nm to 1 mm — Thermal imaging

The frequency range of infrared is higher than microwave and lesser than visible light.

For optical sensing and optical communication, photo optics technologies are used in the near infrared region as the light is less complex than RF when implemented as a source of signal.

Optical wireless communication is done with IR data transmission for short range applications. An infrared sensor emits and/or detects infrared radiation to sense its surroundings.

The working of any Infrared sensor is governed by three laws: Planck's Radiation law, Stephen – Boltzmann law and Wien's Displacement law.

Planck's law states that "every object emits radiation at a temperature not equal to 0 K". Stephen – Boltzmann law states that "at all wavelengths, the total energy emitted by a black body is proportional to the fourth power of the absolute temperature". According to Wien's Displacement law, "the radiation curve of a black body for different temperatures will reach its peak at a wavelength inversely proportional to the temperature".

The basic concept of an Infrared Sensor which is used as Obstacle detector is to transmit an infrared signal, this infrared signal bounces from the surface of an object and the signal is received at the infrared receiver.

There are five basic elements used in a typical infrared detection system: an infrared source, a transmission medium, optical component, infrared detectors or receivers and signal processing. Infrared lasers and Infrared LED's of specific wavelength can be used as infrared sources. The three main types of media used for infrared transmission are vacuum, atmosphere and optical fibers. Optical components are used to focus the infrared radiation or to limit the spectral response.

Optical lenses made of Quartz, Germanium and Silicon are used to focus the infrared radiation. Infrared receivers can be photodiodes, phototransistors etc. some important specifications of infrared receivers are photosensitivity, detectivity and noise equivalent power. Signal processing is done by amplifiers as the output of infrared detector is very small.

Types of IR sensors

Infrared sensors can be passive or active. Passive infrared sensors are basically Infrared detectors. Passive infrared sensors do not use any infrared source and detects energy emitted by obstacles in the field of view. They are of two types: quantum and thermal. Thermal infrared sensors use infrared energy as the source of heat and are independent of wavelength. Thermocouples, pyroelectric detectors and bolometers are the common types of thermal infrared detectors.

Quantum type infrared detectors offer higher detection performance and are faster than thermal type infrared detectors. The photosensitivity of quantum type detectors is wavelength dependent. Quantum type detectors are further classified into two types: intrinsic and extrinsic types. Intrinsic type quantum detectors are photoconductive cells and photovoltaic cells.

Active infrared sensors consist of two elements: infrared source and infrared detector. Infrared sources include an LED or infrared laser diode. Infrared detectors include photodiodes or phototransistors. The energy emitted by the infrared source is reflected by an object and falls on the infrared detector.

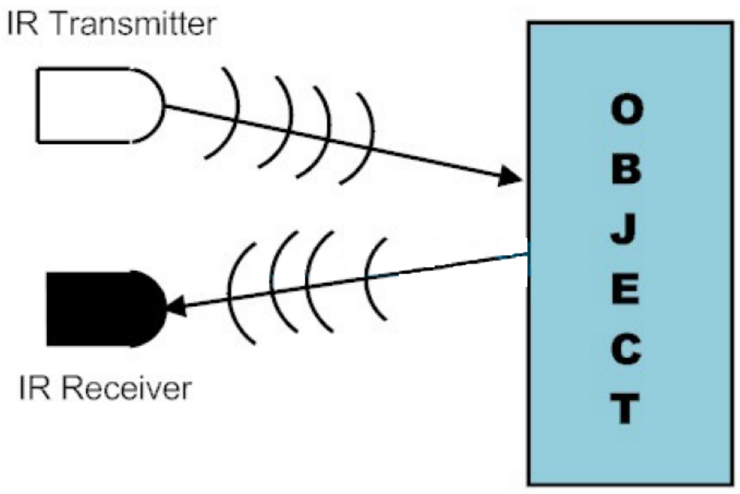


Figure 5.1: Transreceiver system

IR Transmitter

Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations. Hence, they are called IR LED's. Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human eye.

The picture of a typical Infrared LED is shown below:

Figure 5.2: IR Transmitter

There are different types of infrared transmitters depending on their wavelengths, output power and response time.

A simple infrared transmitter can be constructed using an infrared LED, a current limiting resistor and a power supply. The schematic of a typical IR transmitter is shown below.

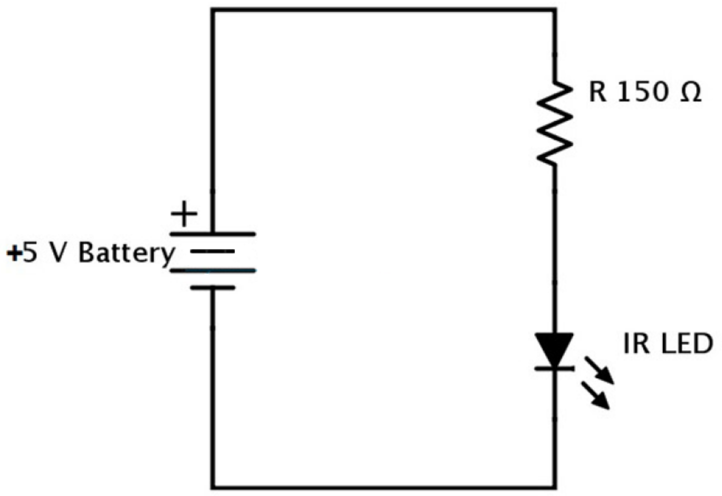


Figure 5.3: IR Transmitter circuit

IR transmitters can be found in several applications. Some applications require infrared heat and the best infrared source is infrared transmitter. When infrared emitters are used with Quartz, solar cells can be made.

IR Receiver

Infrared receivers are also called as infrared sensors as they detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors. Infrared Photodiodes are different from normal photo diodes as they detect only infrared radiation. The picture of a typical IR receiver or a photodiode is shown below:

Figure 5.4: IR Receiver

Different types of IR receivers exist based on the wavelength, voltage, package, etc. When used in an infrared transmitter – receiver combination, the wavelength of the receiver should match with that of the transmitter. A typical infrared receiver circuit using a phototransistor is shown below:

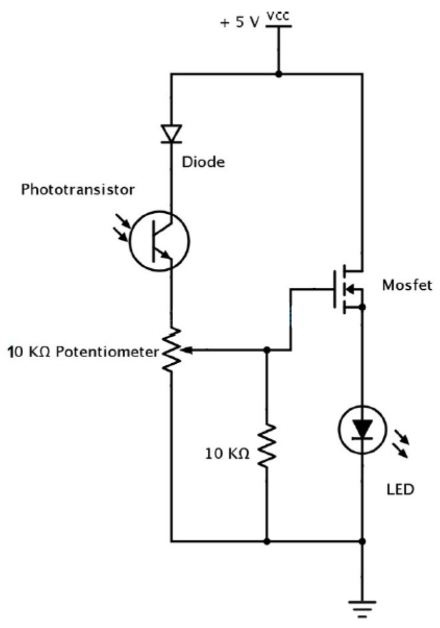


Figure 5.5: IR Receiver circuit

It consists of an IR phototransistor, a diode, a MOSFET, a potentiometer and an LED. When the phototransistor receives any infrared radiation, current flows through it and MOSFET turns on. This in turn lights up the LED which acts as a load. The potentiometer is used to control the sensitivity of the phototransistor.

Principle of Working

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.

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.

Obstacle Sensing Circuit or IR Sensor Circuit

A typical IR sensing circuit is shown below:

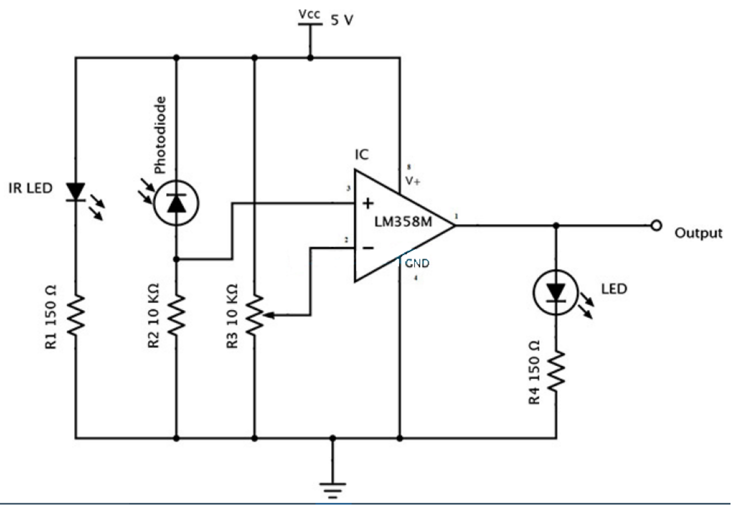


Figure 5.6: IR Transreceiver circuit

IR LED emits infrared light. The Photodiode detects the infrared light. An IC Op – Amp is used as a voltage comparator. The potentiometer is used to calibrate the output of the sensor according to the requirement.

When the light emitted by the IR LED is incident on the photodiode after hitting an object, the resistance of the photodiode falls down from a huge value. One of the input of the op – amp is at threshold value set by the potentiometer. The other input to the op-amp is from the photodiode's series resistor. When the incident radiation is more on the photodiode, the voltage drop across the series resistor will be high. In the IC, both the threshold voltage and the voltage across the series resistor are compared. If the voltage across the resistor series to photodiode is greater than that of the threshold voltage, the output of the IC Op – Amp is high. As the output of the IC is connected to an LED, it lightens up. The threshold voltage can be adjusted by adjusting the potentiometer depending on the environmental conditions.

The positioning of the IR LED and the IR Receiver is an important factor. When the IR LED is held directly in front of the IR receiver, this setup is called Direct Incidence. In this case, almost the entire radiation from the IR LED will fall on the IR receiver. Hence there is a line of sight communication between the infrared transmitter and the receiver. If an object falls in this line, it obstructs the radiation from reaching the receiver either by reflecting the radiation or absorbing the radiation.

Distinguishing Between Black and White Colors

It is universal that black color absorbs the entire radiation incident on it and white color reflects the entire radiation incident on it. Based on this principle, the second positioning of the sensor couple can be made. The IR LED and the photodiode are placed side by side. When the IR transmitter emits infrared radiation, since there is no direct line of contact between the transmitter and receiver, the emitted radiation must reflect back to the photodiode after hitting any object. The surface of the object can be divided into two types: reflective surface and nonreflective surface. If the surface of the object is reflective in nature i.e. it is white or other light color, most of the radiation incident on it will get reflected back and reaches the photodiode. Depending on the intensity of the radiation reflected back, current flows in the photodiode.

If the surface of the object is non-reflective in nature i.e. it is black or other dark color, it absorbs almost all the radiation incident on it. As there is no reflected radiation, there is no radiation incident on the photodiode and the resistance of the photodiode remains higher allowing no current to flow. This situation is similar to there being no object at all.The pictorial representation of the above scenarios is shown below.

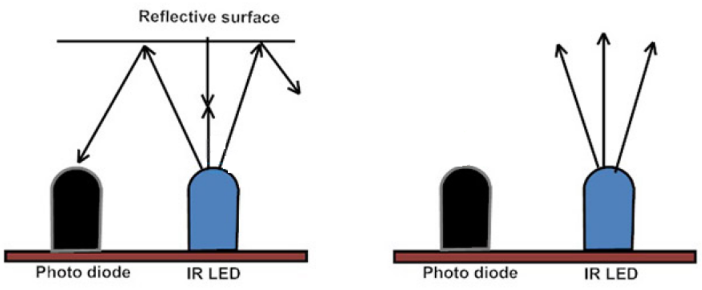


Figure 5.7: Working of the sensors

The positioning and enclosing of the IR transmitter and Receiver is very important. Both the transmitter and the receiver must be placed at a certain angle, so that the detection of an object happens properly. This angle is the directivity of the sensor which is +/- 45 degrees.The directivity is shown below:

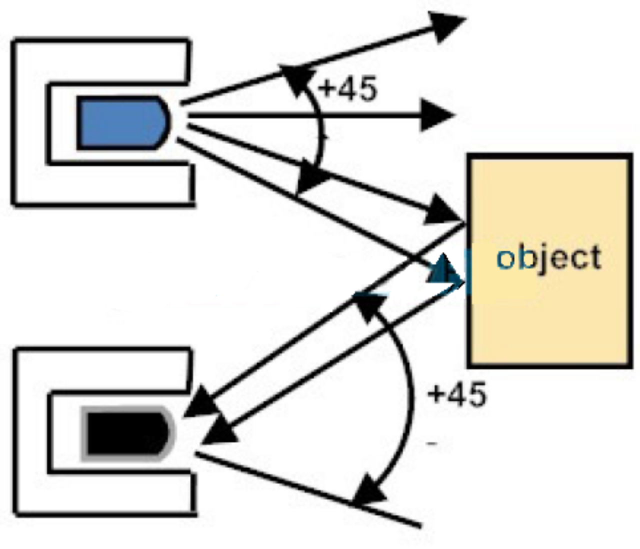


Figure 5.8: Angle of detection

In order to avoid reflections from surrounding objects other than the object, both the IR transmitter and the IR receiver must be enclosed properly. Generally the enclosure is made of plastic and is painted with black color.

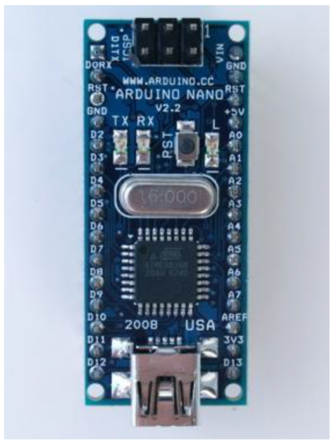
5.1.2 Arduino Micro-controller

This unit can be called the CPU of the whole system. It receives, manipulates, transmits all sorts of data from the sensors to the cloud units and it controls each and every component present in the model.

There are many micro-processor boards available from Arduino manufacturer. For our project we have taken a **Arduino-Nano** board as the controlling unit.

Overview

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.0) or ATmega168 (Arduino Nano 2.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The Nano was designed and is being produced by Gravitech.


Arduino Nano Front

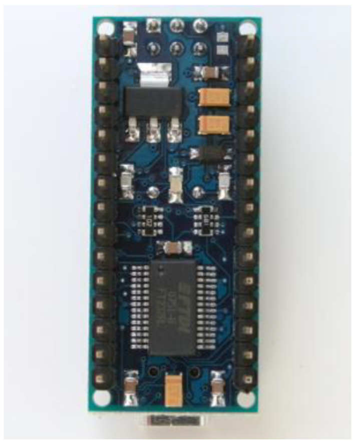
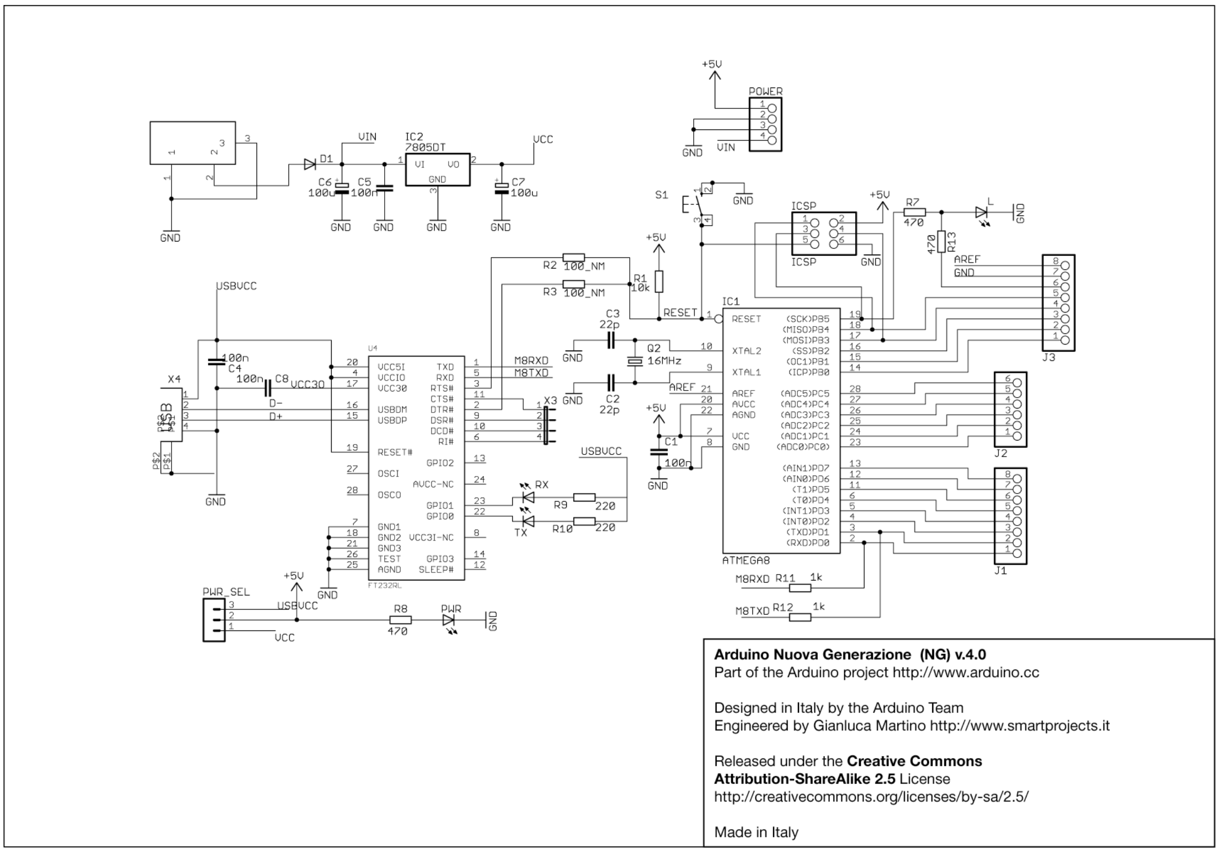

Arduino Nano Rear

Figure 5.9: Front and Rear view

Specifications

- **Microcontroller**: Atmel ATmega168 or ATmega328
- **Operating Voltage (logic level)**: 5 V
- **Input Voltage (recommended)**: 7-12 V
- **Input Voltage (limits)**: 6-20 V
- **Digital I/O Pins**: 14 (of which 6 provide PWM output)
- **Analog Input Pins**: 8
- **DC Current per I/O Pin**: 40 mA
- **Flash Memory**: 16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
- **SRAM**: 1 KB (ATmega168) or 2 KB (ATmega328)
- **EEPROM**: 512 bytes (ATmega168) or 1 KB (ATmega328)

- **Clock Speed**: 16 MHz
- **Dimensions**: 0.73” x 1.70”



Arduino Nuova Generazione (NG) v.4.0
Part of the Arduino project http://www.arduino.cc

Designed in Italy by the Arduino Team
Engineered by Gianluca Martino http://www.smartprojects.it

Released under the **Creative Commons Attribution-ShareAlike 2.5** License
http://creativecommons.org/licenses/by-sa/2.5/

Made in Italy

Figure 5.10: Schematics of Nano

Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

The FTDI FT232RL chip on the Nano is only powered if the board is being powered over USB. As a result, when running on external (non-USB) power, the 3.3V output (which is supplied by the FTDI chip) is not available and the RX and TX LEDs will flicker if digital pins 0 or 1 are high.

Memory

The ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader); the ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the EEPROM library); the ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).**Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.**These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM: 3, 5, 6, 9, 10, and 11.**Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).**These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.**There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the analogReference() function. Additionally, some pins have specialized functionality:

- **I2C: 4 (SDA) and 5 (SCL).**Support I2C (TWI) communication using the Wire library

There are a couple of other pins on the board:

- **AREF**: Reference voltage for the analog inputs. Used with analogReference().
- **Reset**: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

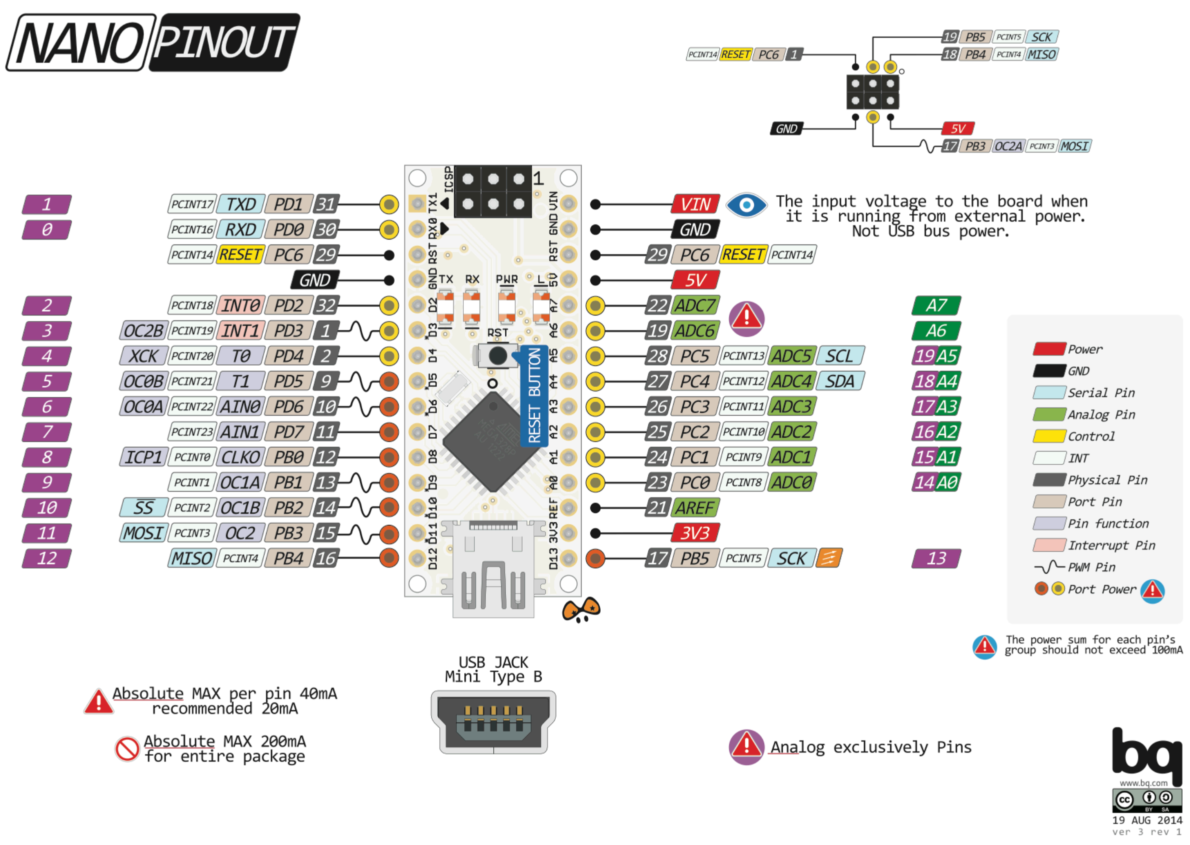

NANO PINOUT
The input voltage to the board when it is running from external power. Not USB bus power.
Absolute MAX per pin 40mA recommended 20mA
Absolute MAX 200mA for entire package
USB JACK Mini Type B
Analog exclusively Pins
The power sum for each pin's group should not exceed 100mA
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
RESET BUTTON
19 AUG 2014
ver 3 rev 1

Figure 5.11: Pinout of Nano

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega168 and ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

Automatic Software Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL is connected to the reset line of the ATmega168 or ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Nano is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Nano. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

5.1.3 Servo Motors

These motors generally consume a very less power and provide an accurate movement as per the programming. So these motors have a very high demand in the field of robotics. However, for our model, we have used these motors to act as gate controllers which allow or restrict the entrance of vehicles into the parking lots.

What is a Servo Motor

A servo motor is an electrical device which can push or rotate an object with great precision. If we want to rotate and object at some specific angles or distance, then we use servo motor. It is just made up of simple motor which run through servo mechanism. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages. Due to these features they are being used in many applications like toy car, RC helicopters and planes, Robotics, Machine etc.

Servo motors are rated in kg/cm (kilogram per centimeter).Most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the lesser the weight carrying capacity.

The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.

Figure 5.12: Servo Motor

Servo Mechanism

It consists of three parts:

1. Controlled device
2. Output sensor
3. Feedback system

It is a closed loop system where it uses positive feedback system to control motion and final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

The reference input signal is compared to reference output signal and the third signal is produces by feedback system. And this third signal acts as input signal to control device. This signal is present as long as feedback signal is generated or there is difference between reference input signal and reference output signal. So the main task of servomechanism is to maintain output of a system at desired value at presence of noises.

Working Principle

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly and a controlling circuit. First of all we use gear assembly to reduce RPM and to increase torque of motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now difference between these two signals, one comes from potentiometer and another comes from other source, will be processed in feedback mechanism and output will be provided in term of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with potentiometer and as motor rotates so the potentiometer and it will generate a signal.

So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

Controlling the Servo Motor

All motors have three wires coming out of them. Out of which two will be used for Supply (positive and negative) and one will be used for the signal that is to be sent from the MCU.

Servo motor is controlled by PWM (Pulse with Modulation) which is provided by the control wires. There is a minimum pulse, a maximum pulse and a repetition rate. Servo motor can turn 90 degree from either direction form its neutral position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90 degrees position, such as if pulse is shorter than 1.5ms shaft moves to 0 degrees and if it is longer than 1.5ms than it will turn the servo to 180 degrees.

Servo motor works on PWM (Pulse width modulation) principle, means its angle of rotation is controlled by the duration of applied pulse to its Control PIN. Basically servo motor is made up of DC motor which is controlled by a variable resistor (potentiometer) and some gears. High speed force of DC motor is converted into torque by Gears. We know that $WORK = FORCE \times DISTANCE$, in DC motor Force is less and distance (speed) is high and in Servo, force is High and distance is less. Potentiometer is connected to the output shaft of the Servo, to calculate the angle and stop the DC motor on required angle.

Servo motor can be rotated from 0 to 180 degree, but it can go up to 210 degree, depending on the manufacturing. This degree of rotation can be controlled by applying the Electrical Pulse of proper width, to its Control pin. Servo checks the pulse in every 20 milliseconds. Pulse of 1 ms (1 millisecond) width can rotate servo to 0 degree, 1.5ms can rotate to 90 degree (neutral position) and 2 ms pulse can rotate it to 180 degree.

All servo motors work directly with a +5V supply rails but we have to be careful on the amount of current the motor would consume, if we are planning to use more than two servo motors a proper servo shield should be designed.

5.1.4 Bread Board

A breadboard is a construction base for prototyping of electronics. Originally it was literally a bread board, a polished piece of wood used for slicing bread. In the 1970s the solderless breadboard (a.k.a. plugboard, a terminal array board) became available and nowadays the term ”breadboard” is commonly used to refer to these.

Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason, solderless breadboards are also extremely popular with students and in technological education. Older breadboard types did not have this property. A stripboard (Veroboard) and similar prototyping printed circuit boards, which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

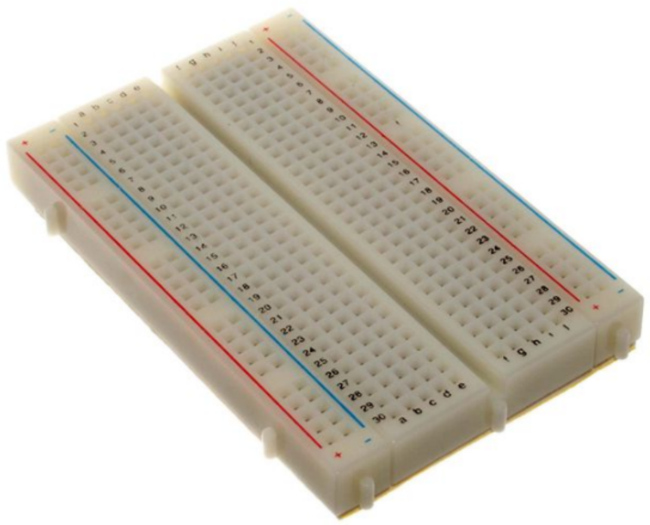


Figure 5.13: A typical BreadBoard

Solderless breadboard

A modern solderless breadboard socket consists of a perforated block of plastic with numerous tin plated phosphor bronze or nickel silver alloy spring clips under the perforations. The clips are often called tie points or contact points. The number of tie points is often given in the specification of the breadboard.

The spacing between the clips (lead pitch) is typically 0.1 in (2.54 mm). Integrated circuits (ICs) in dual in-line packages (DIPs) can be inserted to straddle the centerline of the block. Interconnecting wires and the leads of discrete components (such as capacitors, resistors, and inductors) can be inserted into the remaining free holes to complete the circuit. Where ICs are not used, discrete components and connecting wires may use any of the holes. Typically the spring clips are rated for 1 ampere at 5 volts and 0.333 amperes at 15 volts (5 watts). The edge of the board has male and female dovetail notches so boards can be clipped together to form a large breadboard.

Bus and terminal strips

Solderless breadboards are available from several different manufacturers, but most share a similar layout. The layout of a typical solderless breadboard is made up from two types of areas, called strips. Strips consist of interconnected electrical terminals.

- **Terminal strips**

  The main areas, to hold most of the electronic components.

  In the middle of a terminal strip of a breadboard, one typically finds a notch running in parallel to the long side. The notch is to mark the centerline of the terminal strip and provides limited airflow (cooling) to DIP ICs straddling the centerline.

  The clips on the right and left of the notch are each connected in a radial way; typically five clips (i.e., beneath five holes) in a row on each side of the notch are electrically connected. The five rows on the left of the notch are often marked as A, B, C, D, and E, while the ones on the right are marked F, G, H, I and J. When a ”skinny” dual in-line pin package (DIP) integrated circuit (such as a typical DIP-14 or DIP-16, which have a 0.3-inch (7.6 mm) separation between the pin rows) is plugged into a breadboard, the pins of one side of the chip are supposed to go into row E while the pins of the other side go into row F on the other side of the notch. The columns are numbered 1 - 50 or whatever number of columns there are.

- **Bus strips**

  To provide power to the electronic components.
  A bus strip usually contains two rows: one for ground and one for a supply voltage. However, some breadboards only provide a single-row power distributions bus strip on each long side. Typically the row intended for a supply voltage is marked in red, while the row for ground is marked in blue or black. Some manufacturers connect all terminals in a column. Others just connect groups of, for example, 25 consecutive terminals in a column. The latter design provides a circuit designer with some more control over crosstalk (inductively coupled noise) on the power supply bus. Often the groups in a bus strip are indicated by gaps in the color marking.

Bus strips typically run down one or both sides of a terminal strip or between terminal strips. On large breadboards additional bus strips can often be found on the top and bottom of terminal strips.

Some manufacturers provide separate bus and terminal strips. Others just provide breadboard blocks which contain both in one block. Often breadboard strips or blocks of one brand can be clipped together to make a larger breadboard.

In a more robust variant, one or more breadboard strips are mounted on a sheet of metal. Typically, that backing sheet also holds a number of binding posts. These posts provide a clean way to connect an external power supply. This type of breadboard may be slightly easier to handle. Several images in this article show such solderless breadboards.

Jump Wires

Jump wires (also called jumper wires) for solderless breadboarding can be obtained in ready-to-use jump wire sets or can be manually manufactured. The latter can become tedious work for larger circuits. Ready-touse jump wires come in different qualities, some even with tiny plugs attached to the wire ends. Jump wire material for ready-made or homemade wires should usually be 22 AWG (0.33 mm2) solid copper, tin-plated wire - assuming no tiny plugs are to be attached to the wire ends. The wire ends should be stripped $\frac{3}{16}$ to $\frac{5}{16}$ in (4.8 to 7.9 mm). Shorter stripped wires might result in bad contact with the board's spring clips (insulation being caught in the springs). Longer stripped wires increase the likelihood of short-circuits on the board. Needle-nose pliers and tweezers are helpful when inserting or removing wires, particularly on crowded boards.

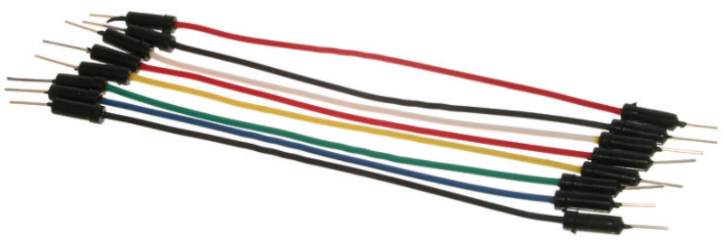


Figure 5.14: Jump Wires

Differently colored wires and color-coding discipline are often adhered to for consistency. However, the number of available colors is typically far fewer than the number of signal types or paths. Typically, a few wire colors are reserved for the supply voltages and ground (e.g., red, blue, black), some are reserved for main signals, and the rest are simply used where convenient. Some ready-to-use jump wire sets use the color to indicate the length of the wires, but these sets do not allow a meaningful color-coding schema.

5.1.5 ESP8266 Wi-Fi Module

ESP-12E WiFi module is developed by Ai-thinker Team. core processor ESP8266 in smaller sizes of the module encapsulates Tensilica L106 integrates industry-leading ultra low power 32-bit MCU micro, with the 16-bit short mode, Clock speed support 80 MHz, 160 MHz, supports the RTOS, integrated Wi-Fi MAC/BB/RF/PA/LNA, on-board antenna.

The module supports standard IEEE802.11 b/g/n agreement, complete TCP/IP protocol stack. Users can use the add modules to an existing device networking, or building a separate network controller.

ESP8266 is high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed Wi-Fi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

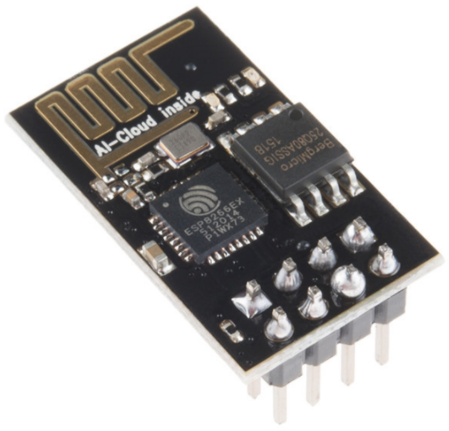


Figure 5.15: ESP8266 Module

ESP8266EX offers a complete and self-contained Wi-Fi networking solution; it can be used to host the application or to offload Wi-Fi networking functions from another application processor. When ESP8266EX hosts the application, it boots up directly from an external flash. In has integrated cache to improve the performance of the system in such applications.

Alternately, serving as a Wi-Fi adapter, wireless internet access can be added to any micro controllerbased design with simple connectivity (SPI/SDIO or I2C/UART interface).

ESP8266EX is among the most integrated WiFi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices

Micro-controller unit

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. ESP8266EX is often integrated with external sensors and other specific devices through its GPIOs.

Internal SRAM and ROM

ESP8266EX WiFi SoC is embedded with memory controller, including SRAM and ROM. MCU can visit the memory units through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

- RAM size < 36kB, that is to say, when ESP8266EX is working under the station mode and is connected to the router, programmable space accessible to user in heap and data section is around 36kB.
- There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

5.2 System Setup and Working

The mechanism of our proposed system is quite simple, initially an user by registering can opt for an parking space. The web application will provide the best optimal parking lot where the user has to park the car. Once the car is parked the sensor will detect and automatically update the data in firebase. After the value is updated in database web app will simply fetch the data and will update itself for the next user. The process goes same for the time when car leaves the parking space.

5.2.1 Setup

Our system depends on a set of hardware and software to function.

Hardware Setup

In our hardware we are using 2 microprocessors i.e. **Node-MCU** and **Arduino NANO**. Along with that we are using **Infra-red Proximity(IR) sensor** to detect the presence of vehicle in the parking space. Both the microprocessors are connected to each other using master slave connection otherwise known as I^2C connection where Node-MCU is the master and Arduino NANO is the slave node. Across the Arduino NANO, all the IR sensors are connected. So whenever the IR sensor will detect some presence, it will be transmitted to the Arduino micro-controller, which will send this data to the master node that is Node-MCU. Across the Node-MCU Wireless LAN(ESP8266) is present which will transmit data to the cloud database i.e. Firebase.

Software Setup

To store the data in database we are using **FIREBASE** a online database software provided by GOOGLE. The job of the database is that it simply acts as an intermediate between our hardware setup and our web app. It stores the values i.e. detected by the sensors and these values are fetched by web app.

In our software setup, we have designed a web application named **PARK IT**. The web application is entirely written using JSP, HTML, CSS, Java Script and BOOTSTRAP. For deployment purpose, we are using Apache Tomcat as our web server. In the application, basically a user has to do registration first by providing his/her personal details. After that we have implemented the simulated algorithm to find the best optimal parking lot by considering the values of distance to the parking lot and space occupied (which data is constantly being updated from the firebase). When user is directed for a parking space he/she is provided with an option to reserve it. Finally when the user parks the car, the data regarding to the status is being fetched by the web application from the firebase. In the end, bill for parking service is displayed to the user.

5.2.2 Working

As discussed in the software setup initially, users who want to avail this opportunity have to first register themselves by providing his personal details. Once that is done as per the algorithm he/she will be directed towards a best optimal parking lot. Then the user will be provided with an option whether he wants reserve that parking space or not. If the user reserves one space, a particular time interval will be provide to the user. Within that time interval the user has to park the car or else he will be faced with some penalties.

Once the user parks the car in directed parking lot then the IR sensor which is present across each parking space will detect the presence of the car once it reaches the territory or range of IR sensor. When the sensor detects the data it will send it to the Arduino micro-processor, where it will forward the data to the master i.e. node-mcu in a wired I^2C connection.

When the node-mcu receives the data sent by Arduino now it will send it to Firebase which is an online database provided by google. Once the data is updated across the firebase, the web app will immediately fetch the data and update the value of the occupied variable. So once the value is updated in the web app, we will have a new value of $f(a, b)$.

In this manner when we park the car or leave the parking lot, the values will constantly increment and decrement and we can make all the parking lots of the network completely balanced along with that solve the issue parking space availability.

5.3 Summary

This chapter explains how the implementation of the simulated system can be done in real-life scenarios. We have implemented the whole system with the help of sensory network, IoT and cloud systems.

Chapter 6

Conclusion and FutureWork

This study has proposed a parking system that improves performance by reducing the number of users that fail to find a parking space and minimizes the costs of moving to the parking space. Our proposed architecture and system has been successfully simulated and implemented which can be further extended to be applicable in a real system.

The results show that our algorithm perfectly distributes the vehicular density amongst the parking nodes. Our results closely agree with those of our proposed mathematical models. The simulation of our system achieved the optimal solution when most of the vehicles successfully found a free parking space.

For future-work, we have considered four things.

1. RFID implementation: RFID stands for "Radio Frequency Identification". This technology can be implemented in our proposed system which can ensure that the person who reserved a parking lot will be the one to park the car in that spot. It can be used to improvise the security measures of the system.

2. Image Processing: Instead of using a large number of sensors in parking lots with a high capacity, image processing techniques can be implemented which can be used to keep track of multiple parking lots at a time without installing a sensor in each individual slot. This can prove to be much cost efficient and less power consuming.

3. Wireless Connections: Some areas that can't be covered with the help of image processing, i.e., areas which are outside of the Field-of-view of the camera will still require sensors to be installed. But in that case, a wired connection might make the system more clumsy and complex. A wireless connection of sensors will prove beneficial rather than a wired connection.

4. Range of IR senors: The effective coverage area of an IR sensor is very less, about 2 to 30 cm. Again it requires an elevation angle of 35 degrees of detection. For these limitations we might switch to ultra-sonic sensors or pressure transducers as the basic sensing elements.