CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for any degree or diploma.

Sonkolon Barua (1110151)

Irfan Mohammad Al Hasib (1110165)

`

CERTIFICATE OF APPROVAL

The thesis titled "**Developing a Vehicular Speed Measuring Device in order to Develop a Driving Cycle",** submitted by **Irfan Mohammad Al Hasib** (Student no. 1110165), and **Sonkolon Barua** (Student no. 1110151) has been accepted as satisfactory in partial fulfillment of the requirement for the degree of **BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING**.

Dr. Md. Ehsan

Professor
Department of Mechanical Engineering,
Bangladesh University of Engineering & Technology.

`

Acknowledgement

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our supervisor Prof. Dr. Md. Ehsan for his supervision throughout our dissertation work. He has been a source of inspiration for us and his knowledge and expertise in this field helped us to learn a lot. We are very lucky that we have got the opportunity to work under his supervision. And we earnestly hope that he will always be with us in our future research works.

We would also like to thank all the staffs of Auto Shop and Heat Engine Lab for their help in the project and accept all our disturbances in smiling faces.

We are also indebted to our parents, brother, sisters, friends and well-wishers who are taking lot of pains for progress in our life and for their sacrifices, blessing and constant prayers for our advancement.

Finally it was a great opportunity for us to get involved with this project.

Irfan Mohammad Al Hasib

(1110165)

Sonkolon Barua

(1110151)

`

Abstract

A driving cycle corresponding to the driving conditions of a particular country is of decisive importance for fuel economy evaluation of vehicles and automobile engines. It's in narrow definition relation graphical representation of Speed, Acceleration versus time. Driving Cycle differs from country to country, state to state and most importantly it's different for different time of data collection or considered time in a day.

For building driving cycle recording thousands of data is the actual task. Possible best data storing method is to be introduced for having the best Standard Driving Cycle in a city. Existing discontinuous speed recording methods being unable to provide instantaneous speed, continuous speed recording methods are best suited here. Continuous voltage output for respective speeds is a special way to be employed for recording speeds with respect to time to build driving cycle; this process is free from limitation of Discontinuous speed recording method as in any instant operator gets voltages which represent speeds linearly.

Analyzing Discontinuous speed measuring methods: GPS Device Data Recording, Android Mobile GPS, IR Obstacle Detector Sensor, Reed Switch System and Continuous methods: Tracking Speeds from Vehicle Speedometer, Add-on method or Adding a system with a DC Motor to be worked as generator – decision can be taken that, above all the last process of Data Recording (Adding a system with a DC Motor to be worked as generator) to build driving cycle is the best suitable method.

`

Table of Contents

`

`

1. Introduction

Dhaka, the capital city of Bangladesh, like the other small cities of the world, has adopted lots of changes to keep pace with the development processes of the modern world. These changes not only make life comfortable, but also often cause various new problems. For the last few decades, Dhaka has been providing the dwelling for more than 15 million people. And for proper transportation and communication of this huge number of people it contains thousands of vehicles, which continuously emit tones of exhaust gases every day. The number of the vehicles in the city has drastically been increased in the last half decade. As a result various problems have been appearing day by day. So far not much research work has been done on the transportation system and driving cycle of Dhaka city. But for avoiding the unwanted problems, to enjoy better performances of the vehicles with longer life time of the vehicles and better environmental conditions, studying the driving cycle of Dhaka city is very crucial. To construct a driving cycle first focused on different speed tracking methods, so that the best possible system can be applied to prepare a typical driving cycle for Dhaka city.

1.1 Literature review

1.1.1 Existing Vehicle Speed Measuring Methods:

So far for centuries there are numerous systems developed to measure speed of automotive vehicles. The best are:

I. Torsion Spring Method:

Originally patented by Otto Schulze on October 7, 1902, it uses a rotating flexible cable usually driven by gearing linked to the output of the vehicle's transmission. The early Volkswagen Beetle and many motorcycles, however, use a cable driven from a front wheel.

When the car or motorcycle is in motion, a speedometer gear assembly turns a speedometer cable, which then turns the speedometer mechanism itself. A small permanent magnet affixed to the speedometer cable interacts with a small aluminum cup (called a speed cup) attached to the shaft of the pointer on the analogue speedometer instrument. As the magnet rotates near the cup, the changing magnetic

`

field produces eddy currents in the cup, which themselves produce another magnetic field. The effect is that the magnet exerts a torque on the cup, "dragging" it, and thus the speedometer pointer, in the direction of its rotation with no mechanical connection between them.

The pointer shaft is held toward zero by a fine torsion spring. The torque on the cup increases with the speed of rotation of the magnet. Thus an increase in the speed of the car will twist the cup and speedometer pointer against the spring. The cup and pointer will turn until the torque of the eddy currents on the cup is balanced by the opposing torque of the spring, and then stop. Given the torque on the cup is proportional to the car's speed, and the spring's deflection is proportional to the torque, the angle of the pointer is also proportional to the speed, so that equally spaced markers on the dial can be used for gaps in speed. At a given speed the pointer will remain motionless and pointing to the appropriate number on the speedometer's dial.

The return spring is calibrated such that a given revolution speed of the cable corresponds to a specific speed indication on the speedometer. This calibration must take into account several factors, including ratios of the tail shaft gears that drive the flexible cable, the final drive ratio in the differential, and the diameter of the driven tires.

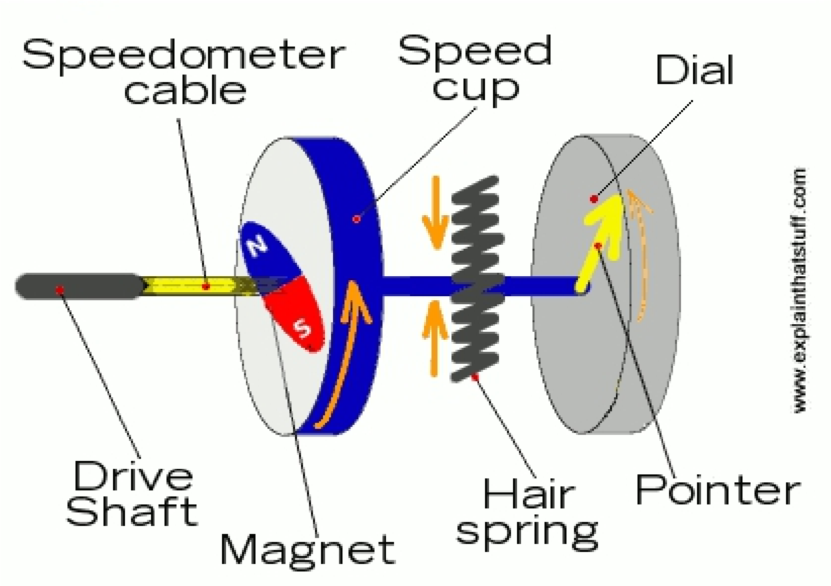
One of the key disadvantages of the eddy current speedometer is that it cannot show the vehicle speed when running in reverse gear since the cup would turn in the opposite direction - in this scenario the needle would be driven against its mechanical stop pin on the zero position.

On newer vehicles, the physical cable is replaced by a rotation sensor. This sends pulses of electricity to the speedometer, the frequency being proportional to the road speed. The electronics inside the speedometer counts the pulse frequency, and converts this into a reading on the scale- a servo or stepper motor will move the needle accordingly. On a digital speedometer, the needle and servo are replaced with a circuit driving the

`

display.

How speedometers work



II. **Transmission Shaft Magnetic method:**

An electronic speedometer receives its data from a vehicle speed sensor (VSS), not a drive cable. The VSS is mounted to the transmission output shaft or to the crankshaft and consists of a toothed metal disk and a stationary detector that covers a magnetic coil. As the teeth move past the coil, they "interrupt" the magnetic field, creating a series of pulses that are sent to a computer. For each 40,000 pulses from the VSS, the trip and total odometers increase by one mile. Speed is also determined from the input pulse frequency. Circuit electronics in the car are designed to display the speed either on a digital screen or on a typical analog system with a needle and dial.

Cars with modern ABS systems measure the speed of each wheel with a tone wheel in each hub. Alternatively they could put a speed sensor at the outlet of the transmission or in a differential. Since the speed sensor is after the transmission the engine speed being high or low isn't going to mess up the accuracy of the speedometer. The math is simple, you can calculate tire circumference and measure the number of times the tire spins to determine the speed in a given time (mph or kph). If you get oversized tires you may have to adjust the tire size in the computer or change a gear ratio in the speedometer cable if it's a mechanical system to ensure your speedometer is accurate.

1.1.2 Driving Cycle:

In the seventies and the eighties the term ‘driving cycle’ was introduced in America. During that time the state California gave an impulse to the automobile industry to develop engines with higher efficiency and lower emissions, by the establishment of emission laws. As a result there arose the need for test procedures to compare several engines with each other. These test procedures are called driving cycles.

A driving cycle is a standardized driving pattern. This pattern is described by means of a velocity-time table. The track that is to be covered is divided in time-steps, mostly seconds. The acceleration during a time step is assumed to be constant. As a result the velocity during a time step is a linear function of time. Because velocity and acceleration are known for each point of time, the required mechanical power as a function of time can be determined with formulas. This function integrated over the duration of the driving cycle produces the mechanical energy needed for that driving cycle.

The world-wide used driving cycles can be divided into three groups:

- European driving cycles
- US driving cycles
- Japanese driving cycles

I. European driving cycles: (NEDC: ECE R15 (1970) / EUDC (1990))

The New European Driving Cycle (NEDC) is a driving cycle, last updated in 1997, designed to assess the emission levels of car engines and fuel economy in passenger cars (which excludes light trucks and commercial vehicles). It is also referred to as MVEG cycle (Motor Vehicle Emissions Group). The NEDC, which is supposed to represent the typical usage of a car in Europe, is repeatedly criticized for delivering economy figures which are unachievable in reality. It consists of four repeated ECE-15 urban driving cycles (UDC) and one Extra-Urban driving cycle (EUDC).

The test procedure is defined in UNECE R101 for the measurement of CO2 and fuel consumption and/or the measurement of electric energy consumption and electric range in hybrid and fully electric M1 and N1 vehicles, and UNECE R83 for the measurement of emission of pollutants of M, N1 and M2 vehicles. It is maintained by the UNECE World Forum for Harmonization of Vehicle Regulations (WP.29), which also works on its successor, the Worldwide Harmonized Light Vehicles Test Procedures (WLTP). Although originally designed for petrol-based road vehicles, the driving cycle is now also used for diesel vehicles and to estimate the electric power consumption and driving range of hybrid and battery electric vehicles.

European driving cycles belong to the modal cycles. This means there are parts in these cycles where the speed is constant. Because modal cycles don't represent real driving patterns an additional group of driving cycles has been developed: the HYZEM cycles. These cycles will be discussed later in this paragraph. For the European driving cycles there are subdivisions. ECE 15 (represents urban driving. It is characterized by low vehicle speed (max. 50 km/h), low engine load and low exhaust gas temperature), EUDC (describes a suburban route. At the end of the cycle the vehicle accelerates to highway-speed), EUDCL (a suburban cycle for low-powered vehicles. It is similar to the EUDC but the maximum speed is 90 km/h), NEDC (a combined cycle consisting of four ECE 15 cycles followed by a EUDC or EUDCL cycle. The NEDC is also called the ECE cycle) and HYZEM are the prime subdivisions. Except HYZEM all the cycles mentioned above are stylistic cycles. These cycles cannot represent real driving patterns. The HYZEM cycles are transient cycles. The parts at which the speed is constant are much smaller than in modal cycles. The HYZEM cycles are derived from real driving patterns throughout Europe. Therefore, they are a better representation for driving conditions than the standard European cycles. The HYZEM cycles are often used but they are not official. The HYZEM cycles consist of an urban cycle, an extra-urban cycle and a highway cycle.

`

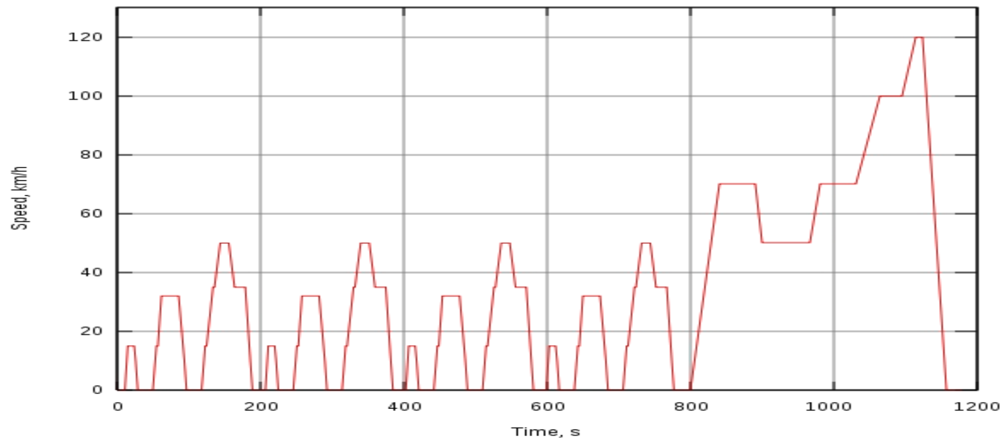


Fig1.2: New European Driving Cycle

❖ Criticism:

- Inability to represent real-life driving

  The NEDC was conceived at a time when European vehicles were lighter and less powerful. The test offers a stylized driving speed pattern with low accelerations, constant speed cruises, and many idling events, however the transient accelerations are much steeper and more dynamic in practice, in part caused by the power surplus of modern engines, as the 0–100 km/h (0–62 mph) average time decreased from 14 seconds in 1981 to 9 seconds in 2007. The UK consumer group Which Criticize the NEDC as being out of date, having been first introduced in the 1970s and not updated since 1997 - before hybrid cars and stop-start technology was generally available. They also criticize it for not replicating real-world driving conditions and having numerous loopholes which mean the test figures are unachievable in practice. Weaknesses noted are: that tests are not necessarily repeatable and comparable, the test cycle does not include sustained motorway driving, test cycles can be performed using optional economy settings which will not typically be selected by drivers, the test cycle is performed with ancillary equipment such an air conditioning and heated windows switched off, the tests can be carried out at 1.2 mph below the required speed thus using less fuel, roof rails and passenger door mirror can be removed for the test, tyre pressures for the test can be set above the recommended values thus reducing rolling resistance, no official body polices the tests and the car companies can arbitrarily reduce their results by 4% at the end of the cycle.

- Cycle beating

`

For the emission standards to deliver real emission reductions it is crucial to use a test cycle that reflects real-world driving style. However the fixed speeds, gear shift points and accelerations of the NEDC offer possibilities for manufacturers to engage in what was called 'cycle beating' to optimize engine emission performance to the corresponding operating points of the test cycle, while emissions from typical driving conditions would be much higher than expected, undermining the standards and public health. In one particular instance, research from two German technology institutes found that for diesel cars no 'real' NOx reductions have been achieved after 13 years of stricter standards.

Successors:

UNECE(United Nations Economic Commission for Europe) WP29(World Forum for Harmonization of Vehicle Regulations) is currently developing a new global harmonized driving cycle, the World Light Test Procedure (WLTP) with participation of experts from the European Union, India, and Japan; it will apply to light duty vehicles (i.e. passenger cars and light commercial vans).

The European Commission intends to introduce a type approval procedure to measure the energy efficiency of mobile air conditioning systems in vehicles, as well as its effect on fuel consumption and emissions, and display an efficiency label on the vehicle; the work is being performed by TU Graz (Institute for Internal combustion engines and Thermodynamics), Aristotle University of Thessaloniki (Laboratory of Applied Thermodynamics), TNO(Netherlands Organization for Applied Scientific Research) and ACEA(European Automobile Manufacturers Association).

II. US driving cycles:
(EPA Federal Test: FTP 72/75 (1978) / SFTP US06/SC03 (2008))

US driving cycles belong to the transient cycles. Like the HYZEM cycles they give a better representation of real driving patterns than the modal cycles. In the early seventies FTP 72 cycle has been developed to describe an urban route. The cycle consists of a cold start phase. This phase is followed by a transient phase with many speed peaks which start from rest. The emissions are measured. In the United States weight factors are used for both phases to norm the emissions. The FTP 72 is often

`

called FUDS, UDDS or LA-4. The SFUDS has been developed to simulate battery discharge and charge during a trip with an EV. The SFUDS was derived for a specific vehicle, the improved dual shaft electric propulsion. The velocity profile is adapted to this vehicle, to obtain a battery discharge- and charge profile that consists of constant power phases. The FTP 75 is the FTP 72 with an extra third phase. This phase is identical to the first phase of the FTP 72 but is executed with a hot engine. HFEDS represents extra urban and highway driving. IM 240 is a cycle is used for inspection purposes. With this cycle, the emissions are determined during the periodic maintenance test. The LA-92 represents like the FTP 72 an urban route. The LA-92 has been developed in 1992, because the existing FTP 72 turned out to be a non-realistic representation of urban driving patterns. For example the LA-92 has a higher average speed. NYCC cycle represents an urban route through New York. A characteristic of this cycle is the low average speed. US 06 is the so called aggressive driving cycle. It is developed to describe a driving pattern with high engine loads.

The EPA Federal Test Procedure, commonly known as FTP-75 for the city driving cycle, are a series of tests defined by the US Environmental Protection Agency (EPA) to measure tailpipe emissions and fuel economy of passenger cars (excluding light trucks and heavy-duty vehicles).

The testing was mandated by the Energy Tax Act of 1978 in order to determine the rate of the guzzler tax that applies for the sales of new cars.

The current procedure has been updated in 2008 and includes four tests: city driving (the FTP-75 proper), highway driving (HWFET), aggressive driving (SFTP US06), and optional air conditioning test (SFTP SC03).

- City driving
    - **UDDS**

      The Urban Dynamometer Driving Schedule is a mandated dynamometer test on tailpipe emissions of a car that represents city driving conditions.

      It is also known as FTP-72 or LA-4, and it is also used in Sweden as the A10 or CVS (Constant Volume Sampler) cycle and in Australia as the ADR 27 (Australian Design Rules) cycle.

      The cycle simulates an urban route of 12.07 km (7.5 mi) with frequent stops. The maximum speed is 91.2 km/h (56.7 mi/h) and the average speed is 31.5 km/h (19.6 mi/h).

The cycle has two phases: a "cold start" phase of 505 seconds over a projected distance of 5.78 km at 41.2 km/h average speed, and a "transient phase" of 864 seconds, for a total duration of 1369 seconds.

- **FTP-75**

The "city" driving program of the EPA Federal Test Procedure is identical to the UDDS plus the first 505 seconds of an additional UDDS cycle.
Then the characteristics of the cycle are:

Distance travelled: 17.77 km (11.04 miles)
Duration: 1874 seconds
Average speed: 34.1 km/h (21.2 mph)

The procedure is updated by adding the "hot start" cycle that repeats the "cold start" cycle of the beginning of the UDDS cycle. The average speed is thus different but the maximum speed remains the same as in the UDDS. The weighting factors are 0.43 for the cold start, 1.0 for the transient phase and 0.57 for the hot start phase.

Though it was originally created as a reference point for fossil fuelled vehicles, the UDDS and thus the FTP-75, are also used to estimate the range in distance travelled by an electric vehicle in a single charge.

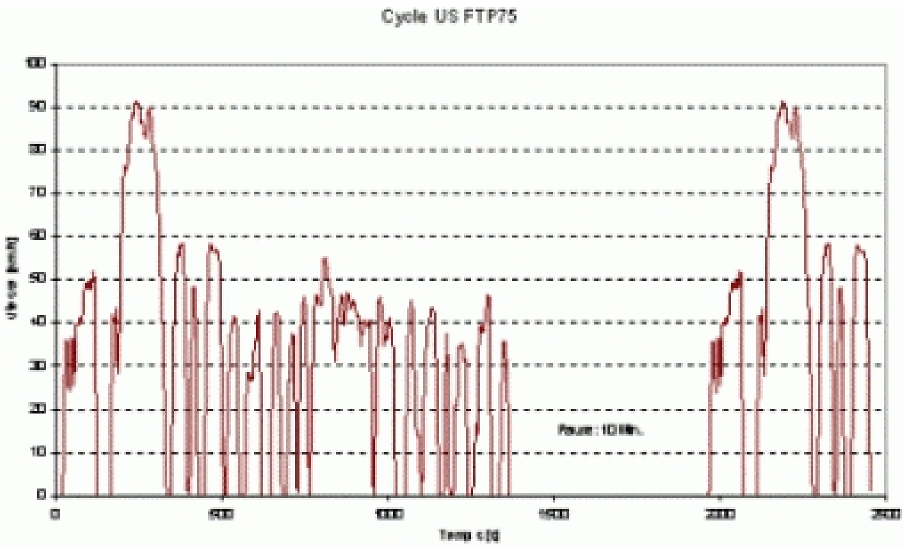


Fig 1.3: FTP-75 Driving Cycle

III. The Japanese driving cycles

`

(10 mode / 10-15 Mode (1983) / JC08 (2008))

The Japanese driving cycles belong to the modal cycles. 10 Mode cycle represents an urban route. 15 Mode is a combination of an urban and an extra-urban route. The maximum speed is 70 km/h. 10-15 Mode is a combination of five cycles. First the 15-Mode, then three times 10-Mode and at last again the 15 Mode.

Japanese 2005 emission regulation introduced a new JC08 chassis dynamometer test cycle for light vehicles (< 3500 kg GVW). The test represents driving in congested city traffic, including idling periods and frequently alternating acceleration and deceleration. Measurement is made twice, with a cold start and with a warm start. The test is used for emission measurement and fuel economy determination, for gasoline and diesel vehicles.

The JC08 test had been fully phased-in by October 2011. In the transitional period emissions were determined using weighted averages from different cycles, as follows:

2005.10: 12% of 11 mode cold start + 88% of 10-15 mode hot start;

2008.10: 25% of JC08 mode cold start + 75% of 10-15 mode hot start;

2011.10: 25% of JC08 cold start + 75% of JC08 hot start.

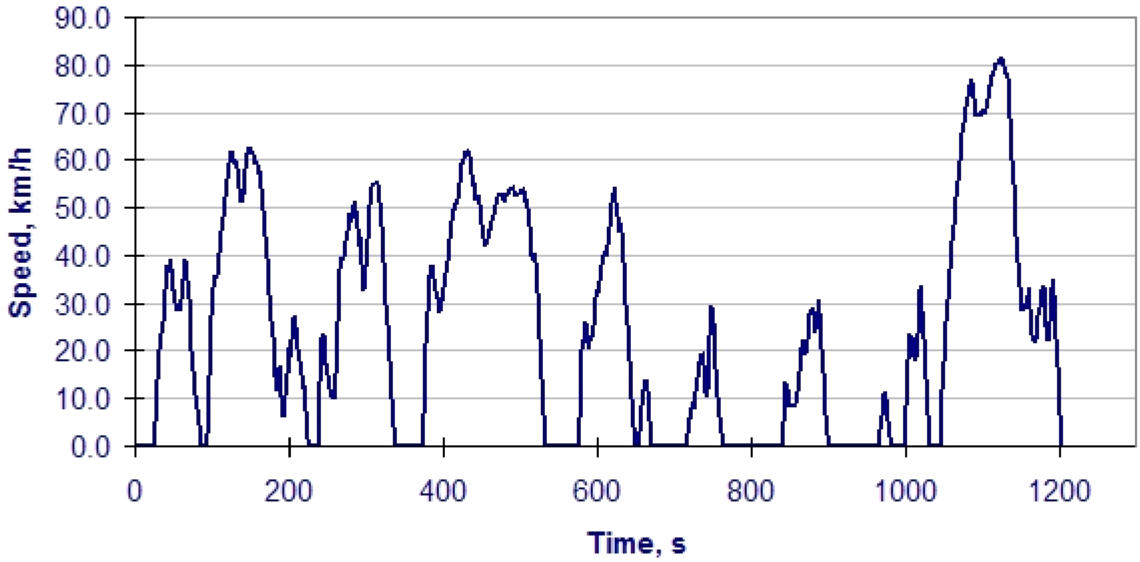


Fig 1.4: The JC08 Test Cycle schematic.

`

The following are selected parameters of the JC08 driving schedule:

Duration: 1204 s

Total distance: 8.171 km

Average speed: 24.4 km/h (34.8 km/h excluding idle)

Maximum speed: 81.6 km/h

Load ratio: 29.7%

- **Global harmonized(WLTP (2015):**

  The Worldwide harmonized Light vehicles Test Procedures (WLTP) define a global harmonized standard for determining the levels of pollutants and CO2 emissions, fuel or energy consumption, and electric range from light-duty vehicles (passenger cars and light commercial vans).

  It is being developed by experts from the European Union, Japan, and India under guidelines of UNECE World Forum for Harmonization of Vehicle Regulations, with final version expected by October 2015.

- Test procedure

  The test procedure provides a strict guidance regarding conditions of dynamometer tests and road load (motion resistance), gear shifting, total car weight (by including optional equipment, cargo and passengers), fuel quality, ambient temperature, and tire selection and pressure.

  Three different WLTC test cycles are applied, depending on vehicle class defined by power-weight ratio PWr in kW/Tonne (rated engine power / kerb weight):

  Class 1 - low power vehicles with PWr <= 22;

  Class 2 - vehicles with 22 < PWr <= 34;

  Class 3 - high-power vehicles with PWr > 34;

`

Most common cars have nowadays power-weight ratios of 40-70 kW/Tonne, so belong to class 3. Vans and buses can also belong to class 2.

In each class, there are several driving tests designed to represent real world vehicle operation on urban and extra-urban roads, motorways, and freeways. The duration of each part is fixed between classes; however the acceleration and speed curves are shaped differently. The sequence of tests is further restricted by maximum vehicle speed Vmax.

Increased variety of manual gearboxes with 4, 5, 6 and 7 gears makes it impossible to specify fixed gear shift points. The WLTP testing procedure instead provides an algorithm for calculating optimal shift points, which takes into account total vehicle weight and full load power curves within normalized engine speeds, covering the wide range of rotation speeds and engine power allowed by current technology. To reflect practical use and fuel efficient driving style, frequent gear changes occurring in less than 5 seconds are filtered out.

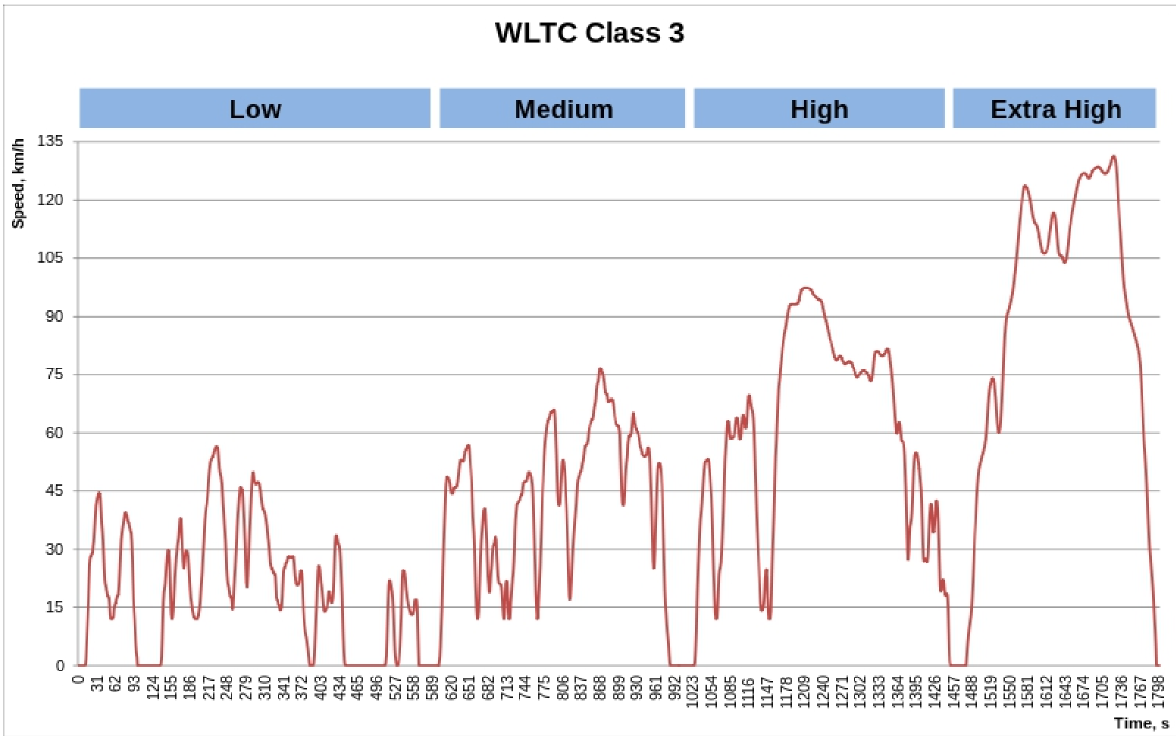


Fig 1.5: WLTC driving cycle for Class 3 vehicles

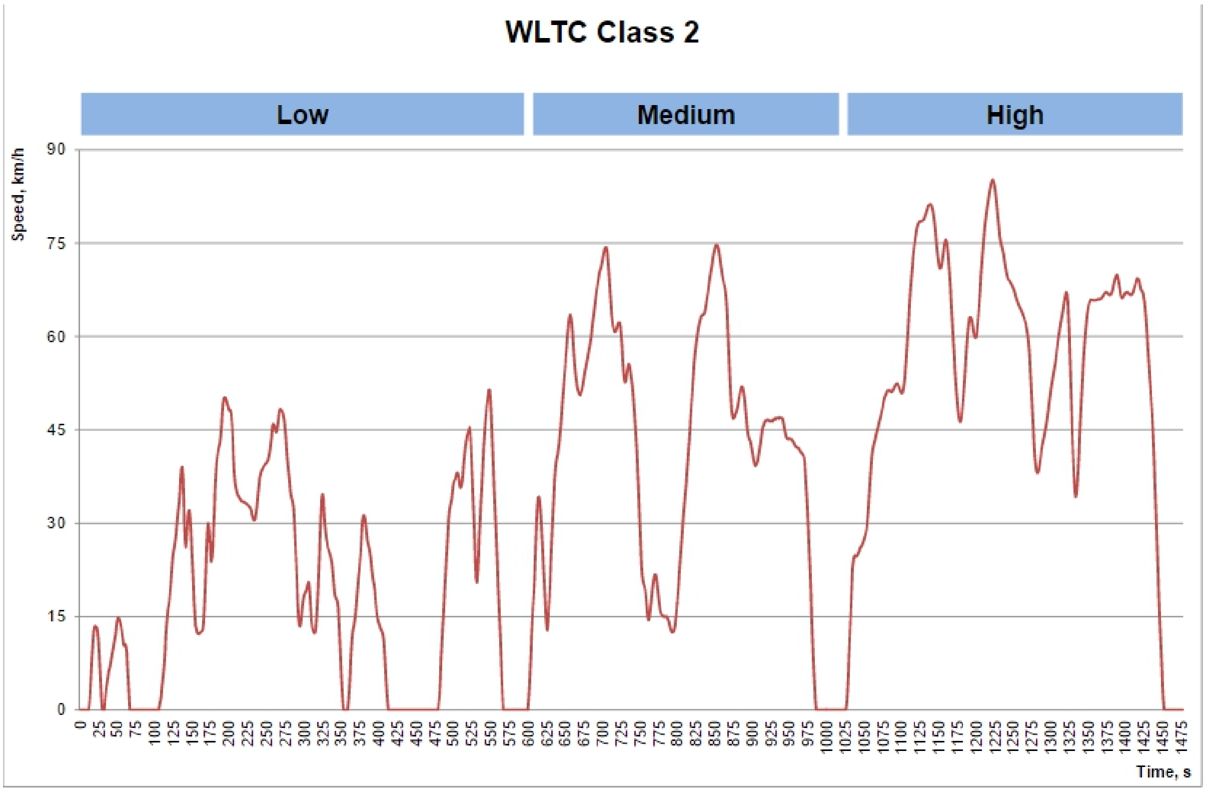


Fig 1.6: WLTC driving cycle for Class 2 vehicles

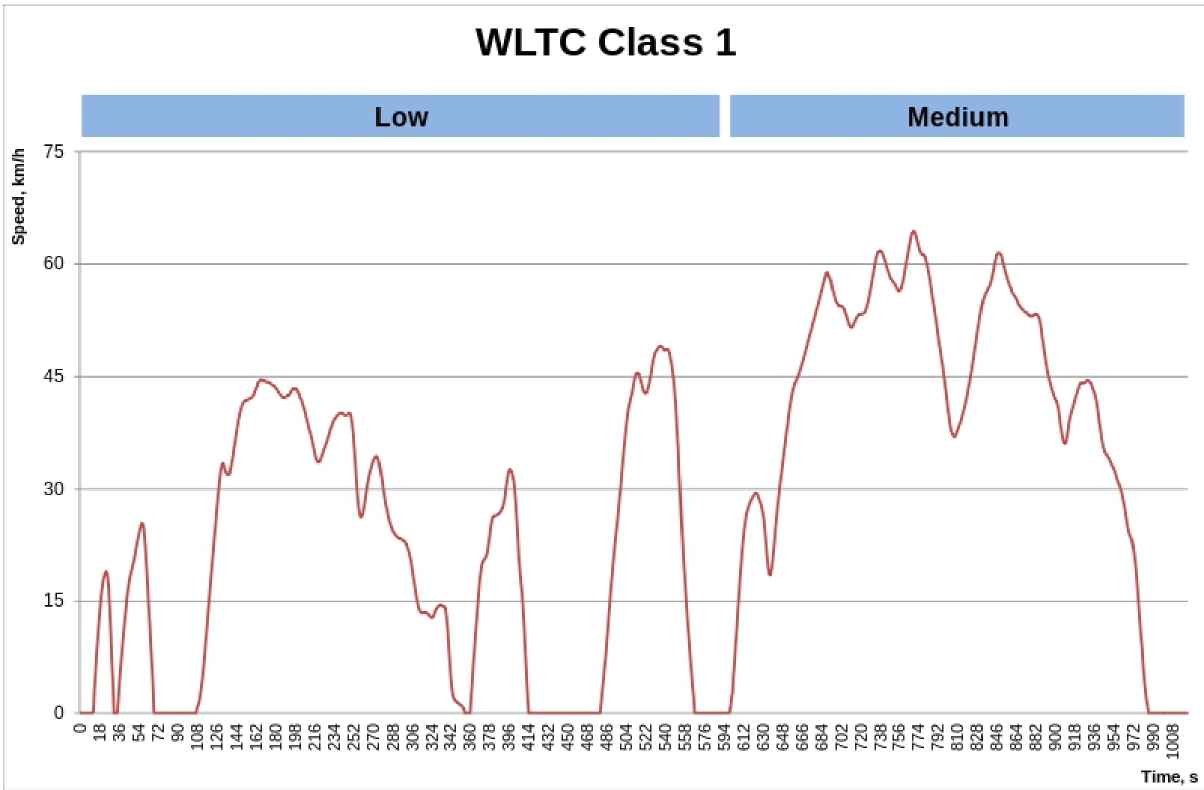


Fig 1.7: WLTC driving cycle for Class 1 vehicles

Objective

The objective of our thesis work was:

- ✓ Analysis and Development of Speed Measuring Methods.
- ✓ Selecting an efficient and economical feasible Speed Measuring Method.
- ✓ By recoding sample data developing a Sample Driving Cycle.

2 Methodology

2.1. Data Accusation Methods:

2.1.1. Using Accelerometer and GPS module:

Inertial Measurement Unit (IMU)

An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and sometimes the magnetic field surrounding the body, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. IMUs are typically used to maneuver aircraft, including unmanned aerial vehicles (UAVs), among many others, and spacecraft, including satellites and landers. Recent developments allow for the production of IMU-enabled GPS devices. An IMU allows a GPS receiver to work when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interference is present.

The IMU is the main component of inertial navigation systems used in aircraft, spacecraft, watercraft, drones, UAV and guided missiles among others. In this capacity, the data collected from the IMU's sensors allow a computer to track a craft's position, using a method known as dead reckoning

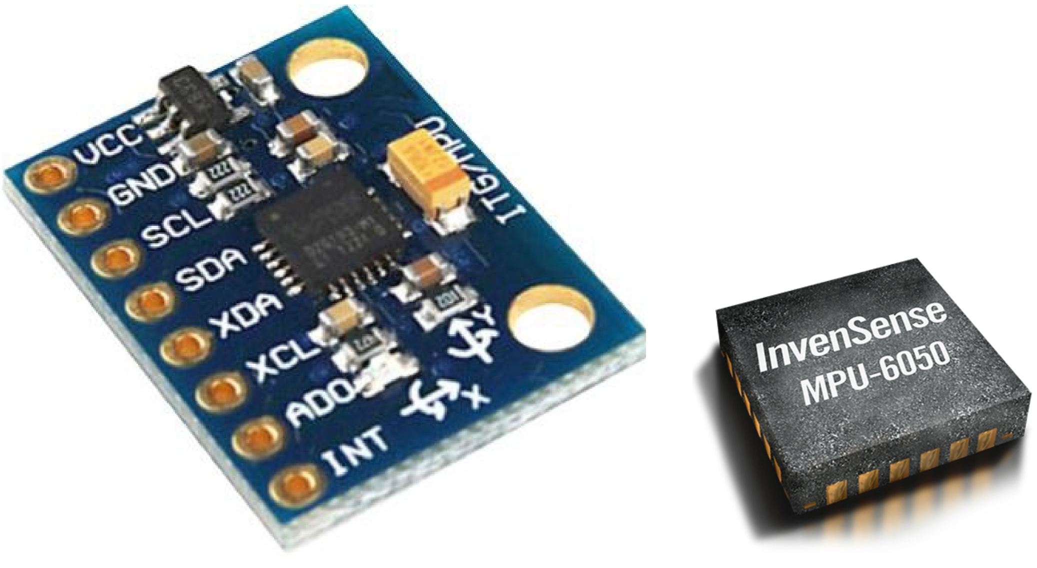


Fig. 2.1: IMU sensor (MPU 6050).

IMU sensors usually consist of two or more parts. Listing them by priority, they are accelerometer, gyroscope, magnetometer and altimeter. The MPU 6050 is a 6 DOF (Degrees of Freedom) or a six axis IMU sensor, which means that it gives six values as output. Three value from the accelerometer and three from the gyroscope. The MPU 6050 is a sensor based

`

on MEMS (Micro Electro Mechanical Systems) technology. Both the accelerometer and the gyroscope are embedded inside a single chip. This chip uses I2C (Inter Integrated Circuit) protocol for communication.

How does an accelerometer work?

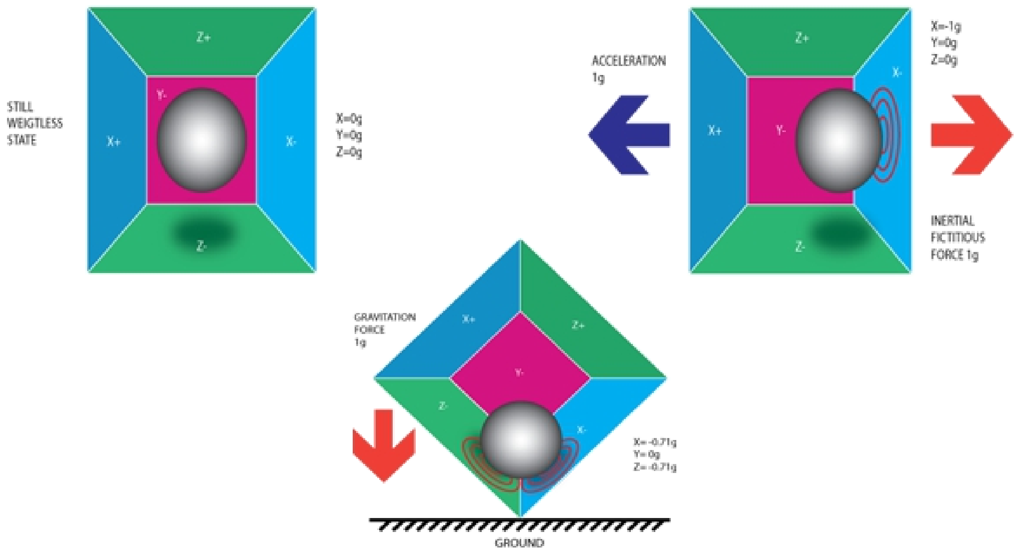


Fig. 2.2: Inside an IMU sensor (MPU 6050).

Piezo Electric Accelerometer

An accelerometer works on the principle of piezo electric effect. Here, imagine a cuboidal box, having a small ball inside it, like in the picture above. The walls of this box are made with piezo electric crystals. Whenever you tilt the box, the ball is forced to move in the direction of the inclination, due to gravity. The wall, with which the ball collides, creates tiny piezo electric currents. There are totally, three pairs of opposite walls in a cuboid. Each pair corresponds to an axis in 3D space: X, Y and Z axes.

`

SENSOR FUSION

Sensor fusion is combining of sensor data or data derived from disparate sources such that the resulting information has less uncertainty than would be possible when these sources were used individually. The term *uncertainty reduction* in this case can mean more accurate, more complete, or more dependable, or refer to the result of an emerging view, such as stereoscopvision (calculation of depth information by combining two-dimensional images from two cameras at slightly different viewpoints).

The MPU-6050 incorporates Sensor Fusion and run-time calibration firmware that enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices in motion-enabled products, guaranteeing that sensor fusion algorithms and calibration procedures deliver optimal performance.

The MPU-6050 devices combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard Digital Motion Processor (DMP), which processes complex 6-axis Motion Fusion algorithms.

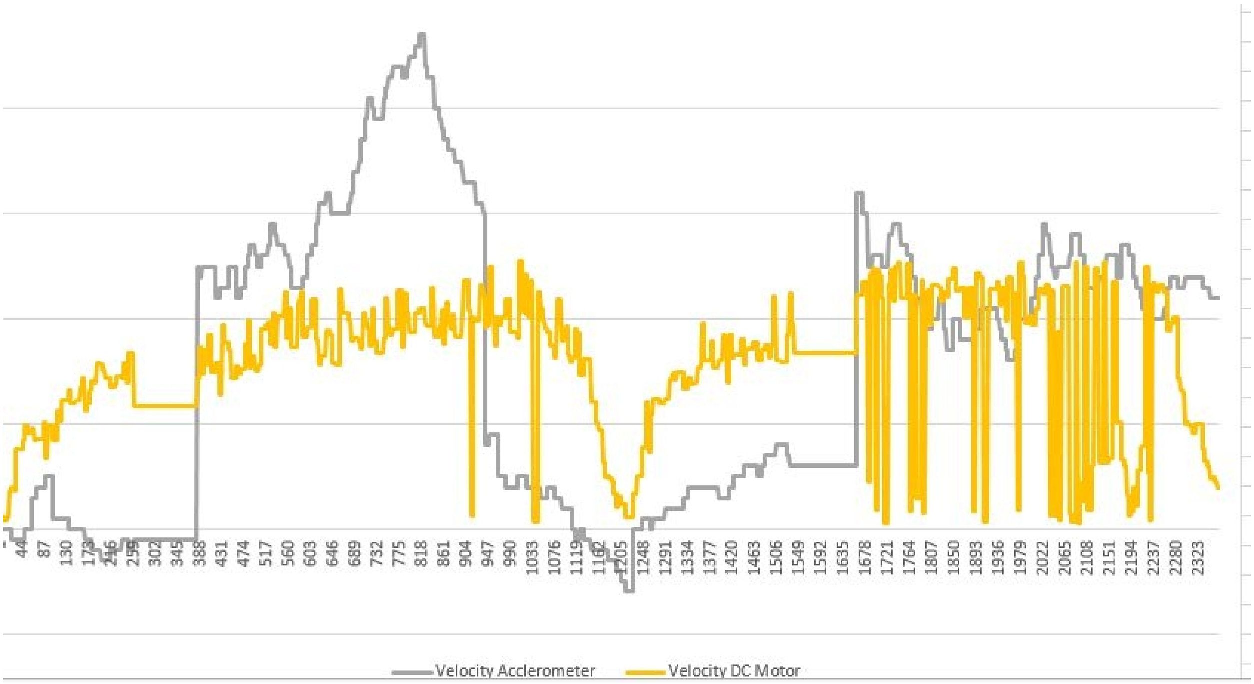


Fig. 2.3: Comparison of velocity from accelerometer Data and velocity from DC motor.

Here, the acceleration data was taken from IMU sensor and integrated over time and compared with calibrated velocity data taken from a DC motor.

Observations

1. Data is shifted as errors are also continuously added during integration.
2. Fast response with change of velocity.

How GPS Works:

The GPS receiver gets a signal from each GPS satellite. The satellites transmit the exact time the signals are sent. By subtracting the time the signal was transmitted from the time it was received, the GPS can tell how far it is from each satellite. The GPS receiver also knows the exact position in the sky of the satellites, at the moment they sent their signals. So given the travel time of the GPS signals from three satellites and their exact position in the sky, the GPS receiver can determine your position in three dimensions - east, north and altitude.

There is a complication. To calculate the time the GPS signals took to arrive, the GPS receiver needs to know the time very accurately. The GPS satellites have atomic clocks that keep very precise time, but it's not feasible to equip a GPS receiver with an atomic clock. However, if the GPS receiver uses the signal from a fourth satellite it can solve an equation that lets it determine the exact time, without needing an atomic clock.

If the GPS receiver is only able to get signals from 3 satellites, you can still get your position, but it will be less accurate. As we noted above, the GPS receiver needs 4 satellites to work out your position in 3-dimensions. If only 3 satellites are available, the GPS receiver can get an approximate position by making the assumption that you are at mean sea level. If you really are at mean sea level, the position will be reasonably accurate. However if you are in the mountains, the 2-D fix could be hundreds of meters off.

A modern GPS receiver will typically track all of the available satellites simultaneously, but only a selection of them will be used to calculate your position.

The GPS satellite system:

The 24 satellites that make up the GPS space segment are orbiting the earth about 12,000 miles above us. They are constantly moving, making two complete orbits in less than 24 hours. These satellites are travelling at speeds of roughly 7,000 miles an hour.

GPS satellites are powered by solar energy. They have backup batteries onboard to keep them running in the event of a solar eclipse, when there's no solar power. Small rocket boosters on each satellite keep them flying in the correct path.

U-blox NEO 6 GPS Module

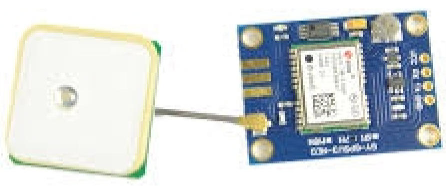


Fig. 2.4: U-blox NEO 6 GPS Module.

`

Horizontal position accuracy

- GPS 2.5 m
- SBAS(satellite-based augmentation system) 2.0 m

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints.

NEMA Protocol

GPS modules typically put out a series of standard strings of information, under something called the National Marine Electronics Association (NMEA) protocol.

NMEA is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association. It replaces the earlier NMEA 0180 and NMEA 0182 standards. In marine applications, it is slowly being phased out in favor of the newer NMEA 2000 standard.

eg1. $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
eg2. $GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

225446	Time of fix 22:54:46 UTC
A	Navigation receiver warning A = Valid position, V = Warning
4916.45,N	Latitude 49 deg. 16.45 min. North
12311.12,W	Longitude 123 deg. 11.12 min. West
000.5	Speed over ground, Knots
054.7	Course Made Good, degrees true
191194	UTC Date of fix, 19 November 1994
020.3,E	Magnetic variation, 20.3 deg. East
*68	mandatory checksum

GNSS

Augmentation of a global navigation satellite system (GNSS) is a method of improving the navigation system's attributes, such as accuracy, reliability, and availability, through the integration of external information into the calculation process. There are many such systems in place and they are generally named or described based on how the GNSS sensor receives the external information.

SBAS

A **satellite-based augmentation system** (SBAS) is a system that supports wide-area or regional augmentation through the use of additional satellite-broadcast messages. Such systems are commonly composed of multiple ground stations, located at accurately-surveyed

`

points. The ground stations take measurements of one or more of the GNSS satellites, the satellite signals, or other environmental factors which may impact the signal received by the users. Using these measurements, information messages are created and sent to one or more satellites for broadcast to the end users.

HOW GPS MEASURES VELOCITY

Stand-alone single-frequency GNSS (global navigation satellite system) receivers represent the largest slice of the commercial positioning market. Such receivers operate mainly in single point position (SPP) mode and estimate velocity either by differencing two consecutive positions (i.e., approximating the derivative of user position) or by using Doppler measurements related to user-satellite motion.

The former approach is the most simple to implement, but it has a meter per second–level of accuracy due to the dependence on pseudo range - based position accuracy. In contrast, Doppler frequency shifts of the received signal produced by user-satellite relative motion enables velocity accuracy of a few centimeters per second.

Both of the velocity data from SPP and Doppler shift are processed motion fusion algorithm *i.e.* Kalman filter and better precision in velocity measurement is acquired.

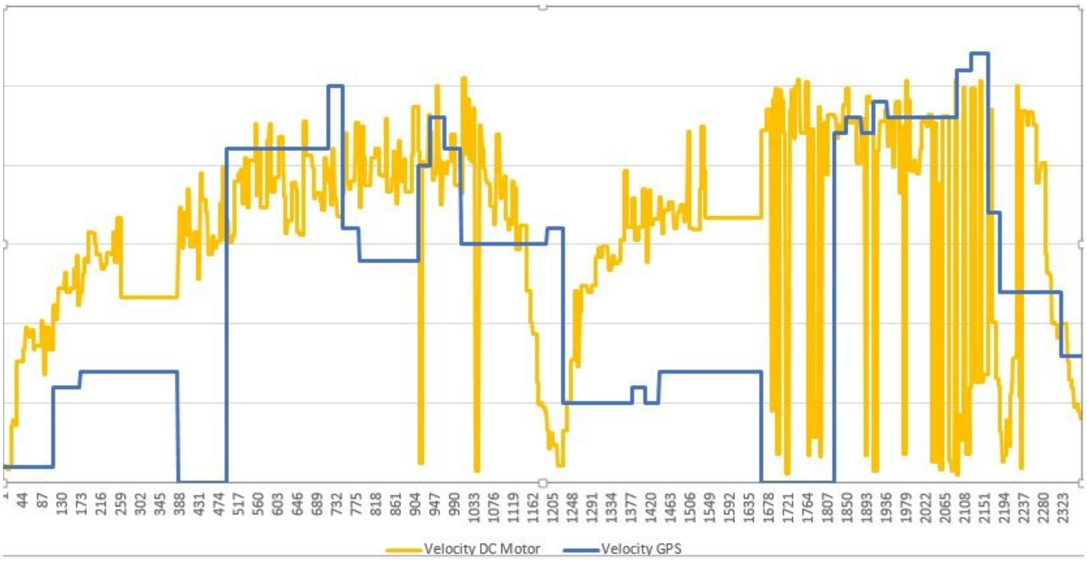


Fig. 2.5: Comparison of velocity from GPS Data and velocity from DC motor.

We obtained GPS velocity data and compared with calibrated

1. Slow response with change in velocity.
2. No drift of data in long time. Low steady state error.

Kalman Filtering

Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies,

`

and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone.

The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

Why kalman filtering?

The vehicle can be equipped with a GPS unit that provides an estimate of the velocity within a few meters/sec. The GPS estimate is likely to be noisy; readings 'jump around' rapidly, though remaining within a few meters of the real velocity. In addition, since the vehicle is expected to follow the laws of physics, its velocity can also be estimated by integrating its acceleration over time, determined by acceleration found from MPU 6050. This is a technique known as dead reckoning. Typically, the dead reckoning will provide a very smooth estimate of the truck's velocity, but it will drift over time as small errors accumulate.

Here Kalman filter can be thought of as operating in two distinct phases: predict and update. In the prediction phase, the vehicle's old velocity will be modified according to the physical laws of motion (the dynamic or "state transition" model). Not only will a new velocity estimate be calculated, but a new covariance will be calculated as well. Perhaps the covariance is proportional to the acceleration of the vehicle because we are more uncertain about the accuracy of the dead reckoning velocity estimate at high acceleration but very certain about the estimate when moving at low acceleration.

Next, in the update phase, a measurement of the vehicle's speed is taken from the GPS unit. Along with this measurement comes some amount of uncertainty, and its covariance relative to that of the prediction from the previous phase determines how much the new measurement will affect the updated prediction. Ideally, as the dead reckoning estimates tend to drift away from the real position, the GPS measurement should pull the position estimate back towards the real position but not disturb it to the point of becoming rapidly jumping and noisy.

`

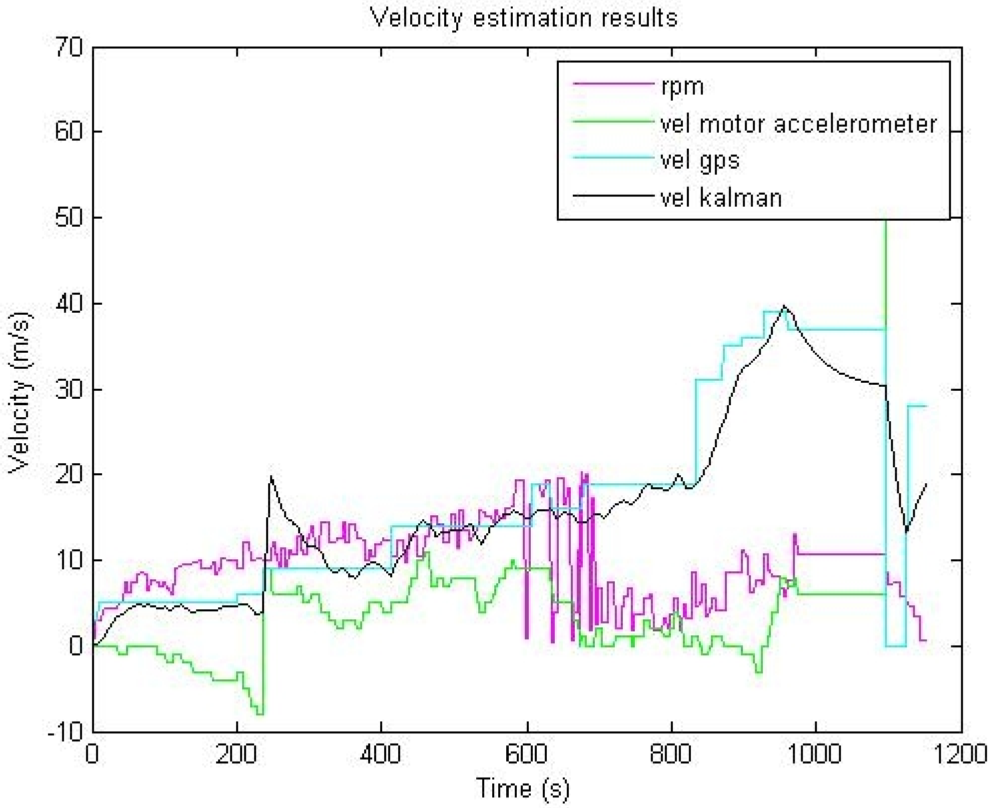


Fig. 2.6: Combining Accelerometer and GPS data using Kalman Filter.

Combining both accelerometer and GPS velocity data

1. increases response behavior.

2 .removes steady state error.

3. But still visibly error 20-30% remains from adc data of DC motor.

ALGORITHM

1. Prediction of velocity data from acceleration.

2. Measured velocity from gps .

3. Estimation of kalman gain.

(According to the error with each data)

4. Filtered data.

2.1.2. Using External Device or Add-on Device: DC Motor- Shaft Wheel Arrangement:

A **DC motor** is a rotary electrical machine that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. The DC motor can also be used as a readymade generator that will generate electricity. It can be used as generators by applying mechanical torque to the output shaft to induce a current.

A DC motor was coupled with a Pressure Hand Drill and then the RPM of the motor and Voltage of the motor was measured by using a Tachometer and a Multimeter respectively.

Fig 2.13: Experimental setup for testing DC Motor for different input rpm for voltage outputs.

By testing the DC Motor found directly a **linear relation between rpm of the motor shaft and voltage output** from it (Graph below):

Voltage [V]	Input RPM
3	1540
3.47	1783
3.7	1907

`

4	2055
4.5	2330
5	2568
5.5	2850
6.1	3155
6.6	3414
7	3626

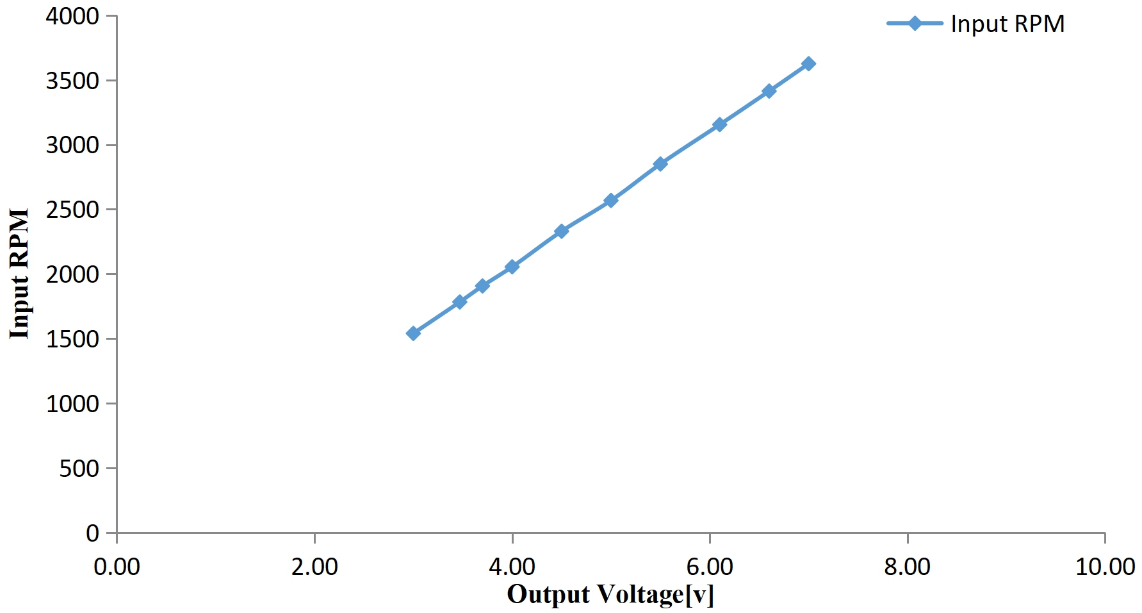


Fig 2.14: A plot of Input RPM vs. Output Voltage

Experimental Setup:

The generated voltage by the dc motor is directly proportional to the rotational motion of its armature. So the armature of the motor was connected with the wheel of a car by the experimental device. A device was developed which can be attached with the bumper of the vehicle. The device has two wheels and a motor connected via a round belt with the wheel shaft.

A list of key Components used in this device is listed below:

- Two Wheels
- Round Rubber belt
- 5 volt dc motor

`

- Bearings
- Two Adjustable Spring Damper
- One DC motor
- Structural frame made of Stainless Steel
- Clamps for Bumper

Fig 2.15: Add-on Device; DC Motor-Wheel Arrangement

Speed Ratio Calculation:

Wheel Shaft Dia., D_w: 2.5 cm.

Wheel Shaft Groove, a: .25cm

Sheave Dia. of Wheel Shaft, d_w: 2cm

Pulley Dia., D_p: 6cm

Pulley Groove, b: .25cm

Pulley Sheave Dia., d_p: 5.5cm

The relationship between the rotational speed of the motor and the Wheel and the Sheave diameter can be expressed as

$$d_p \times n_p = d_w \times n_w \tag{1}$$

Where,

`

n_p= revolutions per minute of the Motor.

n_w= revolutions per minute of the Wheel.

Speed ratio can be calculated as

$$Speed\ Ratio = \frac{n_w}{n_p} = \frac{d_p}{d_w} = \frac{5.5}{2} = 2.75 \quad (2)$$

This means, for the motor shaft to turn **one single rotation** the wheels have to turn **2.75 rotations**.

A general speed calculation data table from the testing of the DC motor mention previous is given below:

Voltage	Motor Rpm	Wheel Rpm	Angular Velocity [rad/s]	Vehicle Speed [m/s]	Vehicle Speed [km/h]
3	1540	4235	443.4892	33.79387704	121.6579573
3.47	1783	4903.25	513.46834	39.12628751	140.854635
3.7	1907	5244.25	549.17786	41.84735293	150.6504706
4	2055	5651.25	591.7989	45.09507618	162.3422742
4.5	2330	6407.5	670.9934	51.12969708	184.0669095
5	2568	7062	739.53264	56.35238717	202.8685938
5.5	2850	7837.5	820.743	62.5406166	225.1462198
6.1	3155	8676.25	908.5769	69.23355978	249.2408152
6.6	3414	9388.5	983.16372	74.91707546	269.7014717
7	3626	9971.5	1044.21548	79.56921958	286.4491905

`

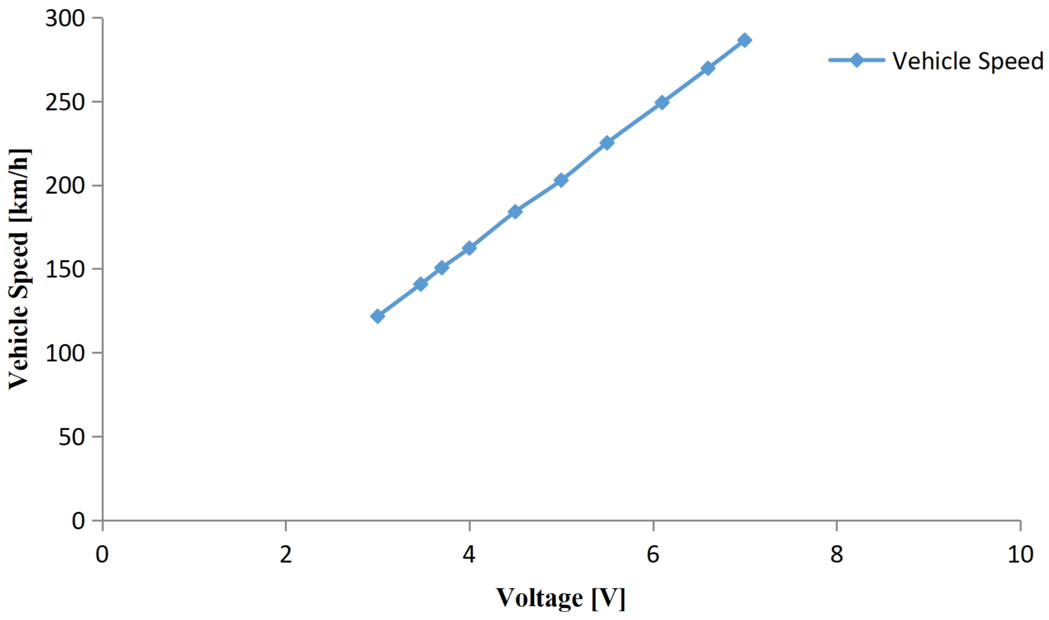


Fig 2.15: A plot of Vehicle Speed vs. Voltage for high RPMs.

From Fig 2.15 it is certain that to generate 3V the velocity of the vehicle have to be about 121 km/h. However, this doesn't represent the general scenario of vehicle speed in cities in Bangladesh. Therefore, the speed of the vehicle has to be calculated by extrapolating from this data.

By Linear curve fitting the above data an equation is generated to predict the speed from voltage. The equation is given below:

$$Speed\ in\ km/h\ =\ 41.187 \times (Voltage)\ -\ 1.9851 \tag{3}$$

According to this equation, if the motor generates 1V then the vehicle speed would be 39.20 km/h.

Again the DC motor was tested by coupling it with a **Bench Drill** for different RPMs to get an idea about the voltage generation in low RPM and to check if the speed is predicted from the equation (3) then how much accuracy the prediction has. The data is given below:

Voltage	Motor Rpm	Wheel Rpm	Angular Velocity [rad/s]	Vehicle Speed [m/s]	Vehicle Speed [km/h]
0.83	400	1100	115.192	8.7776304	31.59946944
1.52	730	2007.5	210.2254	16.01917548	57.66903173
2.73	1330	3657.5	383.0134	29.18562108	105.0682359
4.85	2440	6710	702.6712	53.54354544	192.7567636

`

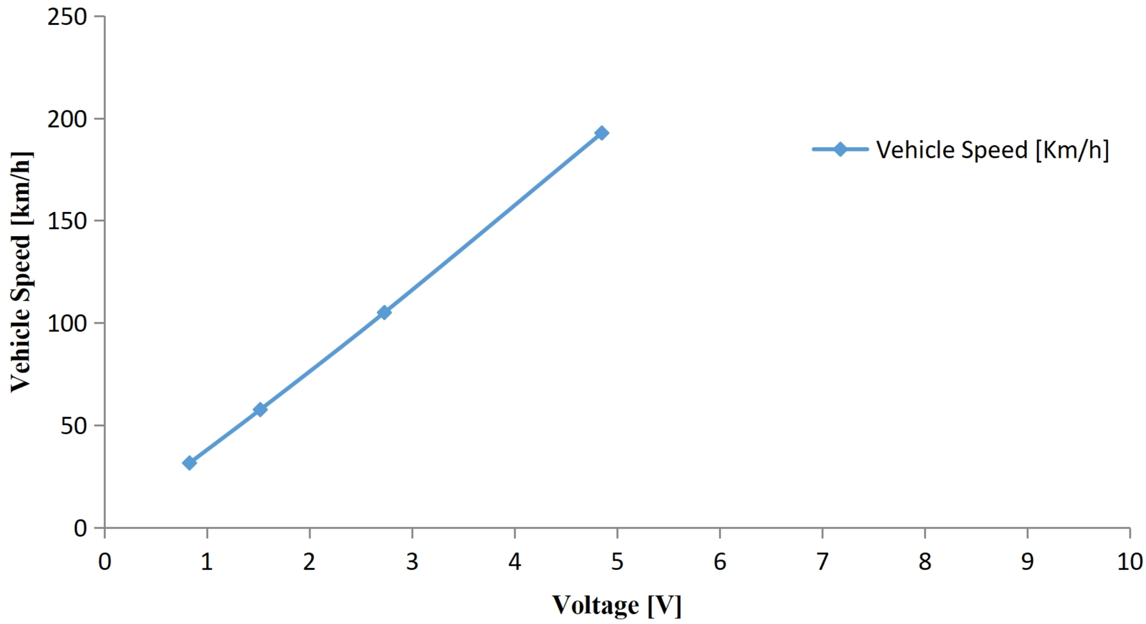


Fig 2.16: A plot of Vehicle Speed vs. Voltage for low RPMs.

By Linear curve fitting the above data an equation is generated to predict the speed from voltage. The equation is given below:

$$Speed\ in\ km/h \ = \ 40.182 \times (Voltage) \ - \ 2.9773 \qquad (4)$$

According to this equation, if the motor generates 1V then the vehicle speed would be 37.20 km/h.

A comparison between equation (3) and (4) for different voltages is given below:

Voltage [V]	Vehicle Speed [km/h]	
	Equation (4)	Equation (3)
0	-2.9773	-1.9851
0.1	1.0409	2.1336
0.15	3.05	4.19295
0.2	5.0591	6.2523
0.25	7.0682	8.31165
0.3	9.0773	10.371
0.35	11.0864	12.43035
0.4	13.0955	14.4897
0.45	15.1046	16.54905

`

0.5	17.1137	18.6084
0.55	19.1228	20.66775
0.6	21.1319	22.7271
0.65	23.141	24.78645
0.7	25.1501	26.8458
0.75	27.1592	28.90515
0.8	29.1683	30.9645
0.85	31.1774	33.02385
0.9	33.1865	35.0832
0.95	35.1956	37.14255
1	37.2047	39.2019

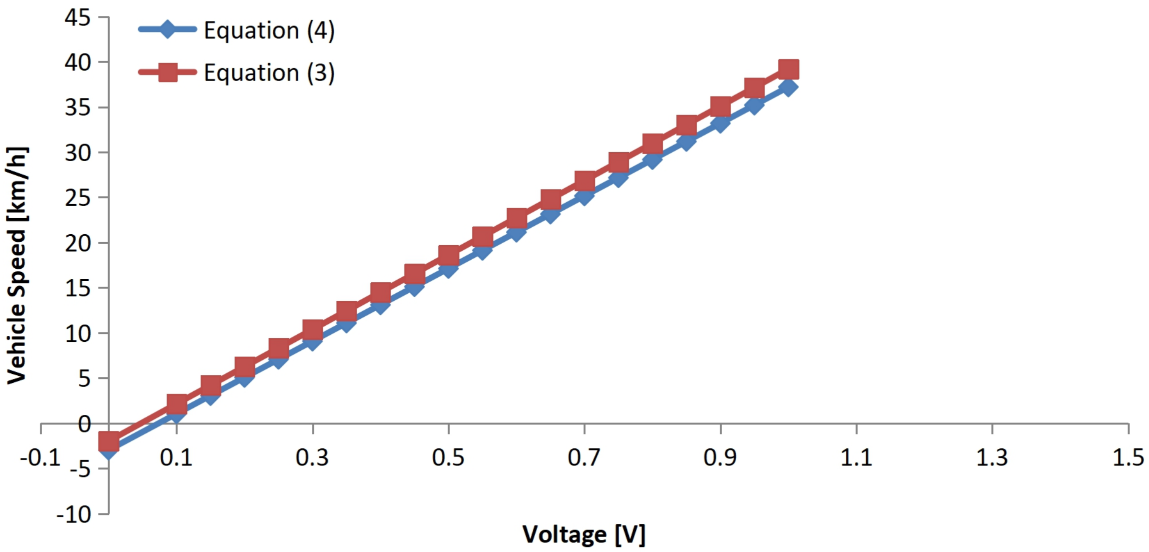


Fig 2.16: A plot of Vehicle Speed vs. Voltage using both equation (3) and (4).

From Fig 2.16 it can be said that as the speed increases the difference between the predictions from these two equations is increasing, thus there might be some level of inaccuracy in these equations.

To check the validity of these equations the multimeter reading and the speedometer reading are recorded simultaneously by video recording using an action camera in a test run.

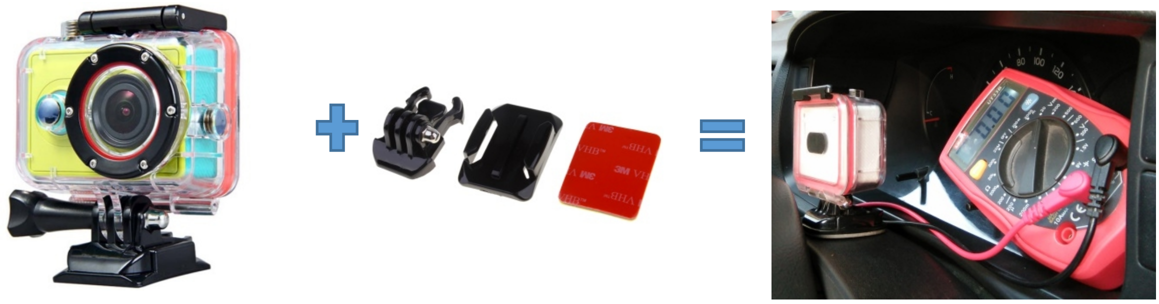
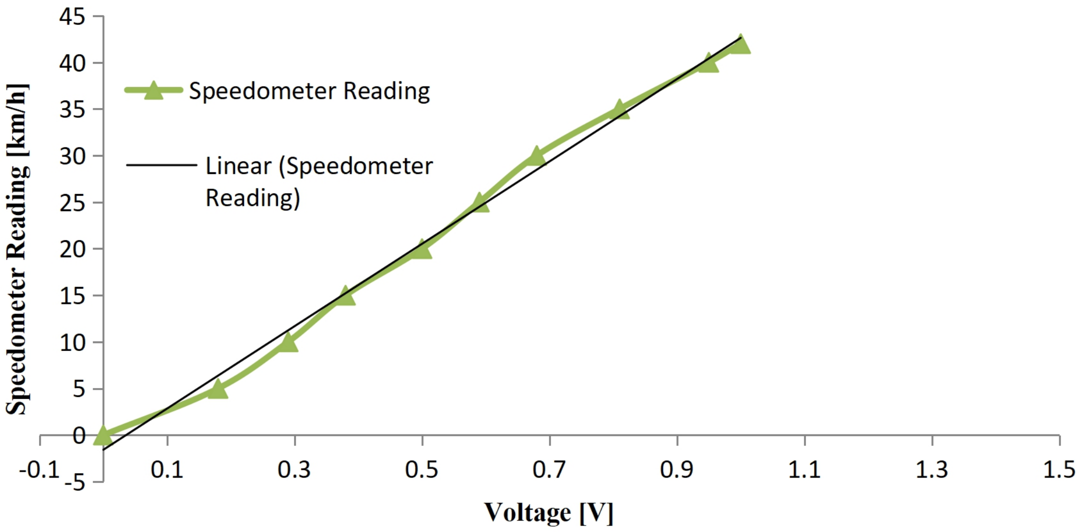


Fig 2.17: Experimental setup for recording simultaneously speedometer and multimeter reading.

A sample data table for this set up is given below:

Voltage [V]	Vehicle Speed [km/h]
	Speedometer Reading
0	0
0.18	5
0.29	10
0.38	15
0.5	20
0.59	25
0.68	30
0.81	35
0.95	40
1	42



`

Fig 2.16: A plot of Speedometer Reading vs. Voltage.

By Linear curve fitting the above data an equation is generated to predict the speed from voltage. The equation is given below:

$$Speed\ in\ km/h = 44.20228 \times (Voltage) - 1.580827 \tag{5}$$

According to this equation, if the motor generates 1V then the vehicle speed would be 42.62 km/h.

A comparison between equation (3), (4) and (5) for different voltages is given below:

Voltage	Vehicle Speed [km/h]		
	Equation (4)	**Equation (3)**	**Speedometer Reading**
0	-2.9773	-1.9851	-1.58083
0.1	1.0409	2.1336	2.839401
0.15	3.05	4.19295	5.049515
0.2	5.0591	6.2523	7.259629
0.25	7.0682	8.31165	9.469743
0.3	9.0773	10.371	11.67986
0.35	11.0864	12.43035	13.88997
0.4	13.0955	14.4897	16.10009
0.45	15.1046	16.54905	18.3102
0.5	17.1137	18.6084	20.52031
0.55	19.1228	20.66775	22.73043
0.6	21.1319	22.7271	24.94054
0.65	23.141	24.78645	27.15066
0.7	25.1501	26.8458	29.36077
0.75	27.1592	28.90515	31.57088
0.8	29.1683	30.9645	33.781
0.85	31.1774	33.02385	35.99111
0.9	33.1865	35.0832	38.20123
0.95	35.1956	37.14255	40.41134
1	37.2047	39.2019	42.62145

`

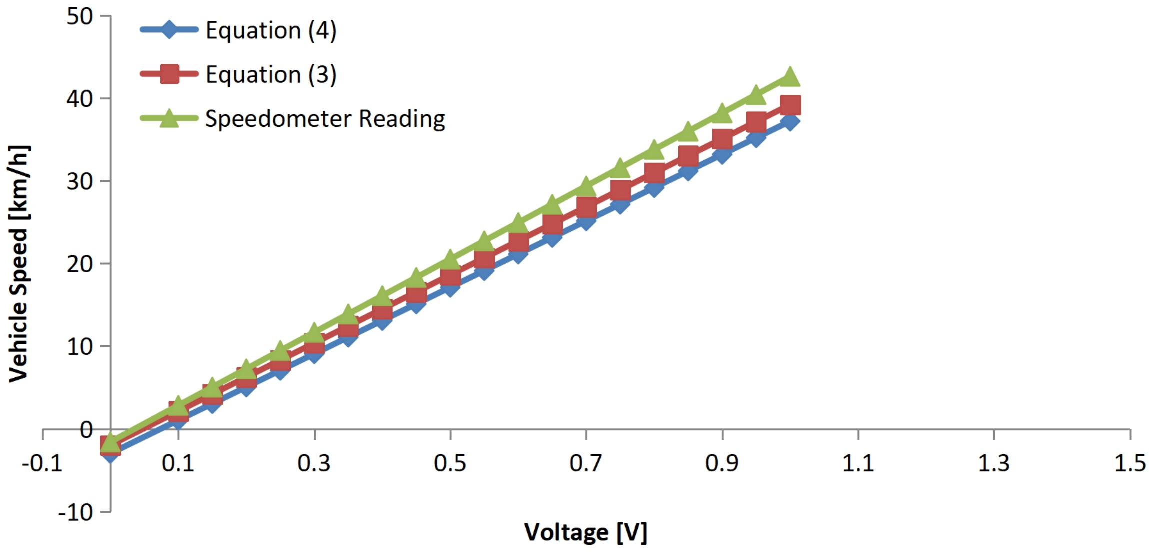


Fig 2.17: A plot of Vehicle Speed vs. Voltage using equation (3), (4) and (5).

Fig 2.17 indicates that as the speed increases the difference between the predictions from equation (3) and (4) with equation (5) is increasing. Therefore, further tests are done to check if the speedometer data is consistent with every run.

To check the consistency of the speedometer further test runs are done and the findings are given below:

Run 1		Run 2		Run 4	
Voltage	**Km/h**	**Voltage**	**Km/h**	**Voltage**	**Km/h**
0.02	0	0.1	2	0.14	4
0.2	8	0.17	5	0.23	8
0.22	10	0.27	10	0.26	10
0.31	12	0.27	10	0.28	10
0.33	14	0.28	12	0.36	15
0.34	15	0.31	11	0.45	20
0.41	18	0.34	15	0.47	20
0.45	20	0.39	15	0.51	21
0.48	21	0.42	18	0.61	25
0.5	21	0.48	21	0.68	30
0.5	21	0.51	22	0.69	30
0.53	23	0.52	22	0.71	30
0.62	28	0.59	25	0.81	35
0.66	28	0.59	25	0.91	38
0.67	29	0.67	29		
0.68	30	0.68	28		

`

0.73	**31**	**0.72**	**30**		
0.81	**35**	**0.72**	**34**		
0.91	**39**	**0.93**	**40**		
0.93	**40**	**0.95**	**40**		
0.98	**42**				

Here, Run 1 is the previously mentioned speedometer data. By Linear curve fitting two equations are generated to predict the speed from voltage for run 2 and run 3. The equation is given below:

For Run 2,

$$Speed\ in\ km/h = 45.597 \times (\text{Voltage}) - 1.8931 \qquad (6)$$

For Run 3,

$$Speed\ in\ km/h = 45.251 \times (Voltage) - 1.8382 \qquad (7)$$

A comparison of equation (3) up to (7) is given below:

Voltage[V]	**Vehicle Speed [km/h]**				
	Equation (5)	**Equation (6)**	**Equation (7)**	**Equation (4)**	**Equation (3)**
0.02	**0.57438**	**-0.9332**	**-0.9812**	**-2.1737**	**-1.1614**
0.2	**8.4132**	**7.212**	**7.2263**	**5.0591**	**6.2523**
0.22	**9.28418**	**8.11702**	**8.13824**	**5.86274**	**7.07604**
0.31	**13.2036**	**12.1896**	**12.242**	**9.47912**	**10.7829**
0.33	**14.0746**	**13.0946**	**13.1539**	**10.2828**	**11.6066**
0.34	**14.5101**	**13.5471**	**13.6099**	**10.6846**	**12.0185**
0.41	**17.5585**	**16.7147**	**16.8017**	**13.4973**	**14.9016**
0.45	**19.3005**	**18.5248**	**18.6256**	**15.1046**	**16.5491**
0.48	**20.6069**	**19.8823**	**19.9935**	**16.3101**	**17.7847**
0.5	**21.4779**	**20.7873**	**20.9054**	**17.1137**	**18.6084**
0.5	**21.4779**	**20.7873**	**20.9054**	**17.1137**	**18.6084**
0.53	**22.7844**	**22.1448**	**22.2733**	**18.3192**	**19.844**
0.62	**26.7038**	**26.2174**	**26.377**	**21.9355**	**23.5508**
0.66	**28.4457**	**28.0275**	**28.2009**	**23.5428**	**25.1983**
0.67	**28.8812**	**28.48**	**28.6569**	**23.9446**	**25.6102**
0.68	**29.3167**	**28.9325**	**29.1129**	**24.3465**	**26.0221**
0.73	**31.4942**	**31.195**	**31.3927**	**26.3556**	**28.0814**
0.81	**34.9781**	**34.8151**	**35.0405**	**29.5701**	**31.3764**

0.91	**39.333**	**39.3402**	**39.6002**	**33.5883**	**35.4951**
0.93	**40.204**	**40.2452**	**40.5121**	**34.392**	**36.3188**
0.98	**42.3814**	**42.5078**	**42.792**	**36.4011**	**38.3782**

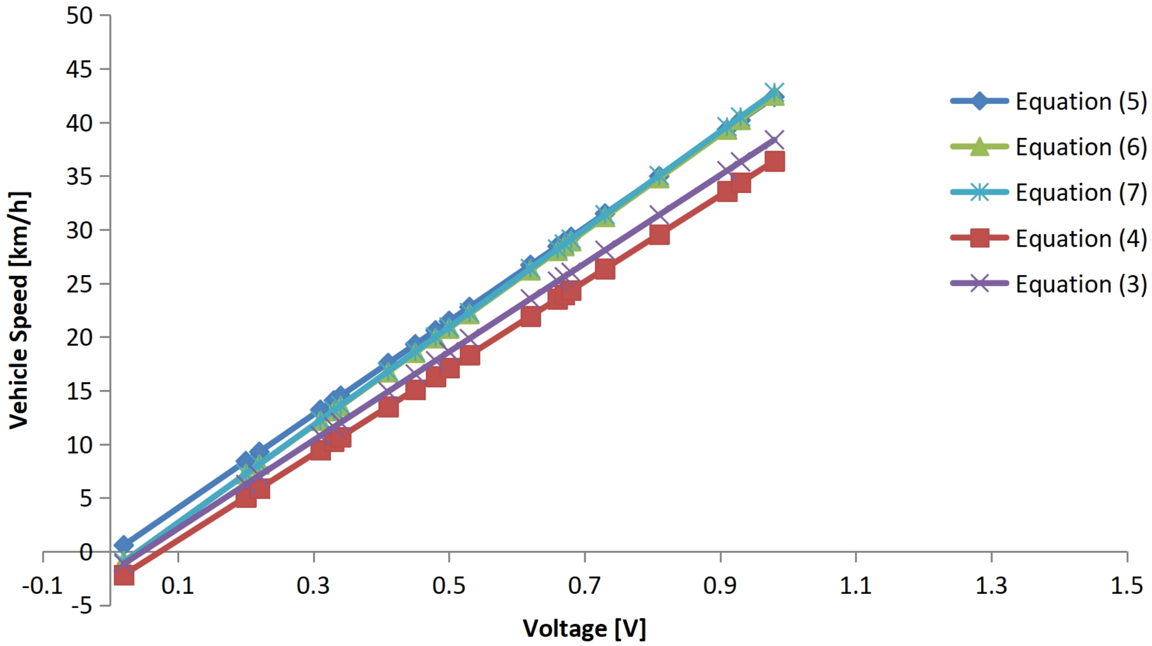


Fig 2.18: A plot of Vehicle Speed vs. Voltage using equation (3), (4), (5), (6) and (7).

Fig 2.18 shows the prediction from equation (5) up to equation (7) has less than 6% error for speeds higher than 15km/h however for low speed there is significant error up to 17%.

Therefore, equation (5) is used to calculate the vehicle speed from the generated voltage.

Software and Programming:

Fig 2.16: Arduino Uno (left), SD Card Module(right)

- Experimental Data:

`

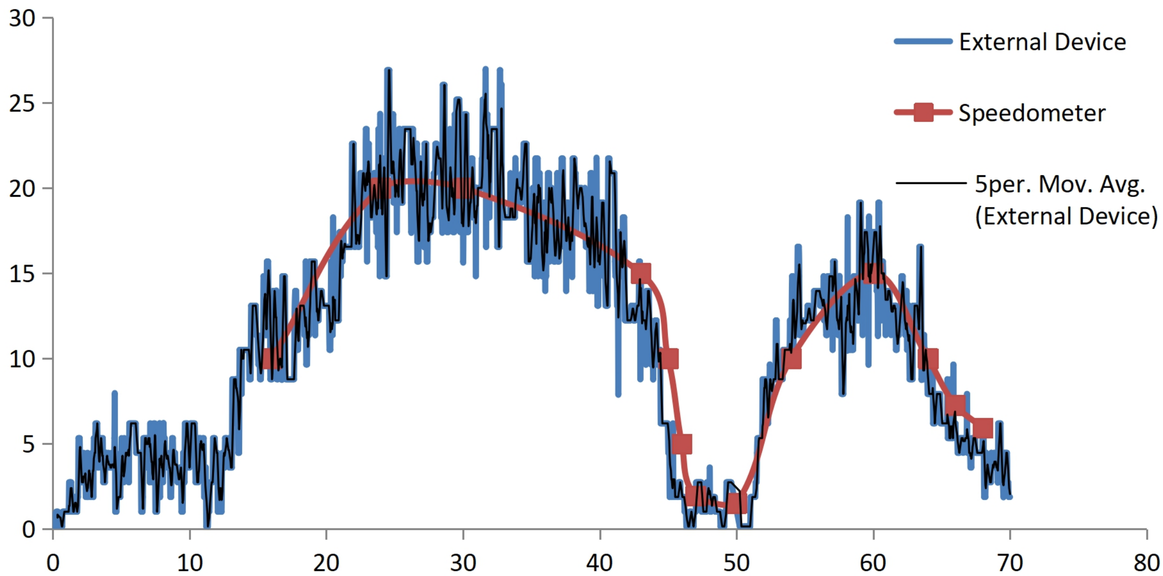


Fig 2.19: A plot of Vehicle Speed vs. Time using data from the external device and video recording of the speedometer.

2.1 Cost Chart for Suggested Add-On Method:

Sl No:	Components	No of pieces	Amount of Cost(BDT)
1	Wheel	2	800
2	Shaft(Galvanized iron)	6feet	150
3	DC Motor	1	200
4	Ball Bearing	2	240
5	Belts	2	80
6	Arduino Mega	1	850
7	Arduino Nano	1	350
9	MPU 6050		650
10	Machining		500
11	Nut-Bolts		50
12	Ss	10 feet	300
Total			5,

Table 2.5: Table of Cost

`

3 Results and Discussions

3.1 Comparison of different Speed measuring methods:

Fig 3.1: GPS Device Speed, Mobile GPS Apps and Speedometer Speed Comparison

Analyzed Measuring Methods	Effectiveness and Compatibility
Accelerometer	1.Fast response with change in velocity 2.Large error occurs with time as errors are continuously added during integration .
GPS Device:	1.Slow response with velocity change 2.less steady state error
Combining Accelerometer and GPS	1. Slow response of GPS is compensated by Accelerometer data. 2. Drift of data due to error adding is corrected by GPS 3. Still then 20-30% error remains.
Add-Device: External Device(DC Motor- Wheel-Shaft Assembly):	1. Least Error 2. Fast Response 3.No drift of data.

Table 3.1: Key points of Different Speed Measuring Method

From decision points it's seen that's our Add-on Device: DC Motor-Wheel-Shaft Assembly meets requirements:

1. It's a very easy and user friendly method
2. Cost is also very low and building instruments available at local market

`

3. It's a continuous speed providing method
4. In this method the device is added to the bumper of Automobile vehicle, so there is no chance of damage of vehicle main body.
5. The programming associated with the method being very easy one can use it to develop driving cycle for different regions and by different automotive vehicles.
6. Like GPS it doesn't have any data missing probability except some very special cases.

So considering these advantages and being the most efficient one relative to other speed measuring systems, DC Motor-Wheel-Shaft Assembly method was chosen to record data to build Driving Cycle of Dhaka City.

3.2. Constructing cycles

Defining the four standard driving modes:

Modes	Condition	Elapsed time
Idling	**velocity = 0** or **(velocity) ≤ 2 km/h** and **(acceleration) ≤ 0.3 m/s^2**	**idling time**
Acceleration	**(acceleration) > 0.3 m/s^2**	**accelerating time**
Cruising	**(velocity) > 2 km/h And -0.3 m/s^2 < (acceleration) ≤ 0.3 m/s^2**	**cruising time**
Deceleration	**(acceleration) < -0.3 m/s^2**	**decelerating time**

- Selection of vehicles: For this study a microbus was selected.
- Route Selection: BUET – TSC - FULLER ROAD – PALASHI - BUET route was considered for driving cycle purpose.
- Selecting data collection time and duration: The traffic situations keep changing at different hours of the day. For Dhaka city, the maximum traffic congestion is seen from 8 am to 11 am, from 1 pm to 3pm and from 4.30 pm to 7.30 pm. So data should be collected in such a way that both the congestion and non-congestion period may be covered. In present case, three cycles were completed by taking data for 25 minutes, starting from 15:27. The traffic condition differs in holidays and weekends. So for

`

getting the real phenomenon of the traffic situation it needs to take thousands of data on the working days.

3.2.1. First Cycle:

3.2.1.1. Speed-time profile:

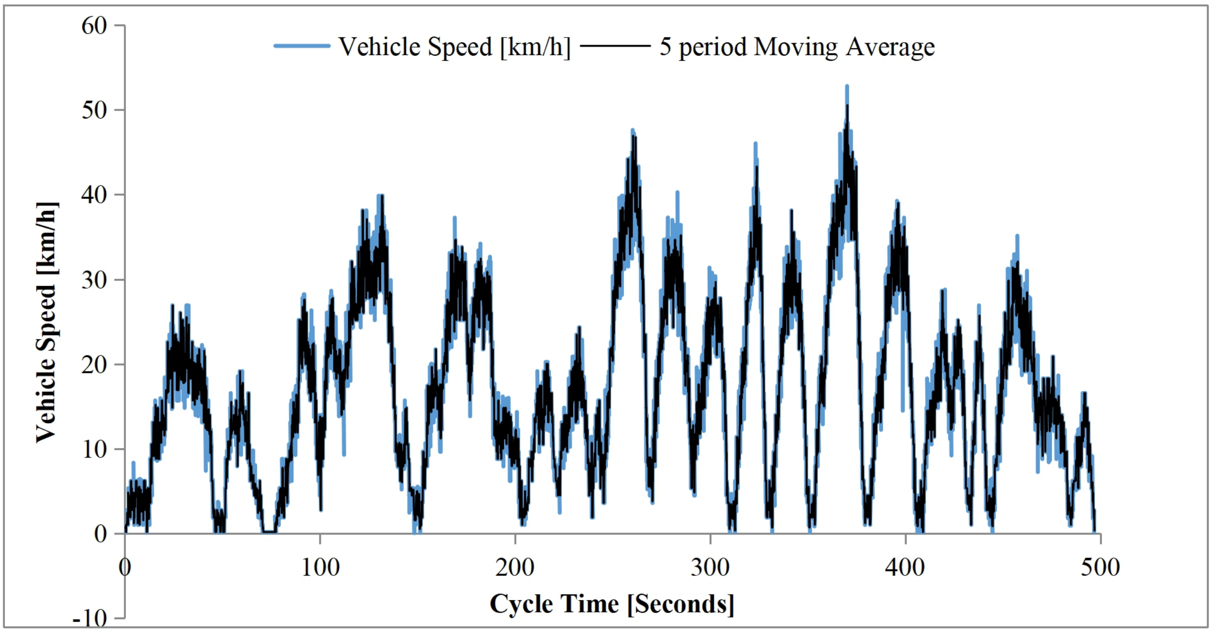


Fig 3.2: Speed-time profile

3.2.1.2. Acceleration-time profile:

`

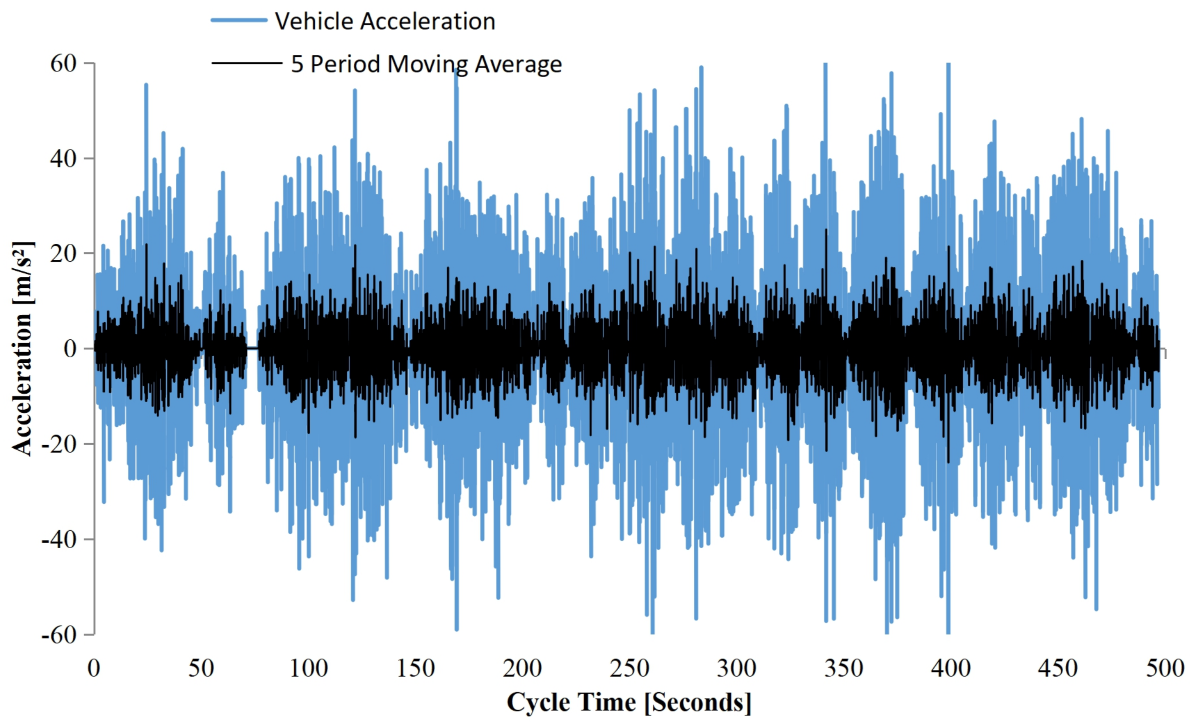


Fig 3.3: Acceleration vs time graph

3.2.1.3. Standard Modes Distribution:

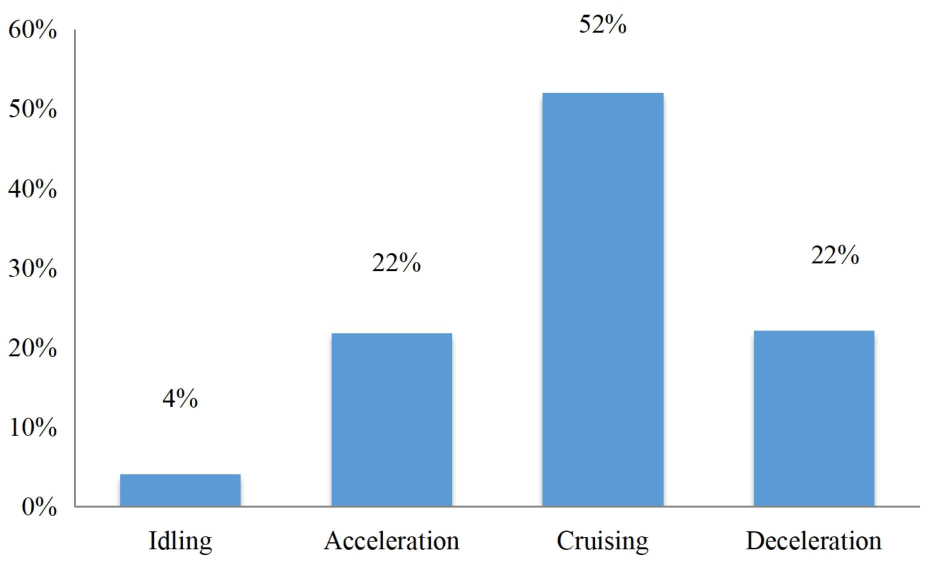


Fig 3.4: Standard Modes Distribution

`

3.2.1.4. Speed Distribution:

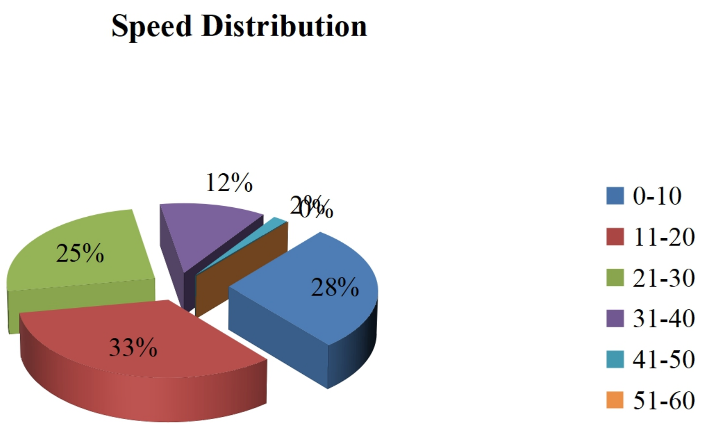


Fig 3.5: Speed Distribution

3.2.2. Second Cycle:

3.2.2.1. Speed-time profile:

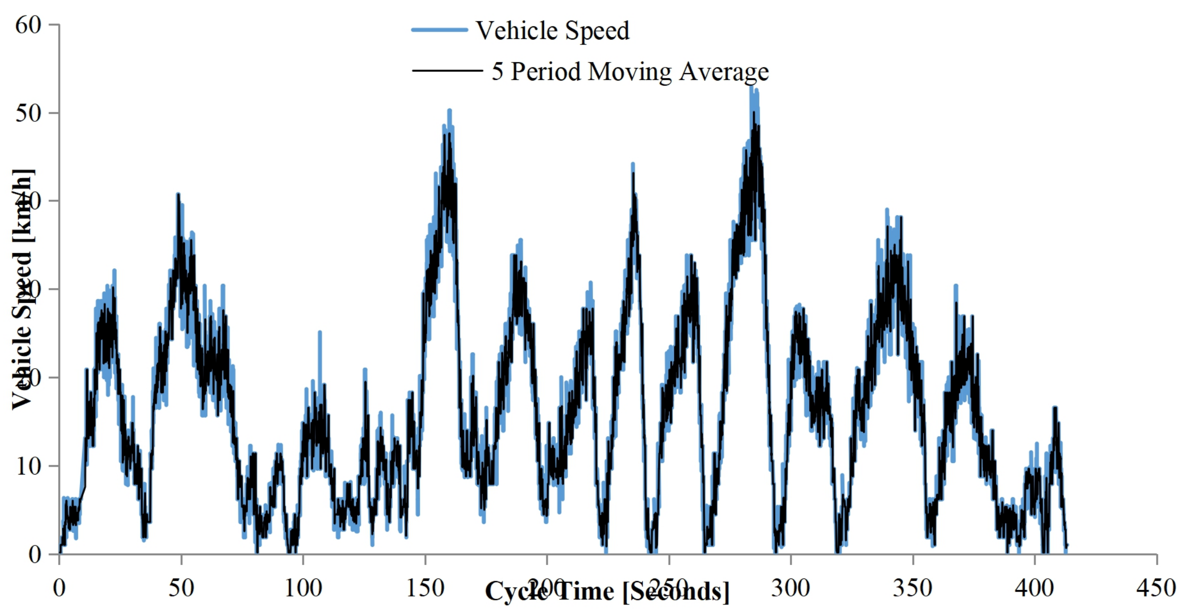


Fig 3.6: Speed-time profile

3.2.2.2. Acceleration-time profile:

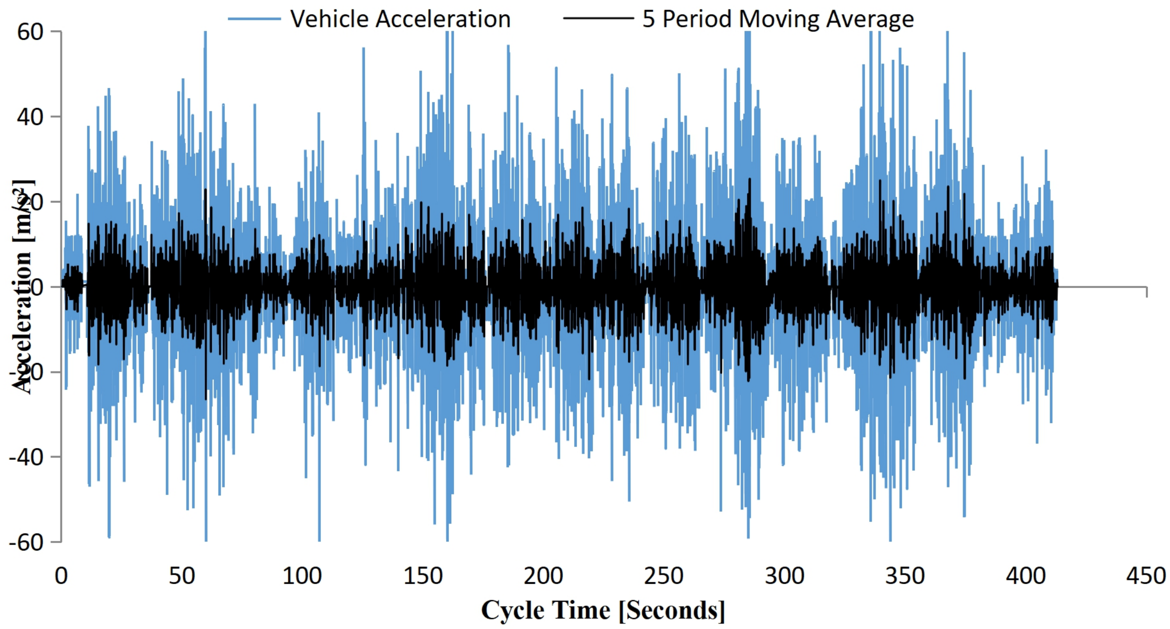


Fig 3.7: Acceleration vs time graph

3.2.2.3. Standard Modes Distribution:

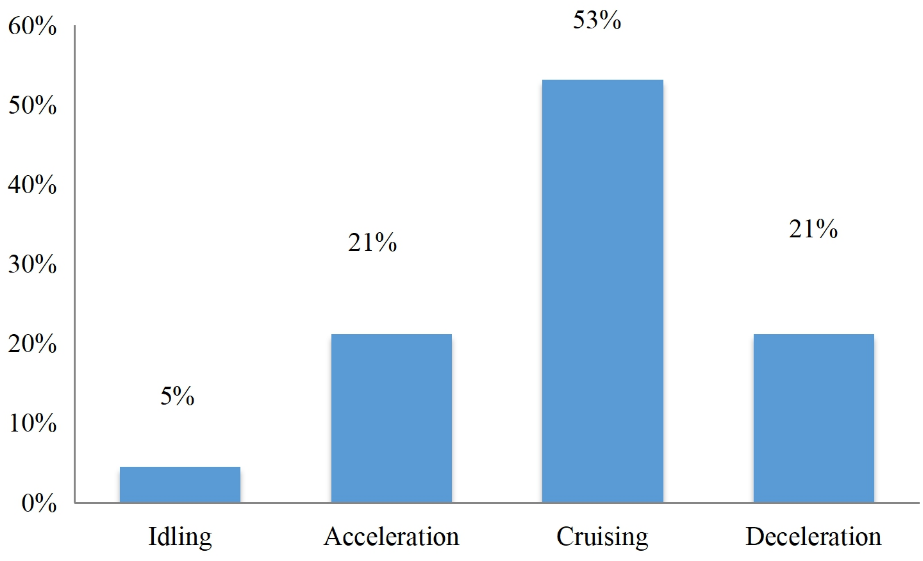


Fig 3.8: Standard Modes Distribution

3.2.2.4. Speed Distribution:

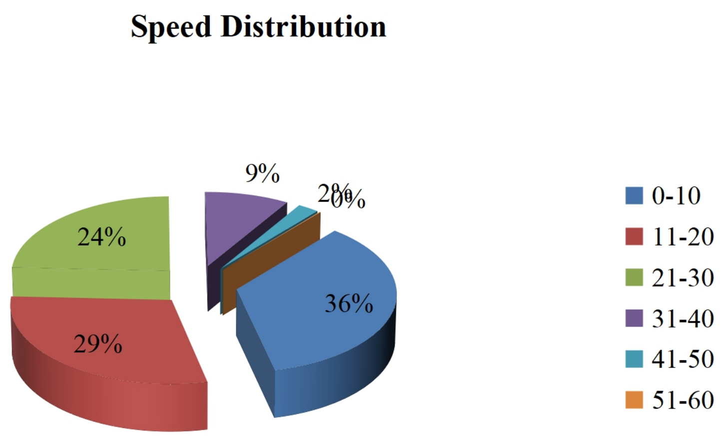


Fig 3.9: Speed Distribution

3.2.3. Second Cycle:

3.2.3.1. Speed-time profile:

`

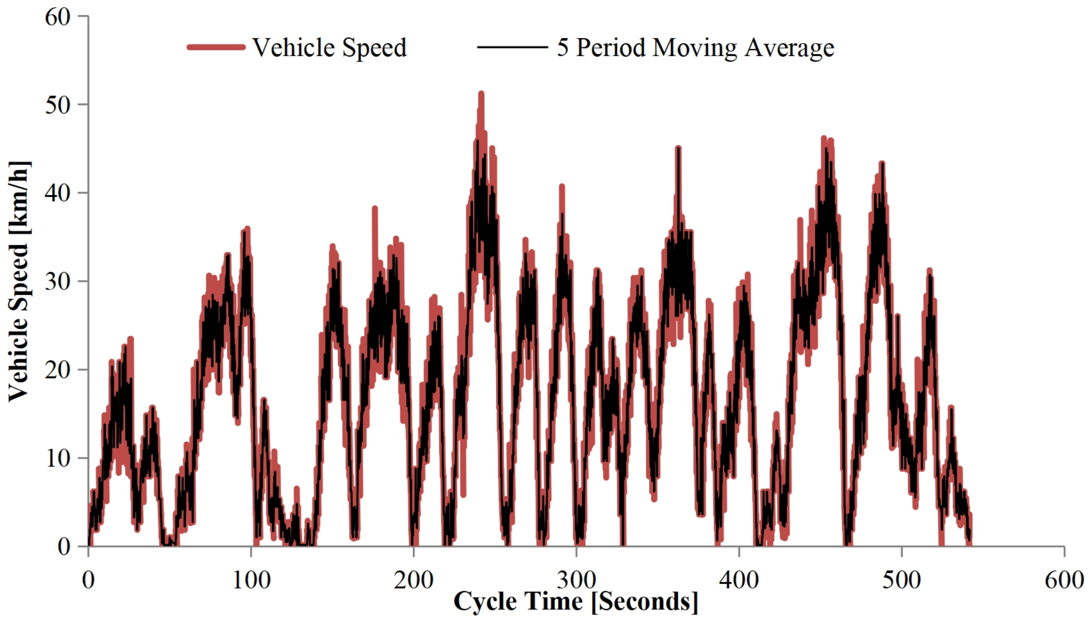


Fig 3.10: Speed-time profile

3.2.3.2. Acceleration-time profile:

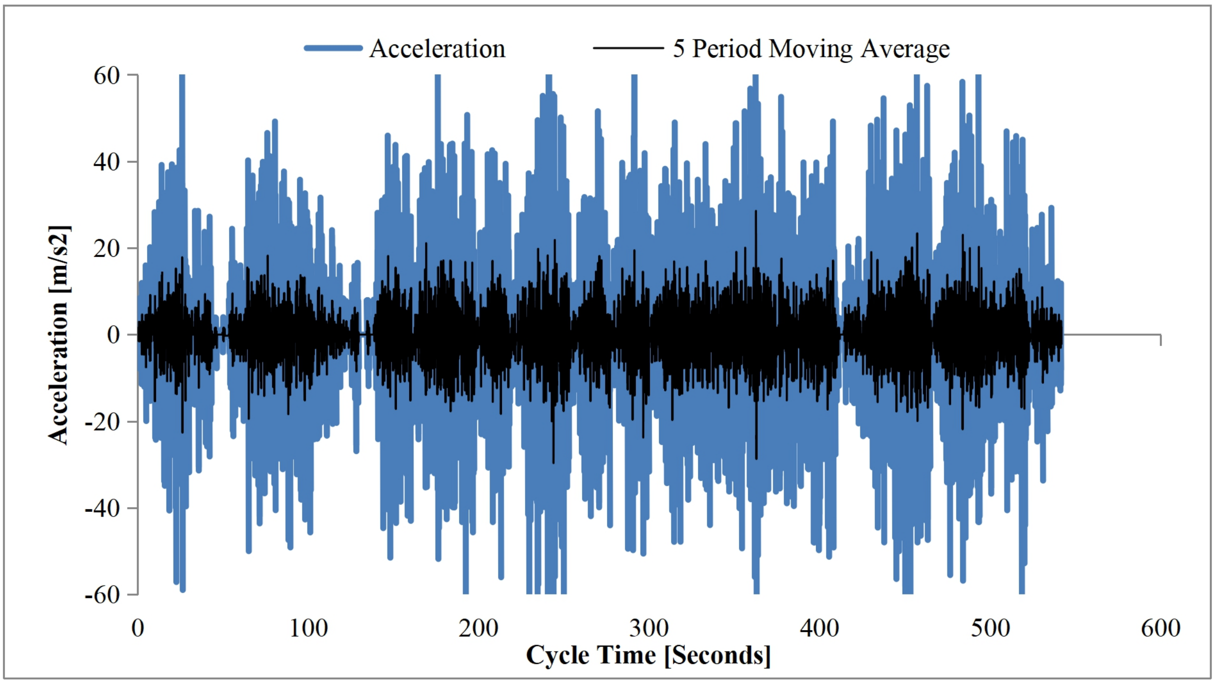


Fig 3.11: Acceleration vs time graph

3.2.3.3. Standard Modes Distribution:

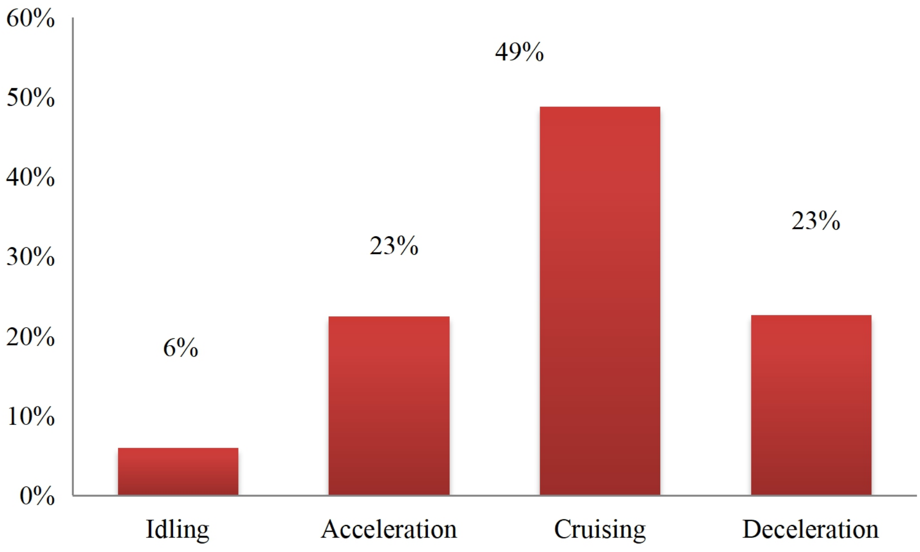


Fig 3.12: Standard Modes Distribution

3.2.3.4. Speed Distribution:

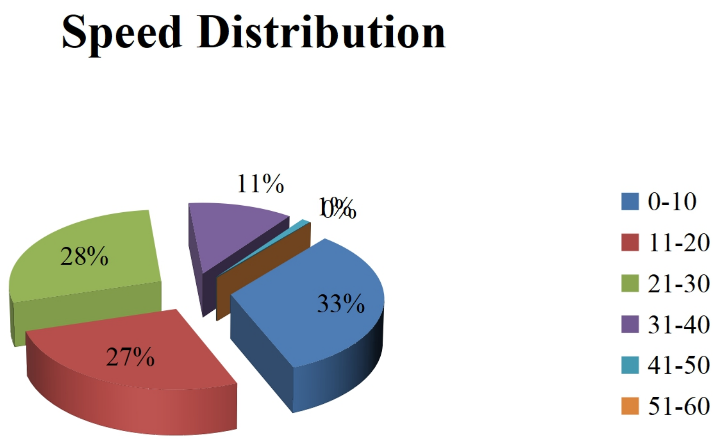


Fig 3.13: Speed Distribution

4 Conclusion and Recommendations

4.1 Conclusion:

- The analysis of speed measurement methods here covered properly. Although it was experienced that using an Expensive GPS -works best, were in quest of cost effective method of speed measuring which is at the same time an

`

efficient one. Arduino and Micontroller based methods are numerously available for speed measuring but most of them are discontinuous methods. Mobile network based data transfer to Operator with Automatic data storing method can be a special addition here.

- For building driving cycle thousands of data to be taken for a particular state, route and time. The more numbers of data the more accurate the driving cycle is.
- For a developing country like Bangladesh for growth of Automobile industry-driving cycles soon to be introduced. The target was to build a complete driving cycle to measure a correlation between speed pattern of vehicles of Dhaka City and Fuel Consumption as well as vehicle emission. Determined to further improvement of this thesis work in future with improved ideas.

`

4.2 Recommendation:

While making the add-on device the following points was determined.

- The main focused point was to build the whole system with temporary joints, so that if any problem is faced then that portion can be replaced.
- The rubber belt used was very flexible so that it can ensure to take extra tension of vibrational shock.
- There was a pivot arm which was made to absorb shaking and adjusting different height of vehicle bumpers
- The rods that were used in making this device were Galvanized Iron (GI) rods. They are heavy in weight and ensure constant contact with roads and wheels.

But situations like:

1. If the road isn't flat and with holes – speed breakers; this device may get damaged and data output will be hampered. The device during operation if faces continuous bouncing, it may not work properly
2. During turning in road corners special design consideration needed for this device.
3. If two wheel of the shaft of the device isn't horizontal, the output voltage will be distorted.
4. At very low speed it may not work properly. So slipping of belt on the wheel shaft and motor shaft needs careful-special design consideration and attention.

Above all for developing Driving Cycle a method of speed measuring which is independent of road condition and atmosphere around will be the best one. Collecting continuous data directly form vehicle wheel, from ECU or from vehicle speedometer are the best options here, as these methods are independent of Road Condition, environment, already existing system of vehicle and close to driver also.

`

References

1: Driving Cycle: https://en.wikipedia.org/wiki/Driving_cycle

2. Requirement of Driving Cycle: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/4247/ppr-354.pdf

3. Driving Cycle: http://www.sciencedirect.com/science/article/pii/0191260783901280

4. Driving Cycle: http://www.obdii.com/drivecycle.html

5. Driving Cycle: https://www.google.com/search?q=driving+cycle&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwjtr4nFt5fLAhULGY4KHd7gDMkQsAQIXA

6. Driving Cycle Abstract: http://papers.sae.org/931804/

7. Driving Cycle: https://www.dieselnet.com/standards/cycles/

8. Driving cycles: a new cycle-building method that better represents real-world emissions-by Zhen Dai*Deb Niemeier, PhD, PE, Douglas Eisinger, PhD

9. Real-world driving cycles for measuring cars pollutant emissions – Part A: The ARTEMIS, European driving cycles - Michel ANDRÉ (June 2004)

10. Development and review of Euro 5 passenger car emission factors based on experimental results over various driving cycles http://www.sciencedirect.com/science/article/pii/S0048969713010814

11.US Driving Cycle: https://en.wikipedia.org/wiki/FTP-75

12. WLTP: https://en.wikipedia.org/wiki/Worldwide_harmonized_Light_vehicles_Test_Procedures

13. NEDC: https://en.wikipedia.org/wiki/New_European_Driving_Cycle

14. Japanese JC08 Cycle: https://www.dieselnet.com/standards/cycles/jp_jc08.php

Appendix

CODE DC MOTOR

```
#include <LiquidCrystal.h>


LiquidCrystal lcd(52, 48,38,36,34,32 );

TinyGPSPlus gps;

String data="hasib12";

int flag=0;



MPU6050 mpu;




char output[100];



//prev_vel.y=0;

//prev_vel.z=0;

float dt=100,t,gps_speed=0;





void setup() {
```

`

```
   Serial.begin(9600);

   pinMode(13, OUTPUT);
   digitalWrite(A0, 0);
   digitalWrite(A1, 0);
   digitalWrite(A3, 0);    ////gnd

   Serial.end();

}

// ================================================================
// ===                MAIN PROGRAM LOOP                 ===
// ================================================================

void loop() {

  t=millis();
```

`

```
 // if programming failed, don't try to do anything

int rpm=analogRead(A3);
data[0]=char(97);
data[1]=char(rpm/4);
data[2]=char(rpm/4);
data[3]=char(rpm/4);
data[4]=char(rpm/4);
data[5]=char(rpm/4);
data[6]=char(int(dt*1000));
//data[4]=char(mz/3);
//data[5]=char(pres);

Serial.begin(9600);
Serial.println(data);
Serial.end();

   // blink LED to indicate activity
```

`

```
        dt=millis();

        dt=dt-t;

        dt=dt/1000;

}
```

CODE ACCELEROMETER AND GPS

```
#include "I2Cdev.h"

#include "MPU6050.h"

#include "HMC5883L.h"

//#include "MPU6050_6Axis_MotionApps20.h"



// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation

// is used in I2Cdev.h

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE

  #include "Wire.h"

#endif

#include <LiquidCrystal.h>

LiquidCrystal lcd(52, 48,38,36,34,32 );

String data="hasib";

// class default I2C address is 0x68

// specific I2C addresses may be passed as a parameter here

// AD0 low = 0x68 (default for InvenSense evaluation board)
```

`

```
// AD0 high = 0x69

MPU6050 accelgyro;

HMC5883L mag;

//MPU6050 accelgyro(0x69); // <-- use for AD0 high


int16_t ax, ay, az;

int16_t gx, gy, gz;

int16_t mx, my, mz;


float tx, ty, tz;

float prev_ax=0,prev_ay=0,prev_az=0;

float prev_gx=0,prev_gy=0,prev_gz=0,prev_my=0;

float prev_sx=0,sx=0;

//VectorInt16 aaReal,aa,gravity;

float prev_vel=0,vel=0,offset=0;

float dt=0,t1=0,t2=0,accl=0;


char output1[30],output2[30],output3[30],output4[30],output5[30];


// uncomment "OUTPUT_READABLE_ACCELGYRO" if you want to see a tab-separated

// list of the accel X/Y/Z and then gyro X/Y/Z values in decimal. Easy to read,

// not so easy to parse, and slow(er) over UART.

#define OUTPUT_READABLE_ACCELGYRO
```

`

```
// uncomment "OUTPUT_BINARY_ACCELGYRO" to send all 6 axes of data as 16-bit
// binary, one right after the other. This is very fast (as fast as possible
// without compression or data loss), and easy to parse, but impossible to read
// for a human.
//#define OUTPUT_BINARY_ACCELGYRO



float x ,y,z,prev_x=0;
int rpm=0,i=0,pres=0,temp=0;
int flag=0;
void setup() {



  pinMode(A0,INPUT);
  lcd.begin(20, 4);
  lcd.print("initializing....");

    // join I2C bus (I2Cdev library doesn't do this automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif
    accelgyro.initialize();
    mag.initialize();

 lcd.print("initializing....complete");
```

`

```
  /* lcd.print("initializing....");
   for(i=0;i<10;i++)
   {
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
   offset=0.5*offset+0.5*ax;
   }
   */

}

void loop() {

   // read raw accel/gyro measurements from device
   t1=millis();
  //for(i=0;i<10;i++)
 //{

  flag=flag+1;
    if(flag==5)flag=0;
//  ax=ax-offset;
   accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```

`

```
mag.getHeading(&mx, &my, &mz);
ax=0.2*prev_ax+0.8*ax+.01*(ax-prev_ax);
ay=0.5*prev_ay+0.5*ay;
az=0.5*prev_az+0.5*az;
prev_ax=ax;
prev_ay=ay;
prev_az=az;


rpm=analogRead(A0);
pres=analogRead(A1);
temp=analogRead(A2);
//rpm=rpm-800;
pres=pres/8;


x=(ax*9.81)/16384;
y=(ay*9.81)/16384;
z=(az*9.81)/16384;
float heading = atan2(my, mz);
```

`

```
//my=my+800;
heading=(heading * 180/M_PI);

if(heading < 0)
   heading += 2 * 180;
float my_1=atan2(mx,my);
my_1=(my_1*180/M_PI);
if(my_1 < 0)
   my_1 += 2 * 180;
 prev_my=my_1;

accl=x*x+y*y+z*z;
accl=sqrt(accl);
accl=accl-10;

gx=gx/16384;
gx=gx*250;
gx = gx * dt + prev_gx;
gy=gy/16384;
gy=gy*250;
gy = gy * dt + prev_gy;

//accl=accl*9810;
//accl=accl/1000;
//accl=accl*dt;
x=x+9.81*sin(((gy-prev_gy)+(my-prev_my))*.5)*dt*dt;
if(accl>1)
```

`

```
{
// if(int(x)>int(prev_x))
  // vel=prev_vel+(ax/abs(ax))*accl*dt*10;
  vel=prev_vel+(x/abs(x))*accl*dt*10;
 //if(int(x)<int(prev_x))
 //   vel=prev_vel-accl*dt*10;
}
//sx=prev_sx+vel*dt;




//tx=tx+gx*dt;


//prev_gx=gx;
//prev_gy=gy;
prev_vel=vel;
prev_gx=gx;
prev_gy=gy;
//prev_sx=sx;


   lcd.clear();
sprintf(output1, "accl:%d  vel: %d ", int(accl*10),int(vel));
lcd.setCursor(0,0);
lcd.print(output1);
sprintf(output2, "ax:%d  err: %d ", int(x),int(981*sin((gy-prev_gy)*dt*dt)));
lcd.setCursor(0,1);
```

`

```
lcd.print(output2);
sprintf(output3, " mx:%d my:%d ", int(heading),int(my_1));
lcd.setCursor(0,2);
lcd.print(output3);
//sprintf(output, " RPM:%d  pres:%d", int(rpm),int(pres));
sprintf(output4, " rpm:%d gy:%d ", int(rpm),int(gy));
lcd.setCursor(0,3);
lcd.print(output4);


//sprintf(output,"%d%d%d%d",int(97),int(accl),int(vel),int(gx));
data[0]=char(97);
data[1]=char(accl+50);
data[2]=char(vel+120);
data[3]=char(gx+128);
data[4]=char(rpm);
data[5]=char(pres);
sprintf(output5, " gx:%d gy:%d mz:%d ", int(accl),int(vel),int(x));
Serial.begin(9600);
Serial.println(output5);
Serial.end();
Serial1.begin(9600);
Serial1.println(data);
Serial1.end();
 // blink LED to indicate activity


 //delay(100);
 prev_x=x;
```

`

```
  t2=millis();

  dt=(t2-t1)/1000;

}
```

CODE PC SOFTWARE (IN PYTHON)

```
#from ser_w import *

#import bluetooth
```

`

```
import sys

import serial

import numpy as np

import time

import cv2

from back_sub import *

#from simulation import simulation

#from matplotlib import pyplot as plt


fh = open("final1.csv","w+")


frame=np.ones((480,640,3),np.uint8)

flag=1

cap=cv2.VideoCapture(0)


acceleration=0

gps=0

h=800

w=1400

graph=np.ones((h,w,3),np.uint8)

graph=graph*255

win=np.ones((800,1600,3),np.uint8)

win=win*255

black=np.ones((300,400,3),np.uint8)

font = cv2.FONT_HERSHEY_SIMPLEX


#graph=graph*125
```

`

```
#graph[0:h,0:w]=0

ser = serial.Serial('COM8', 9600, timeout=0)



time.sleep(.5)
cx=0
cy=0
t2=0
i=0
speed=4
#try:ret,frame=cap.read()
#except:0

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
#gray=cv2.resize(gray,(160,120))
prev_fmm=0
bytes=''
x="hasib12"
x1="hasib12"
gps1=30
cmd=0
num=1
#sim=simulation()
prev_acceleration=0
prev_gps=0
```

`

```
prev_rpm=0
t3=0
t1=time.time()*1000
while(1):

 cmd=cv2.waitKey(6)

 if cmd==113:
    ser.close()
    cv2.destroyAllWindows()
    break

 Y=chr(int(cx/2.5))

 Z=chr(int(cy/2.5))

 #sim.move(0,0,0)
 #bytes[0:1]=X
 #Y=chr(65)
```

`

```
#sock.send(X)






if ser.inWaiting()>100:
   ser.flushInput()
#    ser.flushOutput()
if ser.readable():
    x1=ser.readline()
    #print len(x1)
    if len(x1)==9:
      x=x1
    #if ser.write(chr(65)):
    #  if ser.write(Y):
    #    if ser.write(Y):
    #      ser.write(Z)


    k=0
    rpm=0
    for c in x :
         if k==0:
           if ord(c)!=97:
               print ord(c)
               break
```

`

```
    if k==1:
      acceleration=ord(c)
    if k==2:
      gps=ord(c)
    if k==3:
      rpm=ord(c)
    if k==4:
      gps=ord(c)
    if k==5:
      vel1=ord(c)
    if k==6:
      dt=ord(c)



    k=k+1
#if gps1>20 and gps1<35:
#   gps=gps1
#gps1=gps1*2

acceleration1=acceleration
rpm1=rpm
#print len(x1),gps
```

`

```
    i=i+1

    if i==w-1000:
       i=i-1

       graph[0:h,0:(w-1000)-1]=graph[0:h,1:w-1000]
       graph[0:h,(w-1000)-1:w-1000]=255

    #
    #acceleration=acceleration-97
    #gps=gps+15
    acceleration=acceleration-50
    gps=gps-120
    rpm=rpm*4

    cv2.line(graph,(i-1,int(prev_acceleration)+100),(i,int(acceleration)+100),(255,0,0))
    cv2.line(graph,(i-1,int(prev_gps)+350),(i,int(gps)+350),(0,255,0))
    cv2.line(graph,(i-1,int(prev_rpm/2)+500),(i,int(rpm/2)+500),(0,0,255))

    cv2.line(graph,(600-int(100*np.cos(prev_rpm)),500-
int(100*np.sin(prev_rpm))),(600+int(100*np.cos(prev_rpm)),500+int(100*np.sin(prev_rpm))),(255,2
55,255))
    cv2.line(graph,(600-int(100*np.cos(rpm)),500-
int(100*np.sin(rpm))),(600+int(100*np.cos(rpm)),500+int(100*np.sin(rpm))),(125,125,125))
    cv2.circle(graph,(600,500),100,(255,0,0))

cv2.circle(graph,(600+int(100*np.cos(prev_rpm)),500+int(100*np.sin(prev_rpm))),10,(255,255,255))
    cv2.circle(graph,(600+int(100*np.cos(rpm)),500+int(100*np.sin(rpm))),10,(0,0,255))
    cv2.circle(graph,(900,500),100,(100,100,100),-1)
```

`

```
cv2.line(graph,(900,500),(900+int(100*np.cos(prev_gps/20)),500+int(100*np.sin(prev_gps/20))),(25
5,255,255))

cv2.line(graph,(900,500),(900+int(100*np.cos(gps/20)),500+int(100*np.sin(gps/20))),(125,125,125))
    cv2.circle(graph,(900,500),100,(125,125,125),5)



    #cv2.circle(graph,(500-int(100*np.cos(rpm)),500-
int(100*np.sin(rpm))),(500+int(100*np.cos(rpm)),500+int(100*np.sin(rpm))),(125,125,125))



    #cv2.line(graph,(i,550),(i,int(cy/8)+550),(125,125,0))
    #cv2.line(graph,(0,100),(1600,100),(0,0,0),2)
    #cv2.line(graph,(0,128+150),(1600,128+150),(0,0,0),2)


    graph[30:200,600:800]=(0,0,0)
    cv2.putText(graph,'heading%d '%prev_rpm,(600,50), font,.5,(255,255,255),1)
    cv2.putText(graph,'heading%d '%rpm,(600,70), font,.5,(255,0,0),1)
    cv2.putText(graph,'vel%d km/h'%int(prev_gps/10),(600,90), font,.5,(255,255,255),1)
    cv2.putText(graph,'vel%d km/h'%int(gps/10),(600,110), font,.5,(255,0,0),1)
    cv2.putText(graph,'time%d km/h'%t3,(600,130), font,.5,(255,0,0),1)


    prev_acceleration=acceleration
    prev_gps=gps
    prev_rpm=rpm
```

`

```
#cv2.putText(graph,'vel%d deg'%gps,(300,670), font,.5,(0,255,0),1)
#cv2.putText(graph,'rpm%d'%rpm,(300,350), font,.5,(0,0,255),1)
#cv2.putText(graph,'cmnd%d'%cmd,(300,370), font,.5,(255,0,255),1)
#cv2.putText(graph,'pos_x%dm'%cx,(300,390), font,.5,(125,125,125),1)
#cv2.putText(graph,'pos_y%dm'%cy,(300,410), font,.5,(125,125,0),1)
#cv2.putText(graph,'acceleration',(300,20), font,.5,(255,0,0),1)
#cv2.putText(graph,'gpserature',(300,130), font,.5,(0,255,0),1)
#cv2.putText(graph,'rpm level',(300,270), font,.5,(0,0,255),1)
win[0:h,0:w]=graph
#win[0:480,w:1040]=frame
## win[480:600,w:w+160,0]=fm
## win[480:600,w:w+160,1]=fm
## win[480:600,w:w+160,2]=fm
cv2.rectangle(win,(w,480),(w+160,600),255,3)
#plt.imshow(win,'graph')
#plt.show()
cv2.imshow('graph',win)
cv2.waitKey(1)
t2=time.time()*1000

strn=str(num)+'.jpg'
if cmd==65:
   cv2.imwrite(strn,graph)
   num=num+1
   print 8767547676475876472
   cv2.waitKey(0)
```

`

```
    t3=t2-t1
    if rpm>5:
     fh.write("%d,"%int(t3))
     fh.write("%d,"%int(acceleration)) #accl
     #fh.write("%d,"%int(gps+11))
     fh.write("%d,"%int(gps+11)) #vel
     fh.write("%d,"%int(rpm)) #rpm
     fh.write("%d,"%int(gps))#gps
     fh.write("%d,"%int(vel1-120))
     fh.write("%d\n"%int(dt))



ser.close()
fh.close()
cv2.destroyAllWindows()
```