

# **IoT Communications using LPWAN**

**Report submitted in the partial fulfillment**

**Of**

**Master of Business Administration-Technology Management**

**In**

**Electronics and Telecommunications Engineering**

**By**

**Komal Ashok Mistry (J017)**

Under the supervision of

**Mr. Milap Jhumkawala**

(Head IoT Engineer, GetParking Pvt Ltd)

**Ms. Ashwini Gade**

(Assitant Professor, EXTC Department, MPSTME)

**SVKM's NMIMS University**

(Deemed-to-be University)



**MUKESH PATEL SCHOOL OF TECHNOLOGY  
MANAGEMENT & ENGINEERING  
Vile Parle (W), Mumbai-56**

**2016-17**

**CERTIFICATE**



This is to certify that the project entitled “**IoT Communications using LPWAN**”, has been done by **Ms. Komal Ashok Mistry** under my guidance and supervision for Technical Internship Program as part of curriculum of Master of Business Administration-TechnologyManagement in Electronics and Telecommunications engineering of MPSTME, SVKM’s NMIMS (Deemed-to-be University), Mumbai, India.

Mr.Milap Jhumkawala

Name of Industry Mentor

Ms. Ashwini Gade

Name of Internal Mentor

\_\_\_\_\_

Examiner

**Date: 10/07/2017**

**Place: Mumbai**

\_\_\_\_\_  
**Dr. Vaishali Kulkarni**  
**(H.O.D.EXTC)**

# ACKNOWLEDGEMENT

I take this opportunity to express my regards to my faculty guide **Ms. Ashwini Gade** for her exemplary guidance, monitoring and constant encouragement.

I also take this opportunity to express a deep sense of gratitude to my Industry Mentor **Mr. Milap Jhumkawala**, Head IoT Engineer at GetParking Ltd, for his cordial support, valuable information and guidance, which helped me in completing this task through various stages. I would also like to extend my gratitude to **Mr. Sachin Naik**, CEO at GetParking Ltd for giving me the opportunity and resources for carrying out my project.

I am obliged to all the members of GetParking Ltd, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Komal Ashok Mistry  
MBA.Tech. (EXTC)  
Roll No. J017

## ABSTRACT

The main goal of the project “**IoT Communications using LPWAN**” is to provide a feasible, scalable and economical solution for the existing WiFi enabled sensor communication employed by GetParking Ltd.

The extensive usage of WiFi leads to monumental operational and management costs and thus in the duration of this project the focus is to prototype new sensor assemblies relying on more economical communication technologies. The prototyping is done with the aim of practically testing alternative solutions to WiFi.

GetParking Ltd aims at providing smart solutions for parking by the extensive use of IoT technology streamlined by a cloud based app. This enables efficient ticketing and positional data to reach instantaneously to parking management teams. The generation of the said data is done by a sensor network which constantly monitors the environment and detects vehicle movements.

The proposed work flow is as follows:

1. Understanding the current WiFi dependent sensor architecture in place.
2. Extensive research on alternate communication technologies: GSM and LoRa
3. Estimating test architectural model for GSM communication.
4. Prototyping to understand communication architecture.
5. Prototyping for LoRa based communication.
6. Integrating the gateway assembly with an internet enabled database.

Thus, the major focus is to lay down a hardware structure that can enable Internet Of Things technology

## **INDEX**

<b>Chapter 1</b>	<b>INTRODUCTION</b>	
1.1	Industry introduction	7
1.2	Background of project topic	7
1.3	Motivation and scope of report	7
1.4	Salient contribution	8
1.5	Organization of report	8
<b>Chapter 2</b>	<b>LITERATURE SURVEY</b>	
2.1	Introduction to overall topic	
2.1.1	LPWAN / LoRaWAN	9
2.1.2	LoRa	11
2.1.3	LoRaWAN Architecture	12
2.1.4	Internet Of Things Framework	13
2.1.5	ThingSpeak IoT Database	14
2.2	Exhaustive Literature survey	
2.2.1	Shanon Hartley theorem	15
2.2.2	Spread Spectrum principles	16
2.2.3	LoRa modulation basics	18
2.2.4	Star network topology	18
2.2.5	Link budget comparision	19
<b>Chapter 3</b>	<b>PROBLEM STATEMENT</b>	20
<b>Chapter 4</b>	<b>METHODOLOGY</b>	
4.1	Generation of raw data	21
4.2	LoRa node	24
4.3	LoRa gateway	25
<b>Chapter 5</b>	<b>SYSTEM ANALYSIS</b>	27
<b>Chapter 6</b>	<b>SOFTWARE DESCRIPTION</b>	
6.1	Arduino IDE	28
6.2	Raspberry Pi coding environment	30
<b>Chapter 7</b>	<b>TESTING AND RESULTS</b>	31
<b>Chapter 8</b>	<b>ADVANTAGES AND LIMITATIONS</b>	
8.1	Advantages	32
8.2	Limitations	33
<b>Chapter 9</b>	<b>CONCLUSIONS AND FUTURE SCOPE</b>	34
<b>REFERENCES</b>		35
<b>Appendix A: Softcode</b>		36

**List of the figures**

<b>Fig. No.</b>	<b>Name of the figure</b>	<b>Page No.</b>
1.	LoRaWAN protocol stack	11
2.	LoRa logo	12
3.	LoRaWAN architecture	13
4.	Internet Of Things	14
5.	ThingSpeak database integration	16
6.	Shanon Hartley theorem	17
7.	Shanon Hartley theorem	23
8.	Magnetometer interfacing scheme	24
9.	LoRa shield assembly	25
10.	LoRa gateway	28
11.	Arduino	30
12.	Raspberry Pi	32
13.	Applications of LPWAN	37
14.	Raspberry Pi interface with GSM	38
15.	LoRa node	42
16.	LoRa receiver	46
17.	LoRa gateway	46

**List of tables**

<b>Table No.</b>	<b>Name of the Table</b>	<b>Page No.</b>
1.	LPWAN characteristics	<b>10</b>
2.	LPWAN specifications	<b>10</b>
3.	Comparision between LoRa and FSK	<b>19</b>
4.	Magnetometer interface	<b>22</b>
5.	Magnetometer pin scheme	<b>22</b>

**Abbreviations**

<b>Abbreviations</b>	<b>Full Form</b>
LoRa	Long Range
LPWAN	Low power wide area network
GSM	Global system for mobile communication

# **Chapter 1**

## **Introduction**

### **1.1 Introduction of the Industry**

GetParking Ltd is a smart solutions provider for commercial parking management and control. The parking management methods used in India are extremely dependent on traditional pen and paper monitoring done by onsite staff. This leads to intensive dependency on the credibility and reliability of labour. In order to reduce this redundancy smart technology like IoT (Internet Of Things) along with machine learning and cloud based app technology has been employed by GetParking to improve operational efficiency and reduce costs.

### **1.2 Background of the project topic**

The topic of this project is IoT communication using LPWAN (Low Power Wide Area Network). LPWAN technology is based upon the LPWAN protocol which uses LoRa. LoRa is a proprietary chirp spread spectrum digital modulation technique. The unique feature of LoRa is that it trades data rate for increased sensitivity within a given fixed bandwidth. The data rate lies within 0.3kbps to 50kbps which is extremely low as compared to WiFi and cellular technologies. Thus, only small bitstreams are optimally transmitted through LoRaWAN or LPWAN. But at the same time greater range (Upto 15 km in optimal environment) can be achieved using very less power.

This property of LPWAN makes it very attractive for use in machine to machine communication which doesn't involve the handling of huge data loads and dynamic data.

### **1.3 Motivation and scope of the report**

The main end goal of the project is to develop prototypes which use LPWAN and 2G GSM modules for communication rather than rely upon WiFi. For the fabrication of the

same, detailed study and analysis of the existing framework as well as the new technology is required. LoRa concepts and their implementation to the specific use case is focused upon. Also towards the end a comparative study between LoRa and 2G GSM technology will give better insight of which technology is better suited and why.

## **1.4 Salient contribution**

- 1.4.1 Deeper insight into Internet of Things technology and the possible use cases which can be developed.
- 1.4.2 Study of machine to machine communication and the basic requirements of the same.
- 1.4.3 Brief study and introduction to working of sensors and the medium of data collection
- 1.4.4 Introduction to new hardware components like Arduino Uno Development board, Raspberry Pi computers, magnetometers, ESP8266, LoRa transceivers etc.
- 1.4.5 Introduction to software coding environments and programming languages like C++, python and Java.
- 1.4.6 Introduction to ThingSpeak online IoT database.

## **1.5 Organization of report**

The report starts with a literature survey based on the Shanon Hartley theorem for understanding spread spectrum digital modulation technique.

Next it delves into the technical basics of LoRa technology followed by a brief study of the LPWAN protocol. Then a brief description of the Internet Of Things technology will be undertaken.

Major emphasis will be laid upon the LoRa technology. GSM prototyping was done with the aim of understanding the work flow of communication architecture, knowledge of which is then used for establishing communication for LoRa enabled device



## **Chapter 2**

### **Literature survey**

#### **2.1 Introduction to overall topic**

##### **2.1.1 LPWAN/ LoRaWAN**

LPWAN stands for Low Power Wide Area Network, it is a wireless wide area network technology that is specialized for interconnecting devices with low-bandwidth connectivity, focusing on range and power efficiency. Low-power WAN technologies are designed for machine-to-machine (M2M) networking environments. With decreased power requirements, longer range and lower cost than a mobile network, LPWANs are thought to enable a much wider range of M2M and Internet of Things (IoT) applications, which have been constrained by budgets and power issues.

LPWAN data transfer rates are very low, as is the power consumption of connected devices. LPWAN enables connectivity for networks of devices that require less bandwidth than what the standard home equipment provides. Furthermore, LPWANs can operate at a lower cost, with greater power efficiency. The networks can also support more devices over a larger coverage area than consumer mobile technologies and have better bi-directionality.

Bluetooth, ZigBee and Wi-Fi are adequate for consumer-level IoT implementations. The need for a technology such as LPWAN is much greater in industrial IoT, civic and commercial applications. In these environments, the huge numbers of connected devices can only be supported if communications are efficient and power costs low.

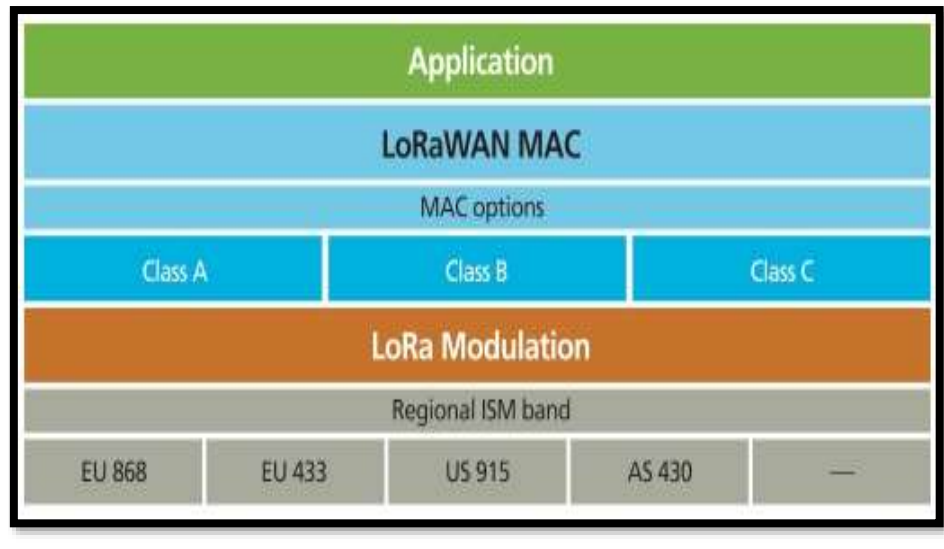
**Table 1.** LPWAN characteristics

Characteristic	Target Value for LPWAN Technologies
Long range	5 – 40km in the open field
Ultra low power	Battery lifetime of 10 years
Throughput	Depends on the application, but typically a few hundred bit / s or less
Radio chipset costs	\$2 or less
Radio subscription costs	\$1 per device and year
Transmission latency	Not a primary requirement for LPWAN. IoT applications are typically insensitive to latency.
Required number of base stations for coverage	Very low. LPWAN base stations are able to serve thousands of devices.
Geographic coverage, penetration	Excellent coverage also in remote and rural areas. Good in-building and in-ground penetration (e.g. for reading power meters).

**Table 2.** LPWAN specifications

LPWAN Technology	Standard / Specification	Range	Spectrum
LoRaWAN	LoRa Alliance LoRaWAN	2-5km in urban areas <15km in suburban areas	Any unlicensed spectrum 868MHz (Eu) 915MHz (US) 433MHz (Asia)

LoRaWAN is a protocol specification built on top of the LoRa technology developed by the LoRa Alliance. It uses unlicensed radio spectrum in the Industrial, Scientific and Medical (ISM) bands to enable low power, wide area communication between remote sensors and gateways connected to the network. This standards-based approach to building a LPWAN allows for quick set up of public or private IoT networks anywhere using hardware and software that is bi-directionally secure, interoperable and mobile, provides accurate localization, and works the way you expect.



**Figure 1.** LoRaWAN protocol stack

### 2.1.2 LoRa (Long Range)

LoRa is a chirp spread spectrum, digital modulation technique which enables long range communication. It was developed by Semtech Corporation and has since been making major strides into the IoT market.

LoRa is a PHY layer implementation and is independent of higher-layer implementations. This allows LoRa to coexist and interoperate with existing network architectures. These features have been major attractors as far as the Internet Of Things Movement (IoT) is concerned. LoRa can be optimised for achieving Machine to Machine connectivity without the use of internet or Bluetooth.



**Figure 2.** LoRa logo

### 2.1.3 LoRaWAN Architecture

LoRaWAN network architecture is typically laid out in a star-of-stars topology in which gateway is a transparent bridge relaying messages between end-devices and a central network server in the backend. Gateways are connected to the network server via standard IP connections while end-devices use single-hop wireless communication to one or many gateways. All end-point communication is generally bi-directional, but also supports operation such as multicast enabling software upgrade over the air or other mass distribution messages to reduce the on-air communication time.

Communication between end-devices and gateways is spread out on different frequency channels and data rates. The selection of the data rate is a trade-off between communication range and message duration. Due to the spread spectrum technology, communications with different data rates do not interfere with each other and create a set of "virtual" channels increasing the capacity of the gateway. LoRaWAN data rates range from 0.3 kbps to 50 kbps.

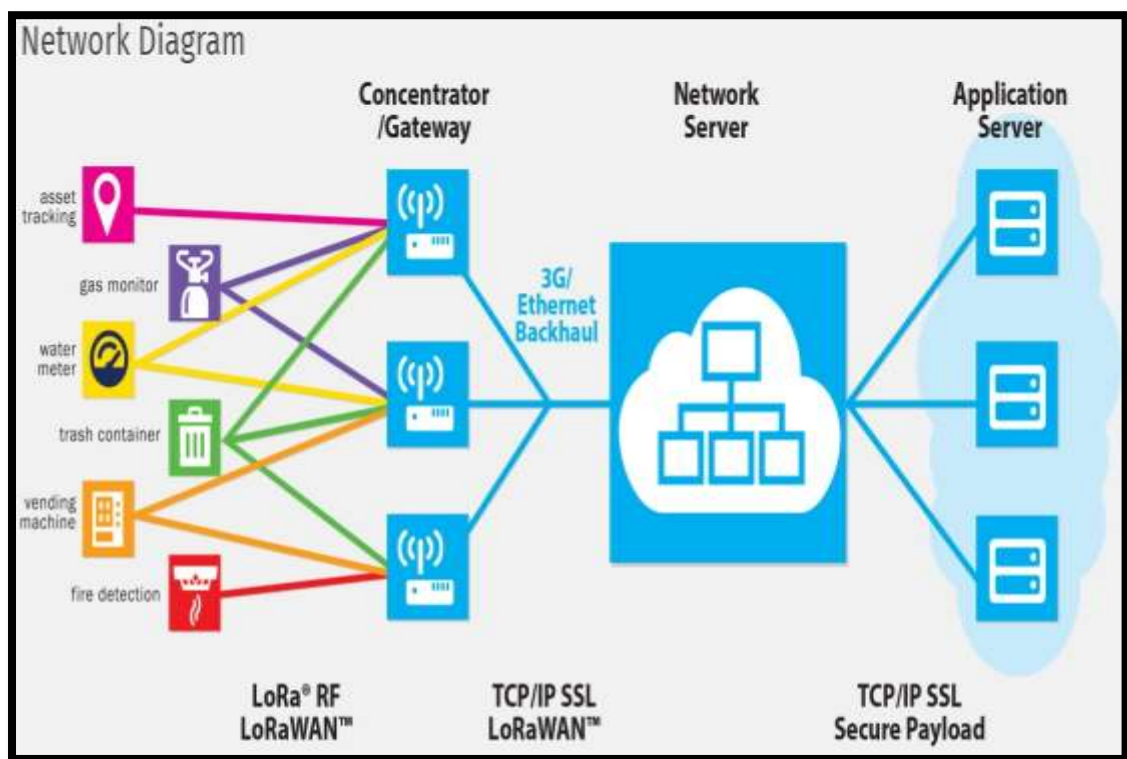


Figure 3. LoRaWAN architecture

### 2.1.4 Internet Of Things Framework

The backend systems of LoRa Network are responsible for routing Internet of Things data between devices and applications. A typical Internet of Things network requires gateways as a bridge between specific radio protocols and the Internet. In cases where the devices themselves support the IP stack, these gateways only must forward packets to the Internet. Non-IP protocols such as LoRaWAN require some form of routing and processing before messages can be delivered to an application. LoRa Network is positioned between the gateways and the applications and takes care of these routing and processing steps.

Nodes broadcast LoRaWAN messages over the LoRa radio protocol. These messages are received by several Gateways.

The Gateway is a piece of hardware that forwards radio transmissions to the backend. It is connected to a Router.

The Router is responsible for managing the gateway's status and for scheduling transmissions. Each Router is connected to one or more Brokers.

Brokers are the central part of LoRa Network. Their responsibility is to map a device to an application, to forward uplink messages to the correct application and to forward downlink messages to the correct Router (which forwards them to a Gateway). The Network Server is responsible for functionality that is specific for LoRaWAN.

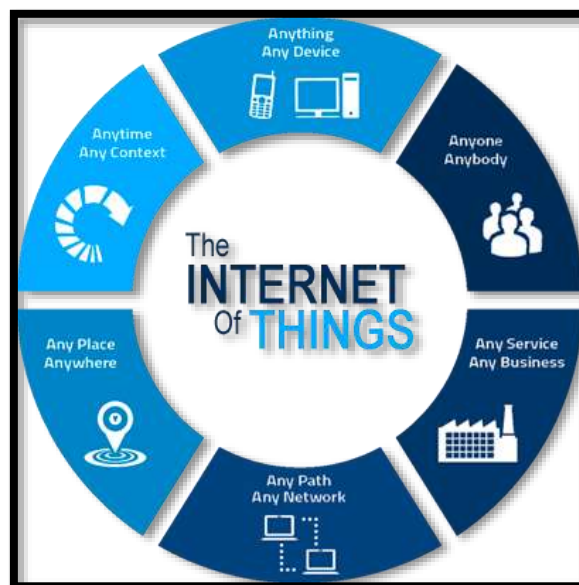


Figure 4. Internet Of Things

### 2.1.5 ThingSpeak IoT database

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize your sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of your IoT data.
- Run your IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.

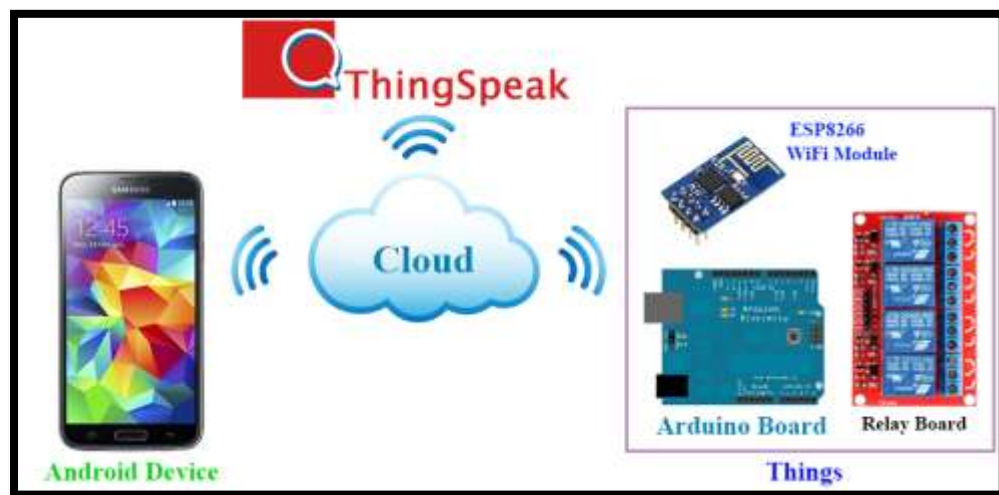


Figure 5. ThingSpeak database integration

## 2.2 Exhaustive literature survey

### 2.2.1 Shannon Hartley Theorem

The Shannon–Hartley theorem states the maximum rate at which information can be transmitted over a communications channel of a specified bandwidth in the presence of noise.

The theorem establishes Shannon's channel capacity for a communication link and defines the maximum data rate (information) that can be transmitted within a specified bandwidth in the presence of noise interference:

$$C = B * \log_2 \left( 1 + \frac{S}{N} \right) \quad \text{Equation 1}$$

Where:

C = channel capacity (bit/s)

B = channel bandwidth (Hz)

S = average received signal power (Watts)

N = average noise or interference power (Watts)

S/N = signal to noise ratio (SNR) expressed as a linear power ratio

By rearranging Equation 1 from log base 2 to the natural log, e, and by noting that  $\ln$  we can manipulate the equation as follows:

$$\frac{C}{B} = 1.433 * \frac{S}{N} \quad \text{Equation 2}$$

For spread spectrum applications the signal to noise ratio is small, since the signal power is often below the noise floor. Assuming a noise level such that  $S/N \ll 1$ , Equation 2 can be re-written as:

$$\frac{C}{B} \approx \frac{S}{N}$$

Or:

$$\frac{N}{S} \approx \frac{B}{C}$$

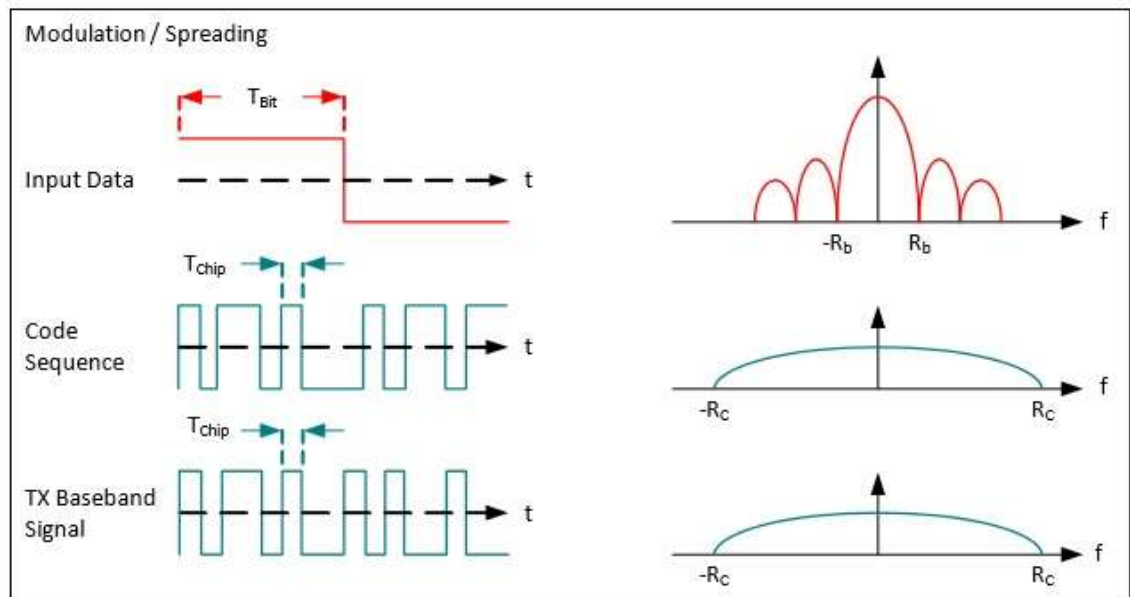
Equation 3

From equation 3 to transmit error free information in a channel of fixed noise-to-signal ratio, only the transmitted signal bandwidth need be increased.

### 2.2.2 Spread Spectrum principles

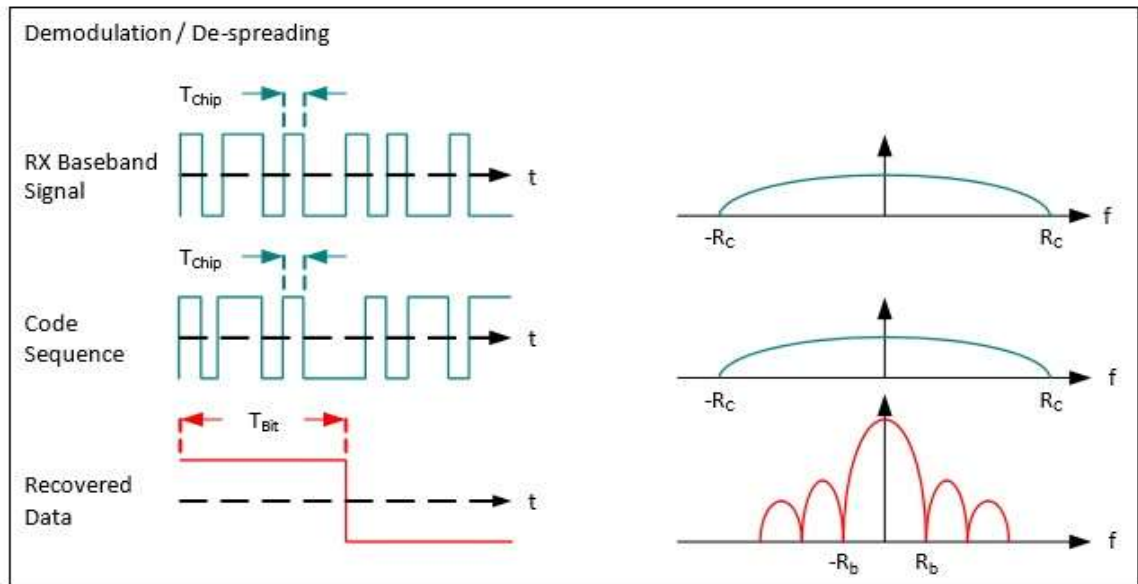
By increasing the bandwidth of the signal, we can compensate for the degradation of the signal-to-noise (or noise-to-signal) ratio of a radio channel.

In traditional Direct Sequence Spread Spectrum (DSSS) systems, the carrier phase of the transmitter changes in accordance with a code sequence. This process is generally achieved by multiplying the wanted data signal with a spreading code, also known as a chip sequence. The chip sequence occurs at a much faster rate than the data signal and thus spreads the signal bandwidth beyond the original bandwidth occupied by just the original signal. Note that the term chip is used to distinguish the shorter coded bits from the longer un-coded bits of the information signal.



**Figure 6** Shannon Hartley theorem 1 shows graphically how spread spectrum signals are modulated





**Figure 2** Shanon Hartley theorem 2 shows spread spectrum de-modulation graphically

The amount of spreading, for direct sequence, is dependent on the ratio of "chips per bit" - the ratio of the chip sequence to the wanted data rate, is referred to as the processing gain ( $G_p$ ), commonly expressed in dB.

$$G_p = 10 * \log_{10} \left( \frac{R_c}{R_b} \right) (dB)$$

Where:

$R_c$  = chip rate (Chips/second)

$R_b$  = bit-rate (bits/second)

As well as providing inherent processing gain for the wanted transmission (which enables the receiver to correctly recover the data signal even when the SNR of the channel is a negative value); interfering signals are also reduced by the process gain of the receiver. These are spread beyond the desired information bandwidth and can be easily removed by filtering.

DSSS is widely used in data communication applications. However, challenges exist for low-cost or power-constrained devices and networks.

### 2.2.3 LoRa modulation basics

In LoRa modulation the spreading of the spectrum is achieved by generating a chirp signal that continuously varies in frequency. An advantage of this method is that timing and frequency offsets between transmitter and receiver are equivalent, greatly reducing the complexity of the receiver design. The frequency bandwidth of this chirp is equivalent to the spectral bandwidth of the signal.

The wanted data signal is chipped at a higher data rate and modulated onto the chirp signal.

The relationship between the wanted data bit rate, symbol rate and chip rate for LoRa modulation can be expressed as follows:

We can define the modulation bit rate,  $R_b$ , as:

$$R_b = SF * \frac{1}{\left[ \frac{2^{SF}}{BW} \right]} \text{ bits/sec}$$

Where:

SF = spreading factor (7..12)

BW = modulation bandwidth (Hz)

### 2.2.4 Star network topology

A star network is the most common form of network topology for power constrained end-point nodes and is relatively simple to implement. Typically, a central coordinator or concentrator acts as the conduit for all network traffic. All network transmissions are routed via the central coordinator.

A star-network topology helps minimize the amount of network traffic. For a network that is not linkconstrained only 3 devices and two links are involved in any communications between two nodes. In addition, nodes are isolated from one another and provides for ease of replacing nodes. Centralization allows for inspection of all network traffic at a single point.

A disadvantage of this topology is that failure of the coordinator will disable all network communications.

### 2.2.5 Link Budget Comparison

**Table** Indicates the comparison between FSK and LoRa

Mode	Equivalent bit rate (kb/s)	Sensitivity (dBm)	$\Delta$ (dB)
FSK	1.2	-122	-
LoRa SF = 12	0.293	-137	+15

Even when transmitting at greater than 4 times the equivalent data rate, LoRa modulation offers similar sensitivity to a conventional FSK system. When the data rate is approximately equivalent the improvement with LoRa is between 7 and 10 dB. If we consider our channel capacity scenario above, we can see with an equivalent link budget, LoRa can actually transmit a data packet in a quarter of the time required for the FSK system.

Thus, assuming a simple TDD or time division multiplexing of the radio channel, LoRa can communication with x4 the number of devices as the FSK system.

## Chapter 3

### Problem Statement

The goal of the project was to provide continuous parking status of each individual parking spot to the GetParking App. The App in turn would use this data to show parking attendants how many spots are occupied and which ones are empty.

To achieve this, a sensor node and gateway architecture needs to be deployed. The focus area being to establish and optimize the communication between end points and gateway. Since the collected information was to be accessed by the Android App, the sensor architecture was to reflect the Internet Of Things structure. The IoT communication was fed by the local machine to machine communication data.

Apart from the technical efficiency, the cost efficiency and reliability were also major focuses during prototyping. It was realized that providing WiFi or internet connection was the most expensive part of the process. But at the same time an IoT infrastructure can't be obtained without the internet. So, efforts were to be diverted to elongating the range of sensor node and establishing intermediate gateways far from the sensor end points. Each gateway could respond to many local end points and update data to the internet. This reduces the no. of devices to be connected to the internet and along with it cost of implementation.

Thus, the machine to machine communication was tested by using two communication technologies

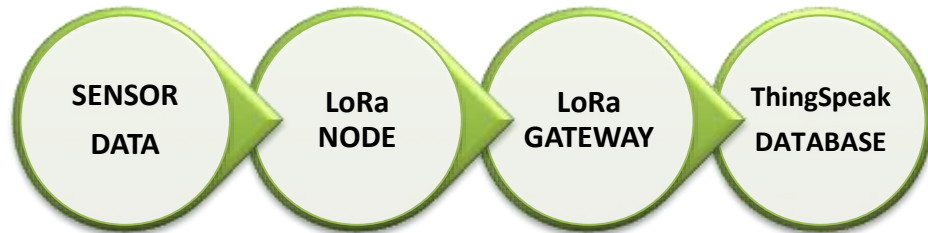
#### 1. GSM      2. LoRa

Through research, it has been established that although GSM has much greater range as compared to LoRa, the latter uses unlicensed spectrum which helps cut down operations cost down to only maintenance cost. The same cannot be said for 2G GSM as each sim card registered in the network has associated rental charges and additionally each SMS sent is charged separately. Although GSM has its economic shortcomings, it is easier to establish communication over this technology. So, to initially test the working and integration of the entire architecture, GSM prototypes were fabricated and tested.

For the main prototyping, LoRa is to be integrated into the architecture already tested out using GSM.

## Chapter 4

### Methodology



#### **[4.1] Generation of raw data**

To generate data, sensors were used which helped in detecting the presence of vehicles. Each individual parking spot is monitored with the help of sensors known as magnetometers. A vector magnetometer detects and measures magnetic flux density **B**. Since vehicles like cars and bikes have a metallic chasis, the presence of the same can be detected by the change in the magnetic flux of the environment.

The magnetometer used for prototyping is a 3-axis vectored magnetometer, which can measure the change of flux density on all three dimensions. Since the magnetic flux density of the environment changes from location to location, initially the default readings of magnetometer were analysed and measured. This helped determine the expected readings, when there is no vehicle in the vicinity i.e. when the parking spot is empty.

Next, the changes in the readings were measured when a vehicle was kept in the vicinity, following the earlier procedure a new range of readings was arrived upon. It was recorded that the magnetic flux became more negative when a vehicle was detected and came back to positive range when the vehicle was moved farther away from the sensor.

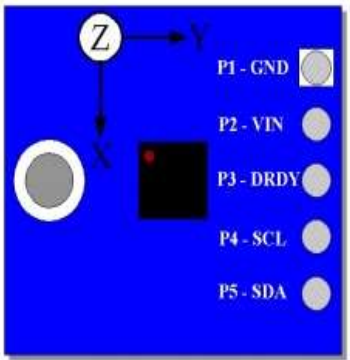
Thus, based on the testing so carried out the magnetometer was serially interfaced with an Arduino development board with the help of jumper wires. Since the magnetometer is an I2C device the outputs were connected to the Arduino in the following fashion:

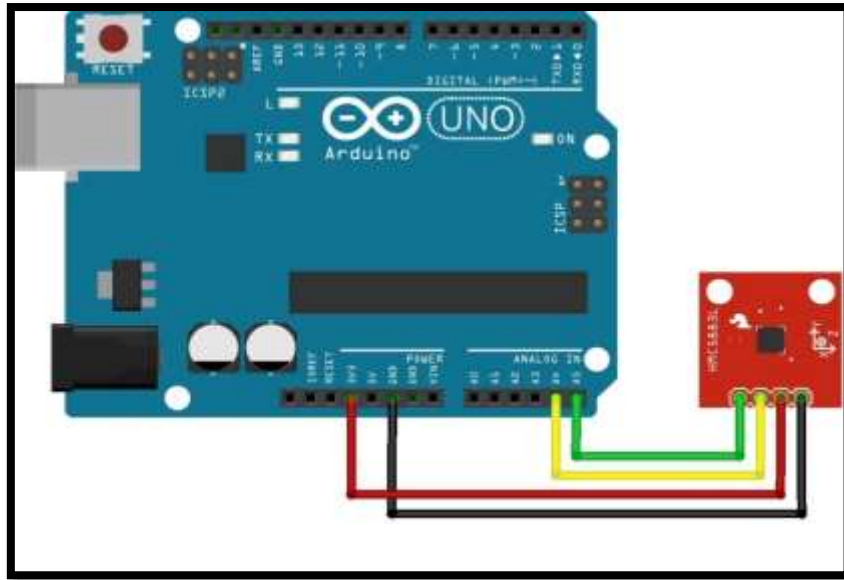
**Table 4.** Magnetometer interfacing

No.	Magnetometer pin	Arduino input
1.	Pin.1 GND	GND pin
2.	Pin.2 VIN	3.1V
3.	Pin.3 DRDY	Not connected
4.	Pin.4 SCL	Analog input pin.5
5.	Pin.5 SDA	Analog input pin.4

**Table 5.** Magnetometer pin descriptions

Pin Descriptions			
Pin	Name	Type	Function
1	GND	G	Ground
2	VIN	P	Supply Voltage – 2.7 to 6.5 VDC (module is regulated to 2.5VDC)
3	DRDY	I	Data Ready, interrupt. (internally pulled high, see datasheet for details)
4	SCL	I	I <sup>2</sup> C Clock (Pulled high on module, Clock 160Hz Default)
5	SDA	IO	I <sup>2</sup> C Data (Pulled high on module)



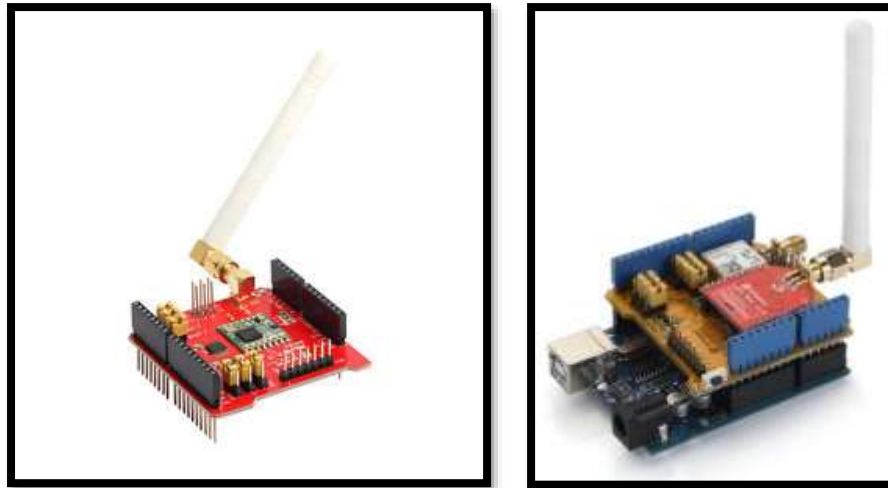
**Figure 8.** Magnetometer interface

The given photo shows how the magnetometer is serially connected to the Arduino. The Arduino development board is then programmed to serially read the 3-axis measurements detected by the magnetometer.

According to the testing carried out earlier, the Arduino is programmed with the logic that enables it to figure out whether the parking space is empty or occupied based on the magnetic flux readings.

The result being that now the magnetometer-arduino assembly can successfully determine the status of parking spot. By introducing appropriate loops and delays, the entire assembly is programmed to carry out constant monitoring.

#### [4.2] LoRa NODE



**Figure 9.** LoRa shield assembly

The status data so generated by the magnetometer-arduino assembly, is transmitted to a LoRa shield attached to the Arduino. The magnetometer wirings are now attached through the LoRa shield. The pins of the LoRa shield are an exact replica of the Arduino Uno pins so the wiring scheme remains the same.

The LoRa shield optimizes the Arduino board to transmit its serial messages using the LoRaWAN protocol. To do this it is optimized with a 868 Mhz high gain antenna. The LoRa shield is a transceiver i.e it can both transmit and receive LoRa packets.

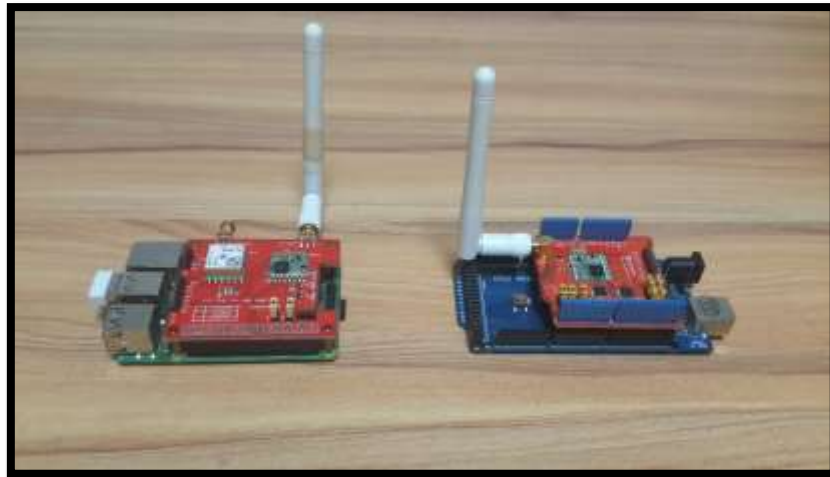
LoRa packet sizes can vary from 0.3 to 50 kb. The assembly is programmed on the Arduino IDE to collect status information and parse LoRa packets which can transmit the status information over the LoRaWAN network. The packets information is as follows:

##### **1A Occupied or 1A Empty**

Each individual LoRa end node is programmed separately. Thus, using this assembly constant monitoring can be done as well as periodic status updates can be issued. These updates are now available on the unlicensed LoRa spectrum and can be picked up by a radio listener tuned to 868 Mhz frequency.



### [4.3] LoRa GATEWAY



**Figure 10. LoRa gateway**

The LoRa node transmits the status packets over the LPWAN network at 868 Mhz. These packets in practical application will be transmitted by all the individual parking spots within the range of 5-10 km. These packets are picked up by a receiving LoRa transceiver which is serially interfaced with a Raspberry Pi 3.

The Raspberry Pi 3 is a credit card sized computer, it can support multiple operating systems. It has a quadcore 1.2 GHz CPU and can support upto 1GB RAM.

The gateway programs are coded on the Raspberry Pi-LoRa Shield assembly. The LoRa shield is programmed using the Arduino Software tools and interfaced with an Arduino Uno just like the LoRa end node. The only difference being that packets are received instead of sending the same. The entire process follows the following steps:



**STEP 1.** The LoRa shield is programmed to continuously keep listening on the 868 MHz band. As soon as a LoRa packet is detected, the packet is broken down to its bit stream. This bit stream is then serially transmitted to the connected Arduino Uno pins over the Rx and Tx pins directly interfaced with the shield.

**STEP 2.** The Arduino Uno simply collects the incoming bit stream and sends it out serially through the hardware USB port. This USB port is in turn connected to the Raspberry Pi3.

**STEP 3.** The Raspberry Pi 3 receives the status data serially over the hardware port ACM0, it then is programmed to continuously read all the data it receives. The serially received data is then stored into a local file on the Raspberry Pi itself along with a time stamp, which allows the creation of a parking data log. The local repository so created serves as a backup in case of internet issues or server breakdown. The repository is continually updated to reflect the incoming data.

**STEP 4.** To truly establish an Internet Of Things infrastructure, the local repository information is in turn updated onto an online database known as ThingSpeak. ThingSpeak, is an IoT optimized online database which helps collect data from remote sensors and it also provides MATLAB analysis features so that more knowledge can be created from the data so obtained.

The Raspberry Pi 3 can connect to the internet over WiFi, the local repository so maintained is thus constantly updated on the ThingSpeak account. This account allocates a channel with its own public API which can be accessed by the Raspberry Pi. Using this API and by performing remote write operation, the local data can be written onto the online database.

The database so generated at ThingSpeak can then be remotely accessed by downloading the channel data. Also, the data can be accessed in an excel format along with a timestamp for the same. This allows clean and structured data that can now be easily used by an Android App, in this case GetParking App.

## **Chapter 5**

### **System Analysis**

The entire system used to fabricate the said prototypes used a combination of both hardware and software tools. The hardware description was given in the preceeding section.

Each element of the system uses a different software coding environment and different languages for the same.

During the prototyping process the following software and coding languages were used:

1. Arduino Uno and LoRa shield assembly- Arduino Software development environment.
2. Raspberry Pi 3 – Raspbian operating system running on python ver3.

The entire system works from end to end in the following stages:

1. Data generation ( Magnetometer – Arduino )
2. Data transmission ( LoRa )
3. Data collection (LoRa)
4. Data management and internet updating system  
( Raspberry Pi , ThingSpeak)

## Chapter 6

### Software Description

As described in the system analysis and methodology sections every hardware component must be individually coded with its application specific programs.

The two software environments used are:

#### 6.1 Arduino IDE

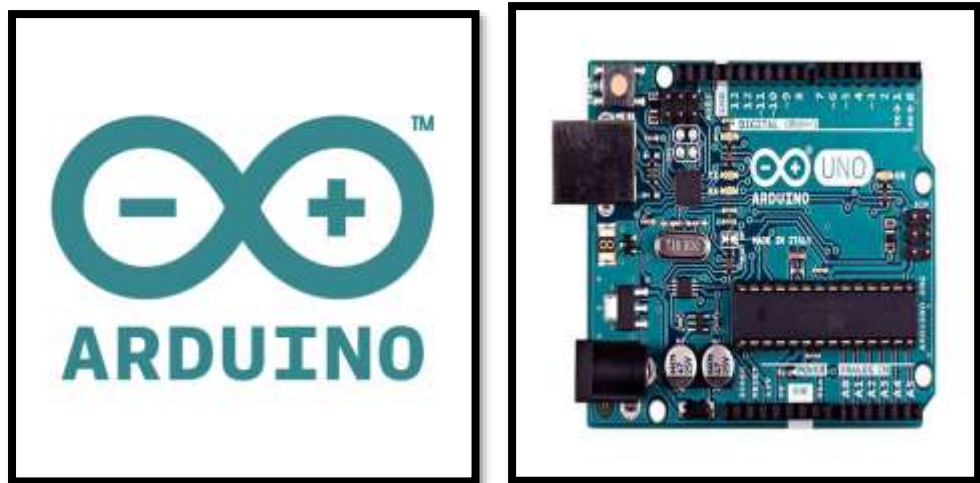


Figure 11. Arduino

The arduino IDE or software is an open source coding tool, which is used to program Arduino products. All Arduino products have an in-built bootloader which enables the burning of programs into the chips through the Arduino IDE.

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.

The programs can be uploaded using a micro USB cable attaching the Arduino board to the PC or laptop running the Arduino IDE. The only requirement is to check whether the appropriate board is selected before uploading and the corresponding serial port it is connected on.

Every Arduino program is known as a sketch. Each sketch can be written in C language. It contains of two loops within its default structure:

1. Void Setup (): This loop contains all the instructions and functions which are to be carried out only once. Usually contains the initialization procedure.
2. Void Loop (): This loop contains the actions that need to be continually carried out through out the lifetime of program execution. This the main execution unit of the code.

Apart from the given structure libraries can also be included as per requirement. Arduino software provides a wide array of libraries with sets of functions, which can be invoked to carry out library specific tasks through the script.

## 6.2. Raspberry Pi coding environment

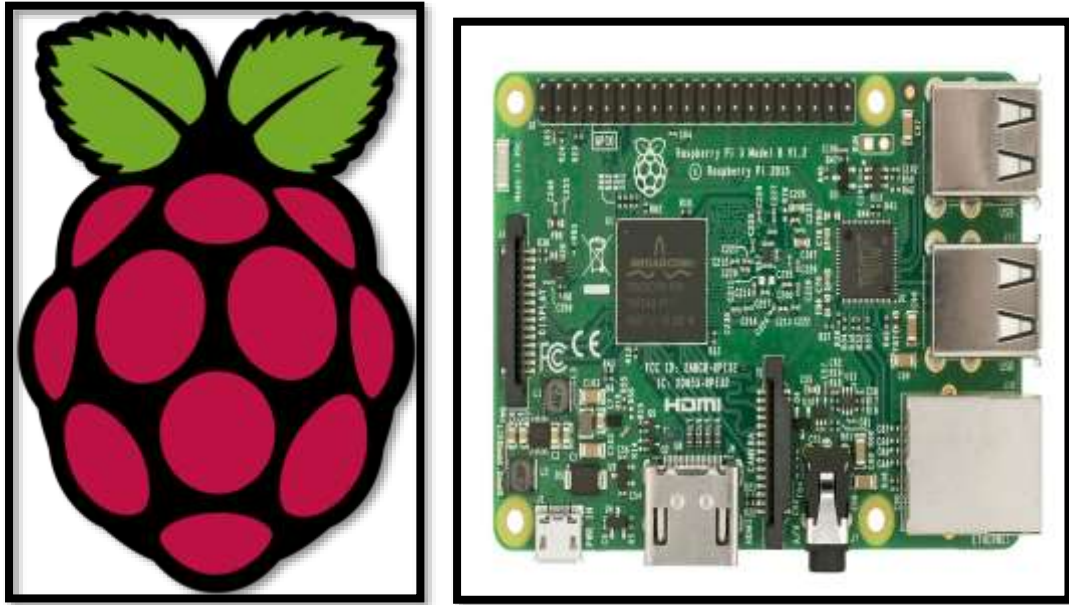


Figure 12 Raspberry Pi 3

The Raspberry Pi is a pocket-sized computer. Just like a conventional computer, the raspi can be executed using different Operating systems like: Raspbian, Fedora, Ubuntu , Windows 10, Debian , Android Things etc.

It is powered by a 1.2 GHz , 64 bit- Quadcore CPU.

For prototyping the Raspberry Pi was powered by the Raspbian OS , which is a Debian based Linux operating system. Using the Python software development tools the Raspi was programmed to serially read incoming data signals and store then in both local and online database.

The various operating systems for the Raspberry Pi can be installed on a MicroSD card which is then lodged into the SD card slot of the board. The Raspberry Pi version 3 is enabled by Bluetooth and WiFi. The WiFi feature helps in the use case, as internet connectivity is essential for uploading data onto the ThingSpeak database.

## **Chapter 7**

### **Testing and Results**

The entire assembly was tested during each stage of the prototyping process. The completion of the same, resulted in the fabrication of a final LoRa based prototype. This prototype consists of a single end node and a gateway corresponding to it.

In practical use, many such end nodes will be required to provide status about every parking spot. The system will in turn use many gateways which can remotely receive status LoRa packets. These gateway devices are connected to the internet over the WiFi.

The end result of the prototyping was to provide a proof of concept, which can be used to propose a LoRaWAN based architecture. These prototypes can be optimised for scalability. The resulting architecture could then use a LoRa based Machine to Machine communication, which helps in increasing the communication range. The Machine to Machine communication can then be fed to the web database. Thus, reducing the no. of devices to be connected to the internet.

During the final testing, I could detect changes in magnetometer readings on the ThingSpeak database with an average time lapse of 15-30 seconds. This time gap was programmed since the updating of each space was to be carried out after every 15 seconds.

So practically the end nodes carry out constant monitoring but send out status signals only after a definite contention period. This was done to avoid an overload of data in the repositories.

Any remote user with the access to the ThingSpeak account can download the database in an excel format. This excel sheet will be later used to integrate new features on to the GetParking App.

## Chapter 8

### Advantages, Limitatons and Applications

During the internship two kinds of prototypes were made. One based upon GSM and the final based upon LoRa. LoRa prototyping was the main focus, which used the GSM architecture as the basic frame work.

#### [8.1] Advantages

- Remote monitoring systems help reduce reliability on human labour.
- LoRa communication is more economical as compared to GSM and WiFi driven systems since it uses unlicensed spectrum.
- The range of LoRa (5-6 km in urban environment) is very small compared to satellite driven GSM technology. But as per the use case, the range provided by LoRa is sufficient to cover an entire parking space.
- LoRa powered end devices do not have any rental charge unlike GSM end nodes which contain an active SIM card for SMS facility.
- LoRa shields have an expected lifetime of upto 7-8 years upon constant usage, thus with such an infrastructure the only cost incurred would be for initial deployment and power supply (5 V for each individual end node)
- GSM network can crash due to network connectivity problems but no such issues are faced while using LoRa devices.
- This framework reduces the no. of devices to be connected to the internet i.e only the gateways need to be connected, thus this greatly reduces the WiFi rental as well.











Speed	1Mbit/s+	~100kbit/s	<10kbit/s
Example technology	4G	2G, LTE-M	LoRa, SIGFOX, NB-IoT
Spectrum	Licenced	Licenced	Licenced or unlicenced
Example use cases	 Smart phone  Connected car  CCTV	 Smart grid  Smart watch  High value object tracking	 Low value object tracking  Smart meter  Smart parking  Smart street lights

Figure 13. Applications of LPWAN



#### [8.2] Limitations

- The LoRa shields used i.e Draguino RN2466 are not available in India for business use and need to be imported from vendors abroad after lengthy registration process.
- LoRa shield are more expensive compared to GSM modules.
- The LPWAN protocol is an Aloha protocol i.e it has no collision detection mechanism. Thus, in the event of packet collision, data is lost and there are no mechanisms to request retransmission.
- The prototypes developed have no security features i.e any unauthorized LoRa transceiver can easily tap into the bit stream and corrupt it as well. Also, the online database at ThingSpeak has a public API, thus collected data is also vulnerable to attacks.

## Chapter 9

### Conclusion and Future Scope

This project “**IoT communication using LPWAN**” gave me an insight to the practical application of both hardware and communication fundamentals that I have studied as a part of my academic syllabus. Also, I had the opportunity to build a prototype and add new features to the same. During the fabrication process I was trained to use hardware components like Arduino Uno, Raspberry Pi, Magnetometers etc.

Apart from hardware, study of the LoRa digital modulation technique and understanding the Internet Of Things architecture helped me see the immense potential that LPWAN brings with it. While integrating data to the ThingSpeak database I had the opportunity to realize the vast no. of features it has to offer and how sensor generated data can be used to create system analysis. Lastly, since each element needed separate programming, I had the chance to learn C and python programming languages which has been very rewarding.

The future scope of the prototypes would include:

- Re-engineering them to account for security and reliability. The prototypes developed can be studied as a base to develop a scalable architecture.
- This new proposed architecture can then have add on features like remote operation ability, where end devices can be remotely set into sleep mode for power conservation.
- Improving the collision detection mechanism by engineering multichannel gateways (prototypes use single channel gateway).
- The magnetometer also can be programmed to measure deviations more accurately, thus increasing reliability of the system
- Lastly instead of ThingSpeak, Firebase online database can be used to store data. As the GetParking App has its entire backend repository on a Firebase system.

All this can be achieved by vigorous testing and design efforts carried upon the basic prototypes.

## REFERENCES

- [1] [https://thingspeak.com/pages/learn\\_more](https://thingspeak.com/pages/learn_more)
- [2] <http://www.semtech.com/images/datasheet/an1200.22.pdf>
- [3] [http://www.semtech.com/wireless-rf/internet-of-things/downloads/Semtech\\_SmartCitiesTransformed\\_WhitePaper\\_FINAL.pdf](http://www.semtech.com/wireless-rf/internet-of-things/downloads/Semtech_SmartCitiesTransformed_WhitePaper_FINAL.pdf)
- [4] <http://www.semtech.com/wireless-rf/internet-of-things/what-is-lora/>
- [5] <http://www.inf.fu-berlin.de/lehre/WS01/19548-U/shannon.html>
- [6] <https://www.arduino.cc/en/Guide/Environment>
- [7] <http://internetofthingsagenda.techtarget.com/definition/LPWAN-low-power-wide-area-network>
- [8] <https://www.leverage.com/research-papers/lpwan-white-paper>
- [9] <https://www.raspberrypi.org/education/>
- [10] [http://wiki.dragino.com/index.php?title=Lora\\_Shield](http://wiki.dragino.com/index.php?title=Lora_Shield)
- [11] <https://www.open-electronics.org/gsm-remote-control-part-4-sim900/>
- [12] <https://rominirani.com/firebase-iot-tutorial-46203a92f869>
- [13] <https://www.adafruit.com/product/1746>

## Appendix A: Soft Code

### GSM PROTOTYPE

#### A. End node code ( Arduino Uno – GSM module )

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);

char msg;
void setup()
{
  Serial.begin(9600);
  mySerial.begin(9600); // Setting the baud rate of GSM Module
  delay(1000);
  //mySerial.print("AT+CMGF=1\r"); // set SMS mode to text
  mySerial.print("AT+CMGF=1\r"); // set SMS mode to text
  //Serial.print("111111");

  delay(100);
  //mySerial.println("AT+CNMI=2,2,0,0,0"); // AT Command to receive a live
  SMS
  //mySerial.print("AT+CNMI=2,2,0,0,0\r");

  mySerial.print("AT+CNMI=1,2,0,0,0\r"); // AT Command to receive a live
  SMS
  delay(100);
  //Serial.println("Ready");
}

void loop()
{
  if (mySerial.available()>0)
  {
    msg=mySerial.read();
    Serial.print("+919930446638 : 1A occupied");
    delay(2000);

  }
}
```

## B. Gateway code ( GSM module – Raspberry Pi)

```
import os
import serial
import time
from time import sleep
from datetime import datetime

ser=serial.Serial("/dev/ttyACM0",baudrate=9600,timeout=1)

while 1:
    stra=ser.readline()

    if(len(stra)>0):
        now=datetime.now()
        nowstr=str(now)
        str1=stra+" "+nowstr

        f=open("parkingdata.txt","a+")
        f.write("%s/n" %(str1))
        print(str1)
        f.close()
```

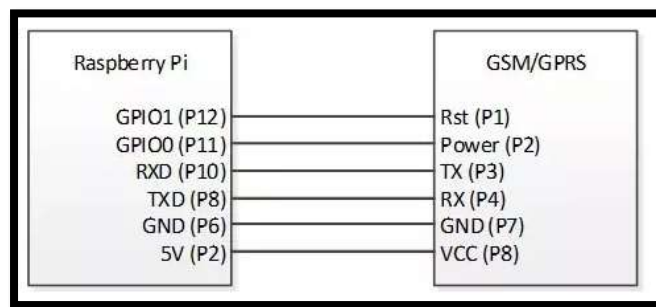


Figure 14. Raspberry Pi interacting with GSM module

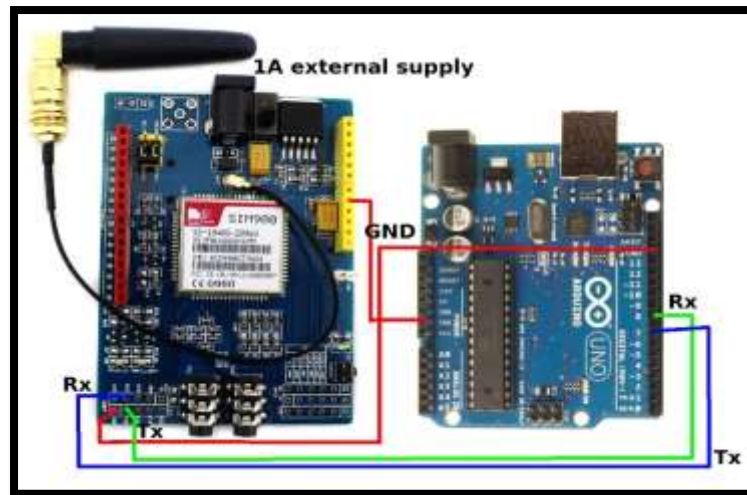


Figure 15. GSM Module with Arduino Uno

## LoRa PROTOTYPE

### A. Lora End node code ( LoRa shield- Arduino- Magnetometer)

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>

#define HMC5883      0x1E    // 7-bit R/W I2C address for The HMC5883
#define MODE_REGISTER 0x02    // HMC8553's Mode Register address
#define X_MSB_REGISTER 0x03    // HMC8553's Data Output X-MSB Register
#define CONT_MEASURE_MODE 0x00 // Continuous-measurement Mode

int counter=0;
int o=0;
int e=0;

void setup()
{
  Wire.begin();
  Wire.beginTransmission(HMC5883);
  Wire.write(MODE_REGISTER);    // Set the HMC8553's address pointer to
the Mode Register
  Wire.write(CONT_MEASURE_MODE); // Set the Mode Register to
Continuous-measurement Mode
  Wire.endTransmission();

  Serial.begin(9600);
```

```
while (!Serial);

    Serial.println("LoRa Sender");

if (!LoRa.begin(915E6))
{
    Serial.println("Starting LoRa failed!");
    while (1);
}
}

void loop()
{
    int x, y, z;

    Wire.beginTransmission(HMC5883); //identifies which slave
    Wire.write(X_MSB_REGISTER);    // Set address pointer to the 1st data-output
    register
    Wire.endTransmission();

    // Read in the 6 bytes of data and convert it to 3 integers
    representing x, y and z
    Wire.requestFrom(HMC5883, 6);

    if (Wire.available() == 6)
    {
        x= Wire.read() << 8;    // X-MSB
        x|= Wire.read();        // X-LSB
        z= Wire.read() << 8;    // Z-MSB
        z|= Wire.read();        // Z-LSB
        y= Wire.read() << 8;    // Y-MSB
        y|= Wire.read();        // Y-LSB
    }

    // Send the results back via the serial port
    Serial.print("X: ");
    Serial.println(x);
    Serial.print("Y: ");
    Serial.println(y);
    Serial.print("Z: ");
    Serial.println(z);
}
```

```
Serial.println();
delay(500);

if(x<=-4096|y<=-4096|z<=-4096)
{
  Serial.print("car detected: ");
  o++;
  notification1();

  Serial.println(counter);

}
else
{
  Serial.print("empty");
  e++;
  notification2();

  Serial.print(counter);

}
}

void notification1()
{
  if(o>15)
  {

    Serial.print("Sending occupied notification");
    //Serial.print(hourFormat12());
    LoRa.beginPacket();      // send packet
    LoRa.print("1A occupied");
    LoRa.print(counter);
    //LoRa.print(hourFormat12());
    LoRa.endPacket();
    while(o>0)
    {
      o--;
    }

  }

  counter++;
```



```
    delay(500);
}

void notification2()
{
    if(e>15)
    {

        Serial.print("Sending empty notification");
        //Serial.print(hourFormat12());
        LoRa.beginPacket();    // send packet
        LoRa.print("1A empty");
        LoRa.print(counter);
        //LoRa.print(hourFormat12());
        LoRa.endPacket();

        while(e>0)
        {
            e--;
        }

    }
    counter++;
    delay(500);
}
```

The screenshot shows the Arduino IDE interface. The left pane displays the code for a LoRa node, and the right pane shows the serial monitor output.

**Code (Left Pane):**

```

LoRaMagnum
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>

#define RHCF868 0x1E // RHCF868
#define MODE_REGISTER 0x02 // MODE_REGISTER
#define MMR_REGISTER 0x03 // MMR_REGISTER
#define CONT_MEASURE_MODE 0x00 // CONT_MEASURE_MODE

int counter=0;
int o=0;
int e=0;

void setup() {
  Wire.begin();
  Wire.beginTransmission(0x28);
  Wire.write(MODE_REGISTER);
  Wire.write(CONT_MEASURE_MODE);
  Wire.endTransmission();

  Serial.begin(9600);

  while (!Serial);

  Serial.println("LoRa Sender");

  if (!LoRa.begin(915E6)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
}

void loop() {
  // ... (rest of the code)
}

```

**Serial Monitor Output (Right Pane):**

```

emptyRX: -393
Y: 137
Z: 63
emptyRX: -393
Y: 138
Z: 62
emptyRX: -394
Y: 139
Z: 64
emptyRX: -391
Y: 139
Z: 63
emptyRX: -395
Y: 139
Z: 64
emptyRX: -395
Y: 138
Z: 61
emptyRX: -394
Y: 136
Z: 63
emptySending empty notificationRX: -392
Y: 134
Z: 61
emptyRX: -392
Y: 136
Z: 62
emptyRX: -396
Y: 136
Z: 63
emptyRX: -394
Y: 135
Z: 61
empty

```

Figure 15. LoRa node

## B. LoRa Receiving node (LoRa shield – Arduino)

```

#include <SPI.h>
#include <LoRa.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  //Serial.println("LoRa Receiver");

  if (!LoRa.begin(915E6)) {
    Serial.println("Starting LoRa failed!");
  }
}

```

```

    while (1);
  }
}

void loop() {
  // try to parse packet
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    // received a packet
    //Serial.print("Received packet ");

    // read packet
    while (LoRa.available()) {
      Serial.print(LoRa.read());
    }

    // print RSSI of packet
    //Serial.print(" with RSSI ");
    //Serial.println(LoRa.packetRssi());
  }
}

```

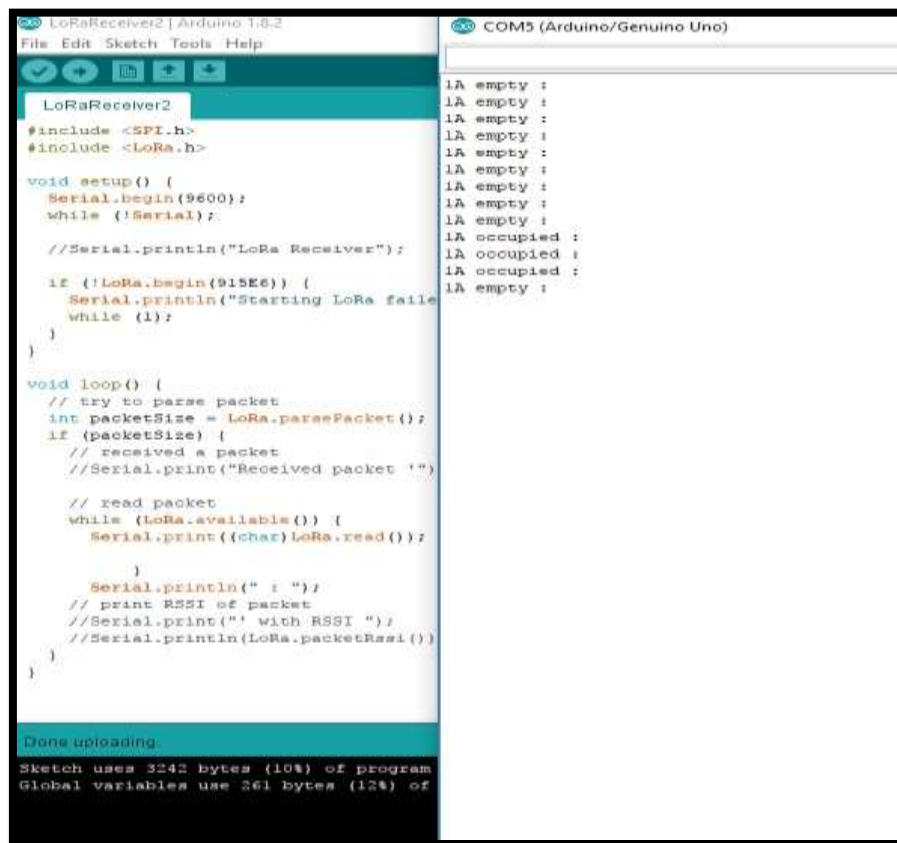


Figure 16. LoRa receiver

## C. LoRa Gateway code (Arduino-Raspberry Pi-ThingSpeak)

### 1. Updating Local file

```
import os
import serial
import time
from time import sleep
from datetime import datetime

ser=serial.Serial("/dev/ttyACM0",baudrate=9600,timeout=1)

while 1:
    stra=ser.readline()

    if(len(stra)>0):
        now=datetime.now()
        nowstr=str(now)
        str1=stra+" "+nowstr

        f=open("parkingdata.txt","a+")
        f.write("%s/n" %(str1))
        print(str1)
        f.close()
```

### 2. Updating ThingSpeak database

```
import os
import serial
import time
import sys
import RPi.GPIO as GPIO
from time import sleep
from datetime import datetime

try:
    import urllib.request as urllib2
except ImportError:
    import urllib2
```

```
ser=serial.Serial("/dev/ttyACM0",baudrate=9600,timeout=1)

count=0

DEBUG=1

RCpin=24

myAPI="QP5I9LCKYDNUV7CH"

myDELAY= 15


def getSensordata():

    while 1:

        stra=ser.readline()

        if(len(stra)>0):

            now=datetime.now()

            nowstr=str(now)

            str1=stra+" : "+nowstr

            f=open("parkingdata.txt","a+")

            f.write("%s\n" %(str1))

            print(str1)

            f.close()

def main():

    print('starting....')

    baseURL='https://api.ThingSpeak.com/update?api_key=QP5I9LCKYDNUV7CH&field1=0'

    print(baseURL)

    while True:
```

```
try:

    mag=getSensordata()

    f= urllib2.urlopen(baseUrl + "&field1=%s" % (mag))

    if(i<10):

        print(f.read())

        print(status)

        i=i+1

    f.close()

    sleep(int(myDELAY))

except:

    print('exiting')

    #break

main()
```

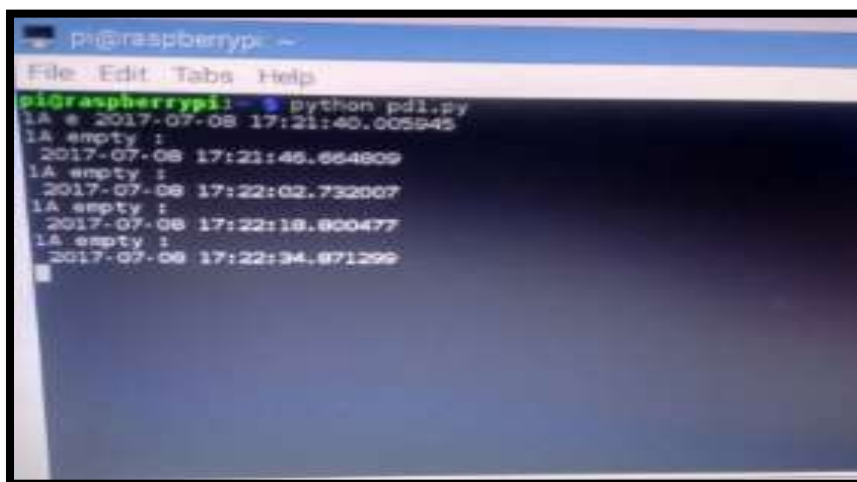


Figure 17. LoRa gateway