Dynamic traffic management system via GPS and localization

BRAC University

Sadman Hoque Sadi

August 2017

Declaration

I hereby declare that the thesis titled "Dynamic traffic management system via GPS and localization" is my own work. Submitted to the department of Electrical and Electronic Engineering, for the partial fulfillment for the degree of Bachelor of Science in Electrical and Electronic Engineering. This is my original work that has not been elsewhere and whatever materials used from other sources have been properly cited.

Thesis supervisor

.....
Shoilie Chakma

.....
Sadman Hoque Sadi
ID:13221039

Acknowledgments

I would like to first thank my supervisor Shoilie Chakma, who's continual guidance and support throughout the 3 semesters enabled me to tackle all the difficulties faced and complete the thesis in a timely manner. Also thankful to Masnur Rahman for helping with choosing the initial hardware which made the project significantly budget friendly.

Abstract

Road intersections using traffic lights are the most common form of crossways that are used throughout the world. Mostly due to space limitations however cities tend to be using intersections in road networks which are rather inadequate for the number of vehicles it serves, causing high levels of congestions. The traditional approach to solving this is to vary the timing of the traffic signals so that roads with greater number of vehicles are allowed to pass through for a longer time. This method works in concept, however in practical applications it has always been somewhat ineffective due to the lack of data regarding the number of vehicles at the intersection. This thesis is based on creating an effective wireless communication channel between vehicles and road infrastructure via a method of localization using GPS, RFID and some algorithms which have been developed to make sure that vehicles only respond to traffic signaling units and road signs which applies to said vehicle.

Contents

1 Introduction **1**
- 1.1 Urban traffic management . . . 1
- 1.2 Existing Techniques . . . 2
- 1.3 Motivation . . . 3
- 1.4 Overview . . . 4

2 System Design **5**
- 2.1 Block diagram and parts selection . . . 5
- 2.2 Microcontroller . . . 6
- 2.3 Transceiver . . . 7
- 2.4 GPS . . . 7
- 2.5 RFID . . . 8

3 How it Works **10**
- 3.1 Overview . . . 10
- 3.2 Communication . . . 10
- 3.3 Localization . . . 14
- 3.4 Conclusion . . . 17

4 Coding **20**
- 4.1 Vehicle Unit . . . 20
- 4.2 Infrastructure Unit . . . 30

5 Simulation **34**
- 5.1 Overview . . . 34
- 5.2 Code . . . 35
- 5.3 Results . . . 38

6 Conclusion **40**
6.1 Feasibility 40
6.2 Future work 41

A Part specifications **46**

B equation derivation **56**
B.1 Deriving equation used for simulation 56

List of Figures

2.1 block diagram of vehicle unit 5
2.2 block diagram of infrastructure unit 6
2.3 an ATmega328P on an Arduino board 6
2.4 image of the NRF24L01(c) 7
2.5 image of the Grove GPS module 8
2.6 image of the RFID module MFRC522 9

3.1 visual depiction of what 'geo-location' and 'radius' refers to on an overhead view of a crossroad 12
3.2 signals codes for specifying the individual roads at an intersection 13
3.3 diagram showing the angle and width at an intersection 14
3.4 overhead view of crossroad with all the necessary measurements used for localization technique 17
3.5 visual depiction of the system working at an intersection . . . 18
3.6 visual depiction of the system working with road signs 19

5.1 graph showing the total number of stationary vehicles in every road at each second 39

List of Tables

3.1 format of transmission signal 11
3.2 signal codes 12
3.3 GPS accuracy 15

Chapter 1

Introduction

1.1 Urban traffic management

A general trend among cities is that they start out rather small, with an adequate road network designed for its population and later on expand dramatically. Which results in much higher land value and greater number of road users. The older city center inevitably becomes the most developed part of the city with expensive land space and an older road network. The outcome is many times more vehicles using its roads than it was ever designed for and not much room for road expansion. And whatever land that is available tends to be inordinately expensive. Typically this problem is met with public transport infrastructure projects such as subway system or elevated railway network, but those are costly projects to build and maintain.

Another way to solve the inner city congestion however, is to instead make the most efficient use of the existing roads. This largely hovers around how well the road intersections are managed, particularly how long every vehicle has to wait due to traffic signals[1]. The focus this thesis is to design an effective system so that road intersections are regulated in such a way that traffic signaling is done through a very clear understanding of the amount of congestion that is present at any road and micro-managing every intersection in order to lower the overall traffic jam in the city.

1.2 Existing Techniques

The concept of better traffic signaling times is not new, there are already several methods that are in use or have been researched upon in order to achieve this. Currently in many scenarios they simply have a fixed timing system, as this is the cheapest method. In some other cases they are using loop detectors that enable traffic lights to know about the presence and speed of the car going on top of the detectors.

A lot of research work is going on in the field of V2V and V2X technology, which allows vehicles and road infrastructure to proper communicate with each other[2]. This is the key aspect towards solving our problem of traffic signals not having adequate data in order to properly vary its signal timing. There have been many systems that are designed based on this concept. Some involve imitating the functionality of the loop detectors but through wireless modules within a vehicle connected to its built in computer,[3] so that it is as if there are numerous loop detectors everywhere on the road. The benefit of this system is that it is relatively simpler to implement. Other similar research works include the traffic lights also taking into consideration the position of the vehicles in order to recommend the vehicle a certain speed, which will lower the likelihood of vehicles needing to actually stop at the intersection[4]. Other systems instead use visual recognition by the traffic light systems[5] which can compare the visual feedback between an empty road and a traffic congested road in order to estimate the level of congestion at each road of the intersection. This method is also relatively simpler to implement, however it has lesser accuracy in terms of the exact number of vehicles.

A common problem with all these methods however is the lack of actual information the traffic lights have in terms of the number of cars that are actually stuck in traffic. And although knowing the vehicle speed can be useful, at intersections where it is expected that every vehicle would have to stop anyways, it becomes rather pointless. These methods are more applicable for crossroads facing light amounts of traffic, where it is possible to mostly avoid vehicles coming to a stop through clever signal timing.

1.3 Motivation

Traffic congestion is a much bigger problem than what it appears to the average individual. Waiting at stop signals for long periods is always annoying, however it collective delay and wastage of time of the millions of people that live in any big city can have much greater implications on things such as the economy, climate and health.

There is a very direct effect of traffic congestion on the economy of any city or country for that matter[6]. Businesses, particularly when it comes to delivery of products and services use the same city roads as everybody else. Higher levels of congestion has all sorts of implications such as greater delivery time, forcing the businesses to work less efficiently at a manufacturing level, the money that is wasted in fuel when trucks are stationary in traffic, and the simple unpredictability that congestion creates that can have numerous implications on the daily functioning of any business. It is not only the producers of products however, the service industry can be greatly affected by traffic congestion as well. The most obvious of which is the public transport sector where time is wasted sitting on traffic when more customers could be served. People who have to go places to do their work such as doctors, repairmen, etc. are all facing lower revenue due to congestion[7]. Basically the entire economy slows down when there is an excessive amount of traffic in any city.

Climate change is one of the biggest problems every city in the world is facing right now, whether it is due to rising sea levels threatening to drown everything, the increased frequency of hurricanes, floods or the extremely high summer temperatures melting roads in some places. A key contributor to all these problems is the emission of CO2 by all the vehicles that people drive[8]. It is impossible to suggest that everyone suddenly find a perfect alternative to a method of transportation people have been using for almost two centuries now, however, the fact that a lot of the times these vehicles are simply stationary in traffic doing nothing, is not helping matters. Right now the transport sector of any city is a key contributor to the overall greenhouse gas emission, in some cities being responsible for more than a quarter of the emissions[9].

Large numbers of stationary vehicles with their engines running can also

have a very negative impact on public health, particularly pedestrians. It is already understood that vehicle emissions effect the climate, but when these emissions are initially concentrated at one area, it also has an adverse effect on people passing by. According to numerous studies, the pollutants emitted from the car exhaust, particularly nitrogen oxides, carbon monoxide and fine particle matter, can irritate our lungs and other parts of our respiratory systems resulting in diseases such as chronic bronchitis, heart disease and pneumonia[10]. The overall result being thousands of urban dwellers facing premature deaths[11].

1.4 Overview

The system that is designed for this thesis is primarily aimed towards reducing the traffic congestion at city center intersections, which tend to face a very high number of vehicles. However it has also been designed with the capability to work with typical road signs and allow exceptions for emergency vehicles[12].
Lowering traffic congestion is done via establishing a communication channel between all the applicable vehicles and traffic signal systems so that the they are aware of the exact number of vehicles at the intersection at any given time. This ensures that the crossroad is being used at its peak efficiency, and the additional feature of exempting emergency vehicles makes sure the system can cope with any unusual situation.
A link between vehicles and road signs are also very important. Although already being used in some cases through image recognition[13], this function being implemented along with the rest of the system makes is much more accessible. The wide usage is very important in this case, as the negligence by inexperienced drivers[14] can easily effect multiple vehicles on the road. And in some scenarios the road signs may not even be visible due to bad weather, which can be extremely dangerous.

Chapter 2

System Design

2.1 Block diagram and parts selection

Part selection was based on the certain criteria that needed to be met in order to carry out the proper functions along with being budget friendly. For the project part of the thesis it was also necessary to interface all the hardware mentioned above in a format so as to make it possible to carry out calculations with the output data from all of them. Each piece of hardware required a different library to make the interfacing possible, and there was an additional complexity faced due to some software limitations of the microcontroller used.

Figure 2.1: block diagram of vehicle unit

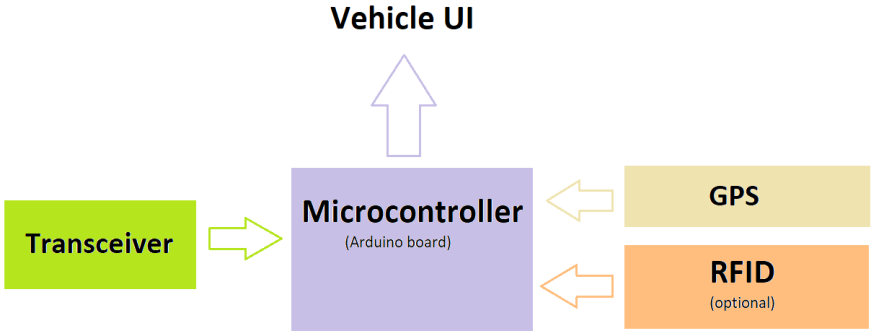
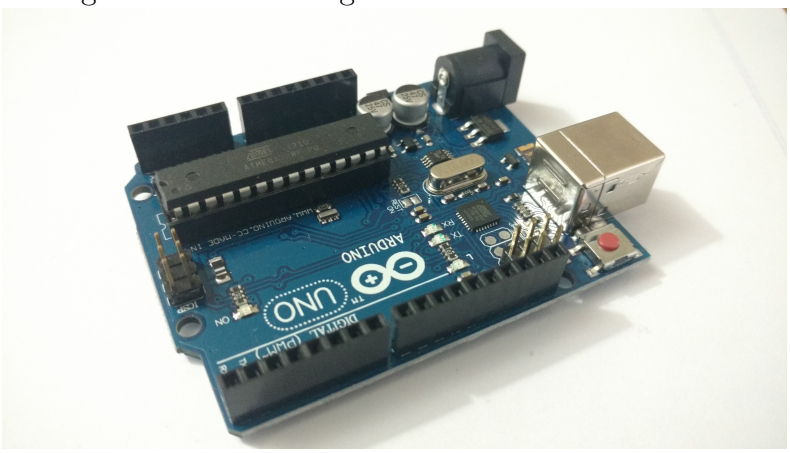


Figure 2.2: block diagram of infrastructure unit

2.2 Microcontroller

This forms the center of all the communication modules and does the actual processing work enabling the entire system to work as intended. Therefore it is very important that the particular microcontroller be something that can be interfaced with all the other parts, and has enough compute power to do the necessary calculations at a pseudo real-time level whilst keeping the cost at a minimum. Based on these necessities and previous research work[15], an ATmega328P microcontroller based on an Arduino board has been used for the thesis.

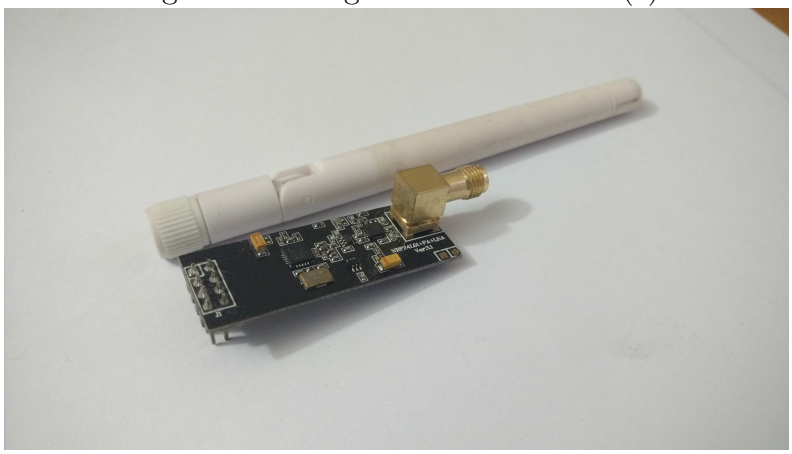
Figure 2.3: an ATmega328P on an Arduino board



2.3 Transceiver

The precise transceivers that were used were NRF24L01(c) modules, chosen due to their relative cost effectiveness and adequate range of 1km; speed of transmission was not of any particular concern since the actual data that gets sent and received is coded and is thus very small. Another thing to keep in mind is that these boards are not capable of both transmitting and receiving at the same time.

Figure 2.4: image of the NRF24L01(c)



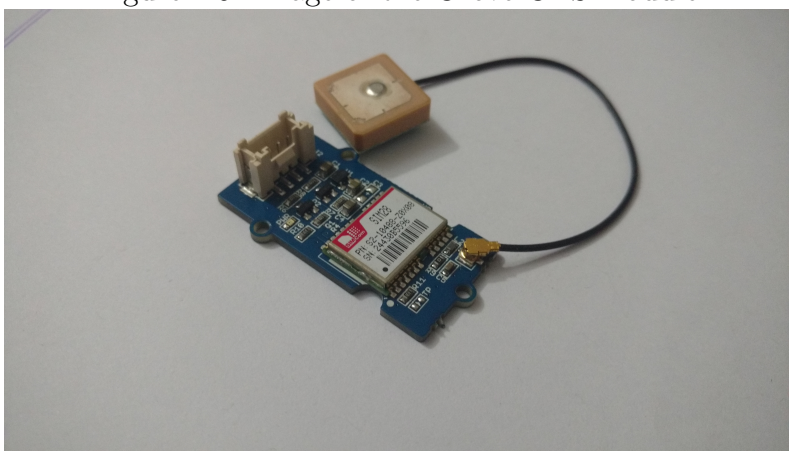
In terms of coding I used a library called RF24[16] which sends and receives codes as character arrays. These arrays, are then decoded for proper use of data by both the infrastructure and vehicle modules and also encoded when transmitting. There is a standardized system of coding in order to make sure both modules understand the actual message being sent.

2.4 GPS

The GPS data is taken using a Grove GPS module, selected due to its simple hardware interface, an external antenna lowering our signal interference, and budget friendliness. By default, the module continually transmits all the data it receives via satellite communication to the Rx pin of the microcontroller. However this is not very practical for its intended use. To collect the data in

a usable manner, a library called 'TinyGPS'[17] has been used, which creates a virtual Rx pin in order to receive the GPS data and then outputs the data in an orderly fashion.

Figure 2.5: image of the Grove GPS module

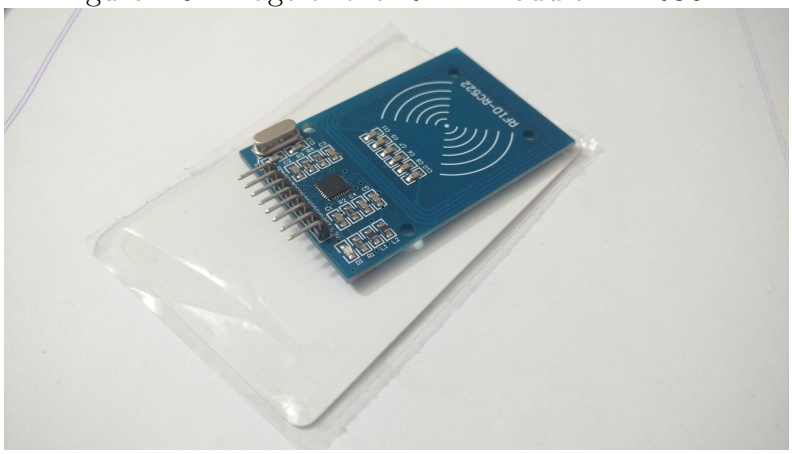


One thing to note here is that GPS signal output generally has 6 decimal places, whilst by default Arduino boards will only utilize values up to 2 decimal places. In order to alleviate this problem a different library called 'BigNumber'[18] has been used which can store such large numbers as character arrays and has built in math functions so that it is possible to do calculations with these numbers.

2.5 RFID

For the RFID module it was very important that the module be capable of reading quickly since vehicles would have read the tags embedded onto the road whilst driving over them. A certain amount of range is necessary because of vehicle height. Due to these reasons it was decided that RFID readers which utilize a relatively high frequency must be used, as they are capable of quick reads over comparatively high distances. The RFID module that was finally selected is the MFRC522, it has an HF (high frequency) rating meaning that, according to the documentation, it can read tags from a maximum range of 15cm.

Figure 2.6: image of the RFID module MFRC522



In terms of using the RFID module a library called 'MFRC522'[19] was used, which outputs the stored value of any card it reads into a string. As the RFID system was only intended in order to tell apart one lane of a road from another, there is no need for any calculation, its use is simply to inform the vehicle in which exact lane it is in. This sort of system is necessary because there are many cities which utilize different traffic signals for different lanes, the most common one being the way vehicles taking left turns need not stop at red lights. For the sake of simplicity however all tests in the thesis has been modeled after an intersection which has no such feature, however the necessary coding and functions are still built in.

Later on in the project however it has been realized that some of the originally intended use of an RFID system can be replaced by the use of GPS data, making the overall proposed system significantly more feasible since embedding RFID tags into roads would be relatively expensive and time consuming. The RFID system still has some use in certain scenarios though, meaning that its complexity of installation needs only to be tackled for a specific few conditions.

Chapter 3

How it Works

3.1 Overview

The key principle of the entire thesis is to use wireless communication instead of relying on image processing. The whole aspect of identifying traffic signals, road signs and figuring out which road has more congestion can be done via image recognition technology, however such systems can be exceedingly expensive, and in the case of vehicle units the cost makes it so that consumers will often avoid it. This situation limits the actual benefits of V2X (vehicle to infrastructure) communication. In my system of direct communication, the infrastructure unit continually transmits its signal data, whilst the vehicle unit receives that information and at times gives a feedback so that the infrastructure unit can calculate the level of congestion.

3.2 Communication

For the communication between the two modules I used a standardized code, one for the transmission signal sent by the infrastructure unit and another that the vehicle unit sends as a feedback. Although it is possible to just use human language like 'stop' and 'go', which can simply be shown to the driver on some display, the coding enables us to add further information which is vital to the proper functioning of the system.

The transmission signal, sent by the infrastructure unit, consists of the geographic coordinates of the center of the intersection, the inner radius, a signal

Table 3.1: format of transmission signal

Latitude	Longitude	Radius	Code	Angle	Width
11 characters	11 characters	5 characters	3 characters	2 characters	6 characters

code, an angle and width parameters.

'Geo-location' refers to the center of the intersection, in case of traffic signs, or center of any road if serving the function of a road sign. Range is meant to be controlled via tuning the transceivers at a hardware level. It is assumed for this thesis however that in any instance where this form of V2X communication is needed the necessary range would need to be at least 1km anyways, if more, then it can be dealt with by using multiple transmitters.

'Radius' refers to the inner radius of a road intersection, in case of transmission by road signs it is to be 0. This is to make vehicles at stop at a certain point before the crossroad. The parameter being circular, and thus curved for each road, unlike the traditional straight line is of particular importance. Realistically, the line up to which vehicles are allowed to go when facing a red light being slightly curved makes no difference. However, if the edges of the corner where two perpendicular roads meet at the intersection gets chamfered, it allows vehicles intending to take a left turn (in case of left-hand driving) be completely exempted from the traffic signal as it is not crossing the inner radius to do so. This is a very convenient method to allow vehicles to take left hand turns without facing traffic, a model that used by road networks of many countries. Lastly, it should be noted that in the transmission code the radius is not given in the form of standard metric measurements, it is in terms of decimal degrees that are used in maps and GPS to project locations.

'Code' is the key part of the whole transmission signal as it is the actual instruction itself. The three digit transmission code is standardized, and allows us to transmit traffic light signals, any road sign and enforce speed limits to vehicles. Basically the range of numbers is allocated in a certain manner which both the infrastructure and vehicle unit understands, a bit like the way ASCII works.

Figure 3.1: visual depiction of what 'geo-location' and 'radius' refers to on an overhead view of a crossroad

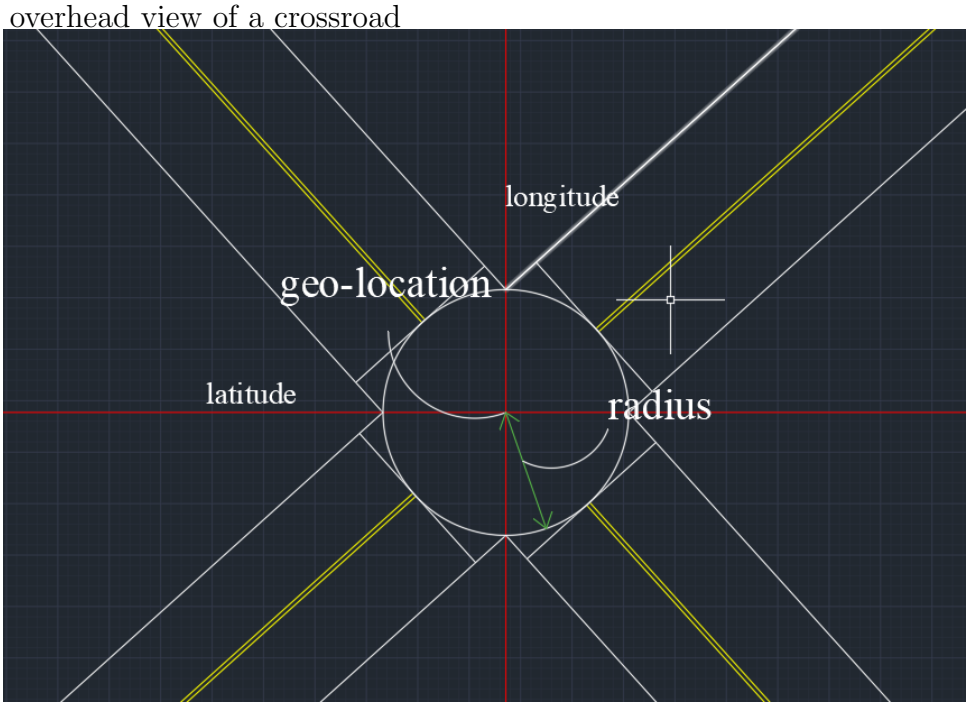


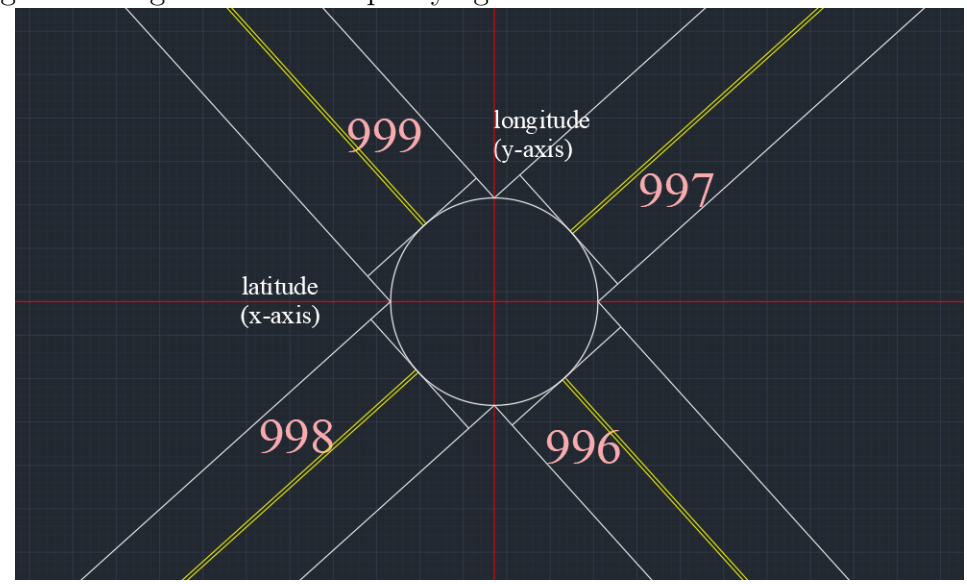
Table 3.2: signal codes

Speed limit	Road signs	Traffic lights
0-300	301-995	996-999

'speed limit' and 'road signs' parts of the signal code are quiet self-explanatory, the traffic light codes however have some technicality in it. The range of only 4 number is due to the simulated intersection having only 4 roads, each represented by a different number which is transmitted when that particular road is allowed to pass through by the traffic signal. The method of forming a uniform coding of this, which is to say that making sure that every vehicle identifies itself to be on the correct road, is through comparing the geographic coordinates of the vehicle and the 'geo-location' transmitted by the infrastructure unit. In visual terms, if the intersection being laid out on a graph with the intersection center representing the central zero point, then all the vehicles on the positive x and y axis identify themselves with the

code 997, the one's with negative y and x axis get 998, positive y-axis and negative x-axis is 999 and the other get 996. The sequence does not make sense when seeing it visually, but in terms of coding it simpler; either way, as long as the system is uniform for all cars, which it is, this method will work.

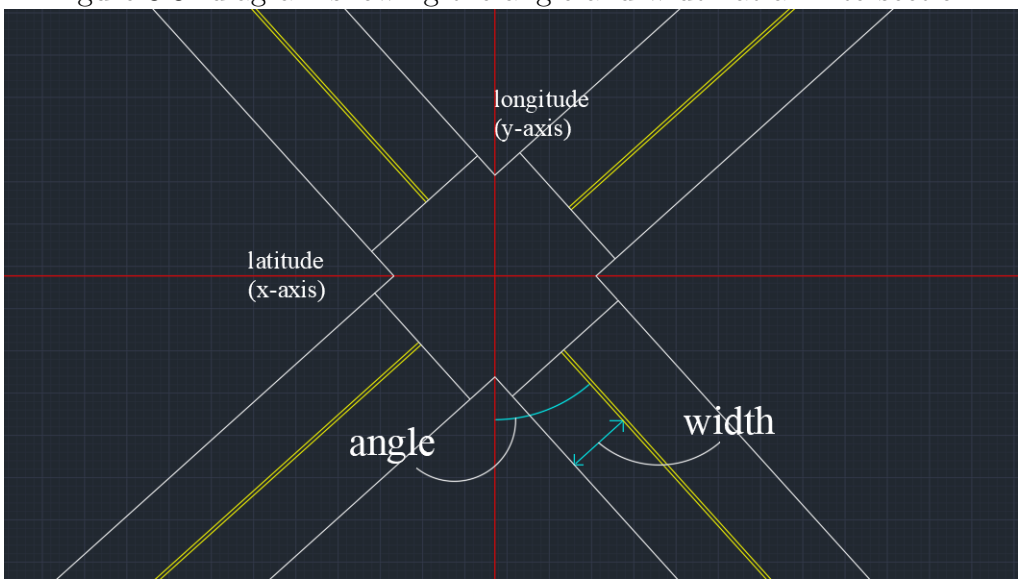
Figure 3.2: signals codes for specifying the individual roads at an intersection



'Angle' is the angle between the center line of the road and longitudinal axis from the intersection center. Exactly which road it is among the four that are in the intersection does not matter, as two of them will have the same angle anyways as they are opposite angles, and the other two are the same but just offset by 90 degrees. These discrepancies have been thought of and accounted for in the code. As to why exactly this parameter is in the code will become clear in the next section.

'Width' is, as the name suggests, the width of the road. However it is only the width of technically half the road, or whatever width the left hand side of the road is (assuming cars drive on the left). The importance of this parameter shall be explained in the next section.

Figure 3.3: diagram showing the angle and width at an intersection



3.3 Localization

A very important aspect of the thesis was to identify exactly which vehicles are supposed to be acting upon the traffic signal transmissions. The transceiver signal coverage is circular, but obviously only the vehicles facing the intersection are supposed to act upon the transmissions, and not those that are in adjacent roads or going in the opposite direction. Another consideration was the fact that GPS location would not be very precise either[20], and the precision can often vary depending on how strong the connection with a satellite is.

The way I devised a solution is rather straight forward, first of all, imagine a random stretch of road with a graph paper grid laid out on top of it. The inaccuracies from the GPS can be both in terms of x-axis and y-axis. And also the transmission signal coverage is circular, so we need to theoretically cut off that signal from outside this road so that vehicles that are in adjacent roads do not respond to this transmission that is not meant for them.

Considering how the system is using GPS coordinates of up to 6 decimal places, we have a best case scenario accuracy of 0.11m, if that goes down to

Table 3.3: GPS accuracy

Decimal places	Degrees	Distance (m)
0	1	111,000
1	0.1	111,00
2	0.01	111,0
3	0.001	111
4	0.0001	11.100
5	0.00001	1.111
6	0.000001	0.111
7	0.0000001	0.011

5 decimal places in the real world for whatever reason our accuracy will go down to 1.1m. For this project we are considering the worst case scenario to be 5 decimal places. When doing testing the particular GPS module that was used, I found that the precision was 7 decimal places, while on the roof of a building. Therefore considering the typical precision on open roads to be 6 decimal places, with 5 being the minimum, makes sense.

In terms of the accuracy when it comes to the length of the road, we do not have much to be concerned for. As vehicles being a meter or two forward or backward makes little to no difference, we are not tackling collision avoidance here. When it comes to road width however, which is represented by the x-axis, 1 or 2 meters of inaccuracy can be a problem as the average road lane is between 2.4m to 3.6m in width[21]. Due to that, in order to actually achieve proper distinction for each lane it would have become rather complicated and possibly expensive, therefore the overall system got slightly modified instead. The consideration was whether exact lane distinction was necessary. It was found that the most common necessity of lane distinction would be for the vehicles making left hand turns (assuming the road network uses left hand driving), whereas in some scenarios vehicles may only be allowed to go straight and not allowed to make right or hand turns. To tackle the first distinction of left hand turns, the novel idea of using an inner radius in the infrastructure transmission code as previously explained solves it very simply. And for all the other lanes, due to their relative infrequency of use,

we are using the RFID tags that will be embedded into the road lanes at the intersection.

Now that all the considerations in terms of GPS precision has been made, we look into the matter of actually figuring out whether any signal is applicable to a particular vehicle or not. To achieve this two equations involving angles have been developed.

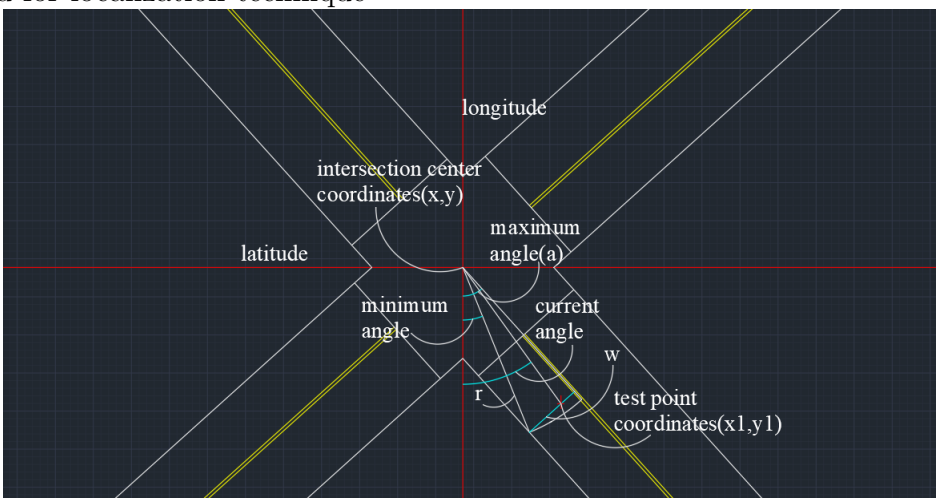
$$minimum\ angle = a - sin^{-1}\frac{w}{r} \tag{3.1}$$

$$currentangle = cos^{-1}\frac{|x| - |x1|}{r} \tag{3.2}$$

Where 'a' is the angle with the road makes with the longitudinal axis going over the center of the intersection, included within the transmission signal. 'w' and 'r' represent the width of the road and the radial distance of the vehicle from the intersection center respectively. And lastly 'x' is the longitudinal coordinate of the intersection center, which is included in the transmission from infrastructure unit and 'x1' is the longitudinal coordinate of the vehicle position.

The way this localization system works is that we consider three principle angles, one being the maximum angle, other is the minimum angle and finally the current angle. Our maximum angle is provided within the signal sent by the infrastructure unit, as already explained, it is the angle between the center line of the road and the longitudinal axis going over the intersection center. The minimum angle, which we are calculating using the given equation, is between the longitudinal axis going over the intersection center and the theoretical position of any vehicle had it been at the edge of the road whilst having the same radial distance. These two form the upper and lower limits, afterwards we simply calculate the angle that a particular vehicle makes with the intersection center, and check whether it is within out limits. If the current angle is less than the minimum angle it means that the vehicle is simply not in the road for which the signal is applicable,

Figure 3.4: overhead view of crossroad with all the necessary measurements used for localization technique



or if the current angle is greater than the maximum angle it means that the vehicle has either gone through the intersection already or in some other road.

Overall, using this process of localization the following has been achieved: not considering the vehicles plying on the side of the road going away from an intersection, not considering any vehicle that is on any adjacent roads, and making sure that every part of the applicable side of a road is covered.

3.4 Conclusion

So overall the system is having two core functions, one is to manage traffic at road intersections, which includes the exception feature for emergency vehicle, and secondly the transmission of road signs.

The traffic management system has an infrastructure unit, continually transmitting information to all vehicles, much like traditional traffic lights, but also takes feedback from stationary vehicles so that it knows the exact number of vehicles at each road. This enables the traffic signs to know exactly how many vehicles are on each road at any time, and thus utilize variable signal timing very accurately. In case of deducting the number of vehicles

however, for the ones that have gone by, we consider a bit of estimation. The average number of vehicles passing through any crossroad per second is to be observed, and then coded into the infrastructure unit. The actual code is arranged in a way so that this process is very easy.

Figure 3.5: visual depiction of the system working at an intersection

Transmission from road signs works much the same way as that from traffic signals, except it does not bother taking in any feedback, since road signs are static. The purpose of this unit is to ward drivers further ahead in time of important road signs earlier than it could have been possible using visual aid only. And we have to consider how sometimes weather, such as fog or rain can obstruct the view of possibly very important signs.

Figure 3.6: visual depiction of the system working with road signs

Chapter 4

Coding

4.1 Vehicle Unit

In order to run the unit, several libraries had been used to make it possible to interface with all the hardware and also to achieve some functionality which is not natively possible on the Atmel microcontroller. Also, with the view to quickly and easily test the system a special code for simulating the function of the GPS without actually connecting the specific hardware was used. The bit of code that was done for the simulation was within the same library that was used to interface with the GPS module, so when running the code in the real world one simply needs to replace the simulation code of GPS with the hardware code, everything else remains constant. For the thesis my GPS module was checked to make sure it worked as intended, and later using the same library, it was simulated in order to make the coding and debugging process much faster as GPS only works when there is a clear line of sight with the satellites. So basically it was impossible to actually use the GPS module when indoors unless the simulation system was used.
For the initialization part of the code, first all the libraries are added to the file, followed by the pin assignments and the fixed string which is the output of the simulated GPS module. The GPS output being fixed was of great help as it enabled easy verification of whether parts of the code was properly working or not. Afterwards we see the numerous variables used throughout the code, declared all at once for simpler debugging, and finally the initializing of all external hardware.

```
#include <SPI.h>
#include<nRF24L01.h>
#include<RF24.h>
#include<RF24_config.h>
#include<MFRC522.h>
#include <TinyGPS++.h>
#include "BigNumber.h"
#include <math.h>



RF24 radio(7,8);                //CE:7; CSE:8
const byte rxAddr[6] = "00001";

#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;

// A sample NMEA stream.
const char *gpsStream =
  "$GPRMC,045103.000,A,3014.1984,N,09749.2872,W,0.67,161.46,030913,,,A*7C\r\n"
  "$GPGGA,045104.000,3014.1985,N,09749.2873,W,1,09,1.2,211.6,M,-22.5,M,,0000*62\r
  \n"
  "$GPRMC,045200.000,A,3014.3820,N,09748.9514,W,36.88,65.02,030913,,,A*77\r\n"
  "$GPGGA,045201.000,3014.3864,N,09748.9411,W,1,10,1.2,200.8,M,-22.5,M,,0000*6C\r
  \n"
  "$GPRMC,045251.000,A,3014.4275,N,09749.0626,W,0.51,217.94,030913,,,A*7D\r\n"
  "$GPGGA,045252.000,3014.4273,N,09749.0628,W,1,09,1.3,206.9,M,-22.5,M,,0000*6F\r
  \n";
TinyGPSPlus gps;


int RFID_code;                  //RFID
String RFID_string_read="";     //RFID
byte readCard[16];              //RFID

char text[32] = {0};            //NRF
```

```
char bla[11]={"00000000000"};        //coordinates
char ha[4]={"0000"};                 //speed/intersection
char he[6]={000000};                 //inner radius
BigNumber x;                         //intersection x-axis (longitude)
BigNumber x1;                        //GPS x-axis (longitude)
BigNumber y;                         //intersection y-axis (latitude)
BigNumber y1;                        //GPS y-axis  (latitude)
BigNumber z;                         //inner radius
int s=0;                             //speedLimit/intersection
int s1;                              //GPS speed;
int a=0;                             //angle from signal
BigNumber d;                         //radial distance
double tempVal;                      //temporary double to store GPS data
double tempVal2;                     //temporary data for calculation
double tempVal3;                     //temporary data for calculation
double a1;                           //calculated angle from the intersection
int interNumber;                     //label of intersections
char gpsTime[]={"0000000000"};       //transmission code
String gt;                           //temp value
BigNumber w;                         //road width
double minAngle;                     //min angle to be inside the road

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  SPI.begin();

  mfrc522.PCD_Init();

  radio.begin();


  Serial.begin(9600);
  BigNumber::begin(6);
}
```

The sequence of the code itself is very straight forward, for simplicity all the key tasks have been broken down into functions, making debugging easy. First the data from the RFID module is taken, in the particular test done for the thesis RFID module was not required so it has been commented out, then transceiver receives signal, which gets parsed into usable data by 'getData' function, afterwards a function for GPS is run and finally a function to calculate the radial distance.

```
void loop() {

  /*RFID SECTION*/                    //RFID not always needed now
  /*RFID();
  Serial.println(RFID_code);*/

  /*RADIO SECTION*/
  receiver();
  //Serial.println(text);     //this is printing an array (for testing)

  getData();                  //pasring the received signal into usable infor

  /*GPS SECTION*/
  while (*gpsStream){
    if (gps.encode(*gpsStream++)){
      displayInfo();
    }
  }
  radialDistance();

  if(x>x1 && y>y1){interNumber=1;}      //bottom-left
  if(x<x1 && y<y1){interNumber=2;}      //top-right
  if(x<x1 && y>y1){interNumber=3;}      //bottom-left
  if(x>x1 && y<y1){interNumber=4;}      //top-left
```

In terms of the algorithm, my code first checks whether the received signal is applicable or not. It then compares the data collected by the vehicle unit

with that in the signal code; checking what kind of instruction the code is giving and then giving the appropriate instruction to driver based on the information collected by GPS and RFID. For the testing done in the this thesis, the focus was on the use of the system at traffic intersections, thus the comparison algorithm for road sign signal code, aside for speed limits, have not been added; however all the functionality is still present and it can be added very easily.

```
  if(checkAngle()){
    Serial.println("valid");

    if(s1>s && s<500){
     Serial.println("slow down");
    }
    if(s=(interNumber+995) && s>995){
      Serial.println("go");
    }
    if(interNumber!=(s-995) && s>995 && d>z){
      Serial.println("control speed");
    }
    if(d=z && interNumber!=(s-995)){
      Serial.println("stop");
    }
    if(s1<3 && interNumber!=(s-995)){
      getTime();                        //SPECIAL CASE: emergencyVehicle()
      transmitter();
      delay (2000);
      while(interNumber!=s){
        receiver();
        getData();
      }
      Serial.println("move");
    }
  }
}
```

The following are all the functions that have been used in the above code. It

should be noted that the code above, involving receiving data and running the algorithm, will run at a continuous loop.

```
void RFID(){
  if ( ! mfrc522.PICC_IsNewCardPresent()){
    return;}
  if ( ! mfrc522.PICC_ReadCardSerial()){
    return;}
  for(int i=0;i<mfrc522.uid.size; i++){
    readCard[i]=mfrc522.uid.uidByte[i];
    RFID_string_read=RFID_string_read+String(readCard[i]);
  }
  if(RFID_string_read.toInt()!=0){
    RFID_code=RFID_string_read.toInt();
  }
  mfrc522.PICC_HaltA();
}

void transmitter(){
    radio.openWritingPipe(rxAddr);
    radio.stopListening();
    //const char text[] = gpsTime;
    radio.write(&gpsTime, sizeof(gpsTime));
  }

void receiver(){
  radio.openReadingPipe(0, rxAddr);
  radio.startListening();
  if (radio.available()){
    radio.read(&text, sizeof(text));
  }
}

void displayInfo(){
  //Serial.print(F("Location: "));
  if (gps.location.isValid()){
    tempVal=gps.location.lat();
```

```
    dtostrf(tempVal,9,6,bla);
    y1=bla;
    Serial.print(y1);
    Serial.print(F(","));
    tempVal=gps.location.lng();
    dtostrf(tempVal,9,6,bla);
    x1=bla;
    Serial.print(x1);
    Serial.print(F(","));
    s1=gps.speed.kmph();
    //Serial.println(s1);
  }
  else{
    Serial.print(F("INVALID"));
  }
  Serial.println();
}

void getData(){
  for(int i=0; i<11; i++){
     bla[i]=text[i];
    }
  x=bla;                          //latitude; x-axis
  //Serial.println(x);

  for(int i=0; i<11; i++){
     bla[i]=text[i+11];
    }
  y=bla;                          //longitude; y-axis
  //Serial.println(y);

  for(int i=0; i<3; i++){
    ha[i]=text[i+27];
    }
  s=atoi(ha);                     //signal code
  //Serial.println(s);
  ha[2]="0";
```

```
  for(int i=0; i<5; i++){
     he[i]=text[i+22];
    }
  z=he;                              //innder radius
  //Serial.println(z);

  for(int i=0; i<2; i++){
    ha[i]=text[i+30];
    }
  a=atoi(ha);                        //road angle
  //Serial.println(a);

  for(int i=0; i<6; i++){
     he[i]=text[i+32];
    }
  w=he;                              //road width
  //Serial.println(z);
  }

void radialDistance(){
  d=((x-x1)*(x-x1))-((y-y1)*(y-y1));
  d=d.sqrt();
  Serial.println(d);
  }

bool checkAngle(){
    tempVal =(double)(y1);                      //y1 intersection y-axis
    tempVal2=(double)(y);                       //y is GPS y-axis
    tempVal=abs(tempVal);                       //d is radial distance from inter
    tempVal2=abs(tempVal2);                     //road width
    tempVal3=double(d);                         //a angle from signal to road
    a1=acos(abs(tempVal-tempVal2)/tempVal3);    //a1 angle from car to signal
    minAngle=a-asin(double(w)/double(d));

    if(interNumber==1 || interNumber==2){a1=90-a1;}
    if(a1>=minAngle && a1<a){
      return true;
      } else{
```

```
        return false;
        }
  }

void getTime(){
   if (gps.time.isValid())
  {
    if (gps.time.hour() < 10){
      gpsTime[0]='0';
      gt=gps.time.hour();
      gt.toCharArray(ha,4);
      gpsTime[1]=ha[0];
    } else{
      gt=gps.time.hour();
      gt.toCharArray(ha, 4);
      gpsTime[0]=ha[0];
      gpsTime[1]=ha[1];
      }
    if (gps.time.minute() < 10){
      gpsTime[2]='0';
      gt=gps.time.minute();
      gt.toCharArray(ha,4);
      gpsTime[3]=ha[0];
    } else{
      gt=gps.time.minute();
      gt.toCharArray(ha, 4);
      gpsTime[2]=ha[0];
      gpsTime[3]=ha[1];
      }
    if (gps.time.second() < 10){
      gpsTime[4]='0';
      gt=gps.time.second();
      gt.toCharArray(ha,3);
      gpsTime[5]=ha[0];
    } else{
      gt=gps.time.second();
      gt.toCharArray(ha, 4);
      gpsTime[4]=ha[0];
```

```
    gpsTime[5]=ha[1];
    }
  if (gps.time.centisecond() < 10){
    gpsTime[7]='0';
    gpsTime[6]='0';
    gt=gps.time.centisecond();
    gt.toCharArray(ha,2);
    gpsTime[8]=ha[0];
  } if(gps.time.centisecond()>=10 && gps.time.centisecond()<100){
    gpsTime[6]='0';
    gt=gps.time.centisecond();
    gt.toCharArray(ha, 4);
    gpsTime[7]=ha[0];
    gpsTime[8]=ha[1];
    } else{
      gt=gps.time.centisecond();
    gt.toCharArray(ha, 4);
    gpsTime[6]=ha[0];
    gpsTime[7]=ha[1];
    gpsTime[8]=ha[2];
      }
    gpsTime[9]=interNumber;
}
else
{
  Serial.print(F("INVALID"));
}
}

void emergencyVehicle(){
  for(int c=0; c<9; c++){
    gpsTime[c]='x';
    }
  gpsTime[9]=interNumber;
  }
```

4.2 Infrastructure Unit

The infrastructure unit of the system also requires some libraries to function, although as many since the only external hardware it is utilizing is a transceiver. In terms of coding however this unit is relatively more complex despite the lesser lines of code.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8);

const byte rxAddr[6] = "00001";

char transmission[]={"0034.123456-094.6543210.002999850.0001"};
char text[32]={0};
int t=10;               //signal countdown
int c=0;                //transceiver switching count
int temp;               //receiver channel no.
int ech;                //channel with emergency vehicle
int ch[4];              //array for channel ratio no.
int chmin;              //channel with minimum number
int chi=0;              //current channel with signal
long rec=0;             //makes sure same car isn't counted twice
long track=000000000;   //for saving the unique ID of transmitting car
int cap=50;             //intersection capacity
int x=3;                //average number of cars that cross intersection per loop
                        //(assuming one loop takes 1 second)
int etime=5;            //time given for roads with emergency vehicles

//char reading

void setup()
{
  radio.begin();
  //radio.setRetries(15, 15);
```

```
  Serial.begin(9600);
}
```

For the part of the code which runs in a continuous loop, there are two key sections, one for transmission and one for receiving. The latter part is where the infrastructure unit takes feedback in order to determine the amount of congestion, whereas in transmission mode it is sending the signal which all vehicles receive. To achieve this swapping of functionality we use a simple counter, when the value is odd the unit is on feedback mode and when it is even the unit transmits its signal. The counter loops after it reaches a particular high number.

In the transmission part of the code we transmit a character array which is the transmission code. Part of that code gets continuously modified when the infrastructure unit is controlling a traffic signal, as the instruction on which road is allowed to pass through has to change. The feedback part of my code, which receives a signal from stationary vehicles, parses the signal in order to get the vehicle's time stamp and intersection number as previously explained. Note how this part of the code contains the special condition for emergency vehicles, adding an exception to the typical cycle to allow vehicles of a particular road to pass through first.

```
void loop()
{
  if(c%2==0){
    radio.openWritingPipe(rxAddr);
    radio.stopListening();
    //const char text[] = "0034.123456-094.6543210.00299985";   //for testing
    radio.write(&transmission, sizeof(transmission));
    //Serial.println("transmitting");                            //for testing
    //Serial.println(c);                                       //for testing
  }
  else{
    radio.openReadingPipe(0, rxAddr);
```

```
  radio.startListening();
  if (radio.available()){
    text[32] = {0};
    radio.read(&text, sizeof(text));
    //Serial.println(text);                                    //for testing

    for(int i=1; i<10; i++){
      rec=(long)text[i-1];
      rec=rec*pow(text[i-1],(9-i));
      }
      temp=text[9];
      if(rec=="xxxxxxxxx"){
        temp=ech;
        emergencyVehicle();
        }
  }
  //Serial.println(c);                                        //for testing
}
```

The algorithm part of infrastructure unit's code is as follows; first is the counter which the transmission and feedback mode cycling relies on, decrementing of the timer counting how long each road is given the 'go' sign and also subtracting the supposed number of vehicles in that road, running a function to add stationary vehicles to the system, and finally two functions which start only once the timer turns to zero to count the ratio of all stationary vehicles at the intersection and calculate the amount of time the next road should receive for its vehicles to pass through.

```
c++;
 if(c==32001){c=0;}
 delay(800);
 t--;
 ch[chi]=ch[chi]-x;
 if(ch[chi]<1){ch[chi]=1;}
 updateTraffic();
 if(t==0){
```

```
    calRatio();
    timerfunction();
    }
}

void updateTraffic(){
  if(rec>track){
      track=rec;
      if(temp==1){ch[0]++;}
      if(temp==2){ch[1]++;}
      if(temp==3){ch[2]++;}
      if(temp==4){ch[3]++;}
  }
}
void calRatio(){
      int m;
      chmin=ch[0];
      for(int i=0; i<4; i++){
        if (ch[i]<chmin){chmin=ch[i];m=i;}
        }
      for(int i=0; i<4; i++){ch[i]=ch[i]/chmin;}
    }

  void timerfunction(){
    if(chi==4){chi=0;}
    chi++;
    transmission[29]=(char)(chi+5);
    t=ch[chi]*t;
    if(t>12){t=12;}
    }

  void emergencyVehicle(){
    transmission[29]=(char)(ech+5);
    etime=5;
    }
```

Chapter 5

Simulation

5.1 Overview

Due to the complexity and expense of actually testing such a system in the real world a computer based simulation was used. As the main focus throughout the thesis had been the usage of the system at road intersection that is what the simulation was based on. Using Matlab the difference in terms of overall congestion over a period of time was observed. The comparison was done between a simulation of the thesis where the traffic signal timing was continuously modified based on the number of stationary vehicles at the intersection, and another instance where the traffic signal gave a constant time to every road as it had no clue about the level of congestion at each road.

The test was done on a simulation of a very busy road intersection where 4 roads meet, each with 6 lanes. In order to keep things realistic the intersection initially starts with a random number of cars, with additional random number of cars getting added to each road after a certain time. And obviously the aspects of initial number of vehicles and the number getting added was the same for both instances that have been simulated.

In order to keep the comparison strictly between the two methods of traffic management, certain real world variables were neglected. It is assumed that in our simulated environment all vehicles are exactly the same and there is no such thing as driver reaction time. The amount of space in between each car is also the same, and so is the driving style of every vehicle, meaning that

they all accelerate and deaccelerate at the same pace.

When it comes to the number of cars that have gone through an intersection, as stated before the system is normally supposed to be adjusted for every specific intersection based on real world observations. For this simulation we obviously do not have any real world reference, therefore a fixed set of presets have been selected, which are the same for every car, and based on that the following equation has been derived.

$$numberofvehicles = 3 + \frac{3}{0.41}t - 5.41 \tag{5.1}$$

This gives us the number of vehicles that have gone through the intersection within time 't'. The preset conditions that have been used to derive this equation are as follows:

$$intersectionlength = 30m$$
$$acceleration = 2m/s^2$$
$$topspeed = 40km/h = 11.11m/s$$
$$vehiclelength = 4.5m$$

For the number of cars in the simulated road, we are making things somewhat realistic by using the random function. Each road of the intersection can start with 0-30 vehicles, and an additional 0-5 vehicles after every 5 seconds in each road.

5.2 Code

Matlab code has been arranged in a very simplistic manner, making debugging as easy as possible, with all the constants and initial congestion at the intersection at the very start, then a loop, and finally a graph showing the results. There are two separate arrays which keep count of the number of vehicles in each road in case of the intersection following a fixed timing system and another that is dynamic timing system, the latter representing my system designed for the thesis.

```
clear all;
close all;
clc;

carsSim=randi(30,1,4);           %number of cars in sim intersection
carsControl=carsSim;             %number of cars in control intersection
carsRatio=[0,0,0,0];             %array to keep ratio of car numbers in each road
roadT=[7,7,7,7];                 %alloted time for each road in seconds
road=1;                          %sim rouad currently facing green light
cRoad=1;                         %control road currently facing green light
realTime=roadT(road);            %counter for green light timing
totalSimCars=zeros(360,1);       %array for total cars in traffic on sim
totalControlCars=zeros(360,1);   %array for total cars in traffic for control
```

The loop part contains 5 key sections, the first is where a random number of vehicles are added to each road of the intersection. Then the ratio of vehicles in each road of the intersection is taken. Following two sections deduct the number of cars in two different methods in order to simulate vehicles leaving the intersection. The two different methods being one that is simulating a regular fixed time traffic signaling and another that constantly changes its timing depending on vehicle ratio. Lastly we take the total number of stationary vehicles in that particular loop for each of the two simulated situations. It should be noted that one loop represents one second, meaning the whole simulation is done for 6 minutes.

```
for t=1:360

   %additional traffic every 5 seconds
   if (mod(t,5))==0;
       carAdd=randi(5,1,4);
       carsSim=carsSim+carAdd;
       carsControl=carsControl+carAdd;
    end
```

```
%taking the ratio of cars
m=min(carsSim);
if m<1
    m=1;
end
carsRatio=carsSim./m;
for a=1:4
    if a~=road
        roadT(a)=6*carsRatio(a);
    end
    if roadT(a)>12
        roadT(a)=12;
    end
end

%deducting simulation car number after alloted time
realTime=realTime-1;
if realTime<1;
    carsSim(road)=carsSim(road)-((3*sqrt((1/0.41)*(roadT(road)-5.48)))+3);
    if carsSim(road)<0
        carsSim(road)=0;
    end
    road=road+1;
    if road>4
        road=1;
    end
    realTime=roadT(road);
end

%deducting cars from control
if (mod(t,7))==0
    carsControl(cRoad)=carsControl(cRoad)-8;
end
if carsControl(cRoad)<0
    carsControl(cRoad)=0;
end
cRoad=cRoad+1;
if cRoad>4
```

```
        cRoad=1;
    end

    %total congestion
    totalSimCars(t)=sum(carsSim);
    totalControlCars(t)=sum(carsControl);
end
plot(1:360,totalSimCars,1:360,totalControlCars),
title('Total congestion between fixed timing and varied timing'),
xlabel('time(seconds)'),ylabel('total no. of cars across all roads'),
legend('varied timing congestion','fixed timing
congestion','Location','northwest'),
grid on;
```

It should be noted that in the codes 'simulation' refers to the simulation of the thesis whereas 'control' is the simulation of the standard system of fixed timing traffic signal, named so because the latter is only there for comparison.

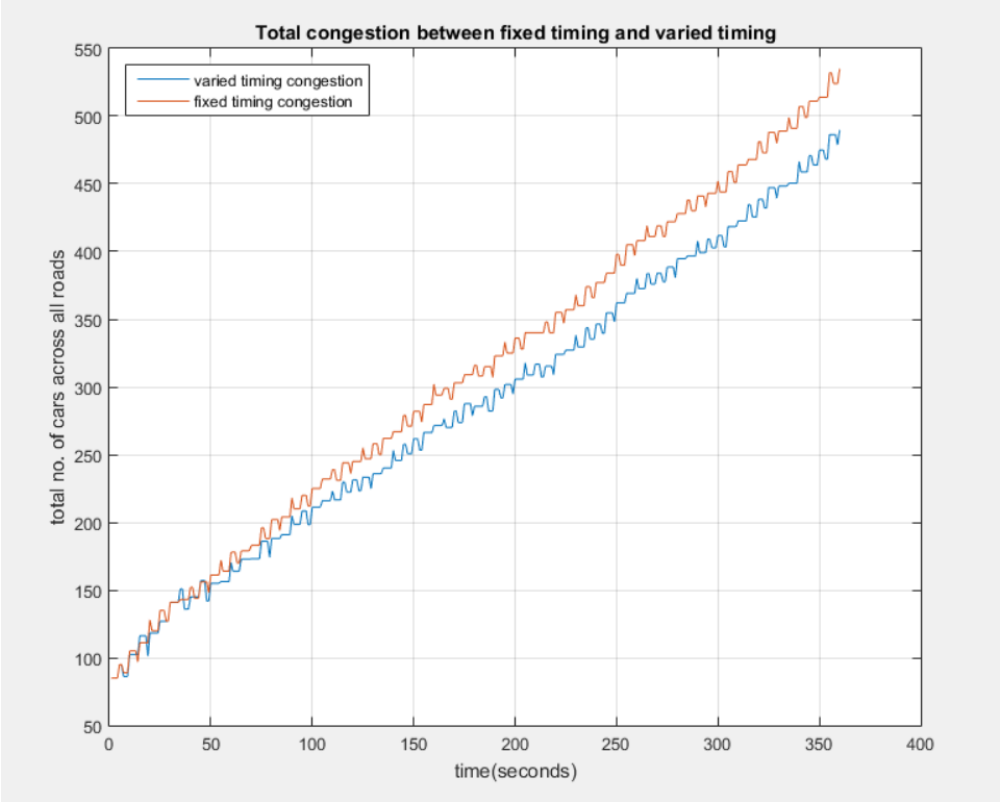
5.3 Results

As per the simulation results, showing the total number of stationary vehicles in each second, it is very much apparent the benefits of the thesis system as opposed to a typical method of not varying the traffic signal timing. It can be seen clearly that despite the same conditions, the constant rise in traffic eventually clogs up the intersection using fixed timing whereas the variable timing system used for the thesis simulation manages to keep the overall congestion at a relatively low level.

One aspect that should be kept in mind however is that my system has been designed to be most effective for intersections that are already facing very large amounts of traffic, which is exactly what the simulation has been based on. In areas where the number of vehicles is not as high the effectiveness of the thesis system will not be as significant as what is observed from the simulation results. However that is to be expected as the system has been designed specifically to meet the congestion problems of core city areas where there is not much room for road network expansion. In other areas this

system may be used in conjunction with road signs in order to raise safety levels.

Figure 5.1: graph showing the total number of stationary vehicles in every road at each second



Chapter 6

Conclusion

6.1 Feasibility

The purpose of the thesis project was to create a system which enables efficient use of existing road networks through automation, with a view to being compatible with future changes as well. To that extent the final system is highly feasible since the area for which the system has been designed for, which is city centers, does not really have much option in terms of expansion and has already shown readiness to installing new hardware technology in order to lower congestion.

One big consideration of this system is obviously the way it is assumed that every vehicle will have a transceiver, RFID module and a GPS. However if looking at the existing situation of the automotive industry, and where appears to be going, it is not an unfair assumption that these pieces of hardware can quiet easily be built into every vehicle. For one, a GPS is already standard equipment in every new car getting released. In case of transceivers, it is a rather old and relatively cheap piece of equipment that can easily be fitted to any vehicle even outside the factory. The only bit part that is somewhat unlikely is the RFID module, however looking at the amount of research done by vehicle manufacturers in order to achieve something like image recognition, a technology that is incredibly more expensive to develop and install to vehicles, something as simple as an RFID module can easily be fitted; and even then, according to the way the system is designed, it will still work without the RFID module, only with slightly lesser functionality.

A very important factor of the designed system has also been making it flexible enough to work with any future changes to the transportation system. From the current news it would appear that the future development in automobiles is heading towards self-driving vehicles, something which can greatly benefit from and easily work with this system. For now the focus of self-driving systems has been detection of the surrounding areas, as they are still made to mainly help the human drivers rather than take over completely. However in the future when they are completely autonomous, those cars can benefit significantly from the wireless communication system of this project as in some cases the image recognition technology of the self-driving vehicles can face the same hurdles as human drivers, such as low visibility in bad weather conditions and low reaction times due to processing speed.

6.2 Future work

Key part of the thesis was obviously managing the congestion at road intersections, however it always been assumed that vehicles will definitely have to stop and wait at some point, the focus was simply to reduce this waiting time. However, if we completely do away with human drivers and start considering self-driving cars, it is possible to entirely eliminate the concept of actually taking turns at crossroads. The fundamental idea, as already established, is that all the vehicles will simply be aware of every other vehicle's presence and will thus avoid each other.

An extension of this concept can be done via the system of this thesis. It is possible, with only changing the transceivers with something that can transmit and receive at the same time only and at a somewhat greater bandwidth, to modify the system so that the infrastructure unit stores in its memory a sort of grid, each block of which is big enough to contain one vehicle. The grid is basically a representation of the intersection center. The oncoming vehicles then simply 'book' certain parts of the grid that they plan to use in order to go their way and inform the infrastructure unit of exactly when it plans to use those specific blocks of the grid. Infrastructure unit, acting as a sort of book keeper, makes sure no two vehicles plan on using the same blocks of the grid at the same time to avoid collisions. It informs any vehicle of overlapping block usage when vehicle is in the booking process so that

vehicles simply change the speed and/or course of movement accordingly.

A key benefit of this system, as opposed to the depth detection using a LIDAR system, is that the vehicles would not need to be constantly checking for the speed and course of movement of all the other automobiles. This not only reduces the processing power needed on every car, thus reducing costs, but also avoids the need of constantly changing the vehicle speed and direction in order to avoid any potential collision. This makes the overall journey much more comfortable for the passenger and also raises fuel efficiency due to the relatively more stable speeds.

Bibliography

[1] "The Benefits of Retiming Traffic Signals - ProQuest", Search.proquest.com, 2017. [Online]. Available: https://search.proquest.com/openview/55e90e85f778e5b828bf54b2017a9295/1?pq-origsite=gscholar&cbl=42116.[Accessed: 14- Aug- 2017].

[2] S. Bhover, A. Tugashetti and P. Rashinkar, "V2X communication protocol in VANET for co-operative intelligent transportation system", 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2017.

[3] S. Ramachandra, K. Reddy, V. Vellore, S. Karanth and T. Kamath, "A novel dynamic traffic management system using on board diagnostics and Zigbee protocol", 2016 International Conference on Communication and Electronics Systems (ICCES), 2016.

[4] J. Erdmann, R. Oertel and P. Wagner, "VITAL: A Simulation-Based Assessment of New Traffic Light Controls", 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015.

[5] T. Osman, S. Psyche, J. Ferdous and H. Zaman, "Intelligent traffic management system for cross section of roads using computer vision", 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), 2017.

[6] "Forbes Welcome", Forbes.com, 2017. [Online]. Available: https://www.forbes.com/sites/federicoguerrini/2014/10/14/traffic-congestion-costs-americans-124-billion-a-year-report-says/#1f573d61c107. [Accessed: 13- Aug- 2017].

[7] "The cost of traffic jams", Economist.com, 2017. [Online]. Available: https://www.economist.com/blogs/economist-explains/2014/11/economist-explains-1. [Accessed: 13- Aug- 2017].

[8] L. Chapman, "Transport and climate change: a review", Journal of Transport Geography, vol. 15, no. 5, pp. 354-367, 2007.

[9] "Sources of Greenhouse Gas Emissions US EPA", US EPA, 2017. [Online]. Available: https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions. [Accessed: 13- Aug- 2017].

[10] "How Vehicle Emissions Affect Us", Env.gov.bc.ca, 2017. [Online]. Available: http://www.env.gov.bc.ca/epd/bcairquality/topics/vehicle-emissions-impacts.html. [Accessed: 13- Aug- 2017].

[11] "Emissions from traffic congestion may shorten lives", News, 2017. [Online]. Available: https://www.hsph.harvard.edu/news/hsph-in-the-news/air-pollution-traffic-levy-von-stackelberg/. [Accessed: 13- Aug- 2017].

[12] A. Chowdhury, "Priority based and secured traffic management system for emergency vehicle using IoT", 2016 International Conference on Engineering and MIS (ICEMIS), 2016.

[13] Supreeth H.S.G and C. Patil, "An approach towards efficient detection and recognition of traffic signs in videos using neural networks", 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2016.

[14] A. Borowsky, D. Shinar and T. Oron-Gilad, "Age, skill, and hazard perception in driving", Accident Analysis and Prevention, vol. 42, no. 4, pp. 1240-1249, 2010.

[15] G. Andrei, T. Serban, S. Alexandru and B. Dorin, "Simulation of traffic management systems using arduino boards", 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2016.

[16] "nRF24/RF24", GitHub, 2017. [Online]. Available: https://github.com/nRF24/RF24. [Accessed: 15- Aug- 2017].

[17] M. Hart, "TinyGPS++ Arduiniana", Arduiniana.org, 2017. [Online]. Available: http://arduiniana.org/libraries/tinygpsplus/. [Accessed: 15- Aug- 2017].

[18] "nickgammon/BigNumber", GitHub, 2017. [Online]. Available: https://github.com/nickgammon/BigNumber. [Accessed: 15- Aug- 2017].

[19] M. Balboa, "miguelbalboa/rfid", GitHub, 2012. [Online]. Available: https://github.com/miguelbalboa/rfid. [Accessed: 15- Aug- 2017].

[20] "Measuring accuracy of latitude and longitude?", Gis.stackexchange.com, 2017. [Online]. Available: https://gis.stackexchange.com/questions/8650/measuring-accuracy-of-latitude-and-longitude. [Accessed: 15- Aug- 2017].

[21] "How wide are freeway lanes?", Dot.ca.gov, 2017. [Online]. Available: http://www.dot.ca.gov/hq/paffairs/faq/faq92.htm. [Accessed: 15- Aug- 2017].

Appendix A

Part specifications

nRF24L01
Single Chip 2.4GHz Transceiver

Product Specification

Key Features

- Worldwide 2.4GHz ISM band operation
- Up to 2Mbps on air data rate
- Ultra low power operation
- 11.3mA TX at 0dBm output power
- 12.3mA RX at 2Mbps air data rate
- 900nA in power down
- 22µA in standby-I
- On chip voltage regulator
- 1.9 to 3.6V supply range
- Enhanced ShockBurst™
- Automatic packet handling
- Auto packet transaction handling
- 6 data pipe MultiCeiver™
- Air compatible with nRF2401A, 02, E1 and E2
- Low cost BOM
- ±60ppm 16MHz crystal
- 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

Applications

- Wireless PC Peripherals
- Mouse, keyboards and remotes
- 3-in-one desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracing systems
- Toys

July 2007

1.1 Features

Features of the nRF24L01 include:

- Radio
  - Worldwide 2.4GHz ISM band operation
  - 126 RF channels
  - Common RX and TX pins
  - GFSK modulation
  - 1 and 2Mbps air data rate
  - 1MHz non-overlapping channel spacing at 1Mbps
  - 2MHz non-overlapping channel spacing at 2Mbps
- Transmitter
  - Programmable output power: 0, -6, -12 or -18dBm
  - 11.3mA at 0dBm output power
- Receiver
  - Integrated channel filters
  - 12.3mA at 2Mbps
  - -82dBm sensitivity at 2Mbps
  - -85dBm sensitivity at 1Mbps
  - Programmable LNA gain
- RF Synthesizer
  - Fully integrated synthesizer
  - No external loop filer, VCO varactor diode or resonator
  - Accepts low cost ±60ppm 16MHz crystal
- Enhanced ShockBurst™
  - 1 to 32 bytes dynamic payload length
  - Automatic packet handling
  - Auto packet transaction handling
  - 6 data pipe MultiCeiver™ for 1:6 star networks
- Power Management
  - Integrated voltage regulator
  - 1.9 to 3.6V supply range
  - Idle modes with fast start-up times for advanced power management
  - 22uA Standby-I mode, 900nA power down mode
  - Max 1.5ms start-up from power down mode
  - Max 130us start-up from standby-I mode
- Host Interface
  - 4-pin hardware SPI
  - Max 8Mbps
  - 3 separate 32 bytes TX and RX FIFOs
  - 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

2.2 Pin functions

Pin	Name	Pin function	Description
1	CE	Digital Input	Chip Enable Activates RX or TX mode
2	CSN	Digital Input	SPI Chip Select
3	SCK	Digital Input	SPI Clock
4	MOSI	Digital Input	SPI Slave Data Input
5	MISO	Digital Output	SPI Slave Data Output, with tri-state option
6	IRQ	Digital Output	Maskable interrupt pin. Active low
7	VDD	Power	Power Supply (+1.9V - +3.6V DC)
8	VSS	Power	Ground (0V)
9	XC2	Analog Output	Crystal Pin 2
10	XC1	Analog Input	Crystal Pin 1
11	VDD_PA	Power Output	Power Supply Output(+1.8V) for the internal nRF24L01 Power Amplifier. Must be connected to ANT1 and ANT2 as shown in Figure 30.
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+1.9V - +3.6V DC)
16	IREF	Analog Input	Reference current. Connect a 22kΩ resistor to ground. See: Figure 30.
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+1.9V - +3.6V DC)
19	DVDD	Power Output	Internal digital supply output for de-coupling purposes. See: Figure 30.
20	VSS	Power	Ground (0V)

Table 1. nRF24L01 pin function

	E-1612-UB
	Ultra High Sensitivity and Low Power
	GPS Receiver Module

General Description

The E-1612-UB module series is a family of stand-alone GPS receivers featuring the high performance u-blox 5 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4mm package. Their compact architecture and power and memory options make E-1612-UB modules ideal for battery operated mobile devices with very strict cost and space constraints.

The 50-channel u-blox 5 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with over 1 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving E-1612-UB GPS receivers excellent navigation performance even in the most challenging environments.

E-1612-UB modules are not designed for life saving or supporting devices or for aviation and should not be used in products that could in any way negatively impact the security or health of the user or third parties or that could cause damage to goods.

Applications

- LBS (Location Based Service)
- Mobile phone
- PND (Portable Navigation Device)
- Vehicle navigation system

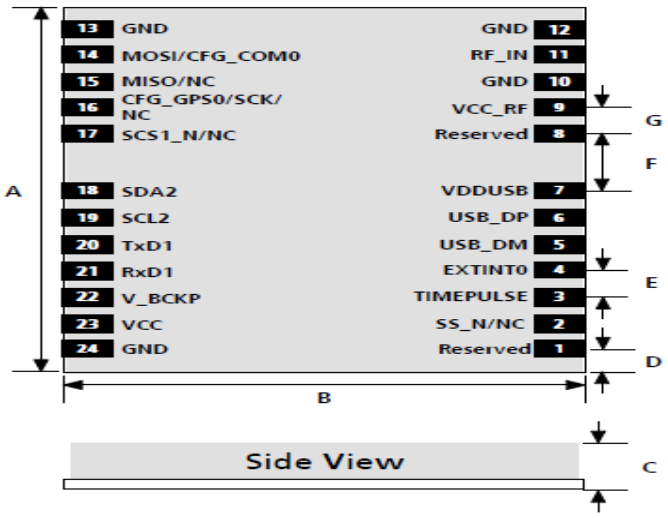
Figure 1: E-1612 -UB Top View

Features

- Build on high performance, low-power UB-5010 chipset
- Ultra high sensitivity: -160dBm
- Extremely fast TTFF at low signal level
- Built in high gain LNA

- Low power consumption: Max 40mA@3.0V
- NMEA-0183 compliant protocol or custom protocol
- Operating voltage: 2.75V to 3.6V
- Operating temperature range: -40 to 85℃
- SMD type with stamp holes
- Small form factor: 16x12.2x2.4mm
- RoHS compliant (Lead-free)

Pin Assignment



Parameter	Specification
A	16.0±0.6/−0.1mm
B	12.2±/−0.1mm
C	2.4±/−0.2mm
D	1.0±0.3/−0.1mm
E	1.1±/−0.1mm
F	3.0±/−0.1mm
G	1.1±/−0.1mm
Weight:	1.6g

Figure 2: E-1612-UB Pin Packag

Performance Specification

Parameter	Specification
Receiver Type	L1 frequency band, C/A code, 50-channels SBAS: WAAS, EGNOS, MSAS, GAGAN
Sensitivity	Tracking -160dBm

	Acquisition	-160dBm
Accuracy	Position	5m CEP without SA
	Velocity	0.1m/s without SA
	Timing (PPS)	10ns RMS
Acquisition Time	Cold Start	29s
	Warm Start	28s
	Hot Start	1s
	Re-Acquisition	<1s
Power Consumption	Tracking	35mA @3V Vcc
	Acquisition	40mA
	Sleep/Standby	TBD
NavigationDataUpdate Rate	1Hz	
Operational Limits	Altitude	Max 18,000m
	Velocity	Max 515m/s
	Acceleration	Less than 4g

Interfaces Configuration

1.1Assisted GPS (A-GPS)

Supply of aiding information like ephemeris, almanac, rough last position and time and satellite status and an optional time synchronization signal will reduce time to first fix significantly and improve the acquisition sensitivity. E-1612-UB modules support the u-blox AssistNow Online and AssistNow Offline A-GPS services8 and are OMA SUPL compliant.

1.2 SuperSense Indoor GPS

MFRC522

Standard performance MIFARE and NTAG frontend

Rev. 3.9 — 27 April 2016
112139

Product data sheet
COMPANY PUBLIC

1. Introduction

This document describes the functionality and electrical specifications of the contactless reader/writer MFRC522.

Remark: The MFRC522 supports all variants of the MIFARE Mini, MIFARE 1K, MIFARE 4K, MIFARE Ultralight, MIFARE DESFire EV1 and MIFARE Plus RF identification protocols. To aid readability throughout this data sheet, the MIFARE Mini, MIFARE 1K, MIFARE 4K, MIFARE Ultralight, MIFARE DESFire EV1 and MIFARE Plus products and protocols have the generic name MIFARE.

1.1 Differences between version 1.0 and 2.0

The MFRC522 is available in two versions:

- MFRC52201HN1, hereafter referred to version 1.0 and
- MFRC52202HN1, hereafter referred to version 2.0.

The MFRC522 version 2.0 is fully compatible to version 1.0 and offers in addition the following features and improvements:

- Increased stability of the reader IC in rough conditions
- An additional timer prescaler, see Section 8.5.
- A corrected CRC handling when RX Multiple is set to 1

This data sheet version covers both versions of the MFRC522 and describes the differences between the versions if applicable.

2. General description

The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO/IEC 14443 A/MIFARE and NTAG.

The MFRC522's internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 A/MIFARE cards and transponders without additional active circuitry. The receiver module provides a robust and efficient implementation for demodulating and decoding signals from ISO/IEC 14443 A/MIFARE compatible cards and transponders. The digital module manages the complete ISO/IEC 14443 A framing and error detection (parity and CRC) functionality.

The MFRC522 supports MF1xxS20, MF1xxS70 and MF1xxS50 products. The MFRC522 supports contactless communication and uses MIFARE higher transfer speeds up to 848 kBd in both directions.

The following host interfaces are provided:

- Serial Peripheral Interface (SPI)
- Serial UART (similar to RS232 with voltage levels dependant on pin voltage supply)
- I^2C-bus interface

3. Features and benefits

- Highly integrated analog circuitry to demodulate and decode responses
- Buffered output drivers for connecting an antenna with the minimum number of external components
- Supports ISO/IEC 14443 A/MIFARE and NTAG
- Typical operating distance in Read/Write mode up to 50 mm depending on the antenna size and tuning
- Supports MF1xxS20, MF1xxS70 and MF1xxS50 encryption in Read/Write mode
- Supports ISO/IEC 14443 A higher transfer speed communication up to 848 kBd
- Supports MFIN/MFOUT
- Additional internal power supply to the smart card IC connected via MFIN/MFOUT
- Supported host interfaces
  - SPI up to 10 Mbit/s
  - I^2C-bus interface up to 400 kBd in Fast mode, up to 3400 kBd in High-speed mode
  - RS232 Serial UART up to 1228.8 kBd, with voltage levels dependant on pin voltage supply
- FIFO buffer handles 64 byte send and receive
- Flexible interrupt modes
- Hard reset with low power function
- Power-down by software mode
- Programmable timer
- Internal oscillator for connection to 27.12 MHz quartz crystal
- 2.5 V to 3.3 V power supply
- CRC coprocessor
- Programmable I/O pins
- Internal self-test

4. Quick reference data

Table 1. Quick reference data

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
V_{DDA}	analog supply voltage	$V_{DD(PVDD)} \leq V_{DDA} = V_{DDD} = V_{DD(TVDD)}$; $V_{SSA} = V_{SSD} = V_{SS(PVSS)} = V_{SS(TVSS)} = 0$ V	[1][2]	2.5	3.3	3.6	V
V_{DDD}	digital supply voltage			2.5	3.3	3.6	V
$V_{DD(TVDD)}$	TVDD supply voltage			2.5	3.3	3.6	V
$V_{DD(PVDD)}$	PVDD supply voltage		[3]	1.6	1.8	3.6	V
$V_{DD(SVDD)}$	SVDD supply voltage	$V_{SSA} = V_{SSD} = V_{SS(PVSS)} = V_{SS(TVSS)} = 0$ V		1.6	-	3.6	V

Table 1. Quick reference data *...continued*

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
I_{pd}	power-down current	$V_{DDA} = V_{DDD} = V_{DD(TVDD)} = V_{DD(PVDD)} = 3$ V					
		hard power-down; pin NRSTPD set LOW	[4]	-	-	5	µA
		soft power-down; RF level detector on	[4]	-	-	10	µA
I_{DDD}	digital supply current	pin DVDD; $V_{DDD} = 3$ V		-	6.5	9	mA
I_{DDA}	analog supply current	pin AVDD; $V_{DDA} = 3$ V, CommandReg register's RcvOff bit = 0		-	7	10	mA
		pin AVDD; receiver switched off; $V_{DDA} = 3$ V, CommandReg register's RcvOff bit = 1		-	3	5	mA
$I_{DD(PVDD)}$	PVDD supply current	pin PVDD	[5]	-	-	40	mA
$I_{DD(TVDD)}$	TVDD supply current	pin TVDD; continuous wave	[6][7][8]	-	60	100	mA
T_{amb}	ambient temperature	HVQFN32		−25	-	+85	°C

[1] Supply voltages below 3 V reduce the performance in, for example, the achievable operating distance.

[2] V_{DDA}, V_{DDD} and $V_{DD(TVDD)}$ must always be the same voltage.

[3] $V_{DD(PVDD)}$ must always be the same or lower voltage than V_{DDD}.

[4] I_{pd} is the total current for all supplies.

[5] $I_{DD(PVDD)}$ depends on the overall load at the digital pins.

[6] $I_{DD(TVDD)}$ depends on $V_{DD(TVDD)}$ and the external circuit connected to pins TX1 and TX2.

[7] During typical circuit operation, the overall current is below 100 mA.

[8] Typical value using a complementary driver configuration and an antenna matched to 40 Ω between pins TX1 and TX2 at 13.56 MHz.

5. Ordering information

Table 2. Ordering information

Type number	Package		
	Name	Description	Version
MFRC52201HN1/TRAYB[1]	HVQFN32	plastic thermal enhanced very thin quad flat package; no leads; 32 terminal; body 5 × 5 × 0.85 mm	SOT617-1
MFRC52201HN1/TRAYBM[2]	HVQFN32	plastic thermal enhanced very thin quad flat package; no leads; 32 terminal; body 5 × 5 × 0.85 mm	SOT617-1
MFRC52202HN1/TRAYB[1]	HVQFN32	plastic thermal enhanced very thin quad flat package; no leads; 32 terminal; body 5 × 5 × 0.85 mm	SOT617-1
MFRC52202HN1/TRAYBM[2]	HVQFN32	plastic thermal enhanced very thin quad flat package; no leads; 32 terminal; body 5 × 5 × 0.85 mm	SOT617-1

[1] Delivered in one tray.

[2] Delivered in five trays.

Appendix B

equation derivation

B.1 Deriving equation used for simulation

First we find out the time taken to go across the length of the intersection.

$$
\begin{aligned}
s &= 0.5u + vt \\
t &= \frac{s}{0.5u + v} \\
t &= \frac{30}{0.50 + 11.11} \\
t &= 5.41s
\end{aligned}
$$

Now we consider the time taken for the second tow of vehicles to cross the intersection, keeping in mind that they have to go a bit further. This additional distance is the length of one vehicle.

$$
\begin{aligned}
t &= \frac{s}{0.5u + v} \\
&= \frac{4.5}{0.511 + 11.11} \\
&= 0.41s
\end{aligned}
$$

The number of rows of vehicles that can pass through in 't' time.

$$n = \frac{1}{0.41n}(t - 5.41) \qquad\qquad n = \sqrt{\frac{1}{0.41n}(t - 5.41)}$$

So finally we get the equation for number of vehicles that pass through.

$$numberofcars = 3 + 3n$$
$$numberofcars = 3 + 3\sqrt{\frac{1}{0.41n}(t - 5.41)}$$