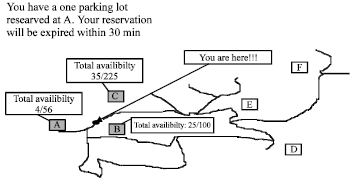
**1 EXISTING SYSTEM**

Searching for a vacant parking space in a metropolitan area is the daily concern for most drivers, and it is time-consuming. It commonly results more traffic congestion and air pollution by constantly cruising in certain area only for an available parking space. For instance, 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. To alleviate such traffic congestion and improve the convenience for drivers, many smart parking systems aiming to satisfy the involved parties (e.g., parking service provider sand drivers) have been deployed. The current smart parking or parking guidance systems only obtain the availability information of parking spaces from deployed sensor networks, and simply publish the parking information to direct drivers. However, since these systems cannot guide the drivers to their desired parking destinations, even sometimes make the situation worse, they are not “smart” enough. For instance, when the number of vacant spaces in an area is limited, more drivers, who obtain the parking information, are heading for these spaces. It will cause severer congestion. It is, therefore, strongly desired to provide an effective strategy to address these concerns. In this paper, we design and implement the prototype of a Reservation-based Smart Parking System (RSPS) not only to broadcast real-time parking information to the drivers as part of a communal application, but also to provide reservation service as part of user-targeted service.



**Fig:1.1.1 Avaliable slots for parking**

**2 PROPOSED METHODOLOGY**

The proposed system is the combination of smart parking and the Slot allocation with the IOT using Banana pi. In the existing system, a dynamic algorithm is carried out, which is a random detection method. It displays the available parking slots to the users.



**Fig 2:Proposed Parking System**

The features of the proposed system are;

▪ Guides drivers find available parking spaces near them

▪ Less number of drivers searching to park, thus reduces the traffic congestion

▪ Avoids air pollution & global warming.

▪ Scalable, robust and reliable.

▪ Reduces the driver stress and improves the urban area.

▪ Provides tools to optimize the parking space management

▪ Accurately find out the vehicle occupancy in real time

.

**3 Banana pi BPI-M1+ (BPI-M1 Plus)**

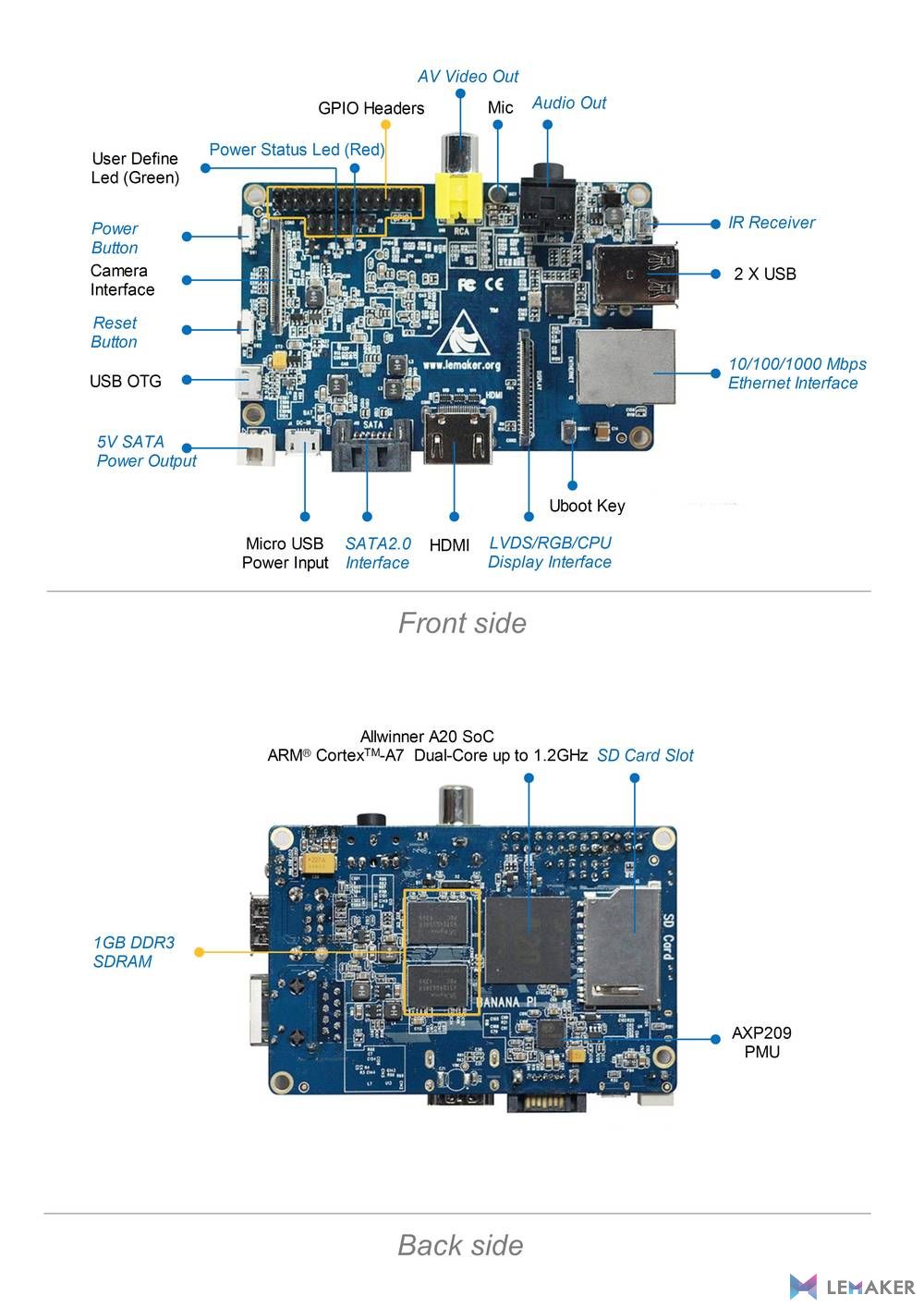
**Banana Pi** - Credit Card Sized computing machine embedded with ARM processor running on Debian Linux OS. Pi Device capable of connecting digital devices to receive and send signals through 26 General Purpose Input/output PINS through voltage modulation of binary formats

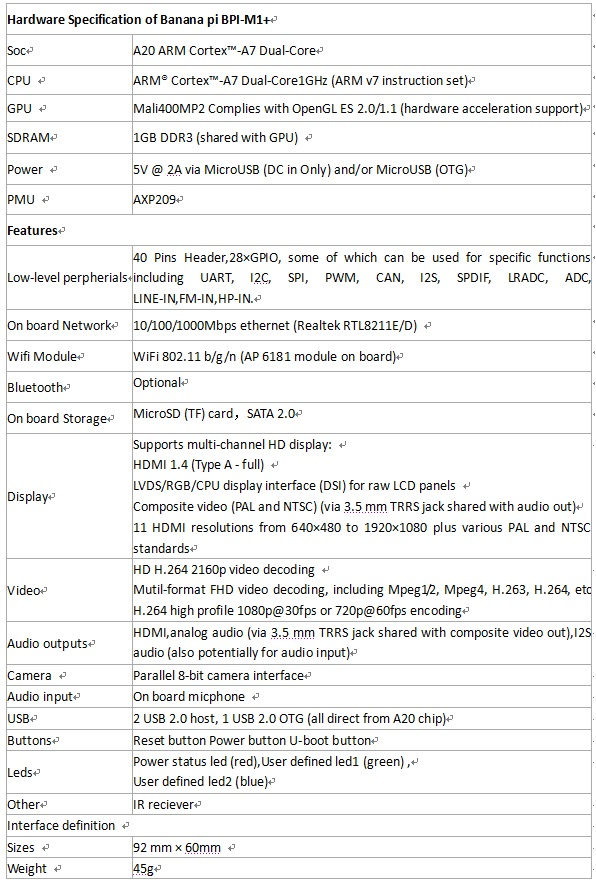
Banana PI BPI-M1+ is the open source hardware platform, Banana PI BPI-M1+ is the dual core Android 4.4 product which more better than the Raspberry Pi. Banana Pi BPI-M1+ series run Android, Debian Linux, Ubuntu Linux, Raspberry Pi image and cubieboard image.

Banana PI BPI-M1+ hardware: 1Ghz ARM7 dual-core processor, 1GB DDR3 SDRAM, WIFI support onboard Banana PI BPI-M1+ with Gigabit ethernet port, SATA Socket. It can run with Android 4.4 smoothly.

The size of Banana PI BPI-M1+ like the credit card, it can easily run with the game it support 1080P high definition video output, the GPIO compatible with Raspberry Pi and can run raspbian image.

**BPI-M1+ hardware interface**

**Fig: 2.3.1 Banana PI**



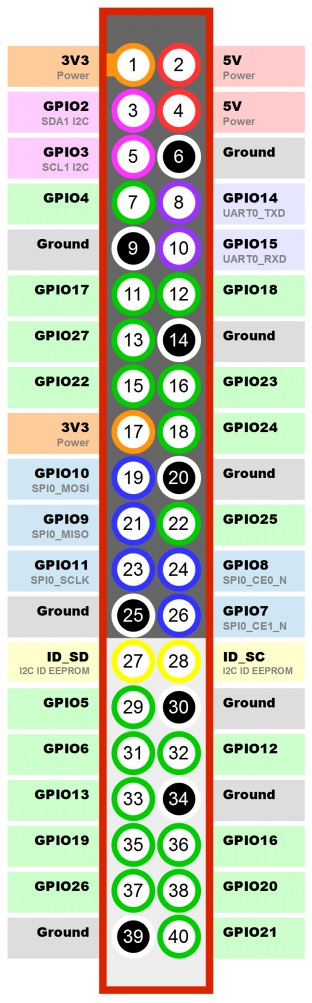
**Fig 2.3.2** :**BananaPI-M1+ hardware specifications**

**BPI-M1+ GPIO pin define**

**GPIO specification**

**Banana Pi 40-pin GPIO Banana Pi has a 40-pin GPIO header that matches th at of the Model B+ Raspberry Pi. Following is the Banana Pi GPIO**

**Pinout:**



**Table: 2.3.3 Pin diagram**

|  |  |  |
| --- | --- | --- |
| GPIO Pin Name  CON3-P01  CON3-P02  CON3-P03  CON3-P04  CON3-P05  CON3-P06 | Default Function  VCC-3V3  VCC-DC  TWI2-SDA  VCC-DC  TWI2-SCK  GND | Function2：GPIO  PB21  PB20 |

**BPI-M1+ micro SD card slot**

BPI-M1+ have support a TF card slot. you can burn image to TF card ,and use it boot BPI-M1+ same as raspberry pi.



**Fig: 2.3.4 Micro SD card**

Note:

Support 8G 16G 32G 64G

Please choose class 10 TF card for banana pi.

**BPI-M1+ HDMI interface**

**BPI-M1+ has a standard HDMI 1.4 interface. so We can use HDMI-to-HDMI cable to connect BPI-M1 to the display monitor that has HDMI interface.**



**Fig: 2.3.5 Cables**

**But If the display monitor doesn't have HDMI interface, only VGA or DVI port. We should use HDMI-to-VGA or HDMI-to-DVI cable to connect the BPI-M1+ to the display monitor.**



**Fig: 2.3.6 Cables**

**Note: if the HDMI-to-VGA/DVI cable is a bad quality cable, it will go wrong on the monitor display. Please choose a good quality cable for BPI-M1+**

**BPI-M1+ IR interface**

**BPI-M1+ support IR interface on board. You can use it as remote control.**

**How to use IR interface**

**Test on bananian image:**

**1**,install lirc

apt-get install lirc

apt-get install evtest

**2**,edit /etc/lirc/hardware.conf as below:

nano /etc/lirc/hardware.conf

# /etc/lirc/hardware.conf

#

# Arguments which will be used when launching lircd

LIRCD\_ARGS="--uinput"

#Don't start lircmd even if there seems to be a good config file

#START\_LIRCMD=false

#Don't start irexec, even if a good config file seems to exist.

#START\_IREXEC=false

#Try to load appropriate kernel modules

LOAD\_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.

DRIVER="UNCONFIGURED"

# usually /dev/lirc0 is the correct setting for systems using udev

DEVICE="/dev/input/event0"

MODULES="sunxi-ir"

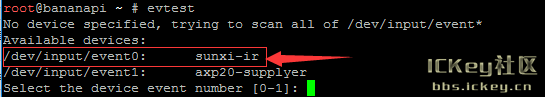
ctrl+O save and ctrl+x exit.

**3**,test lirc

service lirc start

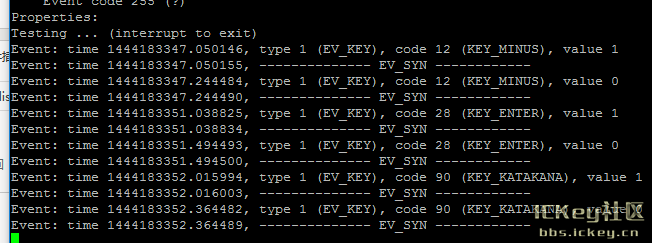
**4** test remote-control unit

Evtest



**Fig 2.3.7 Execution 1.**

Choose "0" must xunxi-ir



**Fig 2.3.7.1: Execution 2**

Please note: value 0 value 1

Press is: 1，unpress is:0

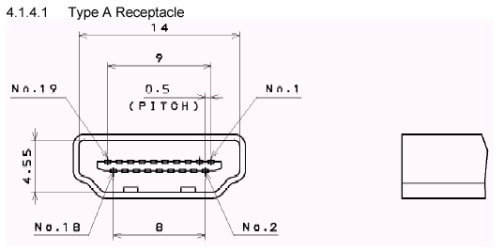
**BPI-M1+ USB interface:**

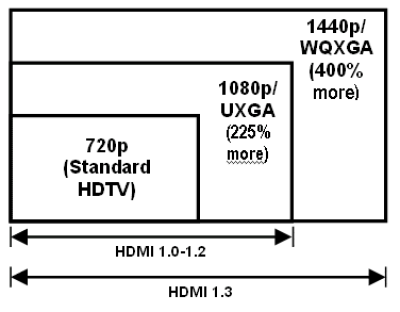
BPI-M1+ have two USB 2.0 interface on board.so you can connect Keyboard,mouse, USB camera and ... on BPI-M1+.

**HDMI:**

HDMI (High-Definition Multimedia Interface) is the first industry-supported, uncompressed, all-digital audio/video interface. HDMI provides an interface between any audio/video source, such as a set-top box, DVD player, and A/V receiver and an audio and/or video monitor, such as a digital television (DTV).

HDMI supports standard, enhanced, or high-definition video, plus multi-channel digital audio on a single cable. It transmits all ATSC HDTV standards and supports 8-channel digital audio, with bandwidth to spare to accommodate future enhancements and requirements.

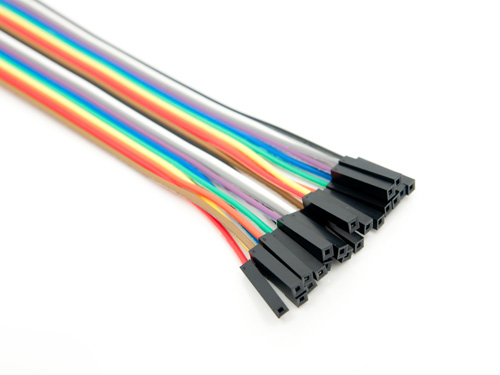




**Fig 2.3.8 HDMI**

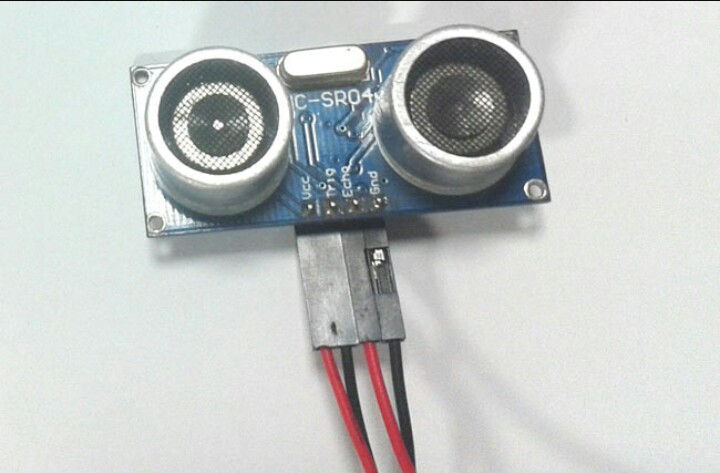
**Jumper Wire:**

A jumper wire is a conducting wire used to transfer electrical signals between two points in a circuit. The wires can either be used to modify circuits or to diagnose problems within a circuit



**Fig 2.3.9 Jumper wires**

**ULTRA SONIC SENSOR**

****

**Fig:2.3.10. UltraSonic Sensor**

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.

The accuracy of Ultrasonic sensor can be affected by the temperature and humidity of the air it is being used in. However, for these tutorials and almost any project you will be using these sensors in, this change in accuracy will be negligible.

It is important to understand that some objects might not be detected by ultrasonic sensors. This is because some objects are shaped or positioned in such a way that the sound wave bounces off the object, but are deflected away from the Ultrasonic sensor. It is also possible for the object to be too small to reflect enough of the sound wave back to the sensor to be detected. Other objects can absorb the sound wave all together (cloth, carpeting, etc), which means that there is no way for the sensor to detect them accurately. These are important factors to consider when designing and programming a robot using an ultrasonic sensor.

**Specifications**

* Echo Pulse Output
* 0V Ground

**Electric Parameter**

* Working Voltage DC 5 V
* Working Current 15mA
* Working Frequency 40Hz
* Max Range 4m
* Min Range 2cm
* Measuring Angle 15 degree
* Trigger Input Signal 10uS TTL pulse
* Echo Output Signal Input TTL lever signal and the range in
* proportion
* Dimension 45\*20\*15mm



# Fig:2.3.11. Pin Configuration

1. VCC: 5V DC
2. GND: ground
3. DO: high/low output AO: analog output

**4 SOFTWARE REQUIREMENTS**

**Operating System** **:** Linux

**Languages** **:** Python

**5 HARDWARE REQUIREMENTS**

**BANANA PI BPI-M1+ – 1**

**ULTRASONIC SENSOR – 1**

**MICRO SD CARD – 4 GB**

**BPI-M1+ HDMI interface – 1**

**HDMI-to-VGA/DVI cable – 1**

**JUMPER WIRE – 4**

## 6 UML DIAGRAMS:

## 6.1 USECASE DIAGRAM

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behaviour means the behaviour of the system when it is running /operating.

So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

So to model the entire system numbers of use case diagrams are used.

Purpose

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because other four diagrams (activity, sequence, collaboration and Statechart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modelled to present the outside view.

So in brief, the purposes of use case diagrams can be as follows:

* Used to gather requirements of a system.
* Used to get an outside view of a system.
* Identify external and internal factors influencing the system.
* Show the interacting among the requirements are actors.

**How to draw Use Case Diagram?**

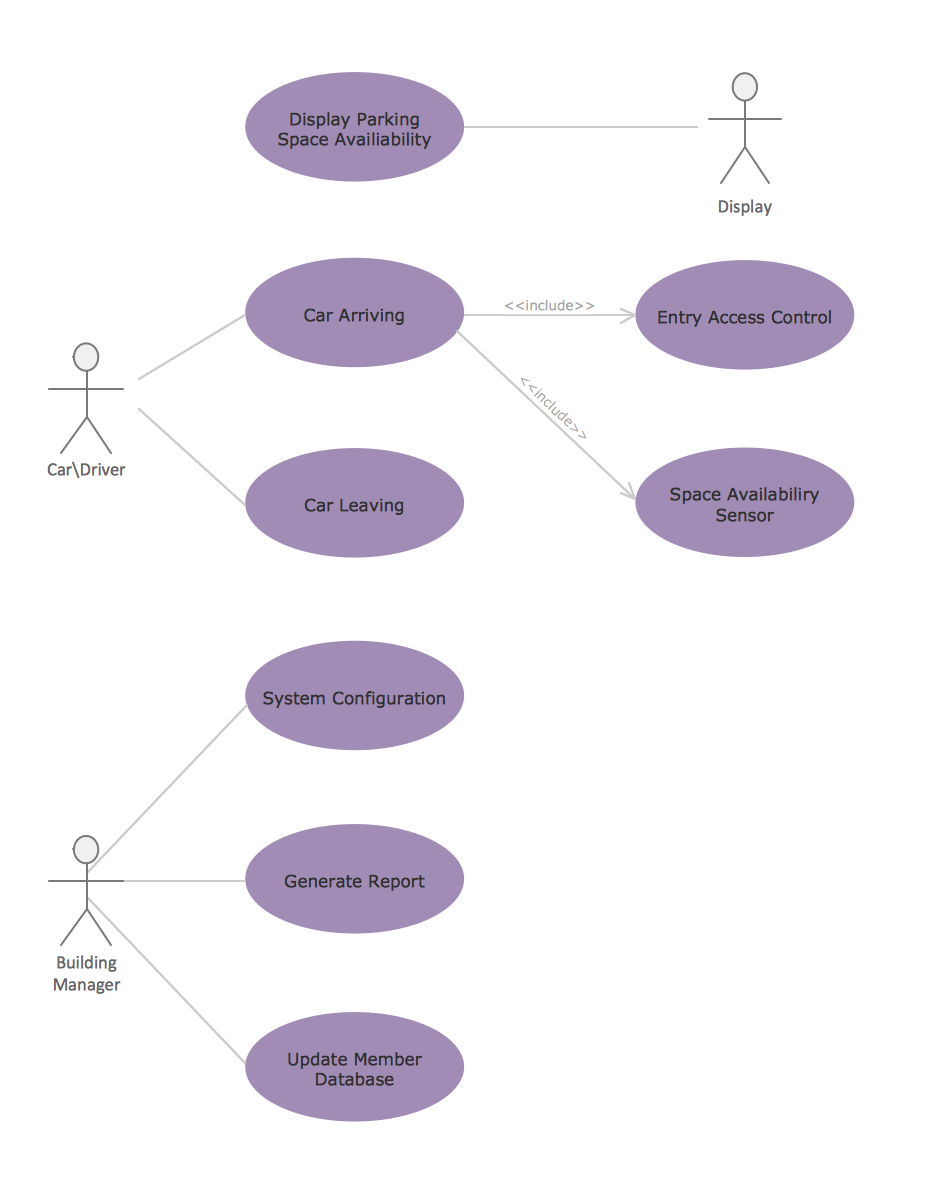
Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases.

So we can say that use cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

The actors can be human user, some internal applications or may be some external applications. So in a brief when we are planning to draw an use case diagram we should have the following items identified.

* Functionalities to be represented as an use case
* Actors
* Relationships among the use cases and actors.

Use case diagrams are drawn to capture the functional requirements of a system. So after identifying the above items we have to follow the following guidelines to draw an efficient use case diagram.

* The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
* Give a suitable name for actors.
* Show relationships and dependencies clearly in the diagram.
* Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
* Use note when ever required to clarify some important points.
* The following is a sample use case diagram representing the Smart Parking System. 

**Fig: 6.1 USECASE DIAGRAM**

**6.2. Class Diagram**

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*.

## Purpose:

The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction.

The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

So the purpose of the class diagram can be summarized as:

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

## How to draw Class Diagram?

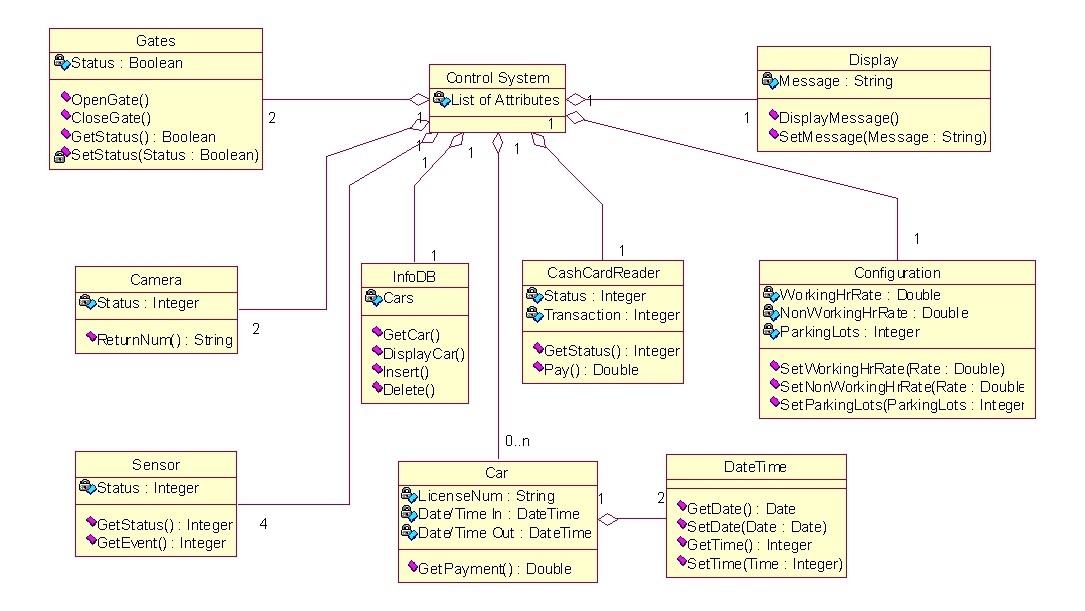
Class diagrams are the most popular UML diagrams used for construction of software applications. So it is very important to learn the drawing procedure of class diagram.

Class diagrams have lot of properties to consider while drawing but here the diagram will be considered from a top level view.

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system.

The following points should be remembered while drawing a class diagram:

* The name of the class diagram should be meaningful to describe the aspect of the system.
* Each element and their relationships should be identified in advance.
* Responsibility (attributes and methods) of each class should be clearly identified.
* For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
* Use notes when ever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
* Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

**Class diagrams **

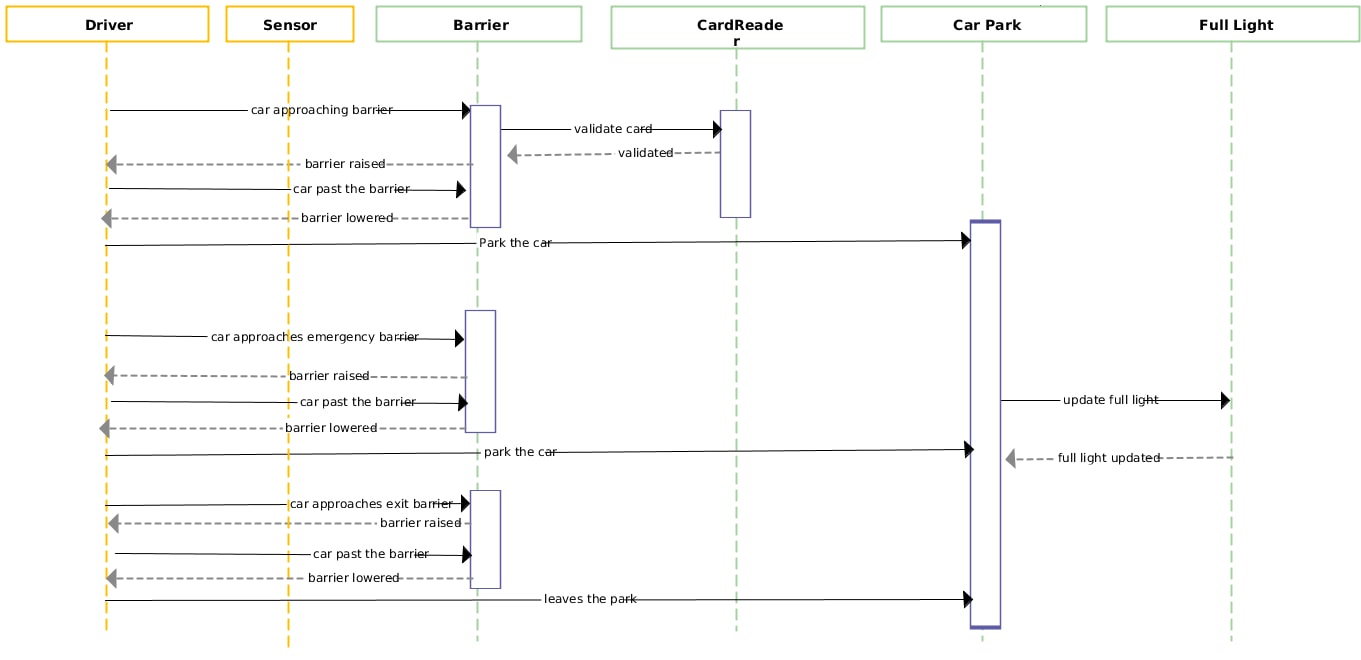
**Fig:6.2 class diagram of Smart Parking System**

**6.3 SEQUENCE DIAGRAM**

UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include [activity diagramming](http://agilemodeling.com/artifacts/activityDiagram.htm), [communication diagramming](http://agilemodeling.com/artifacts/communicationDiagram.htm), [timing diagramming](http://agilemodeling.com/artifacts/timingDiagram.htm), and [interaction overview diagramming](http://agilemodeling.com/artifacts/interactionOverviewDiagram.htm). Sequence diagrams, along with [class diagrams](http://agilemodeling.com/artifacts/classDiagram.htm) and [physical data models](http://agiledata.org/essays/dataModeling101.html) are in my opinion the most important design-level models for modern business application development.

Sequence diagrams are typically used to model:

1. **Usage scenarios**. A usage scenario is a description of a potential way your system is used. The logic of a usage scenario may be part of a use case, perhaps an alternate course. It may also be one entire pass through a use case, such as the logic described by the basic course of action or a portion of the basic course of action, plus one or more alternate scenarios. The logic of a usage scenario may also be a pass through the logic contained in several use cases. For example, a student enrolls in the university, and then immediately enrolls in three seminars.
2. **The logic of methods**. Sequence diagrams can be used to explore the logic of a complex operation, function, or procedure. One way to think of sequence diagrams, particularly highly detailed diagrams, is as [visual object code](http://www.agilemodeling.com/artifacts/sequenceDiagram.htm#VisualCoding).
3. **The logic of services**. A service is effectively a high-level method, often one that can be invoked by a wide variety of clients. This includes web-services as well as business transactions implemented by a variety of technologies such as CICS/COBOL or CORBA-compliant object request brokers (ORBs).

****

**Fig:6.3 SEQUENCE DIAGRAM**

**7. IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered as a most official stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning investigation of the existing system and its constraints on implementation designing of methods to changeover and evaluation of changeover methods.

The parking system is designed in such a way that it is applicable for covered parks,

open parks and street side parking. The fig.1 shows the cloud based IOT architecture for smart parking system which contains cloud service provider which provides cloud storage to store information about status of parking slots in a parking area and etc. [10]. The centralized server which manages to store entire smart parking systems information such as number of slots, availability of vehicles etc. And these information will be accessed through some secured gateways through network.

This smart parking system which consists of several components. And theirs functionality includes:

• Centralized server: maintains databases which contain information about parking spaces present in the city.

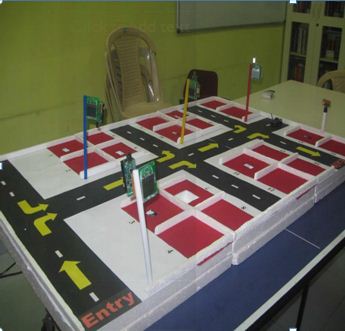
• Raspberry pi: the microcontroller which is used to implement our parking system and it is attached with raspberry pi camera.

• Image capture: Pi-camera is used to capture the picture of parking area continuously to validate the slots which either filled or empty.

• Navigation system: signals the availability of parking slots to the users and navigates to the exact location of nearest parking area from current location.

• Display device: a monitor or tab is used to display the admin side interface and he is capable of modifying the parking lots by observing the device.

• User device: user can connect with the smart parking system with their smart phones or with some browser



**Fig:7.1 Proto Type of Smart Parking System**

1. Banana Pi device configured with Ultrasound Sensor initiates consecutive sound waves to identify object presence to know if it is closer.

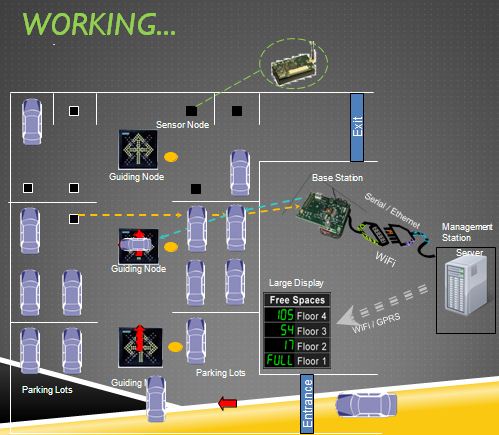
2. Sensor recognizes the location distance and sends back to Pi, further If identified that sensor produces same parameter fur further few seconds it is confirmed as lot to be booked

3. Banana Pi sends notification to the Pi Camera which is hosted approximately straight forward to the vehicle number plate to take snap of the number plate details

4. Banana Pi camera takes the picture of number plate and sends the GIF image to raspberry Pi

5 Banana Pi makes use of wiring Pi to receive incoming image file and invokes OCR4J framework for rasterizing the image to find the numbers on it. Lot details, vehicle number will be sending to the

6. Other users who commute through the traffic could know the availability of remaining slots in the parking lot and could get in easily

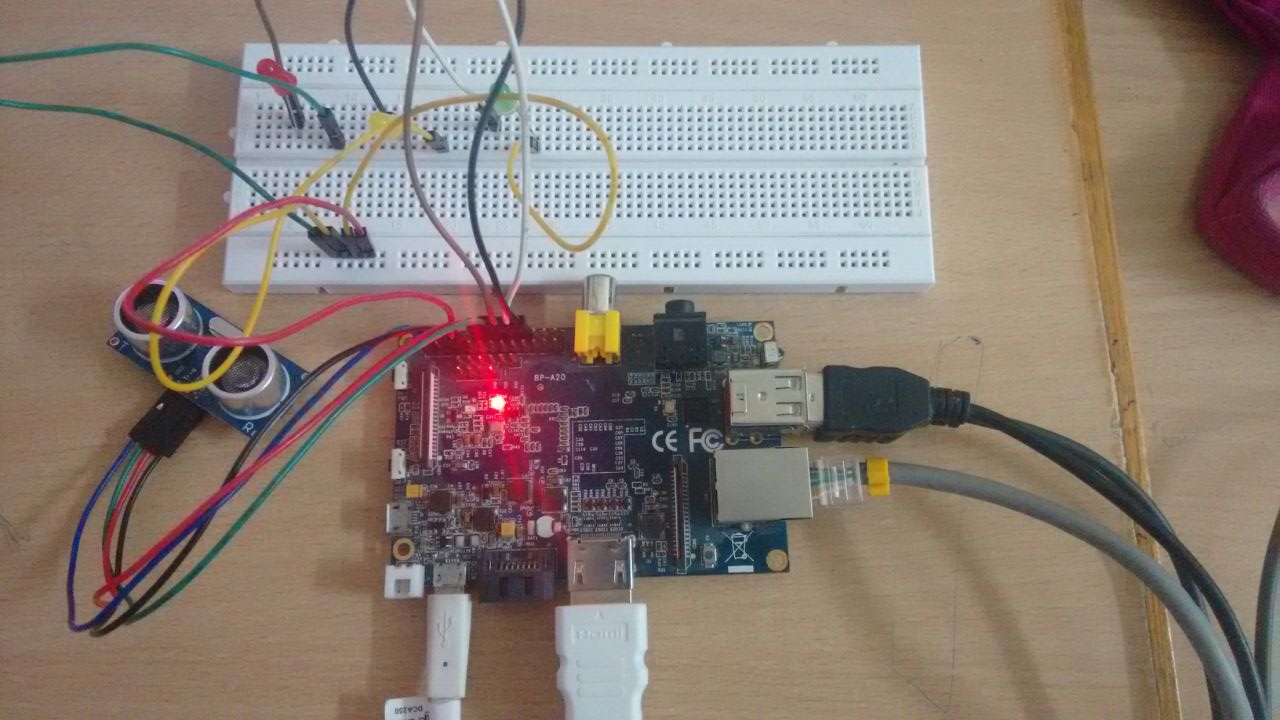


**Fig:7.2.Entry and Exit of Parking Area**

.

**Wiring Pi** - C Library written especially for accessing GPIO PINs for Raspberry Pi. It facilitates to read even analogue signals through the Pi device. There are several frameworks extending the WiringPI to access GPIO signals. PI4J is an example that we leverage here in our proposal for reading sensor signals.

**Circuit diagram**



**Fig: 7.3 Circuit Diagram**

**8. TESTING**

# Testing Your Code

Testing your code is very important.

Getting used to writing testing code and running this code in parallel is now considered a good habit. Used wisely, this method helps you define more precisely your code’s intent and have a more decoupled architecture.

Some general rules of testing:

* A testing unit should focus on one tiny bit of functionality and prove it correct.
* Each test unit must be fully independent. Each test must be able to run alone, and also within the test suite, regardless of the order that they are called. The implication of this rule is that each test must be loaded with a fresh dataset and may have to do some cleanup afterwards. This is usually handled by **setUp()** and **tearDown()** methods.
* Try hard to make tests that run fast. If one single test needs more than a few milliseconds to run, development will be slowed down or the tests will not be run as often as is desirable. In some cases, tests can’t be fast because they need a complex data structure to work on, and this data structure must be loaded every time the test runs. Keep these heavier tests in a separate test suite that is run by some scheduled task, and run all other tests as often as needed.
* Learn your tools and learn how to run a single test or a test case. Then, when developing a function inside a module, run this function’s tests frequently, ideally automatically when you save the code.
* Always run the full test suite before a coding session, and run it again after. This will give you more confidence that you did not break anything in the rest of the code.
* It is a good idea to implement a hook that runs all tests before pushing code to a shared repository.
* If you are in the middle of a development session and have to interrupt your work, it is a good idea to write a broken unit test about what you want to develop next. When coming back to work, you will have a pointer to where you were and get back on track faster.
* The first step when you are debugging your code is to write a new test pinpointing the bug. While it is not always possible to do, those bug catching tests are among the most valuable pieces of code in your project.
* Use long and descriptive names for testing functions. The style guide here is slightly different than that of running code, where short names are often preferred. The reason is testing functions are never called explicitly. square() or even sqr() is ok in running code, but in testing code you would have names such as test\_square\_of\_number\_2(),test\_square\_negative\_number(). These function names are displayed when a test fails, and should be as descriptive as possible.
* When something goes wrong or has to be changed, and if your code has a good set of tests, you or other maintainers will rely largely on the testing suite to fix the problem or modify a given behavior. Therefore the testing code will be read as much as or even more than the running code. A unit test whose purpose is unclear is not very helpful in this case.
* Another use of the testing code is as an introduction to new developers. When someone will have to work on the code base, running and reading the related testing code is often the best thing that they can do to start. They will or should discover the hot spots, where most difficulties arise, and the corner cases. If they have to add some functionality, the first step should be to add a test to ensure that the new functionality is not already a working path that has not been plugged into the interface.

### Unit Test

[**Unit Test**](http://docs.python.org/library/unittest.html#module-unittest) is the batteries-included test module in the Python standard library. Its API will be familiar to anyone who has used any of the JUnit/nUnit/CppUnit series of tools.

Creating test cases is accomplished by sub classing **[unittest.TestCase](http://docs.python.org/library/unittest.html" \l "unittest.TestCase" \o "(in Python v2.7))**.

### Doctest

The **[doctest](http://docs.python.org/library/doctest.html" \l "module-doctest" \o "(in Python v2.7))** module searches for pieces of text that look like interactive Python sessions in docstrings, and then executes those sessions to verify that they work exactly as shown.

Doctests have a different use case than proper unit tests: they are usually less detailed and don’t catch special cases or obscure regression bugs. They are useful as an expressive documentation of the main use cases of a module and its components. However, doctests should run automatically each time the full test suite runs.

# How to Test

The components to be used are:

* Microcontroller (any compatible bananapi)
* Ultra Sonic Sensor module
* Pin connectors
* USB cable

1. Connect the components based on the figure shown in the wiring diagram using pin connectors. VCC power supply is connected to the pin 2, GND is connected to the pin 6, DO pin is connected to a digital I/O pin(pin-11) and the AO pin is connected to the analog output pin(pin-13). Pin number will be based on the actual program code.
2. After hardware connection, insert the sample sketch into the bananapi IDE.
3. Using a USB cable, connect the ports from the microcontroller to the computer.
4. Upload the program.
5. See the results in the serial monitor.

# Testing Results



**Fig:8.1.Available Slots**