King Fahd University of Petroleum and Minerals
College of Computer science and Engineering
Department of Computer Engineering

COE 485 - Term 142

Max Speed Indicator

(عين)

Course instructor: Dr.Ahmad Khayyat

Advisor: Dr.Kamel Chenaoua

Team Members:

Jalal AlRukhaimi	200952450
Hassan Al Shabaan	200968350
Abdul-Aziz Al-Amri	200815940

Table of Contents

1 Introduction 4
2 Problem Statement 4
3 Background 5
4 Requirements and Specifications 5
4.1 Functional Requirement 5
4.2 Non-Functional Requirement 5
4.3 Technical Specification 5
5 System Design 6
5.1 Solution Concept 6
5.2 Architecture 7
5.3 Component Design 10
Project Algorithm: 10
System Components: 11
Off-the-shelf Components: 11
Custom Components: 13
5.4 System Integration 18
6 Testing, Analysis, and Evaluation 18
Speed Limit Sign Detector: ..... **Error! Bookmark not defined.**
7 Issues 20
GPS installation & testing: 20
Python Tkinter 2.7 vs 3.4: 20
OBD2 communication: 20
OpenCV installation 20
DataBase ..... **Error! Bookmark not defined.**
Sign Detection: 21
8 Engineering Tools and Standards 22
8.1 Tools: 22
8.2 Standards: 23
9 Teamwork 24
10 Conclusion 25

List of Figures

Figure 1 Action Sequence _____ 6
Figure 2 System Architecture _____ 7
Figure 3 Hardware Components Figure 4 Software Components _____ 8
Figure 5 System Algorithm _____ 10
Figure 6 Filtering then using templete method _____ 13
Figure 7 Filtering then using circle detection method _____ 14
Figure 8 classification example _____ 16
Figure 9 illustrates the formula of obtaining the angle of a vector of two points. _____ 17
Figure 10 GPS outputs _____ 18
Figure 11 Test Run _____ 19
Figure 12 Digit Recognition _____ 19
Figure 13 Wireshark showing packets _____ 21

List of Tables

Table 1 shows the differences between different platforms _____ 11
Table 2 gps vs accelerometor _____ 17

1 Introduction

Many drivers exceed the speed limit of the streets intentionally or unintentionally. Each street was given a specific speed limit to assure drivers and pedestrian safety. Exceeding this limit can cause car accidents and severe injuries.

The United Nations and road safety reported that as a result of car accidents 16 million people get injured and 700 thousand die annually. In Saudi Arabia there were about 485,000 accidents in 1436 AH. 6142 died in these accidents which mean that on average 17 person die every day. According to the general Department of traffic "45.88% of the accidents were caused by exceeding the speed limit and it is the first cause of car accidents in Saudi Arabia."(Report 1430 AH,p 31)

This project will contribute to solve this problem by reducing the number of car accidents due to exceeding the speed limits. Many people exceed the speed limit without being aware about the actual speed limit or without realizing that they have already exceeded the limit. So this project will help the driver to know the speed limit of a street once he enters the street and will be notified if he exceeded the maximum speed permitted.

The project will help to reduce the number of car accidents due to high speeds. Also, will saves people money by knowing when they exceed the speed limit to avoid a fee by Saher which at least will cost 300 Riyals per fee.

2 Problem Statement

Many people do not focus on the street while driving; hence, they miss the max speed sign. That can lead to violating the speed limit or even worse by getting accidents.

3 Background

There are some existing solutions for this problem. However, only two of them would be the ones that our solutions needs to compete with. They are as follows:

- **Sygic (mobile app):**

This app is similar to google Maps but it is not free and it has more features including road speed limit.

- **Jaguar (car):**

Jaguar cars (some models), have a smart system that provides the driver with speed limits in the streets. The system uses GPS and Jaguar libraries to compare the current speed with the speed limit and notify the driver when exceeding the limit. This solution is built in the car and it cannot be used in other cars as Jaguar holds patents on that technology.

4 Requirements and Specifications

4.1 Functional Requirement

- An Interface to show the max speed of the street
- A notification for exceeding the speed limit
- Mechanism to get the street max speed limit

4.2 Non-Functional Requirement

- System Response time should be sufficient for the driver to react
- Should consume small amount of the car battery
- The cost should be reasonable compared to the car price

4.3 Technical Specification

- The device should be able to get the max speed of the street in less than 10 seconds
- The device should consume less than 8 watt of power

5 System Design

5.1 Solution Concept

General approach of solving the stated problem.

The solution is to have a system that will notify the driver about the maximum allowed speed in any street. The system gets the max speed of current street and car speed. Then having all the necessary data, the driver gets notified through some user interface.

Description of used/developed algorithms.

The solution algorithm is represented in the following diagram:

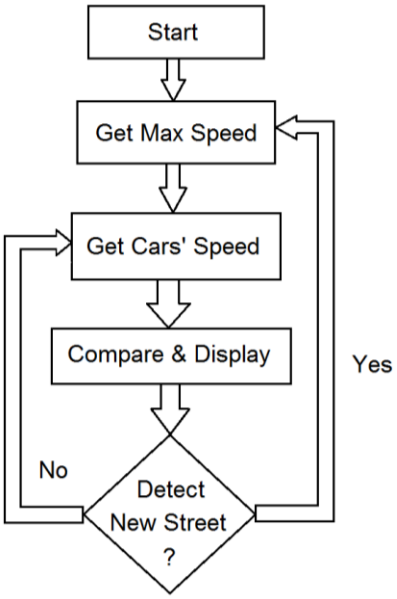


Figure 1 Action Sequence

Alternative approaches and algorithms, comparison, and selection criteria.

There was another approach to solve this problem by forcing the car not to exceed the speed limit. However that will require a different component design for each car. So we will stick to the idea of alerting the driver on a continuous manner.

Sub-function identification.

1) **Speed Limit Detection:** Retrieves max speed from a database that has all streets speed limits. If the data is not available in the database, then the camera is used.
2) **Car Speed Detection:** A real-time updates of the car speed should be fed to the system.
3) **User Interface:** The current and max speed are to be displayed simultaneously on the screen. Also, visible/hearable alerts should be delivered to the driver when exceeding max speed.

5.2 Architecture

System architecture and components:

Different components of our system are integrated together as follows:

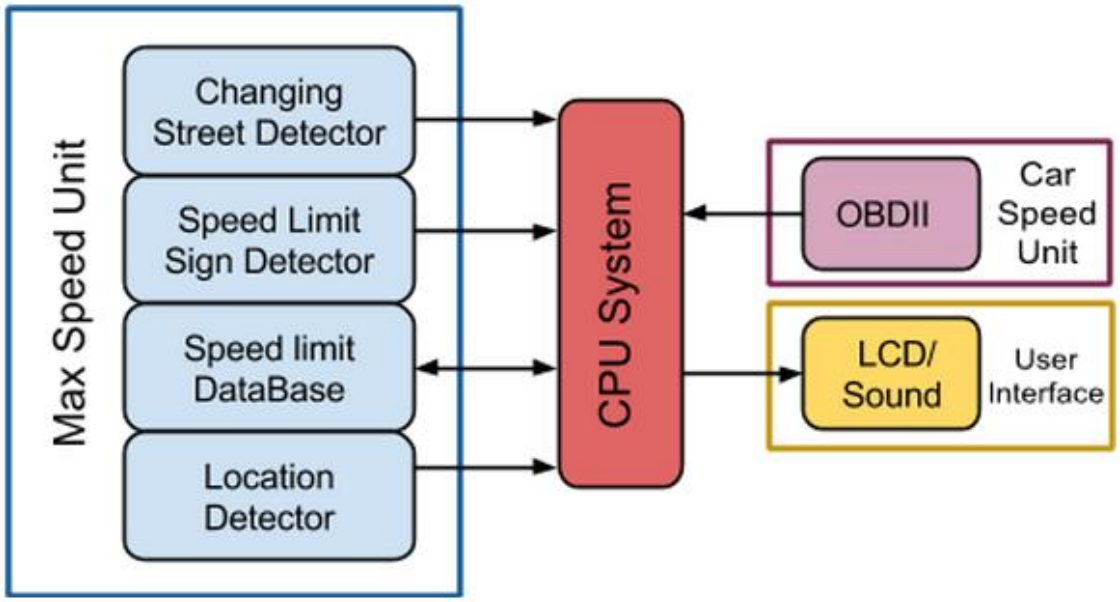


Figure 2 System Architecture

Alternative architectures, comparison, and selection criteria.

RFID-Only Architecture:

A possible design for the system was using RFID. That is, the system will have readers in the car and attached tags in the streets (on the signs or roads intersections). This architecture was an option for us, but we eliminated it for the fact that it would involve changes in the current streets infrastructure.

Database-Only Architecture:

We thought on using only a database to provide speed limit of the current street. But this approach will limit the system when it comes to new streets that are not already updated in the database. That will lead to the need of an always manual updating.

Camera-Only Architecture:

Another design would be using the camera only to read the speed limits from the signs on streets. That way, the camera is an essential component and it has to be up all the time taking pictures.

Figure 3 Hardware Components

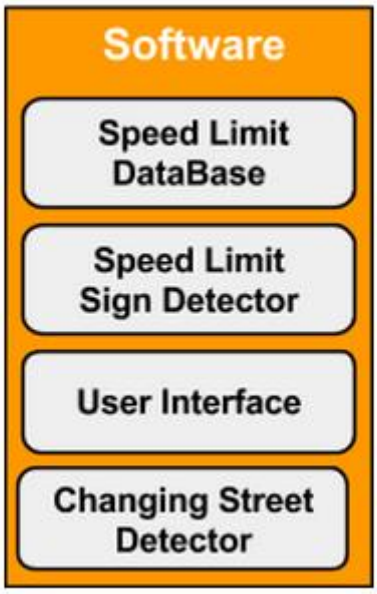
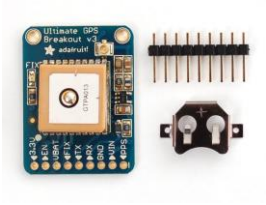


Figure 4 Software Components

GPS:



The component responsible for providing the system with the current location.

Database:

Weather it was locally stored or in the cloud, this component will store the information about all the streets. Then our system will request a speed limit for any street.

OBDII Adapter:

This component is an interface between our system and the car to get real-time information from the car. The current car speed is what matter for us, and we may get some other information later on.

Camera:

The part of the system that will provide us with live pictures from the street. It is supposed to take as much pictures as the system would need.

Display:

This component is the user interface. The driver will get notified when needed via screen and sounds.

Changing street Detector:

This component will be used to detect when the driver change the street to update the speed limit.

5.3 Component Design

Project Algorithm:

The State diagram below shows the flow of the algorithm. when the program starts, it will get the current location and direction of the car. Then the program will send the location to the online database (OpenStreetMap) to acquire the max speed in the street. If the value of the max speed was not found in the Database the program will run the camera and process the picture. After processing, it will get the max speed and update it in the online database. Then, the program will check the car direction to verify if the user is still in the same street or moved to another street. If the user was still in the same street, the program will retrieve the cars' speed and check whether the user exceeded the speed limit or not and alert him if necessary. Finally, the program will check the direction again, if the car has the same direction as before that will indicate that the user is still in same street, otherwise there is a chance that the user moved from a street to another. So the program will get the new location and send it to the database to get the new speed limit. and repeat. (see figure 2)

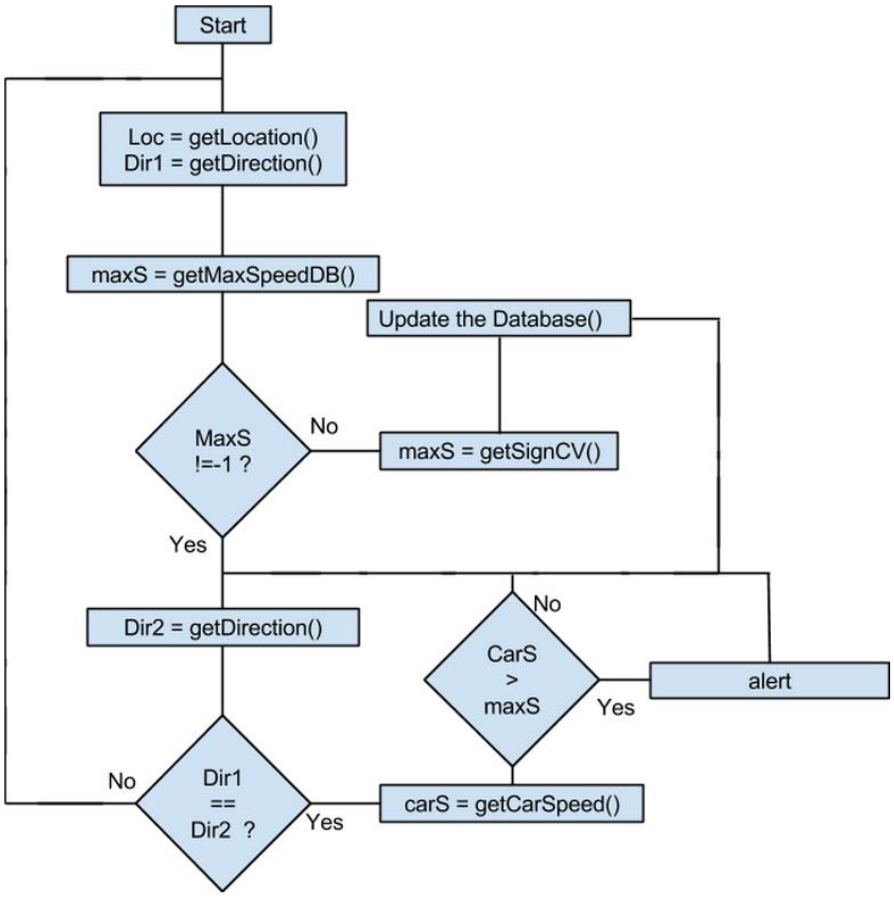


Figure 5 System Algorithm

System Components:

Most of our project components were off-the-shelf, since they do provide good performance and they have a reasonable cost. The components are as follows:

Off-the-shelf Components:

CPU System

We need to do some computer vision tasks. that is why using Microcontroller will not be possible and we have to Use SoC instead. There are 3 possible SoC to be used in the project:
Raspberry pi 2 B (RPI): RPI is a single-board computer developed by Raspberry Pi Foundation in UK.
Beagle bone Black (BBB): BBB is an open source hardware single-board computer developed by Texas instruments to teach and show the capabilities of open source hardware and software as the board can be programmed using many supported open source software.
Intel Edison: Edison is a tiny computer offered by Intel as a development system for wearable devices. The system was initially announced to be the same size and shape as an SD card.
We compared different types of microcontrollers based on the following criteria: Cost, performance CPU and RAM capacity. We eliminated Edison due to the high cost. Then we were left with Raspberry Pi 2 and Beagle Bone Black (BBB). Comparing their CPUs and RAMs resulted that Raspberry Pi 2B is the best option. It is only $35, with 1 GB RAM and most importantly four cores which we need for this project. BBB would have compete with raspberry pi 1 B+ as BBB is better in CPU performance. However RPI 1 B+ GPU is Better than BBB. but the release of RPI 2 with better GPU and CPU than BBB makes it the best choice for the system. Full comparison is shown in Table1.

	Pi 2 B	BBB	Edison
CPU	Cortex A7	Cortex A8	Atom + Quark
Cores	4	1	2 + 1
Clock	900MHz	1000MHz	500MHz
GPU	Videocore IV	PowerVR SGX530	None
Memory	1GB	512MB	1GB
USB Ports	4	2	1*
Flash	None	2GB	4GB
Storage	microSD	microSD	microSD*
Network	10/100	10/100	None
GPIO	40-pin	2x46-pin	70-pin Hirose
Wifi	No	No	Yes
Bluetooth	No	No	Yes
RRP	$35	$49	$85*

Table 1SHOWS THE DIFFERENCES BETWEEN DIFFERENT PLATFORMS

we have initially 3 options :

1. RPI + RPI Camera + Wifi. cost (40+30+12) 82$
2. BBB + Camera Cab + wifi. cost (55+50+12)117$
3. (Edison+board)+ Camera. Cost (110+30) 140 $

Other components are common between the all alternatives.

In total RPI system has better performance and cost as the RPI camera has lower cost than other and has higher fps than BBB camera. the Edison has embedded WIFI but the difference in the cost is much higher than the cost of the WIFI.

GPS: Ultimate GPS Breakout

- Compatible with All Options
- price (40 $)
- capable of receiving up to 10 location updates per second.

Camera: Raspberry Pi NoIR Camera Board - Infrared-sensitive Camera

- Compatible with Raspberry board
- price (30 $)
- Infrared-sensitive Camera (for possible night vision feature)
- capable of shooting 90 frame per second.

Wi-Fi receiver: USB Wi-Fi (802.11b/g/n)

- Compatible with RPI and BBB.

Database:

- We had two options for the database. Either using a remote database (open street map), or having our own local database. We chose Open Street Map for the following reasons:
    1. It is an open source project, that means this project will not be used only to get max speed, but we can also update that database when a street is not registred yet. This project will help in building that open source project.
    2. When a new street is updated, the new information will be available for all users. On the other hand, for local database, each user will have their own version and they have to build their database on their own.
    3. Although it requires internet connection, the overhead is still less than having separate databases for each user (i.e if local database is used, every user will have to recognize all max speed signs, while using remote database the work will be distributed).

- The Online Database (Open Street M) uses PostgreSQL which is object-relational database as data storage format. It also Uses XML to transfer data between the database and the user.

OBD: On-Board-diagnostic

- Display current sensor data, including: Engine RPM, Calculated Load Value, Coolant Temperature, Fuel System Status, Vehicle Speed, Short Term Fuel Trim.

Custom Components:

Speed Limit Sign Detection:

In our project, we use computer vision to get the max speed limit directly from the sign. In order to do that we need two fundamental steps: detect sign in picture and recognize the number in it(as shown in picture).

Sign identification:

Methods:

We can do this in different ways; here are the ones we investigate:

Sign Templates only:

In this method, we get different templates for the sign and get them train by the algorithm to be able to test later. However, this method is complex to implement and time consuming to test all templates. In addition, it is not efficient since the picture will have different brightness and noise levels.

Filtering then using templates:

In this method, we apply some filters on the picture and then use the templates to get the sign. This method gives better performance than the previous one. However, this method is still time consuming to check all the templates.

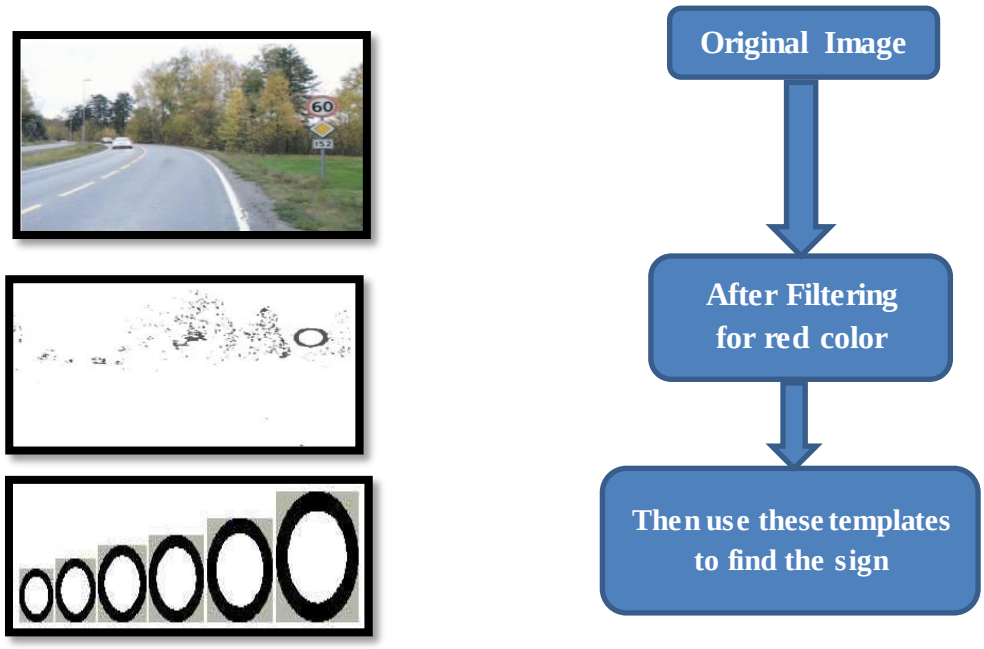
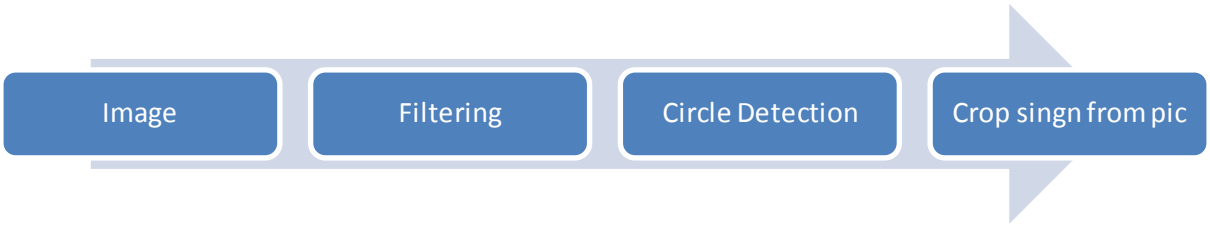


Figure 6 Filtering then using templete method

Filtering then using Circle Detection:

In this method, we filter the picture first to ease circle detection. After filtering, we apply algorithm to identify the circles in the picture. Then crop the circle from the original image. So, in contrast with the previous method we only need to find the circles once which is faster than checking many templates.

Therefore, we choose the third method for our project (as shown in the picture).



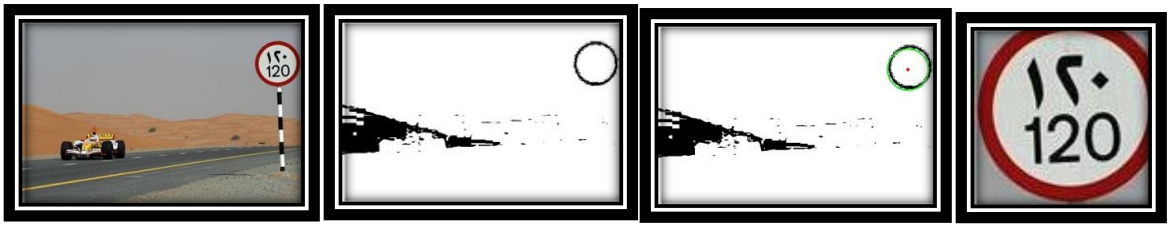


Figure 7 Filtering then using circle detection method

Implementation:

In order for us to implement the method, as mentioned above, we need two things: find suitable filtering and circle detection techniques.

Filtering:

What we did in this part is extracting the red color from the picture. At the beginning, the picture is consisted from three channels red, green and blue (RGB) which means that each pixel has three values (one for each channel). In addition, Red color falls in certain range in each of these channels .Therefore, we got these values and apply threshold for the whole picture pixel by pixel. Specifically, R >100, R>G+40 and R>B+40 means the pixel is red. Then, we change the values of red pixels to black and non-red to white, so we end up with binary picture. After that, we convert the picture to grayscale to minimize the size and facilitate next processes.

Circle Detection:

We used Hough Transform method to detect the circle. The steps of this method are as follows. First, we find the edges of the picture. Second, there are two scenarios. One, we know the radius of the circle, so in this case we only need to find the points that correspond to the position for circle center (as shown in picture 1). Two, we don't know the radios; then the locus of points in parameter space will fall on the surface of a cone. Each point (x, y) on the perimeter of a circle will produce a cone surface in parameter space. The triplet (a, b, R) will correspond to the accumulation cell where the largest number of cone surfaces intersect (as shown in picture 2).

Digit Recognition:

Methods:

Digit recognition also has different methods. here are the ones we investigate all with the assumption that we already identified the sign and now we check inside it :

White to black percentage:

In this method, we compute the percentage of the white background with the black number. After that, we would have an array with each speed and its corresponding percentage to compare with. However, this method has poor accuracy because we might have noise and couple of numbers would have similar percentage such as 110, 120.

Feature extraction:

In this method, we supply the algorithm with sample pictures of the numbers. Then, the algorithm will extract features for each number and save them in an array where each element of the array has the features of specific number. Then we use this array to recognize the numbers we get from picture. This method is more accurate than the previous one because each number has different feature. Although, this method requires more computation than the previous one since it takes one digit at a time, it is still fast enough for our project.
Therefore, we choose second method for accuracy and efficient speed.

Implementation:

Training Phase:

In this phase we supply the code with a picture of black numbers with white background. Then, the code will scan the image to find a shape and put rectangle around it. After that, it will wait for the user to press the number which corresponds to the detected shape and this process will continue for all numbers in picture. Then the features of these numbers along with their values are saved in an array.

Testing phase:

After we finish the training phase, we will have an array containing all the numbers with their corresponding features (where each element is an array). Then, for testing, we use K-nearest algorithm. The algorithm works by making a plane for each feature and project the values for all numbers in it. For example, if we have two features only we will have two planes and when a new number comes it will check for maximum feature match to get the number. Therefore, in the picture below the green is the unidentified number while red and blue are known numbers values. So, since green is close to three blue points and only one red point we classify it as blue.

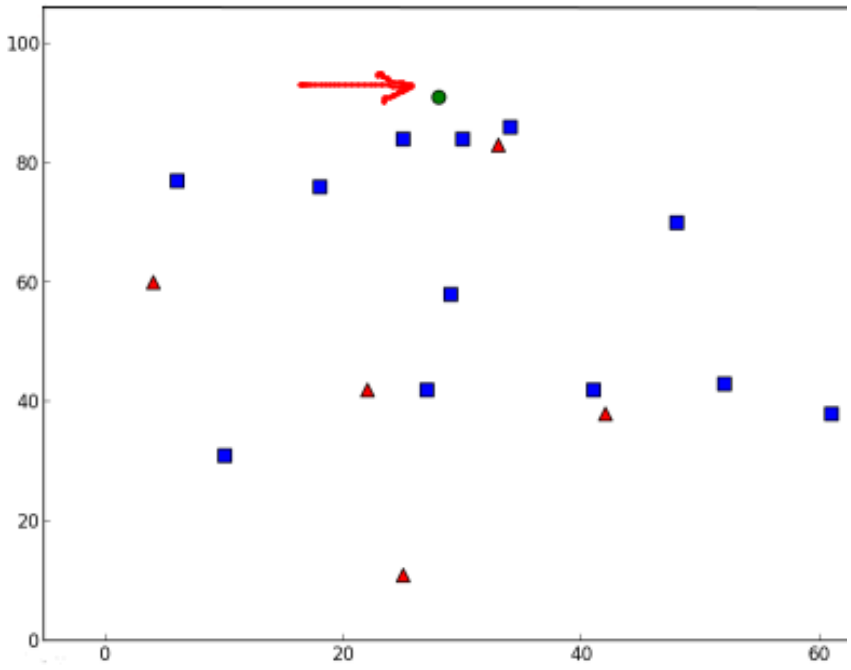


Figure 8 classification example

Changing street detection:

We had different options to detect whether the car has moved to another street or not. That has to be detected to update the speed limit. In fact, it was possible to use a digital compass or an accelerometer. However, this solution will cause additional cost and these chips are usually affected by the surroundings like metal and magnet. On the other hand, using GPS locations with some mathematical equations can obtain the direction from two points and that will be more accurate. Moreover, we already have a GPS to locate the car and no additional cost is required.

Table 2 GPS Vs Accelerometor

Component	Cost	Power	Accuracy
GPS	$0	5V	High
Accelerometer	$20	3.3V	Medium

Acquiring two consecutive point from the GPS can indicate the direction of the car through the the formula of finding the direction of a vector or two points, $\theta = \tan^{-1} \frac{y2-y1}{x2-x1}$. This formula will specify the direction of the car as an angle. This angle can be used to conjecture whether the car is still in the same street or moved to another street. By sensing the difference in the angle within a period of time. if the angle is still within a threshold, the system will assume that the car is still in the same street. Otherwise the system will assume that the car moved to another street and the Speed limit must be updated. This method might has some error margin as it deals with different situations. for example if the street turn, the system will detect changing in street and will issue request for the new speed limit. However if the speed limit changes within the same street the system will not detect that.this problem can be solved by sending signal to the database each fixed distance or time to make sure that the car still in the same street and uses the changing stret detector to send instance request. this solution has communication overhead and need some optimization.

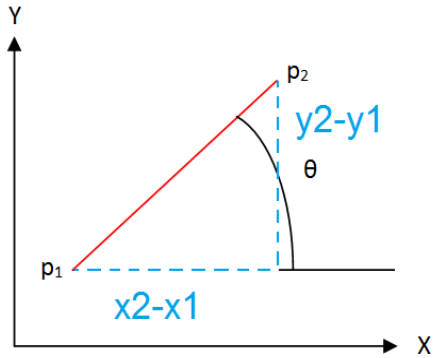


Figure 9 illustrates the formula of obtaining the angle of a vector of two points.

5.4 System Integration

We are using standard Interfaces for two reasons:

- ❖ It is efficient for the project
- ❖ The system to be Easily upgraded in the future.

Our system will be integrated through the raspberry pi board and our python main code through the following standards:

- MIPI- CSI-2: to interface the camera with the board.
- UART : To interface the GPS with the board.
- EvZ USB: to interface the OBD2 with the board.
- XML: To interface the program with the Database.
- OBD: To interface the System with the Car's Systems.

6 Testing, Analysis, and Evaluation

Some working component were tested. The main factor in the project is that the system must respond with that max speed in less than 10 second from the moment a street changing has been detected.

- **DataBase**: Initial results of testing the database showed that we can get a reply in less than 2 seconds (i.e either max speed or -1 if not in the database). That is a good time as we have about 8 second to detect and recognize speed limit signs in case of a negative reply.

- **GPS:** The GPS module is connected to the board through UART. At first we tested the output in the command line. Then we wrote a python code that extracts the main two outputs, Latitude and Longitude from the GPS. The code takes about 0.6 second to obtain the first output then less than 0.2 seconds for every update.

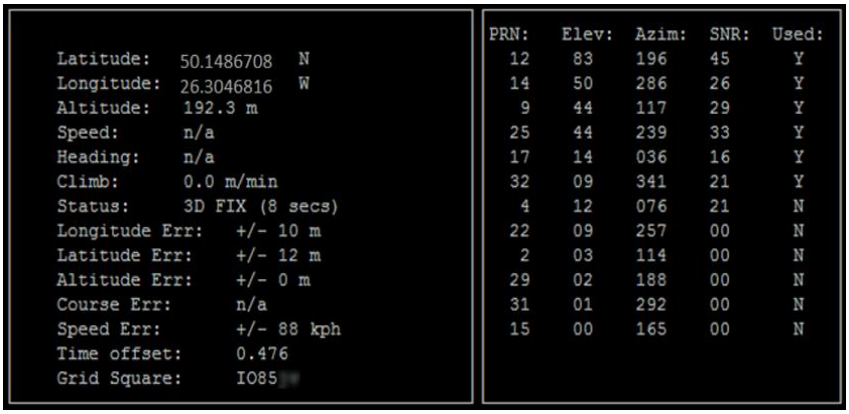


Figure 10 GPS outputs

- **OBD2:** The OBD2 was tested in the car using a python code that obtains the current speed continuously.

- **Camera:** we test the camera library and we were able to get still-images with different resolutions and formats. Also, we could modify the picture before saving it by increasing the brightness, flipping the picture or changing it to grey level.

- **Speed Limit Sign Detecor:**
  In this part we have two things to test: digit recognition and sign identification.

  Sign identification:
  We test different pictures of speed limit signs and found that it was accurate for 7 out of 8 pictures and below a sample run:

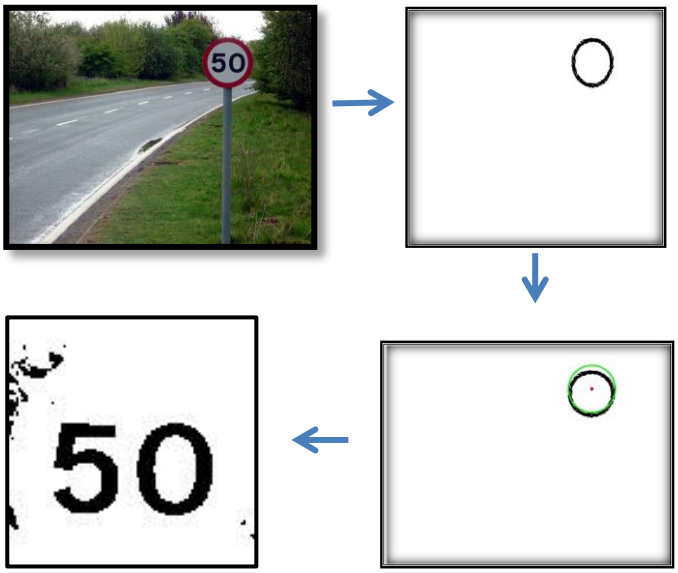


Figure 11 Test Run

Digit Recognition:

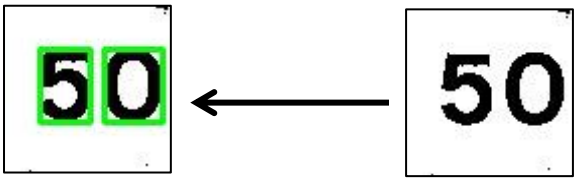


Figure 12 Digit Recognition

7 Issues

GPS installation & testing:

1) When we first purchased the GPS module, we had two options. Either to buy an additional USB cable that easily connects it to the Raspberry, or connect the GPS directly via UART pins. We chose to use the UART to lower the overall cost. In fact, we found out that we need to modify some configurations on the operating system to enable those pins. Unfortunately, that caused the whole operating system to collapse and we had to reinstall it again and we lost all data we had so far. We tried again and it worked. The GPS finally was communicating through UART. That taught us to always backup from time to time.

2) The GPS hardware needed a software to connect it the operating system. We installed a library called "GPSD", but no signal from the satellites was received. However, we found out that the problem was from the walls that was blocking the signal to be received inside the building. Then we had to test it outdoors and for that we needed a screen, keyboard and a mouse. Alternatively, we used a laptop and an Ethernet cable. Then connected to the bard via SSH and GPS signals were captured correctly. Later on, we used a long antenna that solved the problem.

Python Tkinter 2.7 vs 3.4:

The library isn't well documented regarding the difference between python 2 and 3 which caused some problems at the beginning. We had to surf many websites until we were able to add one feature to the GUI that solved it.

OBD2 communication:

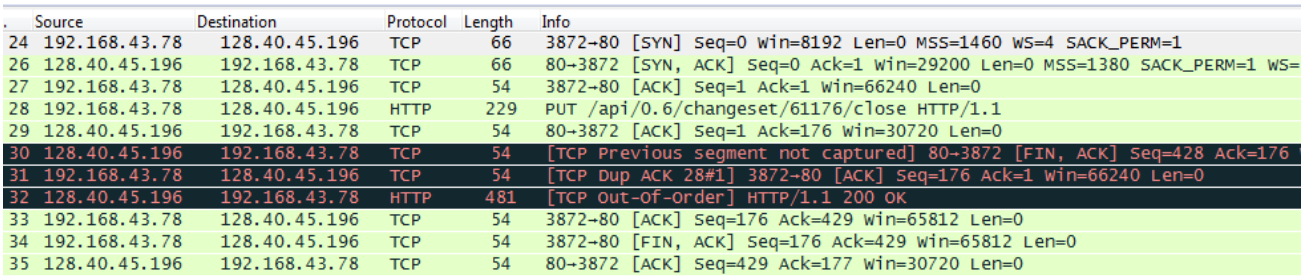
The OBD2 re-establishes the serial connection between the car and the board every time the function getCarSpeed() is called. That causes some delay and we solved it by modifying the library.

OpenCV installation

After installing the openCV and making sure it's running through the command shell, we had a problem importing it from python IDE. We tried reinstalling openCV again but it didn't work. Then, we found out that the problem was that the installation directory and the IDE directory are different. That was solved by changing the directory and the library then worked.

Database

The OpenStreetMap is sort of complicated when it comes to modifying it. Their APIs require OAuth protocol for authentication and that was not a simple work. The python library *httplib* requires special requirements as well as the OpenStreetMap APIs. It took long time and some work to write a code that meets both requirements. Also an error was generated whenever we tried to initiate an HTTP session with the server. After debugging we found out that the header of the HTTP that contains the username and password was blocked by the KFUPM proxy. Switching to another network solved it. A snapshot from Wireshark is shown in figure().



	Source	Destination	Protocol	Length	Info
24	192.168.43.78	128.40.45.196	TCP	66	3872→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
26	128.40.45.196	192.168.43.78	TCP	66	80→3872 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK_PERM=1 WS=
27	192.168.43.78	128.40.45.196	TCP	54	3872→80 [ACK] Seq=1 Ack=1 Win=66240 Len=0
28	192.168.43.78	128.40.45.196	HTTP	229	PUT /api/0.6/changeset/61176/close HTTP/1.1
29	128.40.45.196	192.168.43.78	TCP	54	80→3872 [ACK] Seq=1 Ack=176 Win=30720 Len=0
30	128.40.45.196	192.168.43.78	TCP	54	[TCP Previous segment not captured] 80→3872 [FIN, ACK] Seq=428 Ack=176
31	192.168.43.78	128.40.45.196	TCP	54	[TCP Dup ACK 28#1] 3872→80 [ACK] Seq=176 Ack=1 Win=66240 Len=0
32	128.40.45.196	192.168.43.78	HTTP	481	[TCP Out-Of-Order] HTTP/1.1 200 OK
33	192.168.43.78	128.40.45.196	TCP	54	3872→80 [ACK] Seq=176 Ack=429 Win=65812 Len=0
34	192.168.43.78	128.40.45.196	TCP	54	3872→80 [FIN, ACK] Seq=176 Ack=429 Win=65812 Len=0
35	128.40.45.196	192.168.43.78	TCP	54	80→3872 [ACK] Seq=429 Ack=177 Win=30720 Len=0

Figure 13 Wireshark showing packets

Sign Detection:

- We used a training code to allow the program to recognize different integers However it was recognizing all integers as 0. We found that it represent the 0 as 1114032 no 30 as the ascii representation.
- The brightness and the environment affect the extraction of the sign from the background and cause some noises in the extracted sign. we used HSV color domain and extracted the red from H channel which represent the true color to produce better result.

- The distant and the size of the sign effect the result as the character recognition cannot handle big changes in the character dimensions. This was solved by scaling the character to certain size to ease the recognition of the character.

8 Engineering Tools and Standards

8.1 Tools:

Python IDE: The default python IDE was used mainly on the raspberry pi to run the code as no advanced IDE is required.

Geany: Open source editor that allows to compile different languages. It has the capability of python IDE to run a program with one click, and an Editor that open many files on the same window and shows line number.

Mercurial: A distributed version control software that is used to improve the cooperation of the team members and to restore the software component of the project to a safe state when some modification causes failure.

bitBucket.org: A website to store the repository online to allow access from different machines.

OpenCv(in Python): open source library containing different methods to facilitate image processing. we used it to make sign detection job simpler and not to worry much about the details of implementing the methods that deals with pictures.

Camerapi (library): Library which contains different methods to use the raspberry cam such as: capture, resolution, start and stop video. It was used because its convenient and compatible with the hole system in python.

GPSD (library): GPSD is an open service daemon that monitors one or more GPSs through serial or USB ports.

Wireshark: A program on another machine that we used to test the communication with OpenStreetMap servers. Inspecting all packets allowed us to investigate the HTTP connection.

8.2 Standards:

HTTP: Communication protocol that is used to connect to OpenStreetMap servers to retrieve data from the database and update it. OpenStreetMap's APIs require the changes to be in a form of a PUT or POST method to update the tags inside the XML.

Base64: Encoding schema that was required by the Database API. It was used to encode the username and password when accessing the database.

MIPI- CSI-2 : mobile industry processor interface- camera serial interface facilitates the connection of a small camera to the main processor.

OBD2: is an automotive term referring to a vehicle's self-diagnostic and reporting capability. OBD systems give the vehicle owner or repair technician access to the status of the various vehicle subsystems. We had two choices:

1. OBD2 - Bluetooth.
2. OBD2 - USB.

We used the USB one because it was cheaper, both have the same performance.

UART: A piece of hardware that translates data between parallel and serial forms. We used this as an alternative to the high cost of a USB-GPS.

XML: The major tools in the OSM universe use an XML format following a XML schema definition that was first used by the API only. Basically it is a list of instances of OSM data primitives (nodes, ways, and relations).

9 Teamwork

- **Jalal AlRukhaimi:**
  - Responsibilities:
    - Research & evaluate existing solutions.
    - Getting familiar with the components
    - Test all subsystems individually.
    - Integration of the project.
  - Contributions:
    - Evaluate the feasibility of using a database to retrieve the speed limit is streets.
    - Evaluate and compare different Single board computer to select the best match to the requirements.
    - Contribute with finalizing the project algorithm.
    - Interface, test and integrate the Database to get the speed Limit.
    - Contribution in interfacing and testing OBD2.
    - Interface, test and integrate the GUI.
    - Contribution in writing the code for computer vision algorithm
  - Expertise:
    - Programming in python
    - familiar with Raspberry pi

- **Hassan Alshabaan:**
  - Responsibilities:
    - Map the Requirements to Specifications.
    - Find & Evaluate possible solutions.
    - Work on each subsystem individually.
  - Contributions:
    - Interface, test and integrate openCV on the raspberry pi.
    - Connecting openCV with Camerapi library.
    - Planning the method for detection and Identification of road sign.
    - Evaluating the feasibility of using computer vision to get the max speed limit.
    - Contribution in writing the code for computer vision algorithm
  - Expertise:
    - Dealing with opencv and camera Pi library in python.
    - Image processing.

- **Abdul-Aziz Al-Amri:**
  - Responsibilities:
    - Evaluate and specify the components.
    - Define subsystems and their interfaces.
    - Test & Debugging.
  - Contributions:
    - Choosing and testing a suitable GPS library to establish a layer that connects the GPS to the operating system.
    - Modifying the Raspberry Pi configurations to connect GPS module through UART.
    - Contribution in interfacing and testing OBD2.
    - Contribution in interfacing and testing GUI.
    - Contributed in fixing and debugging database code.
  - Expertise:
    - Networking (i.e TCP and SSH protocols)
    - Programming in python.
    - Working with Linux.

10 Conclusion

In conclusion, this experience taught us a lot of things. Main lesson was the different stages of working on projects. From planning to implementation, testing and debugging. Also, it helped increasing our skills in searching for solutions to a problem and evaluating it against existing ones. In addition, we learned that planning for the project and working on teams is not a trivial task and it has many challenges. Looking back at our work for the project, there is a number of things we would prefer to do better. First, keeping track of the plan and make sure to follow it and trying our best to finish the project as early as possible. Second, document regularly and add up to the report directly. Third, never underestimate any task.