Virtual Sensor For Automotive Engine To Compensate For The Map, Engine Speed Sensors Faults

By

Sohaub S.Shalalfeh Ihab Sh.Jaber Ahmad M.Hroub

Supervisor:

Dr. Iyad Hashlamon

Submitted to the College of Engineering

In partial fulfillment of the requirements for the degree of

Bachelor degree in Mechatronics Engineering

Palestine Polytechnic University

March- 2016

Palestine Polytechnic University

Hebron –Palestine

College of Engineering and Technology

Mechanical Engineering Department

Project Name

Virtual sensor for automotive engine to compensate for the map, engine speed sensors faults

Project Team

Sohaub S.shalalfeh Ihab Sh.Jaber Ahmad M.Hroub

According to the project supervisor and according to the agreement of the testing committee members, this project is submitted to the Department of Mechanical Engineering at College of Engineering in partial fulfillments of the requirements of the Bachelor’s degree.

Supervisor Signature

…………………………..

Committee Member Signature

……………………… ……………………….. ……………………

Department Head Signature

…………………………………

Dedication

To our parents.

To all our teachers.

To all our friends.

To all our brothers and sisters.

To Palestine Polytechnic University.

Acknowledgments

We could not forget our families, who stood by us, with their support, love and care for our whole lives; they were with us with their bodies and souls, believed in us and helped us to accomplish this project.

We would like to thank our amazing teachers at Palestine Polytechnic University, to whom we would carry our gratitude our whole life. Special thanks to our Supervisor Dr. Iyad Hashlamon.

Chapter 4: EXPERIMENTAL AND RESULTS

Chapter 5: Improving For System And Final Result

List of Tables

List of Figures

Table1: Values For Mean Value Engine Model

symbol	description	Unit
Pa	**Ambient pressure**	**101325 pa**
Pm	**Manifold pressure**	**Pa**
$\bar{p}\, m$	**Manifold pressure by virtual sensor**	**Pa**
Tm	**Manifold temperature**	**325k**
Ta	**ambient temperature**	**298k**
α_{cl}	**Throthle angle at close position**	**9.8 degree**
α	**Throthle angle**	**degree**
A_E	**Throthle effective area**	**m^2**
D	**Inlet diameter**	**0.054m**
R	**Specific gas constant**	**0.287J/kg.k**
γ	**Ratio heat of capacities**	**1.4**
V_d	**Displaced volume**	**0.0172409362** m^3
V_m	**Manifold volume**	**0. 1127** m^3
η_{vol}	**Volumetric efficiency**	
ω	**Engine speed**	**rad/sec**
ϖ	**Engine speed by virtual sensor**	**rad/sec**
C_D	**Throttle discharge coefficients**	**0.004**

Abstract

Most of the functions of a modern car directly depend on reading sensors spread over parts of the engine to adjust the efficiency and performance of the engine.These sensors have an important piece of information for the control unit to build a signal that could control the actuator . If the MAP sensor is defectd this will affect the performance of the engine and air fuel ratio, but if the RPM sensor is defect then the vehicle and engine will not start.

To solve this problem we use the virtual sensor principle. The proposed scheme assists in: Sensing of critical unmeasurable parameters like Volumetric and Combustion efficiency, developming of Virtual Sensor from manifold pressure dynamics for rotational speed sensor. Health Monitoring of the manifold pressure and crankshaft sensor. For the suggested scheme, two robust second-order sliding mode observer are employed that require two dynamic equations engine model based on inlet manifold pressure and rotational speed dynamics. The proposed methodology has the potential of online implementation on any automotive engine after minor tuning. This procedure is customized for 1.4 L gasoline engine sensors: Manifold Pressure and Crankshaft. The observer dynamic equation is simulated on matlab using simulink tools and compared with the real time response.

CHAPTER ONE

INTRODUCTION

1.1 Introduction:

Old car is controlled mechanically. With the development of the world and the increasing demand, the car has become an electromechanical system .Its controlled where this control achieved for driver safety, comfort, reduced fuel consumption, more engine performance, and low emissions ..

Vehicles are now computerized machines. This fact has had an enormous effect on possibilities for functionality of vehicles, which together with the needs and requirements from customers and from society have greeted vigorous activities in development.

The modern vehicles contain sensors, actuators and electronic control module (ECM) where the sensors convert physical quantities (pressure, mass, acceleration) into output signal (usually electrical) to the electronic control module to be processed and sent to actuators.

Modern engines utilize an electronic engine control module (ECM) to continuously monitor and control the engine operation to optimize fuel economy, emissions control, and performance. The ECM uses various physical sensors to collect information reflecting current operating conditions. The information is used to generate output signals for various actuators which control the operation of the engine. Using the actuators, the ECM controls the air-fuel ratio, fuel injection, ignition timing, and various other functions to control operation of the engine. Optimal control of the engine over a wide range of engine operating conditions (and ambient conditions) depends on the availability, accuracy, and reliability of data gathered by the engine sensors ..

When a defect occurs in any sensor in the engine it affects the efficiency and performance of the engine. One way to increase the performance of the engine is to increase the use of fault diagnosis system and model based diagnosis.

virtual sensor system will be developed for the air intake manifold system. The air intake manifold contains a number of components and sensors related by reasonably well under stood thermodynamic and physics of gases. The diagnosis system is based on nonlinear semi physical model and use combination of different residual generation methods.

The sensors effect on vehicles in intake manifold are the pressure manifold sensor, engine speed sensor and throttle position sensor. When one is defected the vehicles cannot work then stop,To find a solution and provide comfort To the driver, We will use nonlinear observers to develop fault diagnosis methodologies.These observers provide a facility of virtual sensors development.

1.2 Related work

Virtual sensors of tire pressure and road friction by Fredrik Gustafson and Evaluation of a sliding mode observer for vehicle sideslip angle by Elsevier in Control Engineering Practice , Vehicle Yaw Control via Second-Order Sliding-Mode Technique. we workVirtual sensor for automotive engine to compensate for the map, engine speed sensors faults.

1.3 Motivation

The application of the idea of the virtual sensor in the automotive world is to save time and effort on the driver and the possibility of application in other areas of the car, for example,air bag ,tire pressure system ,side slipe and yaw sensor.

The motivations can be made clear by answering the following question: Why use higher order sliding mode observer ?

In engineering systems, some parameters are difficult to measure or some time even not measurable due to certain reasons including inaccessibility by virtue of harsh environment. Another problem is regarding unknown inputs to the system. A well established solutionfor observation of system states, estimation of parameters, and reconstruction ofunknown inputs is model based observer in modern control theory. An observeris a mathematical dynamical framework that estimates unknown states using certainsystem measurements.

1.4 Problem definition:

Air Intake Manifold system contains a sensors, Any malfunction in the sensors will deffect the efficiency of intake system .The Manifold Absolute Pressure (MAP) sensor is a key sensor because it senses engine load. The sensor generates a signal that is proportional to the amount of vacuum in the intake manifold. The engine computer then uses this information to adjust ignition timing and fuel enrichment.

When the engine is working hard, intake vacuum drops as the throttle opens wide. The engine sucks in more air, which requires more fuel to keep the air/fuel ratio in balance. In fact, when the computer reads a heavy load signal from the MAP sensor, it usually makes the fuel mixture go slightly richer than normal so the engine can produce more power. At the same time, the computer will retard (back off) ignition timing slightly to prevent detonation (spark knock) that can damage the engine and hurt performance.

The Throttle Position sensor measures the throttle position, which is controlled by the gas pedal. It is used to determine engine load and if it fails it can cause automatic transmission shifting problems. It is used by the vehicle's computer to control engine performance by increasing the amount of fuel delivered as the throttle opens.

1.5 Solution methodology:

The solution is based on development of model based fault diagnosis for air intake system. The input and output of the actual system are provided to the developd model. The comparison of the actual system output and the model output help in generating residuals. analytical model technique is used for fault diagnosis. This model relies on dynamical model of each component ie actuator, sensor and process.The dynamical model is utilized to generate residuals based on the same input and output of the actual system to be diagnosed. The last stage involves the evaluation of the generated residuals to detect, quantify and isolate faults.

Parameter estimation is a powerful analytical model based fault diagnosis technique. Sliding mode techniques and Kalman Filter are popular parameter estimation techniques:Sliding mode techniques have been extensively used for estimation and control. The actual system measurements are tracked using sliding mode observer. The uncertainty is modeled under the sliding mode injectors.

1.6 Time table: Time- schedule for secand semester

Week / Process	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Find automotive Engine	■	■	■	■												
Buying automotive				■	■											
Find constant and Engine properties				■	■	■	■									
Starting to loading equations matlab Simulink						■	■	■								
Run Simulation and tuning work								■	■	■						
Buying micro controller electronic part									■	■	■	■				
Real Time connect And take Results												■	■	■		
Writing documentation													■	■	■	
Final presentation																■

CHAPTER TWO

MANIFOLD ENGINE MODELING

MANIFOLD ENGINE MODELING

2.1 Intake manifold dynamics:

In automotive engineering, an inlet manifold or intake manifold is the part of an engine that supplies the fuel/air mixture to the cylinders. The primary function of the intake manifold is to evenly distribute the combustion mixture (or juklst air) to each intake port in the cylinder head(s). Even distribution is important to optimize the efficiency and performance of the engine[1].

The intake manifold is a dynamic environment that can be modeled using several sets of equations and empirical data models. The overall intake manifold model begins with the throttle valve, where air enters the intake manifold, and ends with the intake valves, where air exits the intake manifold and enters the combustion chamber. Figure(2.1) shows the layout of the engine intake system. The throttle valve is used to determine how much air flow passed the throttle, instead of measuring the air directly with the Manifold Air Flow sensor (MAF).

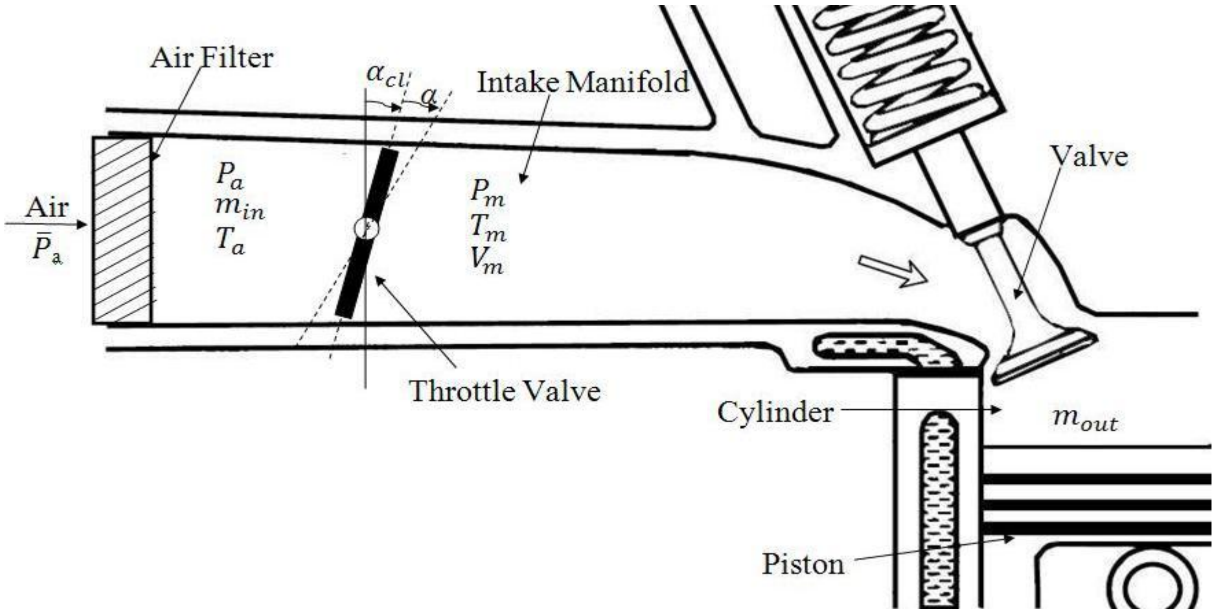


Figure 2.1:The layout of the engine intake system

We get pressure from the following relationship equation(2.1)

$$p_m v_m = mRT_m \tag{2.1}$$

p_m :The Manifold pressure in Pascal (Pa),

v_m The Manifold volume in(m^3),

T_m TheManifold temperature in (K),

R :The Specific gas constant in(J/kg .K,)

m : the mass of the air in (kg).

2.2 Manifold Pressure

The manifold air pressure sensor, Figure 2.2, is used to indicate engine load.It provides indirect measurement for mass of air entering cylinder or the mass air leaving intake manifold.

The difference in the incoming and outgoing mass flow rates represents the net rate of change of air mass with respect to time,Figure 2.3 shows the relationship between Map and Throttle sensors. This quantity, according to the ideal gas law, is proportional to time derivative of the manifold pressure which is writtin as

$$\dot{p}_m = \frac{RT_m}{V_m}(\dot{\mathrm{m}}_{in} - \dot{m}_{out}) \qquad (2.2)$$

Equation(2.2)can be extended as

$$\dot{p}_m = \mathrm{A}_1.\mathrm{A}(\alpha).f\ (\mathrm{P}_m) - \mathrm{A}_2\mathrm{P}_m \qquad (2.3)$$

Where the

A(α): is the throttle effective area in m^2 .

α: is the throttle angle in degrees ($^\circ$).

D: is the diameter of the inlet pipe in meters (m).

α_{cl} : is the throttle angle at close position in degrees($^\circ$).

T_a :Ambient Temperature in kelvin(K)

$$\mathrm{A}_1 = \frac{R}{V_m T_m} C_d P_a \gamma_c \tag{2.4}$$

$$\mathrm{A}_2 = \frac{v_d \omega \eta_{vol}}{4\pi V_m} \tag{2.5}$$

$$\gamma_c = \sqrt{\frac{1}{RT_a} . \gamma (\frac{2}{\gamma + 1})^{\frac{\gamma+1}{\gamma-1}}} \tag{2.6}$$

$$f\ (\mathrm{P}_m) = (1 - e^{(9(\frac{\mathrm{P}_m}{P_a} - 1))}) \tag{2.7}$$

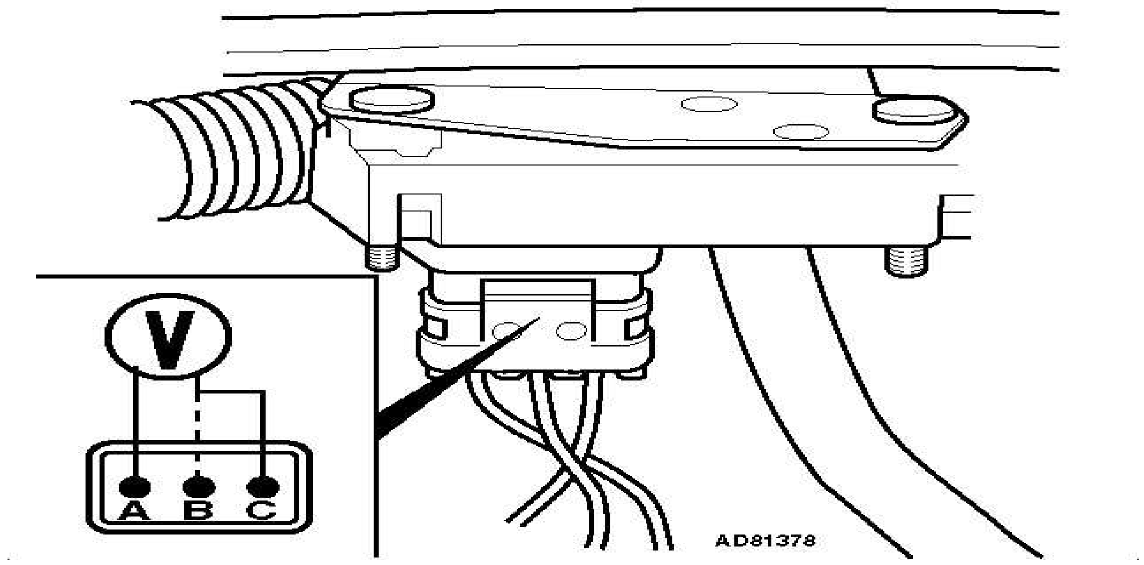


Figure 2.2 :Location of Map sensor

The relationship between intake manifold pressure, throttle angle, and air flow rate is shown in Figure 2.3.

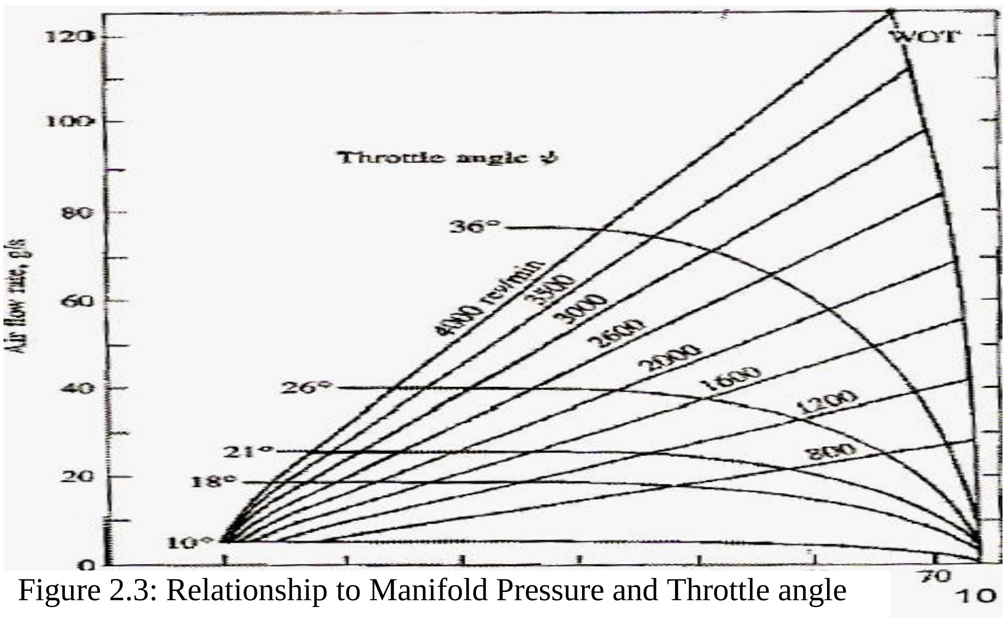


Figure 2.3: Relationship to Manifold Pressure and Throttle angle

2.3 Throttle:

The throttle is a valve, Figure 2.4, that directly regulates the amount of air entering the engine, indirectly controlling the charge (fuel + air) burned on each cycle due to the fuel injector or carburetor maintaining a relatively constant fuel/air ratio. In automobiles, the control used by the driver to regulate power is sometimes called the throttle pedal accelerator. An engine's power can be increased or decreased by the restriction of inlet air.

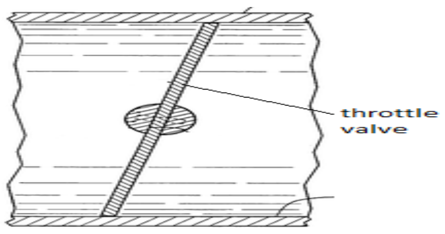


Figure 2.4: Throttle valve

When the driver presses the accelerator pedal, the throttle passage opens to allow more air into the intake manifold[2].

Usually an airflow sensor measures this change in flow of air by measuring the change in the pressure and communicates to the Electronic control unit(ECU). The ECU then increases the amount of fuel being sent to the fuel injectors in order to obtain the desired air-fuel ratio. Often a Throttle Position Sensor (TPS) is connected to the shaft of the throttle plate to provide the ECU with information on whether the throttle is in the idle position, Wide Open Throttle (WOT) position, or somewhere in between these extremes. The effective throttle area can be expressed in simple mathematical form as Equation(2.8)

$$A(\alpha) = \pi \frac{D^{2}}{4} (1-\cos(\frac{\alpha+\alpha_{cl}}{\alpha_{cl}})) \tag{2.8}$$

2.4 Engine speed sensor (rpm)

The rpm sensor is also known as a crank sensor or an engine speed sensor. A crank sensor is an electronic part,shown in Figure 2.5, which is used in an internal combustion engine to monitor the position or rotational speed of the crankshaft. This information is used by engine management systems to control ignition system timing and other engine parameters. Before electronic crank sensors were available, the distributor would have to be manually adjusted to a timing mark on the engine.

The crank sensor can be used in combination with a similar camshaft position sensor to monitor the relationship between the pistons and valves in the engine, which is particularly important in engines with variable valve timing. This method is also used to "synchronizes" a four stroke engine upon starting, allowing the management system to know when to inject the fuel. It is also commonly used as the primary source for the measurement of engine speed in revolutions per minute.

Common mounting locations include the main crank pulley, the flywheel, the camshaft or on the crankshaft itself. This sensor is the most important sensor in modern day engines. When it fails, there is a chance the engine will not start, or cut out while running.

There is also a relationship between Map, speed and the Throtlle degree as show in Table 2.1

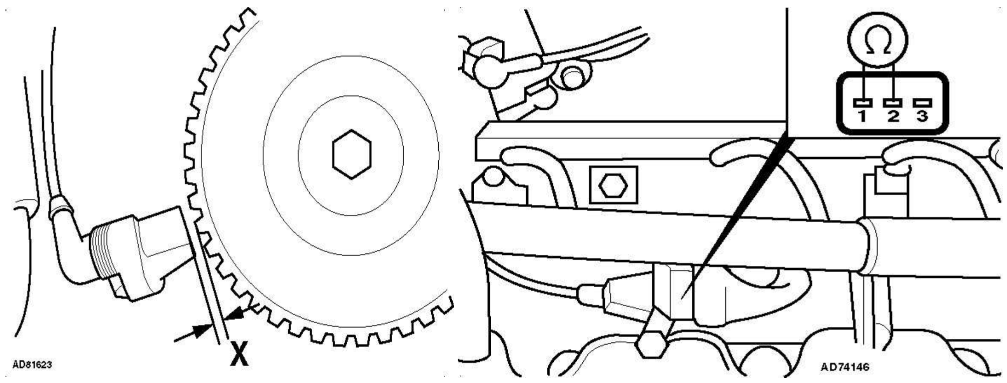


Figure 2.5: Cranck shaft postion sensor

To make sure that project works perfectly, The throttle angle and presure must be entered to obtain the speed and it must be very close to predefined speeds for example take the value of throttle angle in table (2.1) 32 degree at speed 1500 rpm we must the out put matlab take 760 mbar.

Table 2.1: Reading between the throttle position sensor and rpm sensor and MAP sensor.

Angle\Speed	500	1000	1500	2000	2500	3000	3500	4000
22	850	620	435	350	275	225	200	180
32	965	870	760	645	550	480	425	413
42	965	950	870	827	770	710	680	620
52	1000	940	960	900	860	800	780	750
62	1000	996	980	940	920	880	855	828
72	1000	990	975	965	910	890	870	850

Where the:

Speed: is speed of the engine in (rpm).

Pressure: is pressure of intake manifold in (mbar).

Angle :is angle of throttle plate in ($^\circ$) .

2.5 Volumetric Efficiency

Once air is beyond the throttle and the calculated mass air flow(M_{In}) is known,The performance parameter, volumetric efficiency can then be used to determine how much air is exiting the intake manifold into the combustion chambers. Volumetric efficiency can be defined as the volume flow rate of air in the intake manifold divided by the rate at which volume is displaced by the piston . Volumetric Efficiency is the measurement of how close the actual volumetric flow rate is to the theoretical volumetric flow rate. Mathematically, it can be defined as

$$\eta_{vol} = \frac{\dot{m}_{out}\text{ (actual)}}{\dot{m}_{out}\text{ (ideal)}} \qquad (2.9)$$

Where

η_{vol} :Volumetric Efficiency.

$\dot{m}_{out(actaul)}$:is the actual air mass flow rate leaving the manifold.

$\dot{m}_{out\,(ideal)}$:is the ideal air mass flow rate leaving the manifold**.**

CHAPTER THREE

AIR INTAKE SYSTEM PARAMETER ESTIMATION

AIR INTAKE SYSTEM PARAMETER ESTIMATION

This chapter deals with the estimation of speed engine based on the model given in the previous chapter. This estimation provides credible knowledge about engine performance and can be efficiently used for health monitoring of gasoline engine.

3.1 Fault

Is unacceptable deviation of at least one property or parameter of the system from its usual behavior. The faults can occur in each component of the system.

Consider the system *n-th* order continous time system

$$x(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t)$$

where,

x $\in R^{n}$ is the system state vector.

u $\in R^{l}$ is the control input vector. (3.1)

y $\in R^{p}$ is the output vector.

A $\in R^{(n\times n)}$ is the system matrix.

B $\in R^{(n\times l)}$ is the input distribution matrix.

$C \in R^{(p\times n)}$ is the output distribution matrix.

Based on these components the faults can be classified as:

System Faults: faults occur by change in the system itself. Any change in system matrix $\triangle A$ reflects itself in its parameters matrix(A).

$$x(t) = (A + \triangle A)x(t) + Bu(t) \tag{3.2}$$

Actuator Faults: Actuator faults B^{f} correspond to the variation of the control input u(t) applied to the system.

$$x(t) = Ax(t) + B^f u(t) \tag{3.3}$$

Sensor Faults : Faults C^f that appear in the installed sensors are termed as Sensor Faults

$$y(t) = C^f x(t) \tag{3.4}$$

Sensor fault type

1. Degraded Efficiency: In this case,sensor will be showing false values but the Changing trends will be same as the system behavior. Mathematically,loss in Accuracy/degraded efficiency of sensor can be explained as

$$y = c(1 - f_{loss})x \tag{3.5}$$

Where f_{loss} : represents the degraded efficiency.

c : contant number.

2.Freeze

In this case, the measurements taken from installed sensors will Contain either '0` in case of fully damaged or any other value which will Remain constant through the system process. Freezing nature of sensor can be modeled as

$$y = f_{freeze} \tag{3.6}$$

Where f_{freeze} : is constant value that sensor is sharing with the processor.

3. Drift: If the sensor is showing increasing values of the measured state With time, this behavior is termed as drift and it can be modeled as

$$y = f_{drifts}(t)cx \tag{3.7}$$

Where f_{drifts} : is time dependant factor by which the value is drifting from its original value.

4.Bias: The sensor measurements sometimes show biased value from The true Value, this can be modeled as

$$y = f_{bais} + cx \tag{3.8}$$

Where f_{bais} : is the bias added in the sensor readings.

3.2 Fault Diagnsis

Optimum performance of engineering systems heavily relies on the proper functionality of The sub-systems. Each sub-system should be precisely monitored for smooth operation of any engineering system. This motivation lead to the development of various fault diagnosis methodologies as shown in Figure 3.1

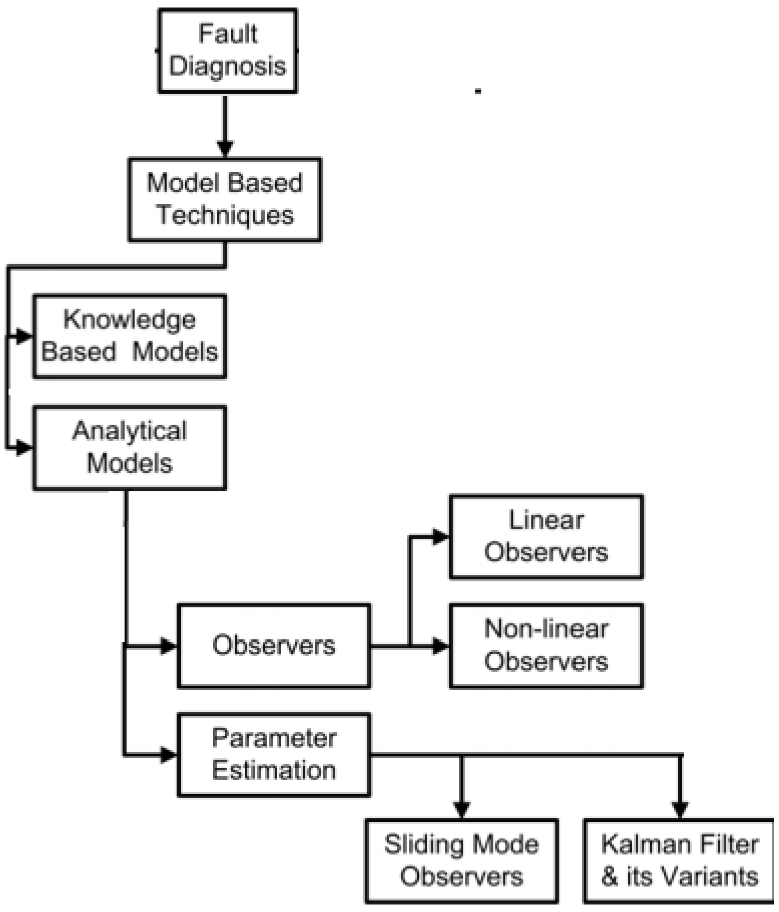


Figure 3.1 Classification of fault diagnosis methodologies

3.3 Model based

Fault diagnosis scheme utilizes an underlying mathematical model that operates in parallel with real time system to monitor system health. The inputs and outputs of the actual system are provided to the developed model[5]. The comparison of the actual system outputs and model outputs helps in generating residuals. These residuals are further explored to analyze and reconstruct faults. Figure 3.2 shows a brief idea about the model-based fault diagnosis schemes.

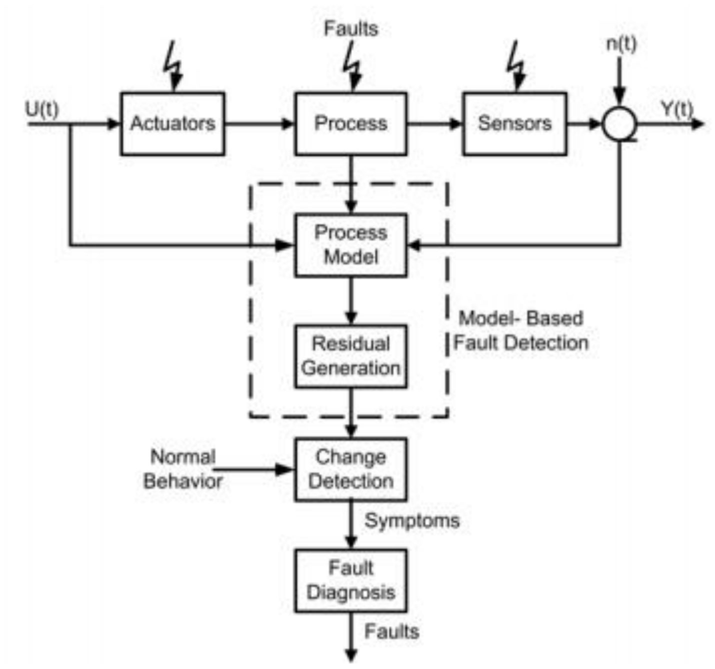


Figure 3.2: Model based fault diagnosis methodology

Following are the popular model based fault diagnosis techniques

Analytical Models:

The analytical models are popular tools for fault diagnosis in control systems community. This model based fault diagnosis scheme relies on a dynamical model of the system developed on the basis of physical laws. Each component i.e. actuators, sensors and process of the system is dynamically modeled. The dynamical model is then utilized to generate residuals based on the same inputs and outputs of the actual system to be diagnosed. The last stage involves the evaluation of the generated residuals to detect, quantify and isolate faults. Some of the popular analytical model based fault diagnosis techniques are:

1. Diagnostic Observers:

Diagnostic observers are employed to reconstruct all of the system states from the available output measurements. Linear diagnostics observers are extensively used to reconstruct the faults and diagnose the system health.

But a linear observer have a lot of limitation and can’t deal with nonlinear system which makes fault diagnosis of highly nonlinear systems very difficult

2. Parameter Estimation:

Estimation of critical system parameters is very useful for the diagnosis of engineering systems. A system whose dynamics are known and time varying parameter need to be estimated can be diagnosed by estimating its health indicating parameters. Figure 3.3 gives a brief idea about the parameter estimation methodology. The input (U(t)) and output (Y(t)) are supplied to parameter estimation technique.

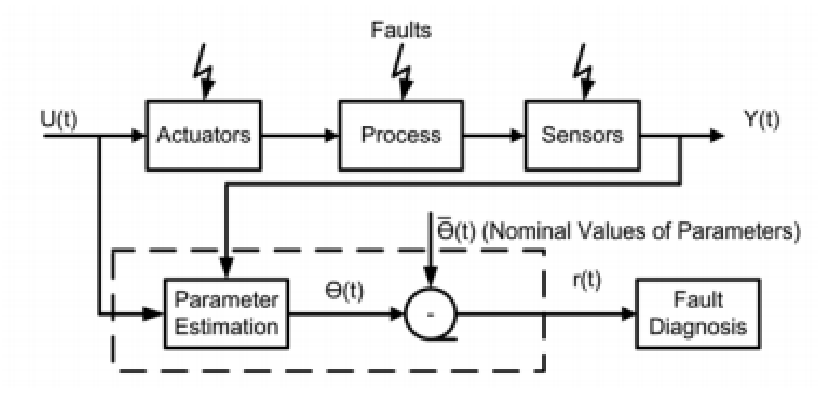


Figure 3.3: Parameter Estimation based Fault Detection

Some popular Some popular parameter estimation techniques are: **Kalman Filter & its variants**: Kalman filter is extensively used for state and parameter estimation of engineering systems. . Standard Kalman filter requires linear model of the underlying system, initial measurement and process noise distribution and parameter/state estimates. Based on this information, the algorithm first predicts the estimates and rectifies the estimates using the measurements and Kalman filter gain. As standard Kalman fillter is limited to linear system, Extended Kalman Filter (EKF), being used for estimation of nonlinear system by linearizing it
Sliding mode techniques:

Sliding mode techniques are very powerful tool to estimate the parameters of uncertain systems. Figure (3.4) explains the sliding mode observer being used for parameter estimation. The actual system measurements are tracked using sliding mode observer. The uncertainty is modeled under the sliding mode injectors. These robust injectors eliminate the difference with actual and observed output. The detailed error analysis helps in estimating critical parameters being used to diagnose the faults of the system. Sliding mode techniques have been extensively used for estimation and control.

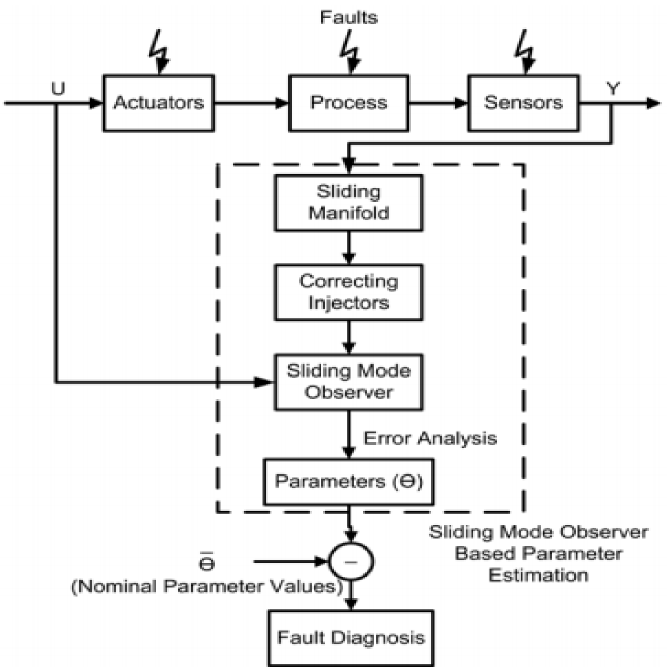


Figure 3.4: Sliding mode observer parameter estimation based fault diagnosis

Benefits of Parameter Estimation:

There are certain system parameters that directly correspond to system health status. Most of the times, these parameters are measurable by installing sensors to sense the respective phenomena. However, in some cases it is almost impossible to use a sensor to assess particular parameter for health diagnosis. In case of control and diagnosis scheme development, precise values of un-measurable parameters are required to develop an accurate mathematical model of a dynamical system. Sometimes, it is required to create redundancy for the installed sensor in order toobey strict safety by laws. These problems can be resolved by employing state-of-art

estimation techniques. The significant benefits of parameter estimation are:

1. Efficient health monitoring can be done by the estimation of un-measurable critical parameter.

2. Instead of installing another sensor, virtual sensors can be developed to sense a phenomena in parallel to real sensors. This reduces dependence on a single sensor and can also help in sensor health monitoring.

3. As model based fault diagnosis techniques heavily depends on precise dynamical models, this model precision can be achieved by estimating unknown parameters.

Keeping in view the various advantages of parameter estimation, there are certain critical parameters in gasoline engine whose estimation can benefit us in number of ways.

Virtual Sensors

Virtual sensors measure a phenomena through another inter-related process, thus providing a redundancy in measurements. The sensors under consideration are: Manifold Air Pressure (MAP) Sensor, Mass Airflow Sensor and Throttle Position sensor . The estimation of pressure from Mass AirflowThrottle Position can help us in the development of virtual sensors. Modern automotive industry employee highly sophisticated sensors. These sensors are heavily relied upon to ensure optimum engine performance. The development of virtual sensors can help in creating redundancy for these sensors and health monitoring of these sensors can be conducted efficiently.

Advantages of Second Order Sliding Mode Technique:

1. Work with nonlinear models which makes fault diagnosis of highly nonlinear systems easy.

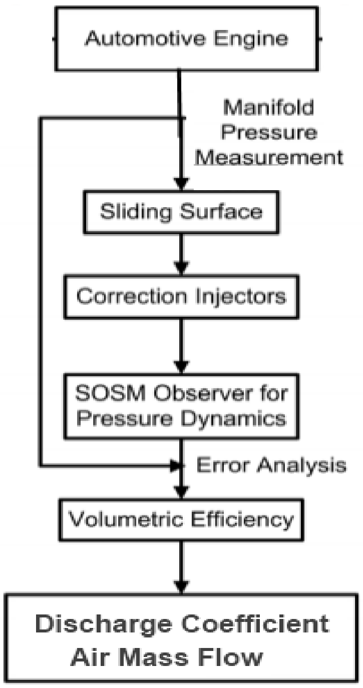
2. It may or may not require complete system information and noise distribution for convergence .as mandatory in Kalman filter.

3. Nonlinear framework of second order sliding mode based technique is effcient

4. Computationally cheap & online implementable as it is free of multiple matrices evaluation, transformations and multiplication.

5. does not involve the system to be Jacobian linearized as required in EKF.

6. linearization can be inaccurate for a system with large nonlinearities for short time period



Automotive Engine
Manifold Pressure Measurement
Sliding Surface
Correction Injectors
SOSM Observer for Pressure Dynamics
Error Analysis
Volumetric Efficiency
Discharge Coefficient Air Mass Flow

Figure 3.5: Proposed second order sliding mode observer based estimationstrategy for gasoline engine parameters.

3.4 Intake Manifold Pressure Dynamics

Consider the following states x, outputs y and input u, as defined in the context of an engine.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} P_m \\ \dot{P}_m \\ \omega_e \\ \dot{\omega}_e \end{bmatrix}$$

$$y = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} P_m \\ \omega_e \end{bmatrix} \qquad (3.9)$$

$$u = \alpha$$

Based on above defined states, output and input, the validated pressure dynamics in Eq (2.3) can be written as,

$$\dot{x}_1 = x_2 = A_1.A(x_5).f(x_1) - A_2.x_1.\eta_{vol} \tag{3.10}$$

Taking time derivative of X_2 we get $\dot{X}_2$ as

$$\dot{x}_2 = A_1 A(x_5) f'(x_1) + A_1 A'(x_5) f(x_1) - A_2.\dot{x}_1.\eta_{vol} \tag{3.11}$$

By replacing pressure dynamic and simplifying, we get

$$\dot{x}_2 = A_4 - A_5\eta_{vol} - A_6\eta_{vol} + A_7\eta^2{}_{vol} \tag{3.12}$$

where

$$A_1 = \frac{RT_m}{V_m} C_d P_a \sqrt{\gamma\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}$$

$$A_2 = \frac{V_d\,\omega}{4\pi V_m}$$

$$A_3 = C_d P_a \sqrt{\gamma\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}$$

$$A_4 = \frac{A_1 A_D}{\alpha_{cl}} f\ (x_1)\sin\left(\frac{x_5+\alpha_{cl}}{\alpha_{cl}}\right)\dot{\alpha} + \frac{9A_1{}^2 A^2(x_5) f\ (x_1)}{P_a} e^{(9(\frac{x_1}{P_a}-1))}$$

$$A_5 = \frac{9A_1 A_2 A(x_5) x_1 \eta_{vol}}{P_a} e^{(9(\frac{x_1}{P_a}-1))}$$

$$A_6 = A_1 A_2 A(x_5) f\ (x_1)$$

$$A_7 = A_2{}^2 \eta^2{}_{vol} x_1$$

$$f\ (x_1) = (1 - e^{(9(\frac{x_1}{P_a}-1))})$$

$$A(x_5) = A_D\left(1 - \cos\left(\frac{x_5+\alpha_{cl}}{\alpha_{cl}}\right)\right)$$

$$A_D = \frac{D\pi^2}{4} \tag{3.13}$$

The inlet manifold pressure dynamics can be written as:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1(x,t,u) + \xi(x,t,u) \end{aligned} \tag{3.14}$$

where, the initially known nonlinear dynamics of pressure manifold are:

$$f_1(x,t,u) = A_4 \tag{3.15}$$

and the dynamics to be estimated are:

$$\xi(x,t,u) = A_7\eta^2{}_{vol} - A_5\eta_{vol} - A_6\eta_{vol} \tag{3.16}$$

3.5 Second Order Sliding Mode Observer (SOSM) for Manifold Pressure Dynamics

After analyzing the pressure dynamics , an observer is proposed to estimate of the state vector x_1 and extract unknown variables

$$\begin{aligned} \dot{\hat{x}}_1 &= \hat{x}_2 + z_1 \\ \dot{\hat{x}}_2 &= f_1(\hat{x},t,u) + z_2 \end{aligned} \tag{3.17}$$

where $\dot{\hat{x}}_1$ represents the corresponding observer states and z_1, z_2 are the observer injectors based on super twisting algorithm. These injectors aim to eliminate the error between the estimated states and the actual states i.e. ($e_1 = x_1 - \hat{x}_1$) & ($e_2 = x_2 - \hat{x}_2$). These injectors are defined as,

$$z_1 = \lambda_1 |e_1|^{1/2} sign(e_1) + \upsilon_1 \tag{3.18}$$

$$\dot{\upsilon}_1 = \alpha_1 sign(e_1) \tag{3.19}$$

And

$$\begin{aligned} z_2 &= 0 && \text{if} \quad e_1 \neq 0 \,\&\, \dot{e}_1 \neq 0 \\ z_2 &= \lambda_2 |z_1|^{1/2} sign(z_1) + \upsilon_2 && \text{if} \quad e_1 = 0 \,\&\, \dot{e}_1 = 0 \\ \dot{\upsilon}_2 &= \alpha_2 sign(z_1) \end{aligned} \tag{3.20}$$

The state x_1 is available for measurement from the sensor installed in the engine inlet manifold. The gains λ_1, α_1 are the observer gains. The above second order sliding mode observer inherits anti-peaking structure, where e_1 and e_2 reach the sliding manifold one by one in a recursive fashion.The choice of λ_1 and α_1 depends on the uncertainty bound and the initial state estimation error in the worst case [1].

Convergence and Error Dynamics Analysis

The error dynamics before converging to the surface figure (1)comes out to be

$$\begin{aligned} \dot{e}_1 &= e_2 - z_1 \\ \dot{e}_2 &= F_1\left(x, \hat{x}, t, u\right) \end{aligned} \tag{3.21}$$

Where

$$F_1\left(x,\hat{x},t,u\right) = f_1\left(x,t,u\right) - f_1(\hat{x},t,u) + \zeta_1(x,t,u) \tag{3.22}$$

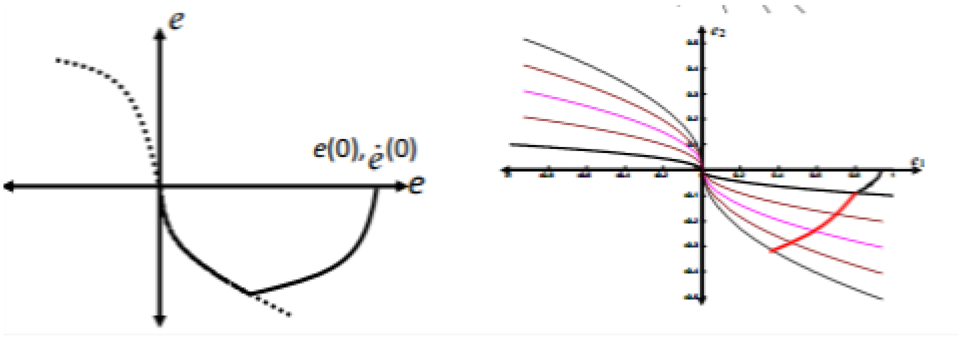


Figure 3.6: *The error* will converge to zero in finite time

When e_1 converges to sliding manifold, the dynamics of e_2 becomes

$$\dot{e}_2 = F_1\left(x, \hat{x}, t, u\right) - z_2 \tag{3.23}$$

After the second order sliding mode has been achieved, the term e_1 approaches tozero

$$\begin{aligned}\dot{e}_1 &= e_2 - z_1 \\ 0 &= e_2 - z_1\end{aligned} \tag{3.24}$$

When e_1 approaches to zero, $\dot{e}_1$ also vanishes and from Eq (3.17) we are left with

$$z_1 = x_2 - \hat{x}_2 \tag{3.25}$$

Or

$$\bar{z}_1 = C_D A_8 - A_2 x_1 \eta_{vol} - \hat{x}_2 \tag{3.26}$$

As the convergence of e_1 is achieved, the convergence of e_2 is also guaranteed. Under convergence i.e $e_2 = 0$ and $\dot{e}_2 = 0$ the following sliding mode dynamics from Eq (3.19) will be available.

$$\dot{e}_2 = F_1\left(x, \hat{x}, t, u\right) - \bar{z}_2 \tag{3.27}$$

Or

$$0 = \xi_1(x, t, u) - \bar{z}_2 \tag{3.28}$$

Or

$$\bar{z}_2 = \xi_1(x, t, u) \tag{3.29}$$

Where

$$\xi_1(x,t,u) = A_7\eta^2{}_{vol} - \eta_{vol}(A_5 + A_6) \tag{3.30}$$

Or

$$\bar{z}_2 = A_7\eta^2{}_{vol} - \eta_{vol}(A_5 + A_6) \tag{3.31}$$

Estimation of Parameters from Pressure Dynamics

Estimation of η_{vol} :

$$\eta_{vol} = \frac{(A_5 + A_6) \pm \sqrt{(A_5 + A_6) + 4A_2 z_2}}{2A_7} \tag{3.32}$$

Estimation of C_D :

$$C_D = \frac{\bar{z}_1 + \hat{x}_2 + A_2.x_1.\eta_{vol}}{A_8} \tag{3.33}$$

Once has been estimated, it can lead to the development of virtual sensor for angular speed of engine as(3.34)

$$x_3 = \bar{\omega}_e = \frac{1}{\eta_{vol} A_2 x_1}(A_9 - (z_1 + x_2)) \tag{3.34}$$

CHAPTER 4

EXPERIMENTAL AND RESULTS

4.1 Automotive specification

After the theoretical study for our project we started the practical application of theoretical ideas. The first step that we have work to bring, a car that has following specifications.

Manufacturer: Opel/Vauxhall Model: Corsa-B 1,4
Engine code: X14XE Output: 66 (90) 6000
Tuned for: R-Cat Year: 1994-00

Types of vehicle sensors on the opel Corsa-B 1.4 Engine

1-Camshaft position (CMP) sensor
2- Data link connector (DLC) - fascia fuse box
3- Engine control module (ECM) - RH A-post
4- Engine coolant temperature (ECT) sensor
5- Engine speed (RPM) sensor
6- Evaporative emission (EVAP)
canister purge valve
7- Exhaust gas recirculation (EGR) solenoid
- X14XE/X16XE/X16XEL
8- Fuel filter - underside rear
9 -Fuel pressure regulator
10- Fuel pump - in tank
11- Fuel pump relay - RH A-post
12- Heated oxygen sensor (HO2S)
13- Idle air control (IAC) valve
14- Ignition coil
15- Injectors
16- Intake air temperature (IAT) sensor

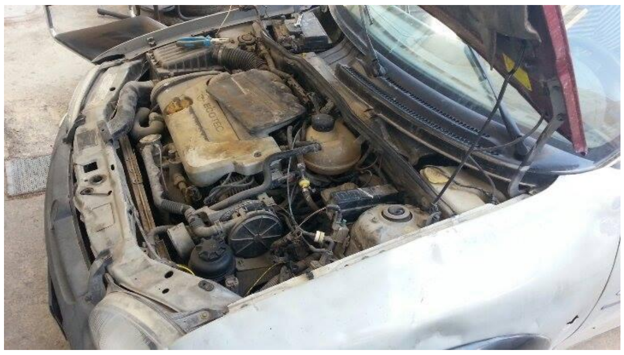


Figure 4.1: Opel Corsa B Engine Power 1.4

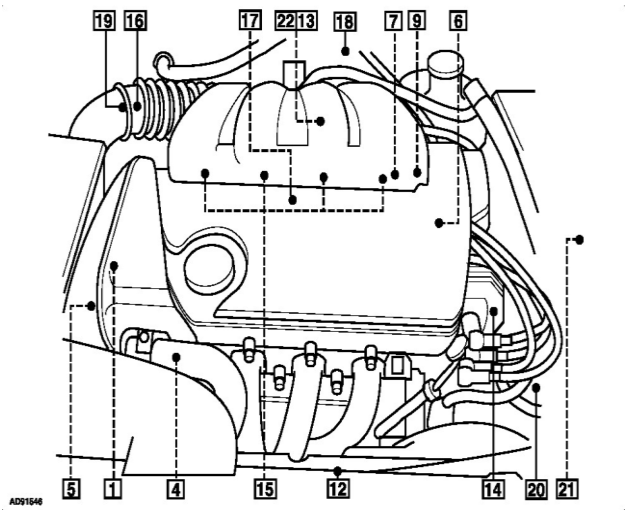


Figure 4.2: Component Engine location

- X14XE/X16XE/X16XEL

17- Knock sensor (KS)

18 -Manifold absolute pressure (MAP) sensor

- X14XE/X16XE/X16XEL

20- Secondary air injection (AIR) pump

- X14XE/X16XE/X16XE

21- Secondary air injection (AIR) solenoid

- X14XE/X16XE/X16XEL

22 -Throttle position (TP) sensor

23- Vehicle speed sensor (VSS) - transmission

4.2 Implementation In matlab Simulink:-

MATLAB is a multi-paradigm numerical computing environment and fourth-generation programming language. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems. So we used matlab to make simulation for virtual sensor and detect diagnosis in air intake manifold system[6]. we make implementation on matlab using imbedded function blocks. Figure (4.3) show the simulation of virtual sensor on matlab.

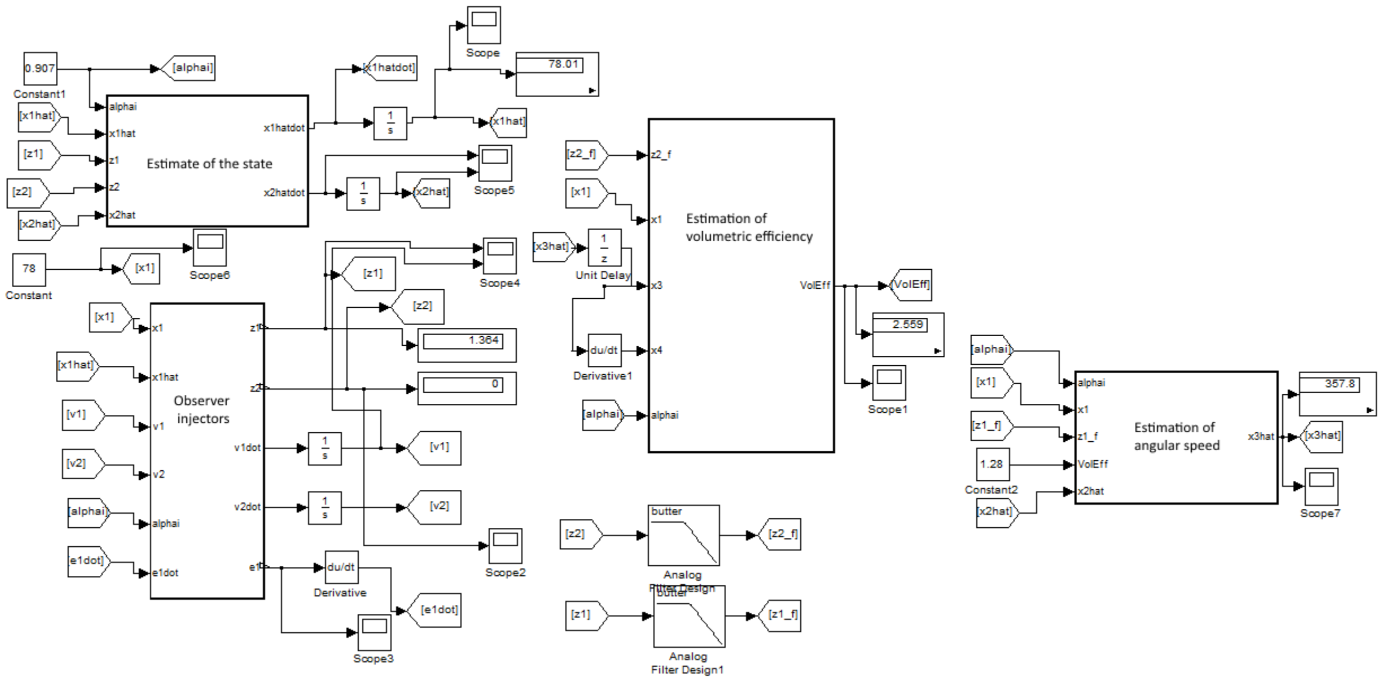


Figure 4.3: Embedded Function blocks

Figure4.4 show response of X_1 state and observed state. the error with percent error 0% as shown in figure (4.5). The SOSM gains were chosen as given in Table 4.1. While tuning observer parameters for the demonstrated results, these gains were manipulated to get healthy values of the parameters as found in the literature.

Table 4.1:SOSM Observer Parameters For Speed Virtual Sensor

Parameter	Value
α_1	0.9
λ_1	5
α_2	1.1
λ_2	135

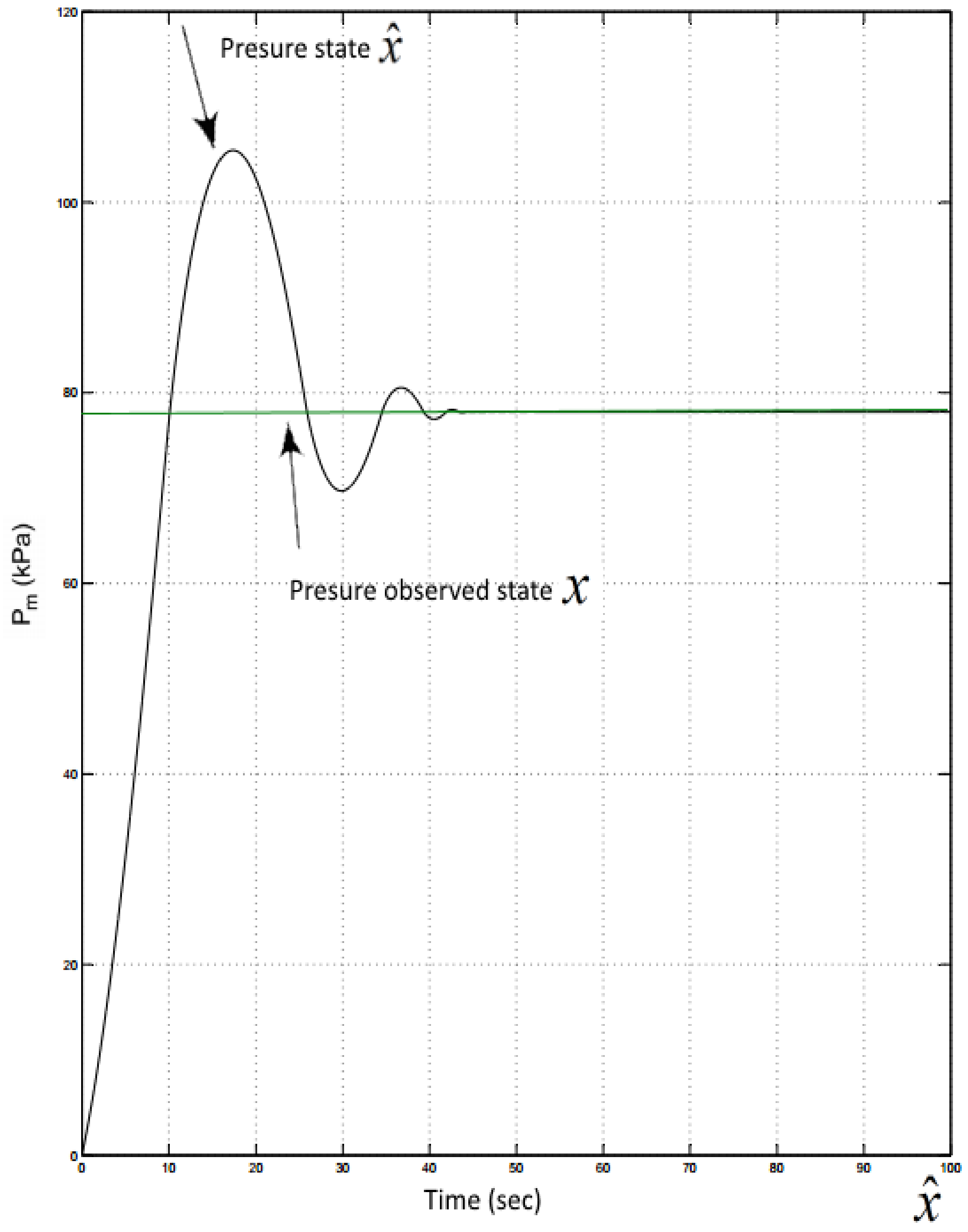


Figure 4.4: Estemation pressure responce

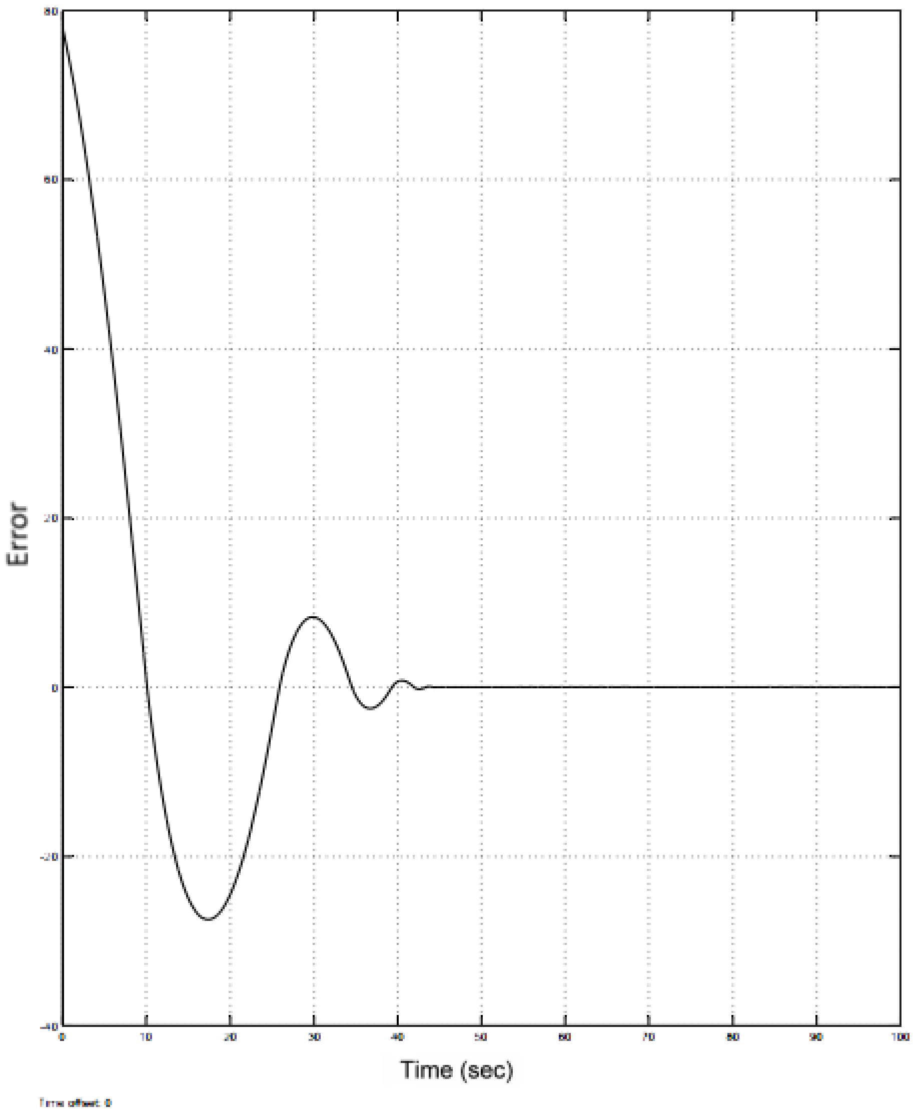


Figure 4.5: Error defference between actual and estimation

The main outcome of the proposed methodology is the redundancy of manifold pressure and crankshaft sensors installed in air intake system (AIS). Equations (3.34) serve as virtual sensors for the measurement of angular speed. Figure 4.6 shows the measurements from virtual sensors and Figure 4.7 shows the measurements from actual sensors.

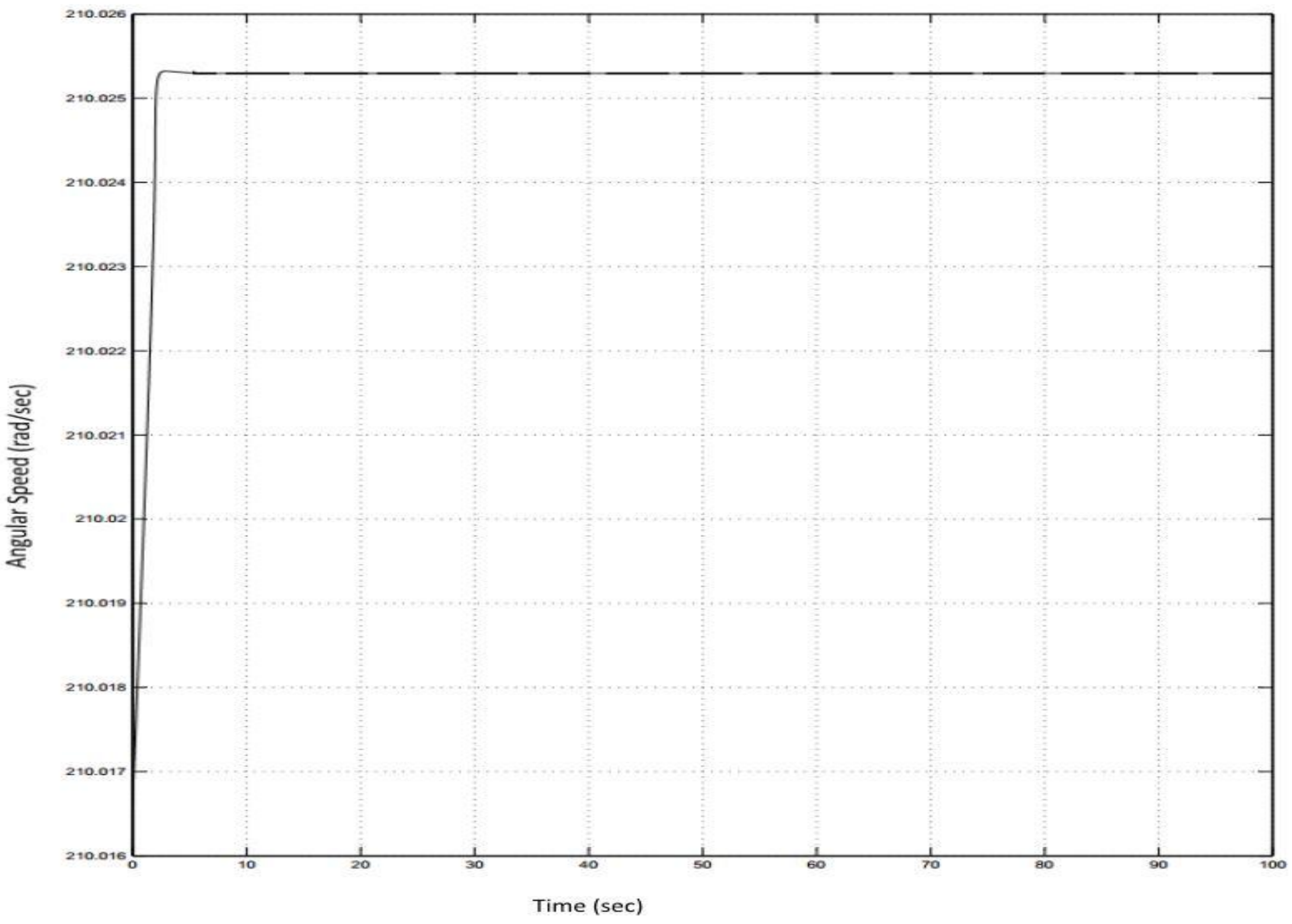


Figure 4.6:Estimation speed responce

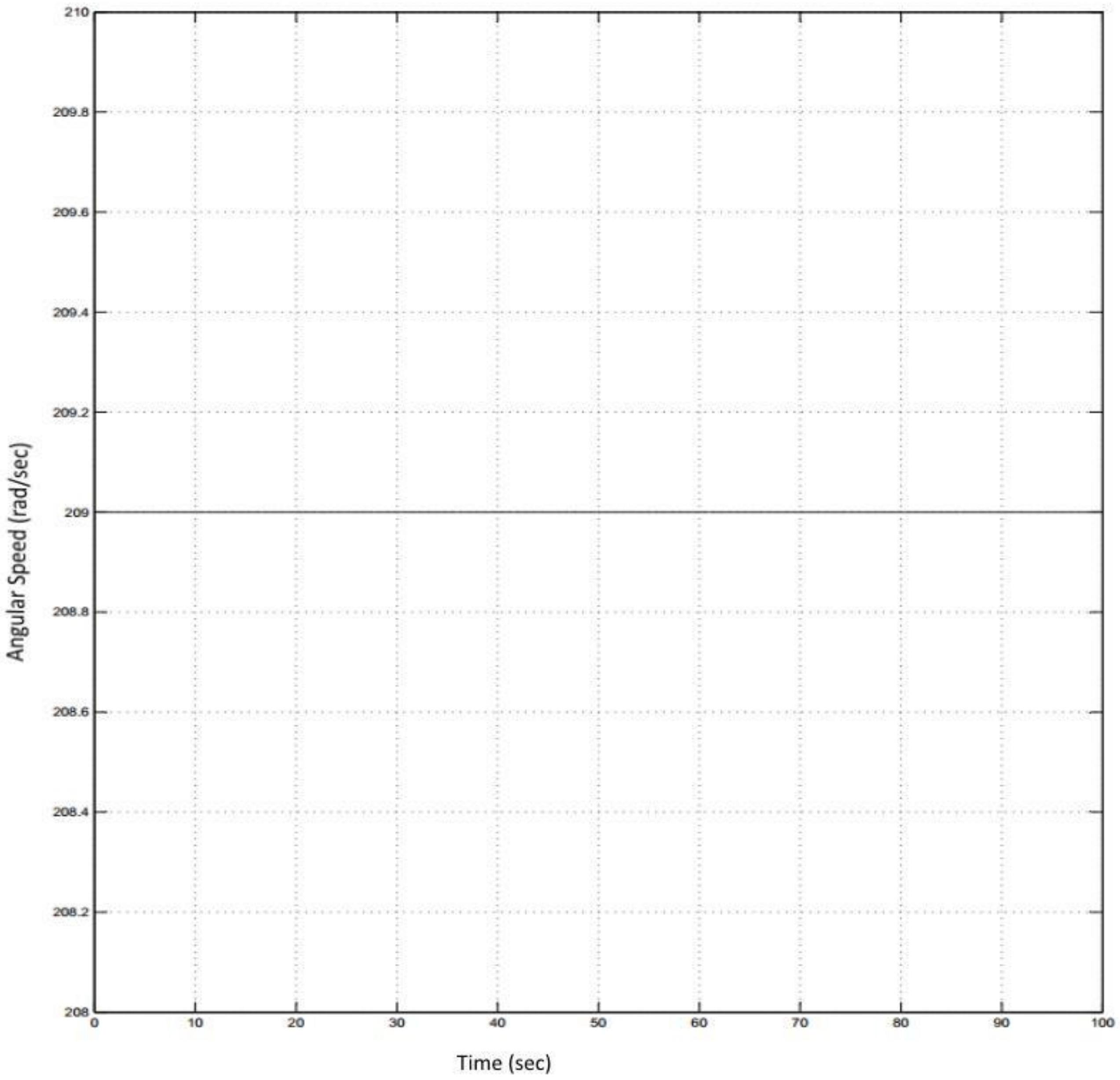


Figure 4.7: Actual speed responce at constant speed

4.3 Real-time And Engine control module:-

In automotive electronics, electronic control unit (ECU) is a generic term for any embedded system that controls one or more of the electrical system or subsystems in a motor vehicle, ECU is a type of electronic control unit that controls a series of actuators on an internal combustion engine to ensure optimal engine performance. It does this by reading values from of a sensors within the engine, interpreting the data using multidimensional performance maps [7],

Figure 4.8, and adjusting the engine actuators accordingly. Before ECUs, air-fuel mixture, ignition timing, and idle speed were mechanically set and dynamically controlled by mechanical and pneumatic means, In the last Figure 4.9 and Figure 4.10 show Wiring diagram sensor that is connected to ECU.

The computer operating program consists of a series of predetermined information cells, these cells hold the equation for proper vehicle operation. If the computer detects that it can't control a particular system it will illuminate the MIL (malfunction inductor lamp), and check engine or service engine soon light. This means the computer has stored a trouble code, There are a lot of programs that support for those import the data and export to and from the ECU .
By Using autodata software we Find out of parameter and location of electrical component and data sheet for ports form sensor and send command to actuators and Screen lamp.

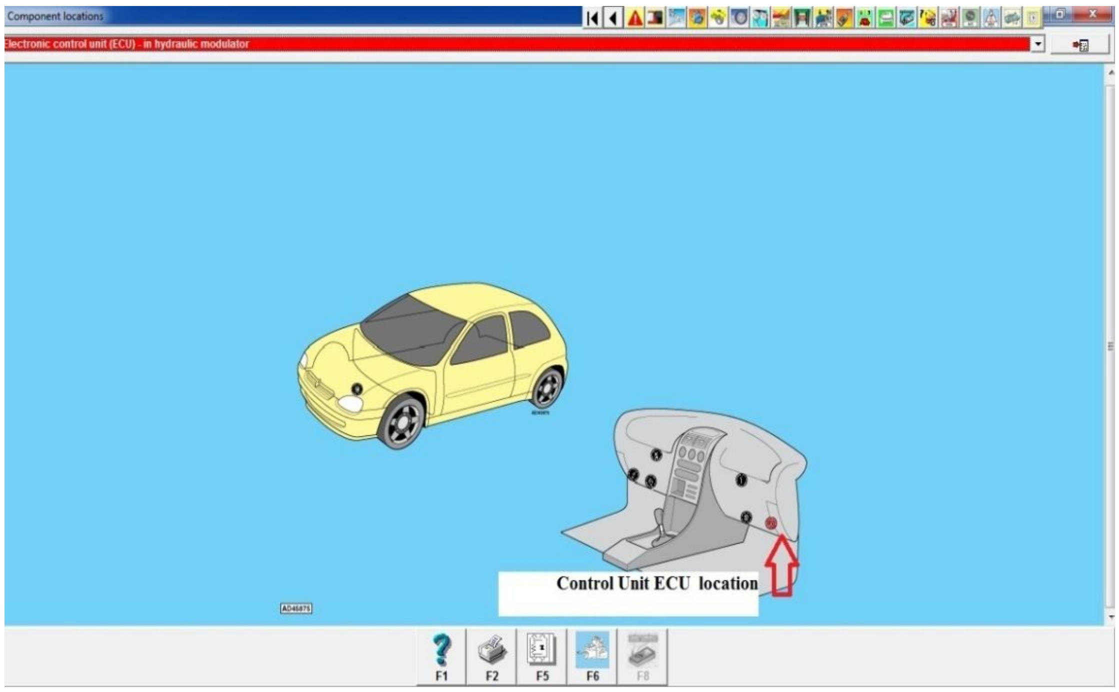

Component locations
Electronic control unit (ECU) - in hydraulic modulator
Control Unit ECU location
AD46875
F1
F2
F5
F6
F8

Figure 4.8: Control Unit Engine Location

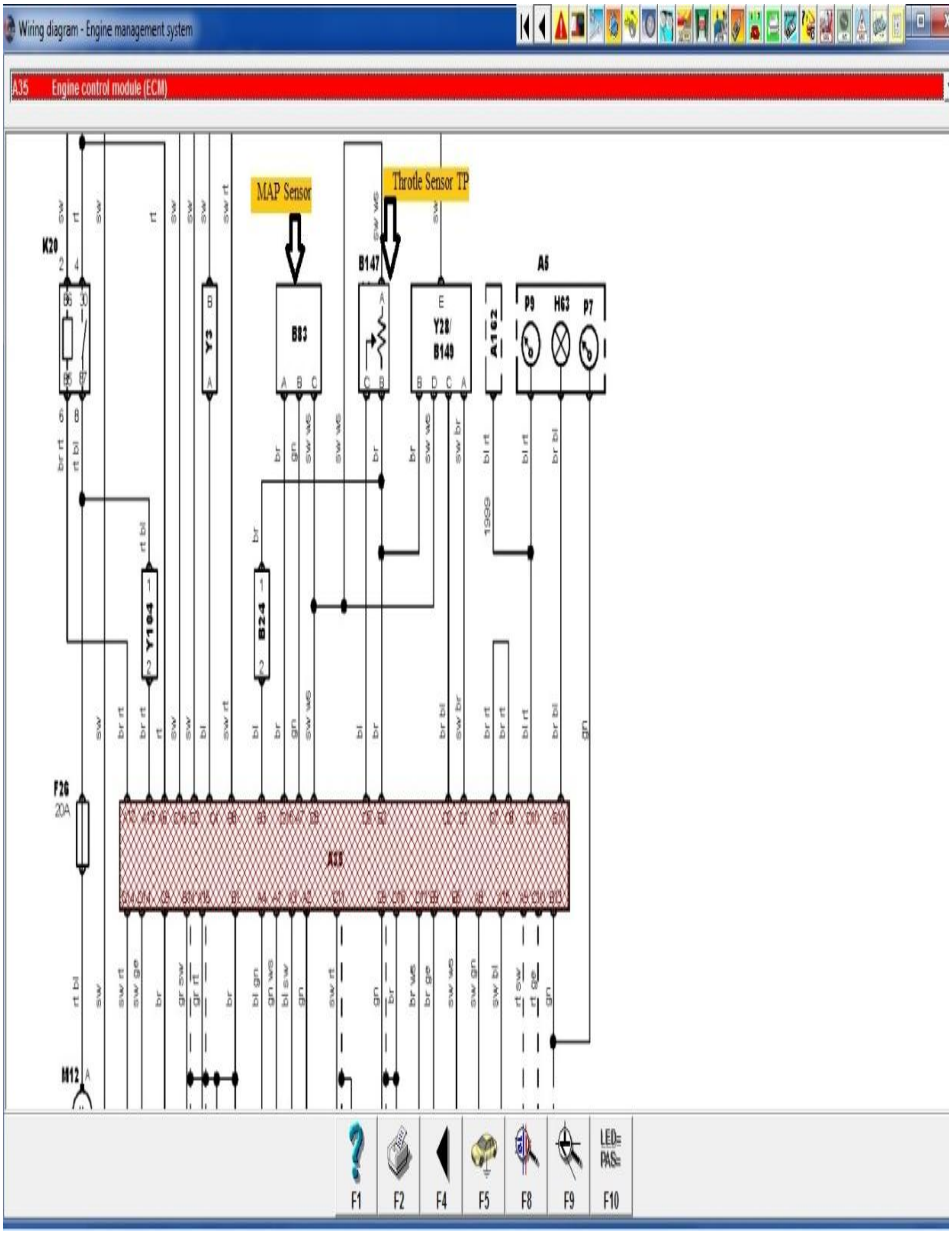


Figure 4.9: Wiring diagram for component of the Engine

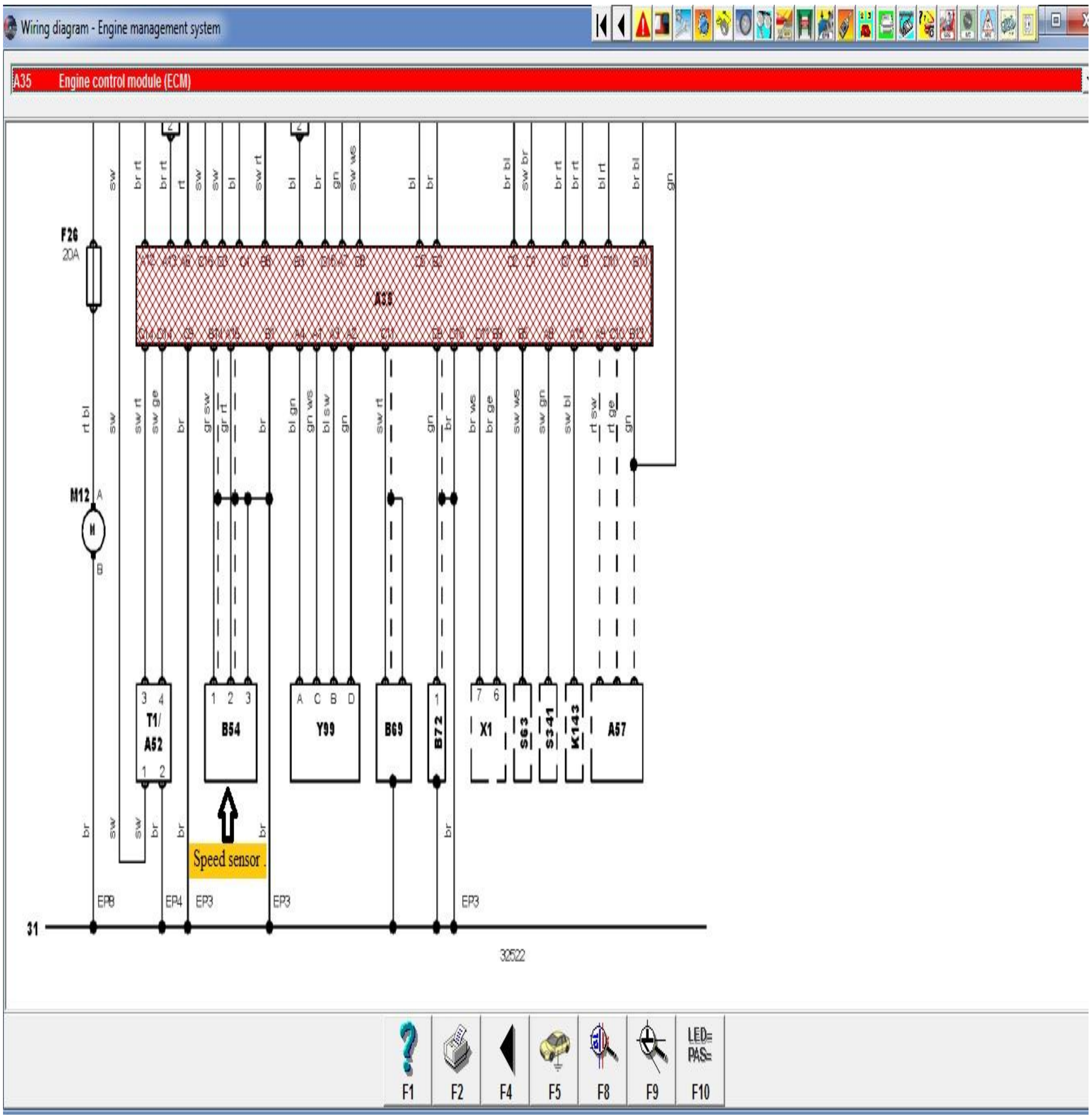


Figure 4.10: Wiring digram show speed sensor connecting

Input and output port: Engine management pin

This figure show The number pin data for map ,TP and speed sensor, in auto data software

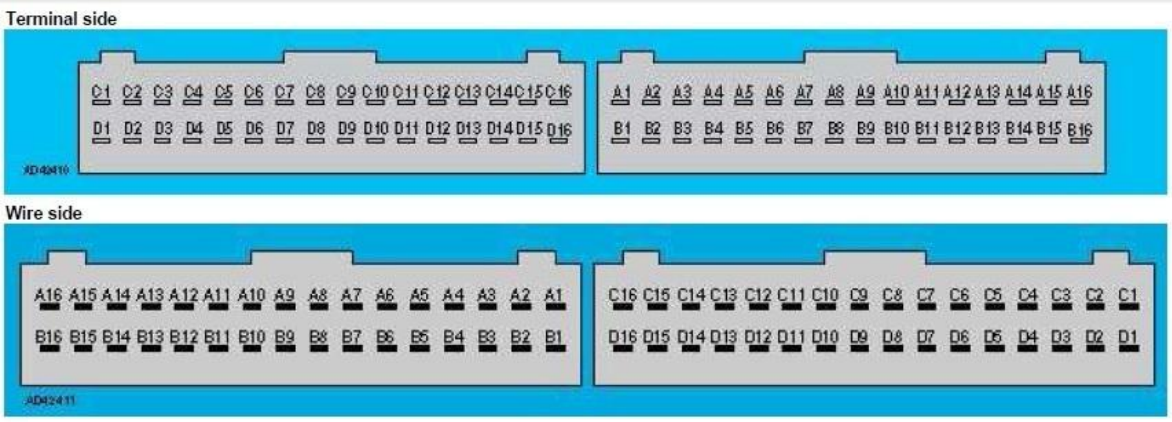
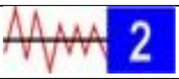

Terminal side
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16
B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 B16
Wire side
A16 A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1
B16 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1
C16 C15 C14 C13 C12 C11 C10 C9 C8 C7 C6 C5 C4 C3 C2 C1
D16 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1

Figure 4.11: Terminal side pin data

Table 4.2: Pin data

Component/circuit description	ECM pin	Signal	Condition	Typical value
RPM sensor	A16	Input signal	Engine cranking	
RPM sensor	B14	Output signal	Engine cranking	
RPM sensor	D16	Earth	Ignition ON	0 V
Throttle position (TP) sensor	B2	Earth	Ignition ON	0 V
Throttle position (TP) sensor	D5	Input signal	Ignition ON - throttle closed	0,7 V
Throttle position (TP) sensor	D5	Input signal	Ignition ON - throttle open	4.6V
Throttle position (TP) sensor	D8	Output signal	Ignition ON	5V
Manifold absolute pressure(MAP) sensor	A7	Input signal	Engine idling - engine hot	1,3 V
Manifold absolute pressure(MAP)	D8	Output signal	Ignition ON	5V
Manifold absolute pressure(MAP) sensor	D15	earth	Ignition ON	0 V

A7 $\longrightarrow$ signal output

D8 $\longrightarrow$ +5V (power supply)

D15 $\longrightarrow$ 0 (ground)

<u>TP Pin</u>

D5 $\longrightarrow$ signal output

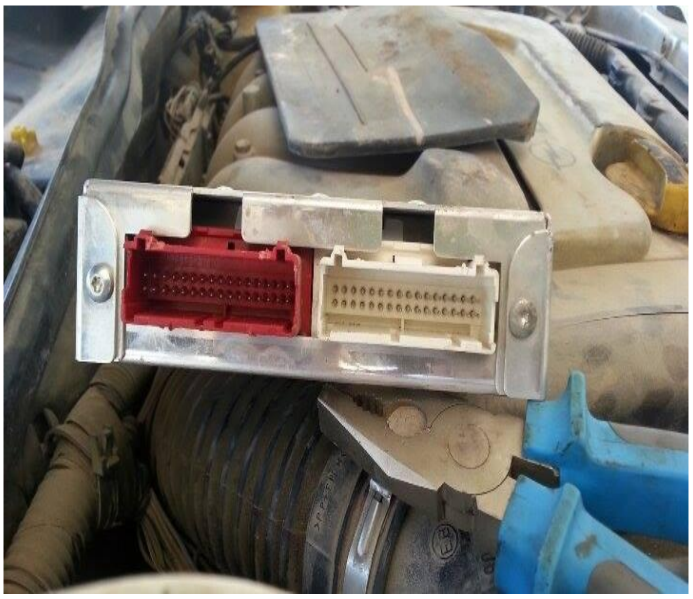
D8 $\longrightarrow$ +5V (power supply)

B2 $\longrightarrow$ 0 (ground)

<u>Crank position sensor pin</u>

A16

B14



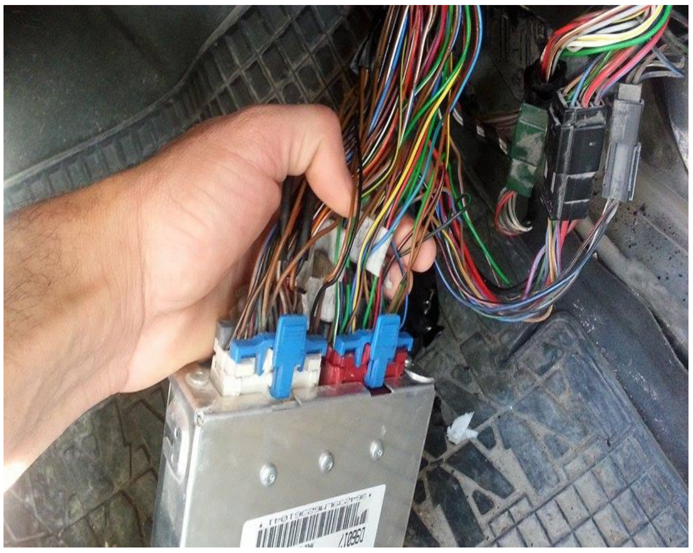


Figure 4.12: Control unit for opel Corsa-B Engine

4.4 Choices of real time controller:-

- PIC microcontroller family (PIC18F4450,PIC18F2450....atc).

Figure 4.13: PIC Family

- Data acquisition card betweenHost and TargetPC

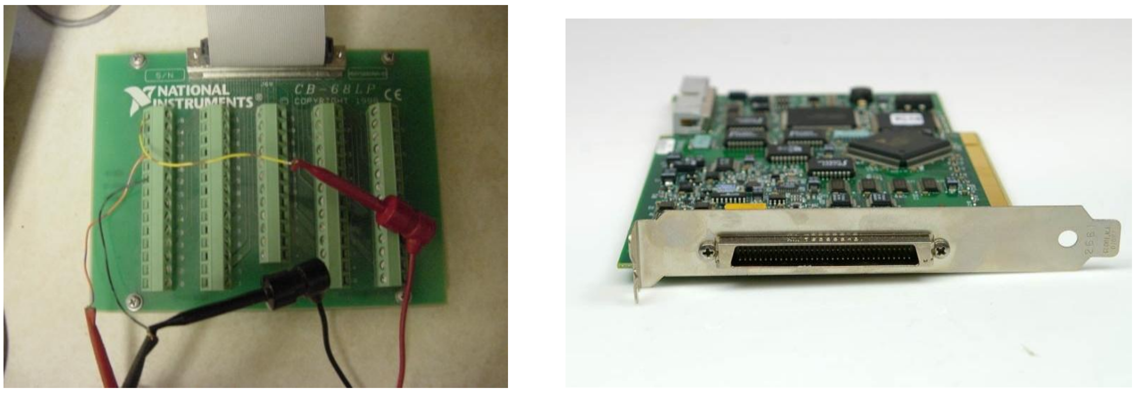


Figure 4.14: DAQ and I/O port

Arduino Controller Family:-

The Arduino microcontroller is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free. For advanced Arduino users, prowl the web; there are lots of resources[9].

a good choice for students and educators. With the Arduino board, you can write programs and create interface circuits to read switches and other sensors, and to control motors and lights with very little effort.

Arduino model

1. Arduino UNO
2. Arduino Micro
3. Arduino Mini
4. Arduino Nano
5. Arduino Mega
6. Arduino Mega ADK
7. Arduino Fio
8. Arduino Robot Motor
9. Arduino Robot Control
10. ...etc

This is what the Arduino board looks like.

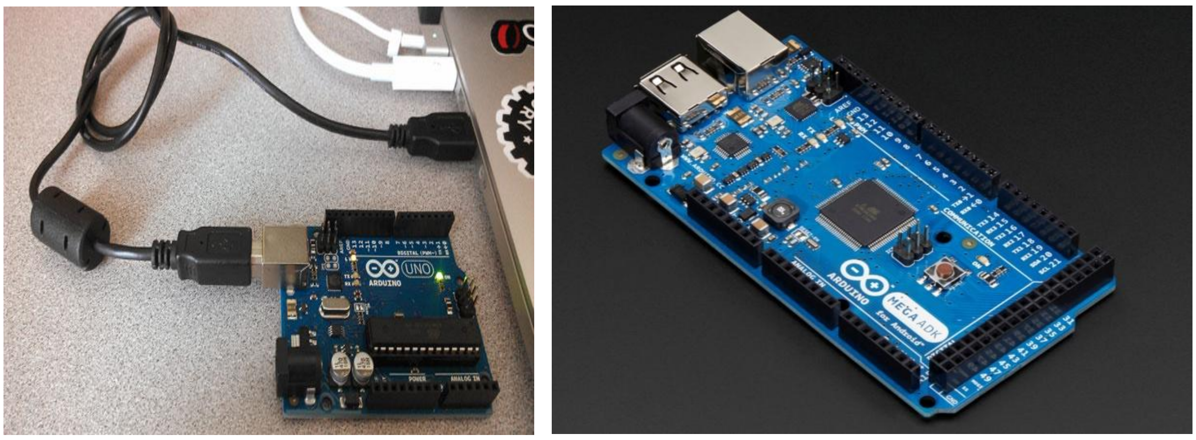


Figure 4.15: Arduino bord

The Arduino programming language is a simplified version of C/C++. If you know C, programming the Arduino will be familiar. If you do not know C, no need to worry as only a fewcommands are needed to perform useful functions, An important feature of the Arduino is that you can create a control program on the host PC, download it to the Arduino and it will run automatically. Remove the USB cable connection to the PC, and the program will still run from the top each time you push the reset button. Remove the battery and put the Arduino board in a closet for six months. When you reconnect the battery, the last program you stored will run. This means that you connect the board to the host PC to develop and debug your program, but once that is done, you no longer need the PC to run the program.

What You Need for a Working System

1. Arduino board
2. USB programming cable (A to B)
3. battery or external power supply (for stand-alone operation)
4. Solderless breadboard for external circuits, wire for connections
5. Host PC running the Arduino development environment. Versions exist for Windows, Mac and Linux.

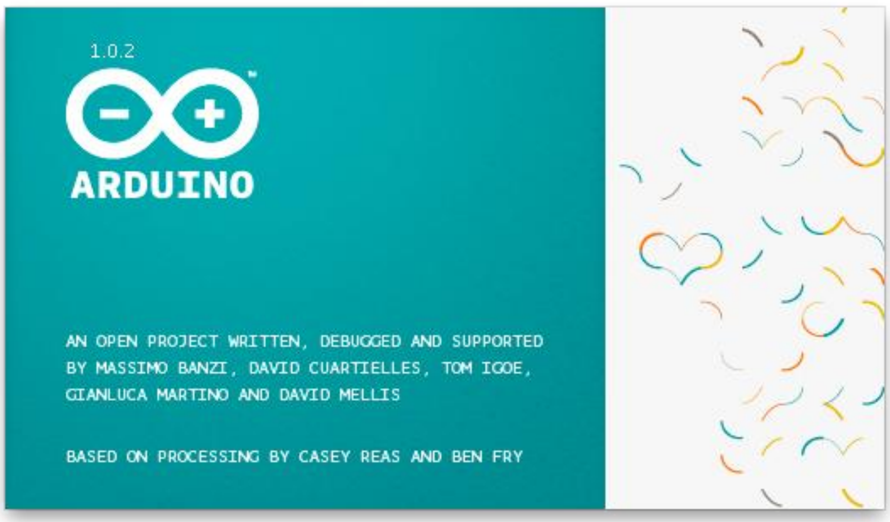
we to used Arduino Mega ADK,to the following specification.

Table 4.3**:** Mega ADK Specifications

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
USB Host Chip	MAX3421E
Length	101.52 mm
Width	53.3 mm
Weight	36 g

Installing the Software

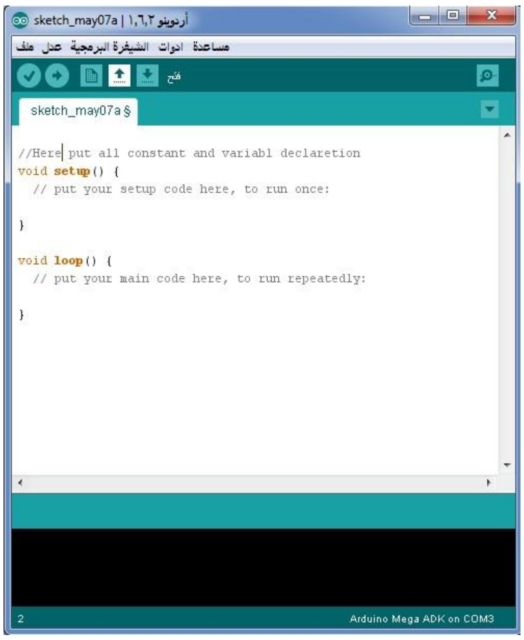
Follow the instructions on the Getting Started section of the Arduino web site, http:arduino.cc/en/Guide/HomePage. Go all the way through the steps to where you see the pin 13 LED blinking. This is the indication that you have all software and drivers successfully installed and can start exploring with your own programs.



AN OPEN PROJECT WRITTEN, DEBUGGED AND SUPPORTED BY MASSIMO BANZI, DAVID CUARTIELLES, TOM IGOE, GIANLUCA MARTINO AND DAVID MELLIS

BASED ON PROCESSING BY CASEY REAS AND BEN FRY

Connect your Arduino to the computer with the USB cable. You do not need the battery for now,The green PWR LED will light. If there was already a program burned into the Arduino, it will Start the Arduino development environment. In Arduino-speak, programs are called “sketches”,but here we will just call them programs.,In the editing window that comes up, enter the following program, paying attention to wheresemi-colons appear at the end of command lines.


sketch_may07a | أردوينو ١,٦,٢
ملف عدل الشيفرة البرمجية أدوات مساعدة
فتح
sketch_may07a §
2
Arduino Mega ADK on COM3

```
//Here put all constant and variabl declaretion
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 4.16: Main Arduino Window

Min Function Language Code

```
{
void setup()
{
Serial.begin(9600);
}
void loop(){
Serial.print(" " );
  Serial.println( value);
}
```

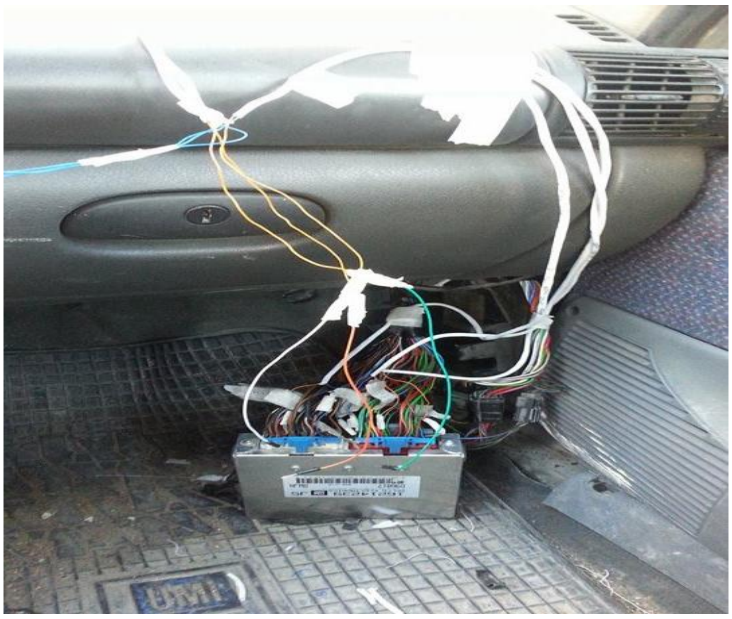


Figure 4.17: ECU Wire Connection

4.5 Analog Signal Reading

1- Analog Read Arduino by serial

Read an input on analog pin , prints the result to the serial monitor, by serial Arduino code Attach the center pin of a sensor to analog pin of Arduino, and the outside pins to + 5V and ground, note the value reading read bit-by-bit it show between1024 bit ,A bit is the basic unit of information in computing and digital communications, we need converting the value of bit to voltage and calibration.

Sensor calibration:-

To work calibration for any Sensor We need the two values if the linear relationship i.e Equation(4.1), For the conversion of the voltage value of the physical quantity measured by sensor, but if you were not a linear relationship we need a set of readings from which to extract the equation, i.e Equation(4.2).

Table 4.4: volt pressure relation

V(volt)	Pressure (Kpa)
0.3	0
0.3	10
0.6	20
1.1	30
1.7	40
2.2	50
2.7	60
3.3	70
3.8	80
4.4	90
4.9	100

The readings returned by the analog inputs are raw binary values (low level functions). An approximate voltage conversion can be performed as:

Volts=(bits/1024)*5V...........................bit to volt (4.1)

Where span is the maximum voltage minus the minimum voltage from the table above. For a proper voltage conversion, though, use the calibration values (Slope) stored in the internal flash on the Control processor.

Figure 4.18 show Map calibration, If the linear relation slope $\frac{\triangle y}{\triangle x}$ = gain converting, sensor value=((bits/1024)*5V)*slop, in non linear relation ,we needed the slope equation.

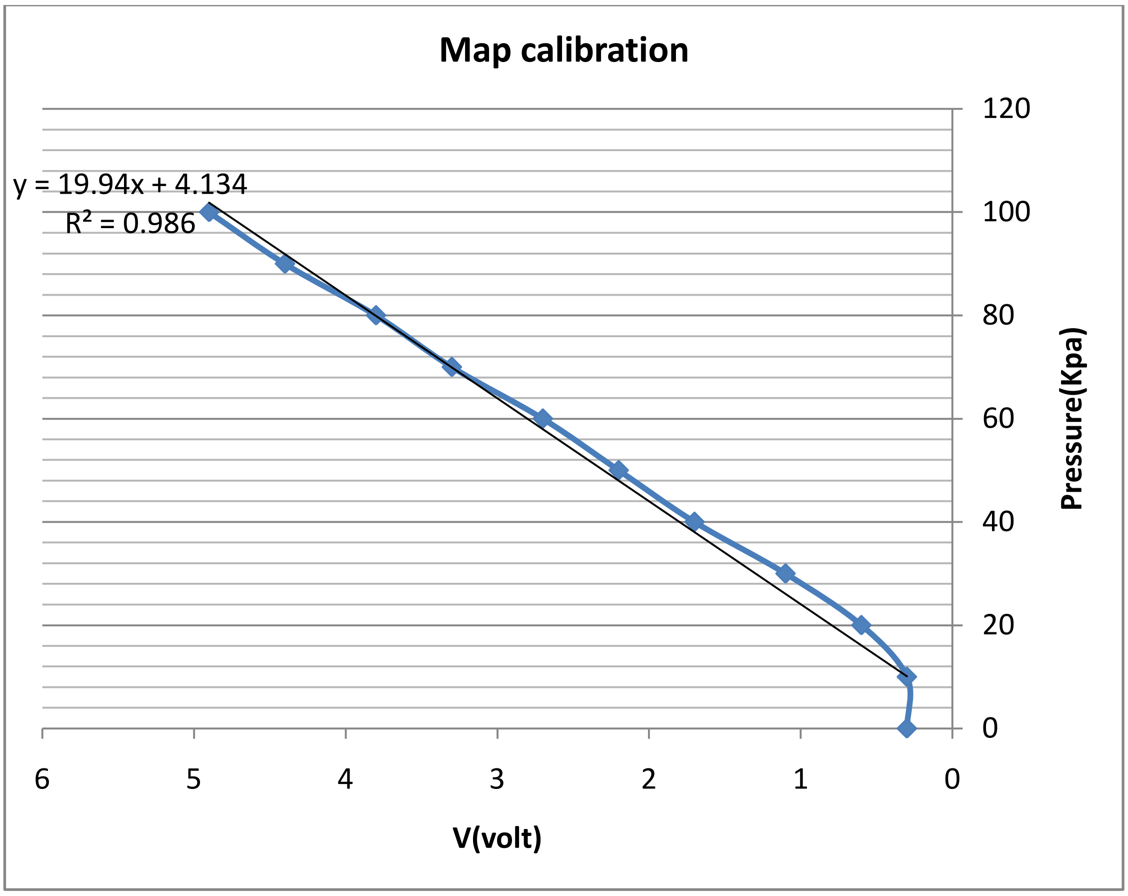


Figure 4.18: Map calibration

Pressure(Kpa)=19.94*volts+4.134 (4.2)

slope of Throtle angle by Minimum and Max at 10-100 degree voltaeg

degree = 2.2 * (5/1024). (4.3)

In speed or crank postion sensor genrating a signal at variable freqancy dependent revolution of cranck disk per time unit.

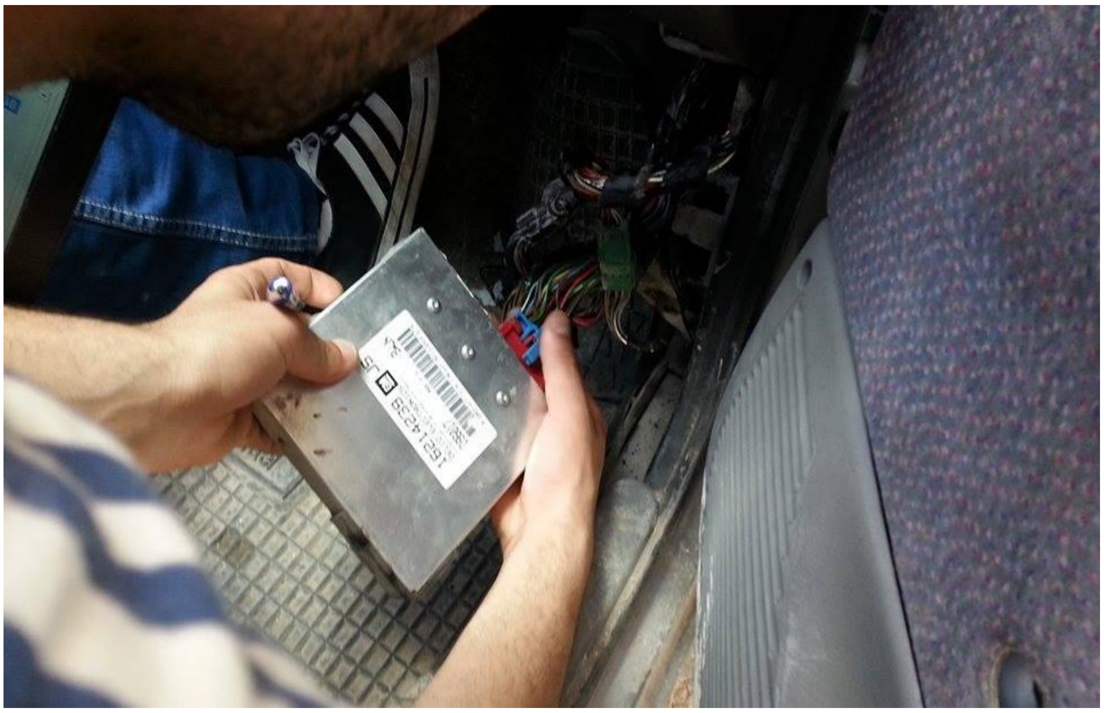


Figure 4.19: Actual ECU for opel Corsa-B 1,4

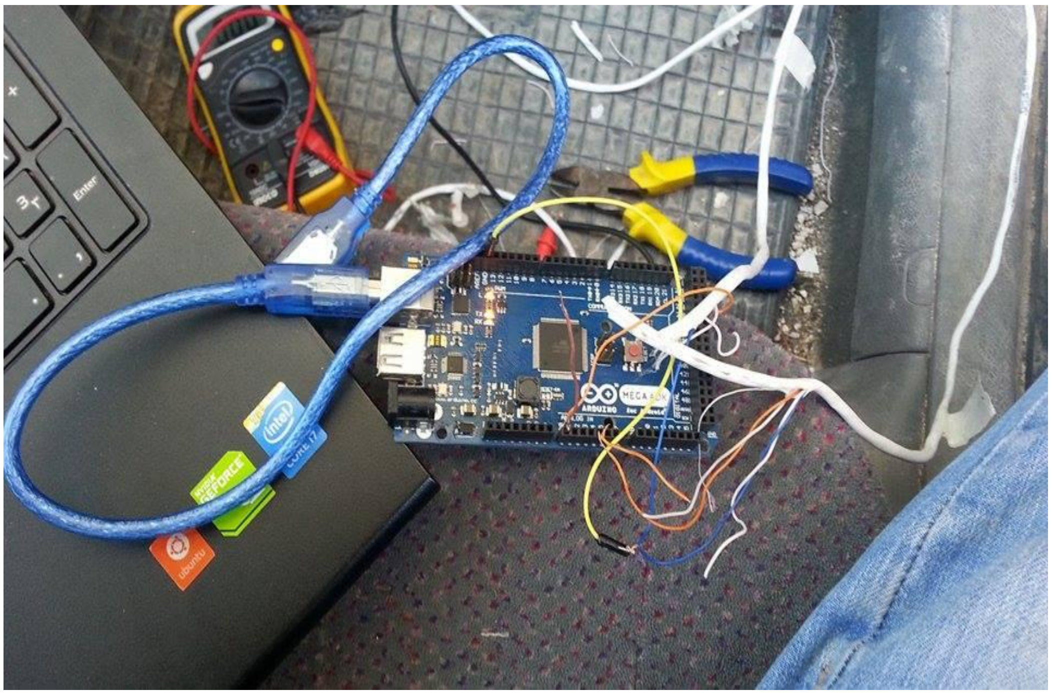
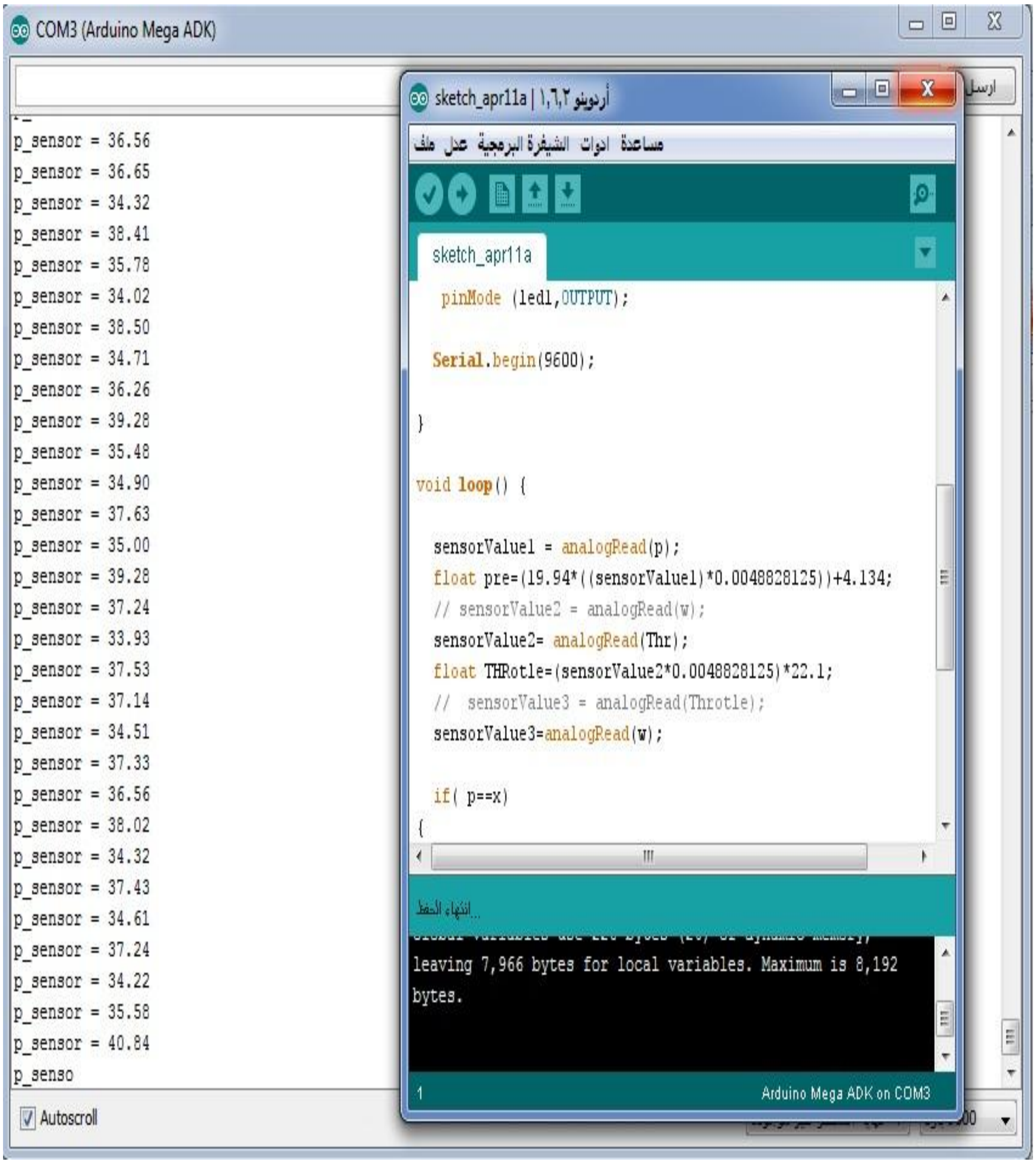


Figure 4.20: Arduino Controller interface bettwen ECU and PC

Figurs 4.19 and 4.20 show our working on ECU of the car and controller interface with it.

Actual sensor value

After calibration process the readings were as follows, In Figure 4.21 show the pressure reading after calibration and that values at ideal speed Engine. In Figure 4.22 and Figure 4.23 The throtlle value closed and full open respectively as a degree. In Figure 4.24 and Figure 4.26 The speed ideal and test road respectively as a voltge. In Figure 4.25 and Figure 4.27 show ideal speed and test road speed curves by matlab

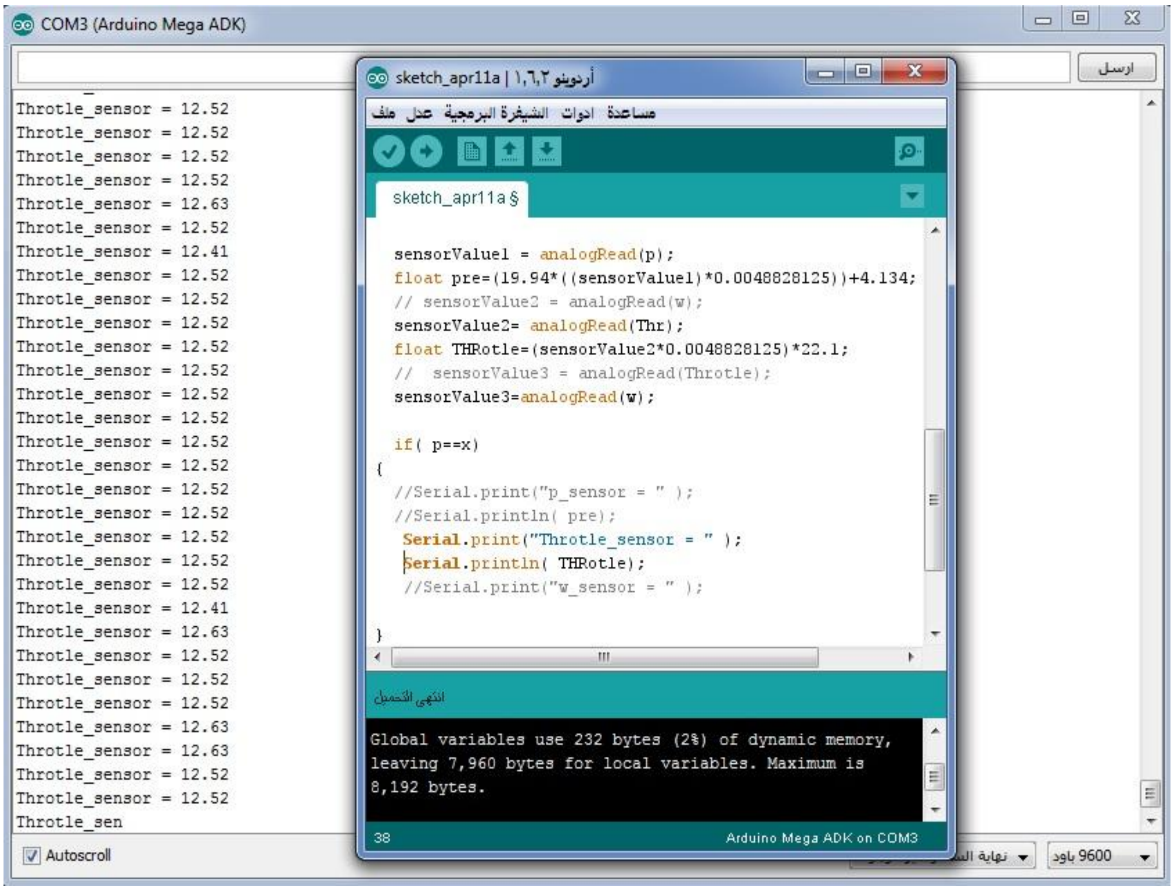


```
p_sensor = 36.56
p_sensor = 36.65
p_sensor = 34.32
p_sensor = 38.41
p_sensor = 35.78
p_sensor = 34.02
p_sensor = 38.50
p_sensor = 34.71
p_sensor = 36.26
p_sensor = 39.28
p_sensor = 35.48
p_sensor = 34.90
p_sensor = 37.63
p_sensor = 35.00
p_sensor = 39.28
p_sensor = 37.24
p_sensor = 33.93
p_sensor = 37.53
p_sensor = 37.14
p_sensor = 34.51
p_sensor = 37.33
p_sensor = 36.56
p_sensor = 38.02
p_sensor = 34.32
p_sensor = 37.43
p_sensor = 34.61
p_sensor = 37.24
p_sensor = 34.22
p_sensor = 35.58
p_sensor = 40.84
p_senso
```

```
  pinMode (led1,OUTPUT);

 Serial.begin(9600);


}


void loop() {

 sensorValue1 = analogRead(p);
 float pre=(19.94*((sensorValue1)*0.0048828125))+4.134;
 // sensorValue2 = analogRead(w);
 sensorValue2= analogRead(Thr);
 float THRotle=(sensorValue2*0.0048828125)*22.1;
 //  sensorValue3 = analogRead(Throtle);
 sensorValue3=analogRead(w);

 if( p==x)
{
```

```
leaving 7,966 bytes for local variables. Maximum is 8,192
bytes.
```

Figure 4.21: pressure at idal speed

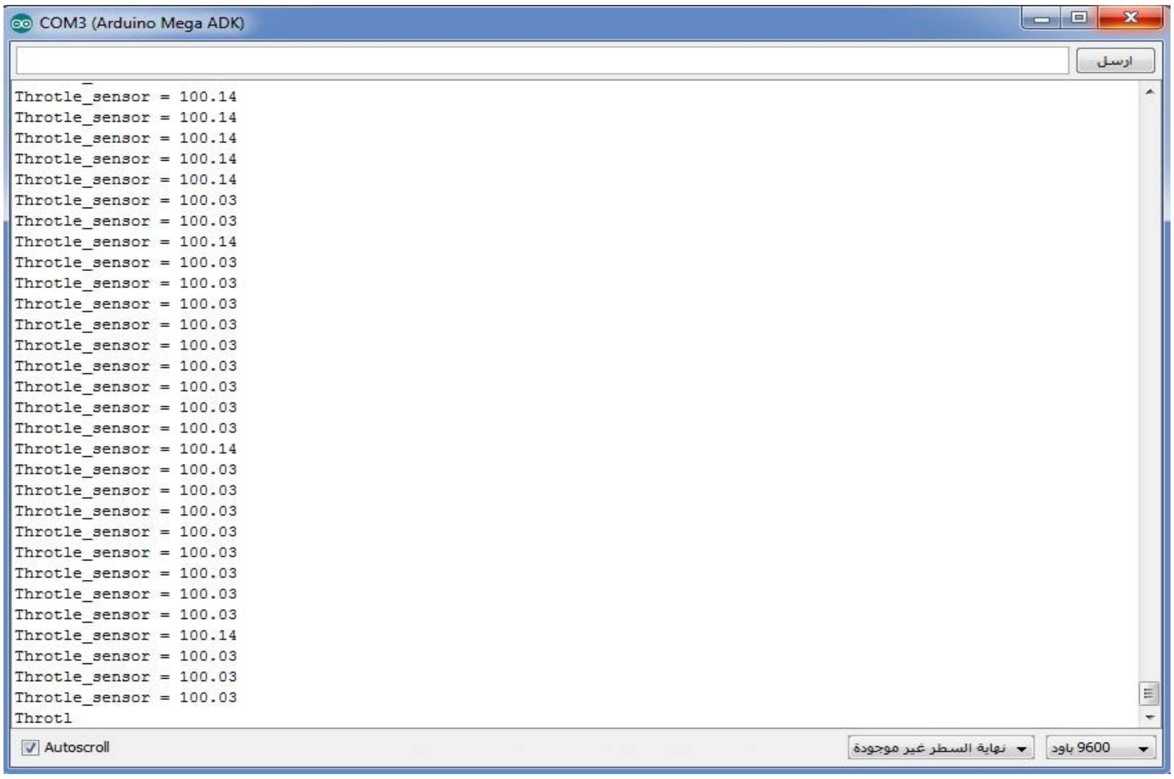


```
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.63
Throtle_sensor = 12.52
Throtle_sensor = 12.41
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.41
Throtle_sensor = 12.63
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sensor = 12.63
Throtle_sensor = 12.63
Throtle_sensor = 12.52
Throtle_sensor = 12.52
Throtle_sen
```

```
  sensorValue1 = analogRead(p);
  float pre=(19.94*((sensorValue1)*0.0048828125))+4.134;
  // sensorValue2 = analogRead(w);
  sensorValue2= analogRead(Thr);
  float THRotle=(sensorValue2*0.0048828125)*22.1;
  //  sensorValue3 = analogRead(Throtle);
  sensorValue3=analogRead(w);

  if( p==x)
{
  //Serial.print("p_sensor = " );
  //Serial.println( pre);
   Serial.print("Throtle_sensor = " );
   Serial.println( THRotle);
   //Serial.print("w_sensor = " );

}
```

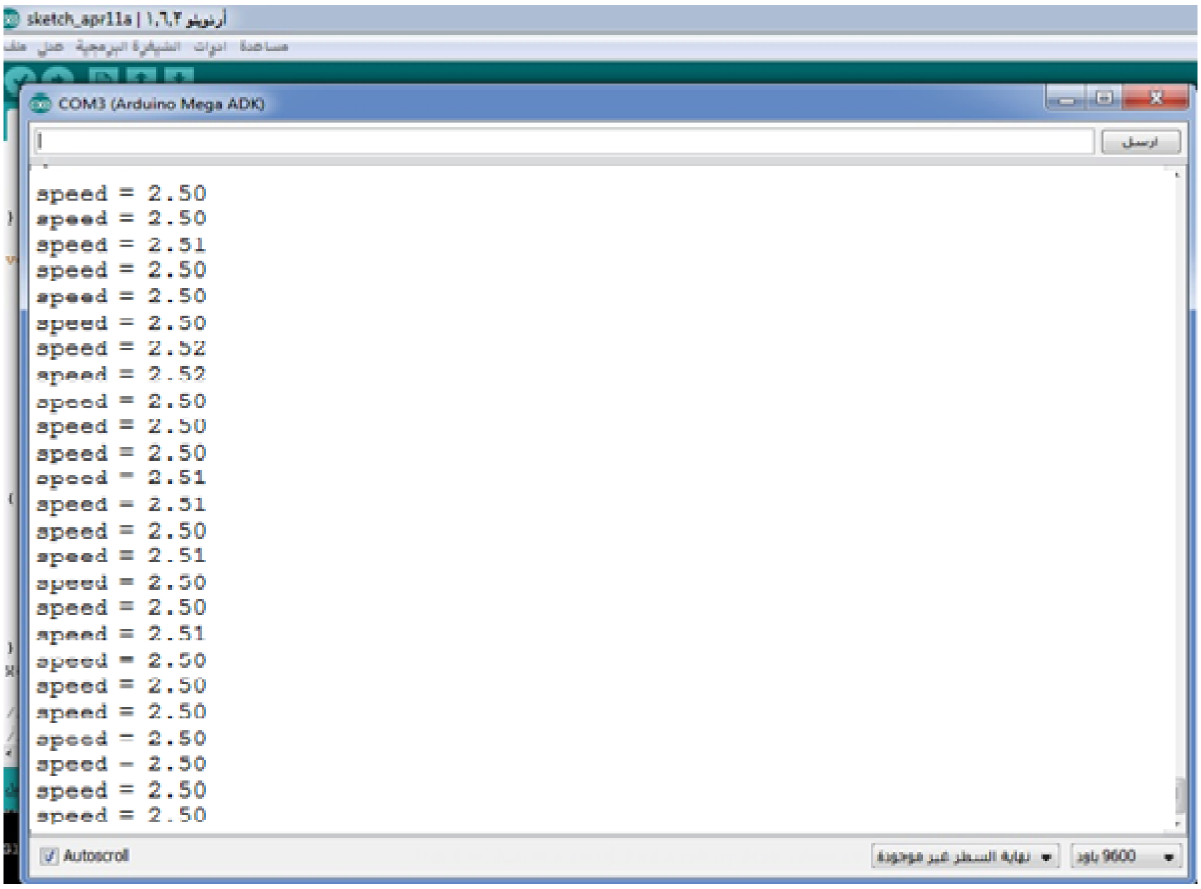
```
Global variables use 232 bytes (2%) of dynamic memory,
leaving 7,960 bytes for local variables. Maximum is
8,192 bytes.
```

Figure 4.22: Throtle closed angle



```
Throtle_sensor = 100.14
Throtle_sensor = 100.14
Throtle_sensor = 100.14
Throtle_sensor = 100.14
Throtle_sensor = 100.14
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.14
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.14
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.14
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtle_sensor = 100.03
Throtl
```

Figure 4.23: Full Throtle open angle



```
speed = 2.50
speed = 2.50
speed = 2.51
speed = 2.50
speed = 2.50
speed = 2.50
speed = 2.52
speed = 2.52
speed = 2.50
speed = 2.50
speed = 2.50
speed = 2.51
speed = 2.51
speed = 2.50
speed = 2.51
speed = 2.50
speed = 2.50
speed = 2.51
speed = 2.50
speed = 2.50
speed = 2.50
speed = 2.50
speed = 2.50
speed = 2.50
speed = 2.50
```

Figure 4.24: voltage at ideal speed engine

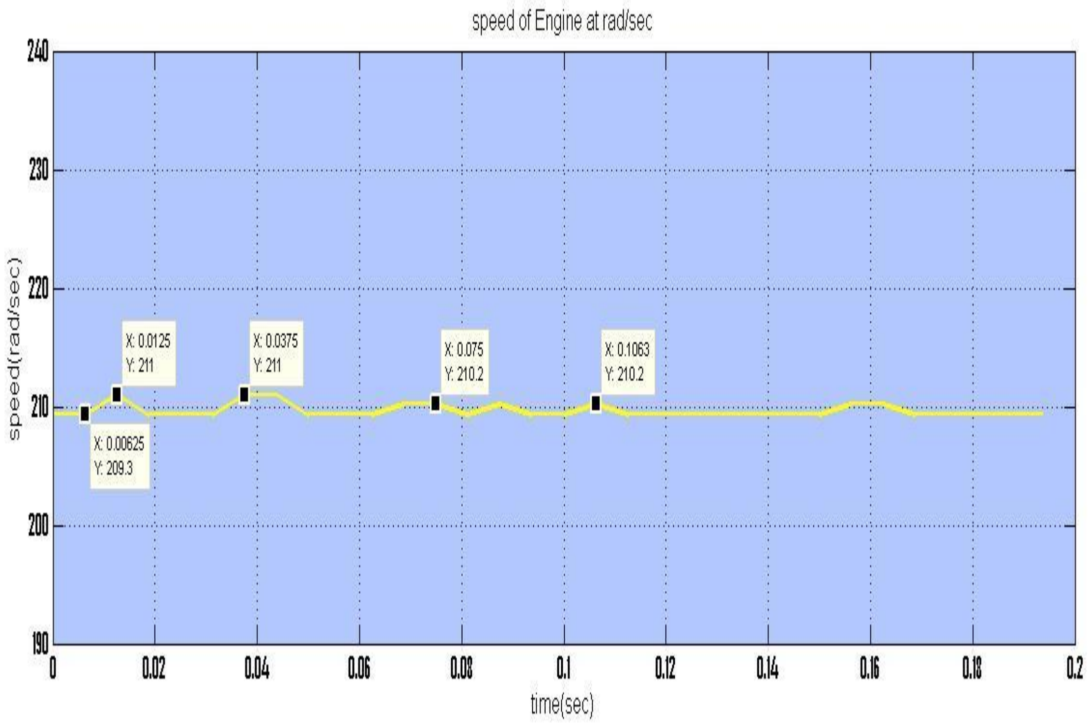
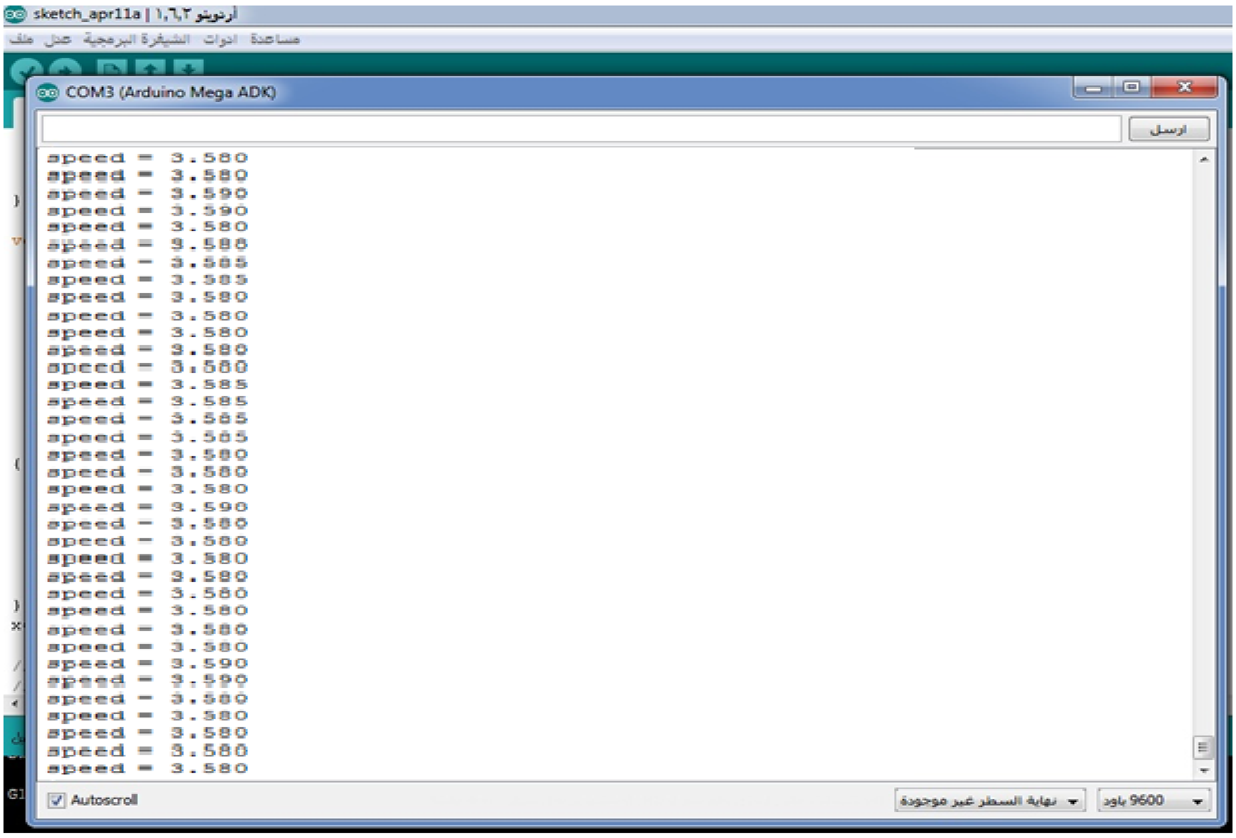


Figure 4.25: ideal speed Engine signal


sketch_apr11a | أردوينو ١,٦,٣
ملف تعدل الشيفرة البرمجية ادوات مساعدة
COM3 (Arduino Mega ADK)
ارسل

```
speed = 3.580
speed = 3.580
speed = 3.590
speed = 3.590
speed = 3.580
speed = 3.580
speed = 3.585
speed = 3.585
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.585
speed = 3.585
speed = 3.585
speed = 3.585
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.590
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.590
speed = 3.590
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
speed = 3.580
```

✓ Autoscroll   نهاية السطر غير موجودة   9600 باود

Figure 4.26: Test road speed at voltage

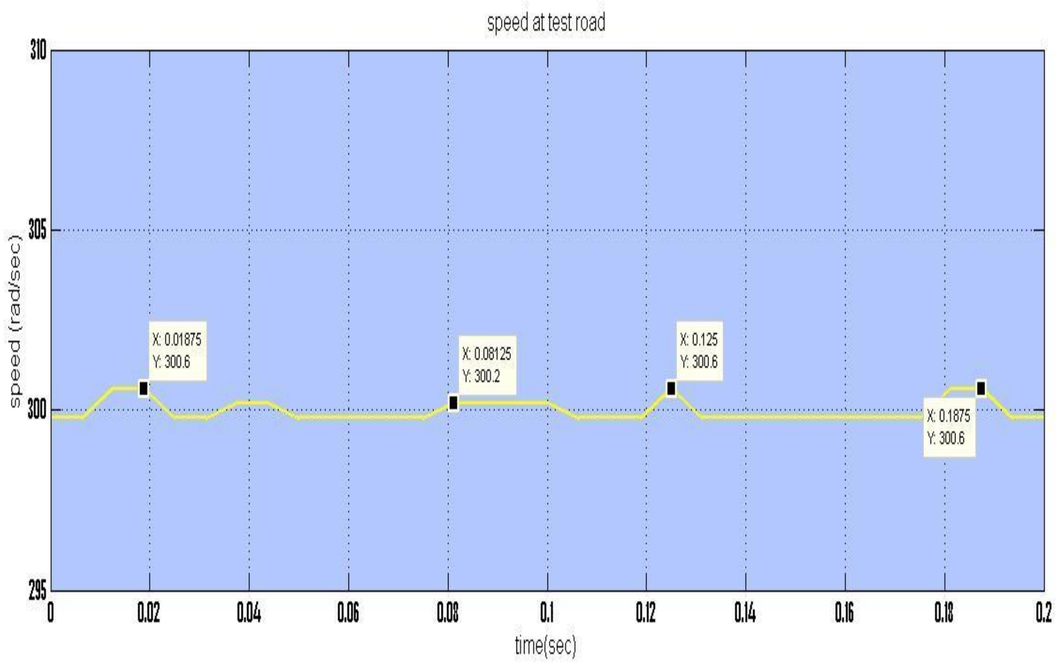


Figure 4.27:Engine speed at test road signal

CHAPTER 5

Improving For System And Final Result

Hardware and tools :

In testing stage of the system we noticed noise in generated signals and this noise cause unsatisfied results and destored signals so we decieded elimination this noise using low pass filter shows in Figure 5.1 and his equation at Equation (5.1), it help us to get satisfaying results. Figure 5.2 shows Additions to improve the hardware connect

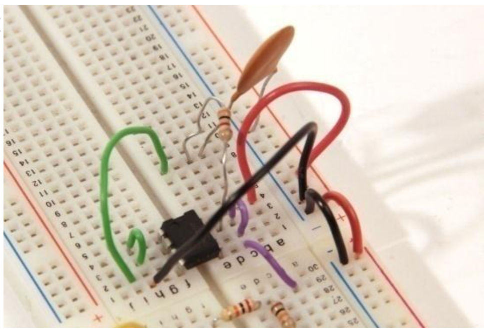


Figure 5.1: Low pass filter

$$f_c = \frac{1}{2\pi C_1 R_1} \tag{5.1}$$

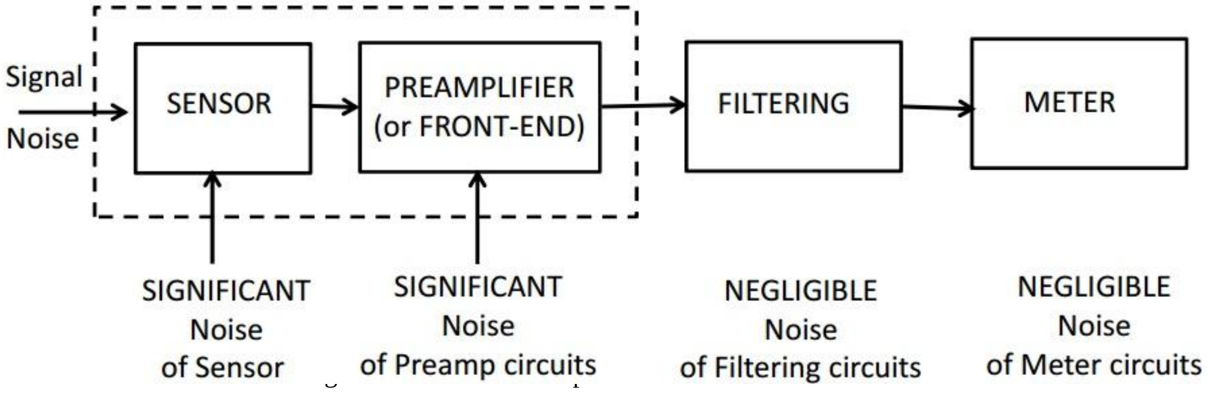
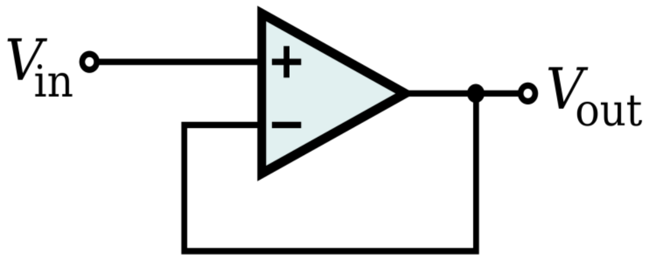

Signal
Noise
SENSOR
PREAMPLIFIER (or FRONT-END)
FILTERING
METER
SIGNIFICANT Noise of Sensor
SIGNIFICANT Noise of Preamp circuits
NEGLIGIBLE Noise of Filtering circuits
NEGLIGIBLE Noise of Meter circuits

Figure 5.2: Additions to improve the hardware connect.

Buffer Amplifier

A buffer amplifier in figure 5.3 provides electrical impedance transformation from one circuit to another. Two main types of buffer exist: the voltage buffer and the current buffer.

A voltage buffer amplifier is used to transfer a voltage from a first circuit, having a high output impedance level, to a second circuit with a low input impedance level. The interposed buffer amplifier prevents the second circuit from loading the first circuit unacceptably and interfering with its desired operation. In the ideal voltage buffer in the diagram, the input resistance is infinite, the output resistance zero (impedance of an ideal voltage source is zero). Other properties of the ideal buffer are: perfect linearity, regardless of signal amplitudes; and instant output response, regardless of the speed of the input signal[11].



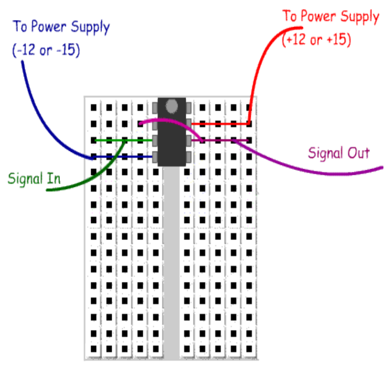


Figure: 5.3 Buffer Amplifier Circuit

Tuning parameters :

after analyses the final result we concluded need to improve and develop the system to get accurate and more suitable results.

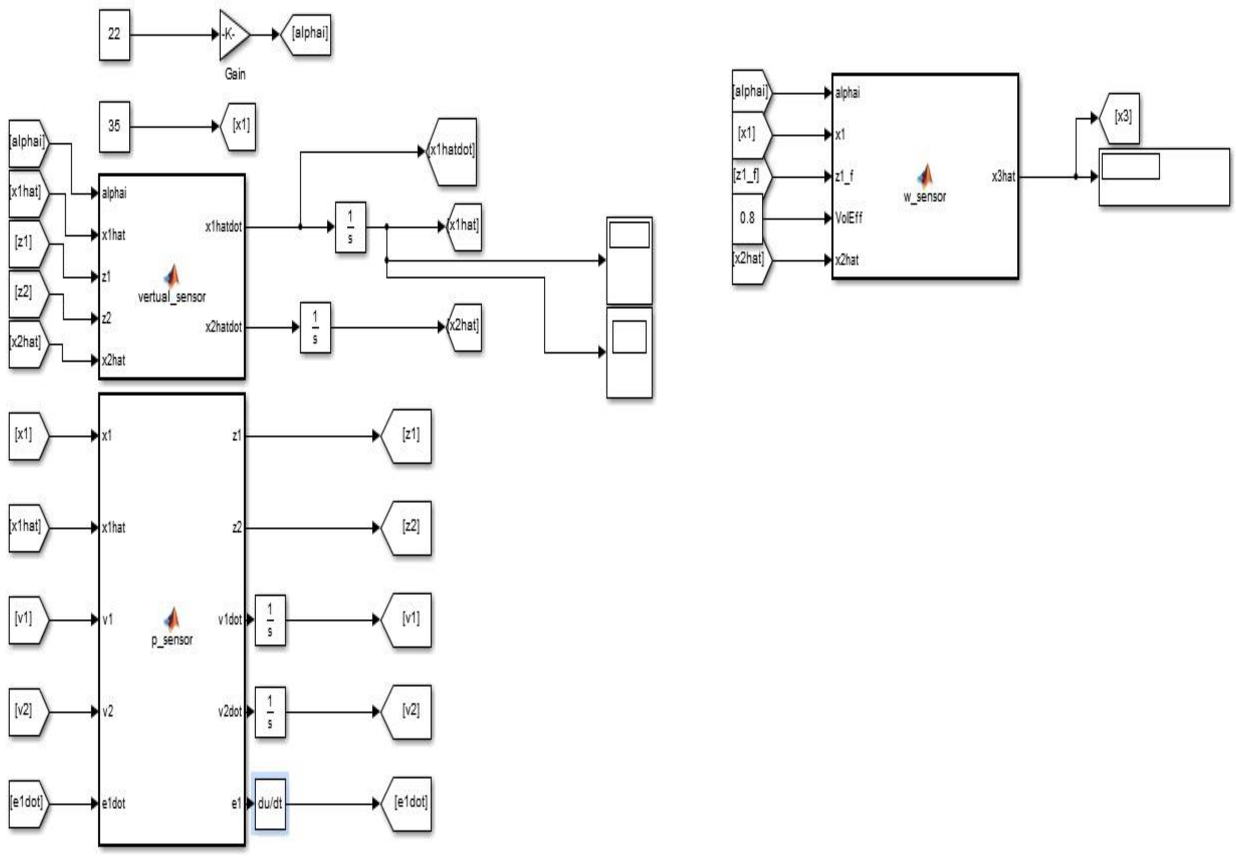


Figure 5.4: matlab simulink block after improving

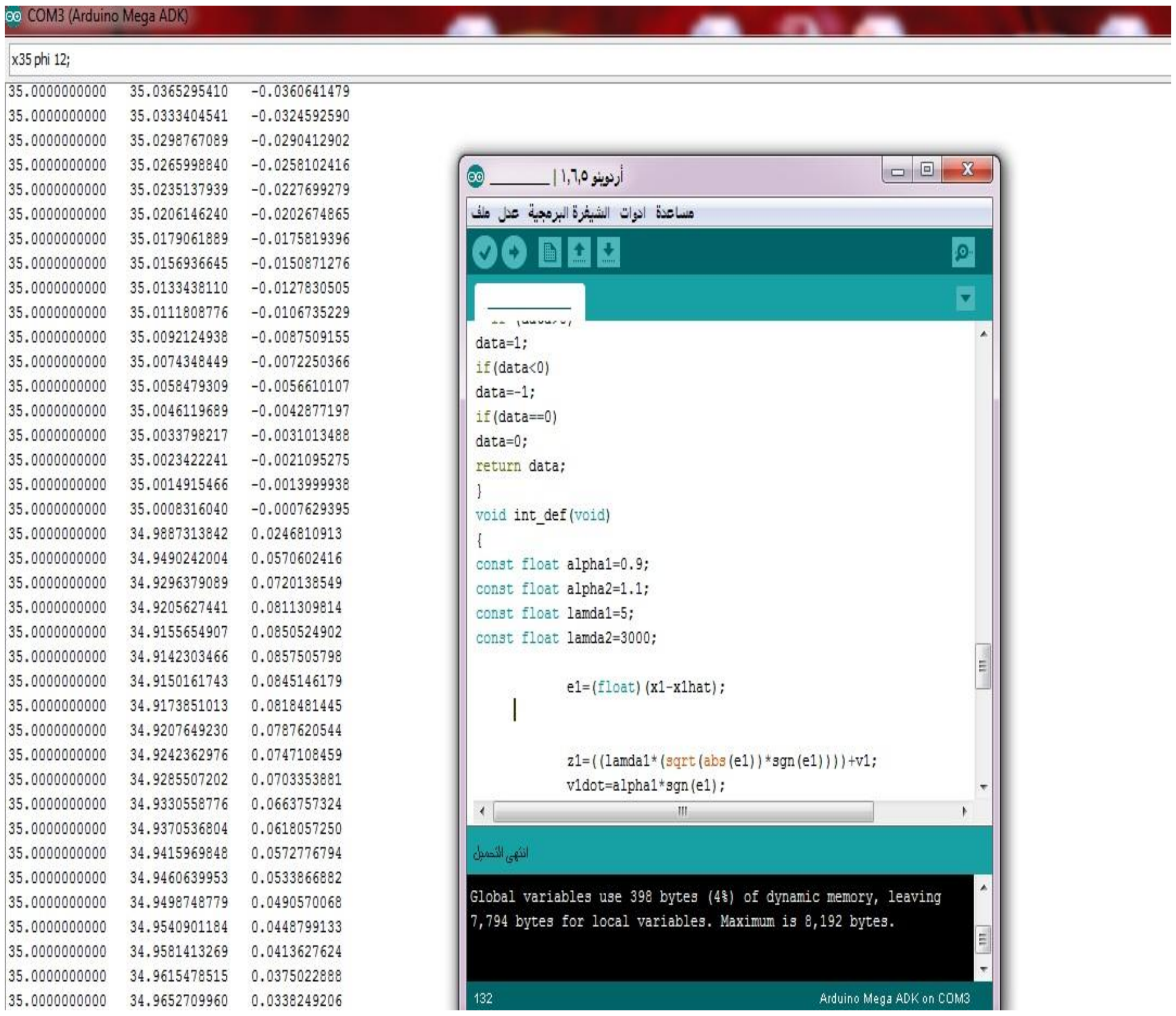
Appendix A shows the Matlab Code After removing non-required, By tuning the parameter to get the best and required results.

SOSM Observer parameters for virtual sensor

Table 5.1: Observer parameter

PARAMETER	VALUE
α1	0.9
λ1	5
α2	1.1
λ2	3000

From parameters in Table 5.2 we got to improve the system in time ,speed ,and error value. in the a new software that make improve for response at all point ,and much less settling time show in Figure 5.5.



```
x35 phi 12;
```

```
35.0000000000   35.0365295410   -0.0360641479
35.0000000000   35.0333404541   -0.0324592590
35.0000000000   35.0298767089   -0.0290412902
35.0000000000   35.0265998840   -0.0258102416
35.0000000000   35.0235137939   -0.0227699279
35.0000000000   35.0206146240   -0.0202674865
35.0000000000   35.0179061889   -0.0175819396
35.0000000000   35.0156936645   -0.0150871276
35.0000000000   35.0133438110   -0.0127830505
35.0000000000   35.0111808776   -0.0106735229
35.0000000000   35.0092124938   -0.0087509155
35.0000000000   35.0074348449   -0.0072250366
35.0000000000   35.0058479309   -0.0056610107
35.0000000000   35.0046119689   -0.0042877197
35.0000000000   35.0033798217   -0.0031013488
35.0000000000   35.0023422241   -0.0021095275
35.0000000000   35.0014915466   -0.0013999938
35.0000000000   35.0008316040   -0.0007629395
35.0000000000   34.9887313842   0.0246810913
35.0000000000   34.9490242004   0.0570602416
35.0000000000   34.9296379089   0.0720138549
35.0000000000   34.9205627441   0.0811309814
35.0000000000   34.9155654907   0.0850524902
35.0000000000   34.9142303466   0.0857505798
35.0000000000   34.9150161743   0.0845146179
35.0000000000   34.9173851013   0.0818481445
35.0000000000   34.9207649230   0.0787620544
35.0000000000   34.9242362976   0.0747108459
35.0000000000   34.9285507202   0.0703353881
35.0000000000   34.9330558776   0.0663757324
35.0000000000   34.9370536804   0.0618057250
35.0000000000   34.9415969848   0.0572776794
35.0000000000   34.9460639953   0.0533866882
35.0000000000   34.9498748779   0.0490570068
35.0000000000   34.9540901184   0.0448799133
35.0000000000   34.9581413269   0.0413627624
35.0000000000   34.9615478515   0.0375022888
35.0000000000   34.9652709960   0.0338249206
```

```
data=1;
if(data<0)
data=-1;
if(data==0)
data=0;
return data;
}
void int_def(void)
{
const float alpha1=0.9;
const float alpha2=1.1;
const float lamda1=5;
const float lamda2=3000;

        e1=(float)(x1-x1hat);


        z1=((lamda1*(sqrt(abs(e1))*sgn(e1))))+v1;
        v1dot=alpha1*sgn(e1);
```

```
Global variables use 398 bytes (4%) of dynamic memory, leaving
7,794 bytes for local variables. Maximum is 8,192 bytes.
```

Figure 5.5: Serial monitor data.

After downloading the code and run microcontroller, from serial monitor we plot the response and error at two point using last matlab code as shows in Figure 5.6 and Figure 5.8.

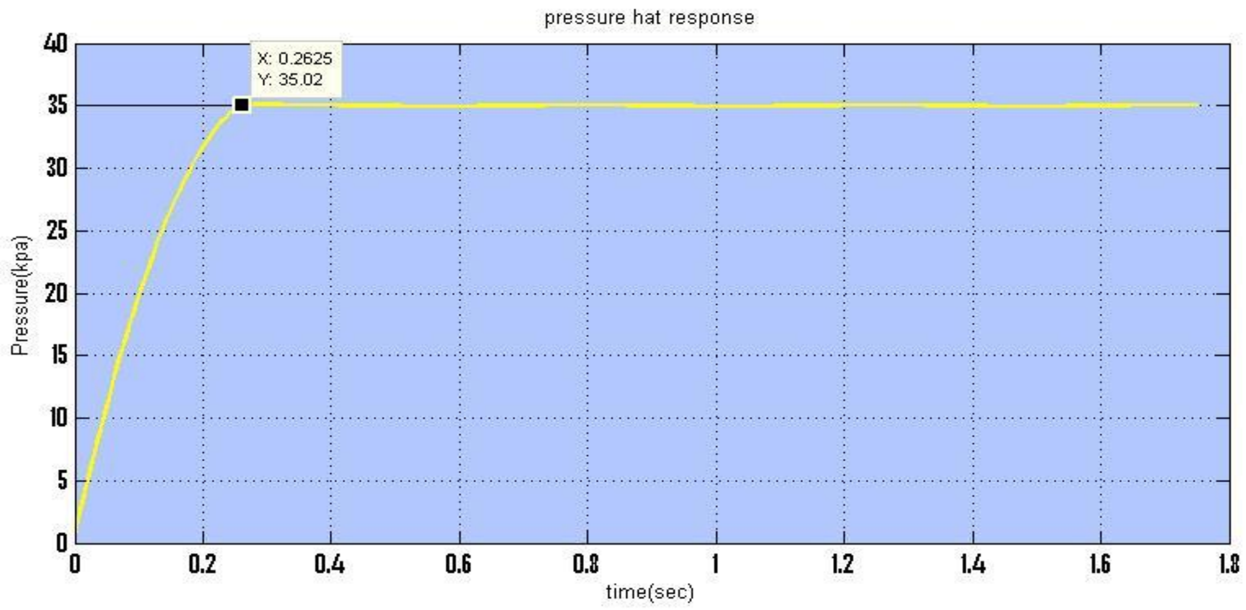
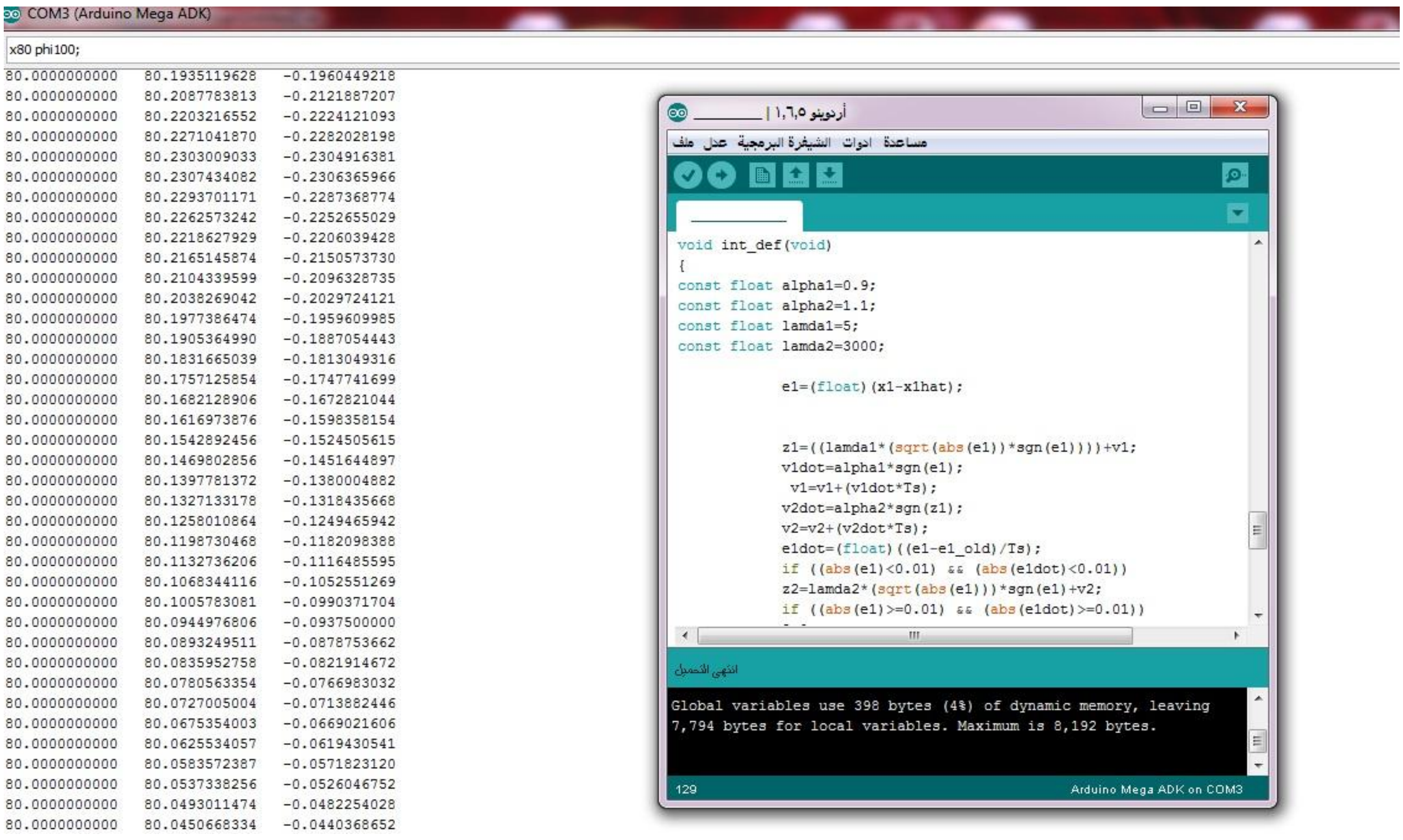


Figure 5.6: pressure hat responce at real pressure35 Kpa.

We also note that It became the best properties, for example the settling time approximate equal 0.25 second at pressure equal 35 kpa and the value of error nearly zero. In figure 5.7 shows the data of serial monitor by arduino software at test road presure 80 kpa.



```
x80 phi100;
80.0000000000   80.1935119628   -0.1960449218
80.0000000000   80.2087783813   -0.2121887207
80.0000000000   80.2203216552   -0.2224121093
80.0000000000   80.2271041870   -0.2282028198
80.0000000000   80.2303009033   -0.2304916381
80.0000000000   80.2307434082   -0.2306365966
80.0000000000   80.2293701171   -0.2287368774
80.0000000000   80.2262573242   -0.2252655029
80.0000000000   80.2218627929   -0.2206039428
80.0000000000   80.2165145874   -0.2150573730
80.0000000000   80.2104339599   -0.2096328735
80.0000000000   80.2038269042   -0.2029724121
80.0000000000   80.1977386474   -0.1959609985
80.0000000000   80.1905364990   -0.1887054443
80.0000000000   80.1831665039   -0.1813049316
80.0000000000   80.1757125854   -0.1747741699
80.0000000000   80.1682128906   -0.1672821044
80.0000000000   80.1616973876   -0.1598358154
80.0000000000   80.1542892456   -0.1524505615
80.0000000000   80.1469802856   -0.1451644897
80.0000000000   80.1397781372   -0.1380004882
80.0000000000   80.1327133178   -0.1318435668
80.0000000000   80.1258010864   -0.1249465942
80.0000000000   80.1198730468   -0.1182098388
80.0000000000   80.1132736206   -0.1116485595
80.0000000000   80.1068344116   -0.1052551269
80.0000000000   80.1005783081   -0.0990371704
80.0000000000   80.0944976806   -0.0937500000
80.0000000000   80.0893249511   -0.0878753662
80.0000000000   80.0835952758   -0.0821914672
80.0000000000   80.0780563354   -0.0766983032
80.0000000000   80.0727005004   -0.0713882446
80.0000000000   80.0675354003   -0.0669021606
80.0000000000   80.0625534057   -0.0619430541
80.0000000000   80.0583572387   -0.0571823120
80.0000000000   80.0537338256   -0.0526046752
80.0000000000   80.0493011474   -0.0482254028
80.0000000000   80.0450668334   -0.0440368652
```

```
void int_def(void)
{
const float alpha1=0.9;
const float alpha2=1.1;
const float lamda1=5;
const float lamda2=3000;

            e1=(float)(x1-x1hat);


            z1=((lamda1*(sqrt(abs(e1))*sgn(e1))))+v1;
            v1dot=alpha1*sgn(e1);
             v1=v1+(v1dot*Ts);
            v2dot=alpha2*sgn(z1);
            v2=v2+(v2dot*Ts);
            e1dot=(float)((e1-e1_old)/Ts);
            if ((abs(e1)<0.01) && (abs(e1dot)<0.01))
            z2=lamda2*(sqrt(abs(e1)))*sgn(e1)+v2;
            if ((abs(e1)>=0.01) && (abs(e1dot)>=0.01))
```

```
Global variables use 398 bytes (4%) of dynamic memory, leaving
7,794 bytes for local variables. Maximum is 8,192 bytes.
```

Figure 5.7: serial monitor data at pressure 80Kpa

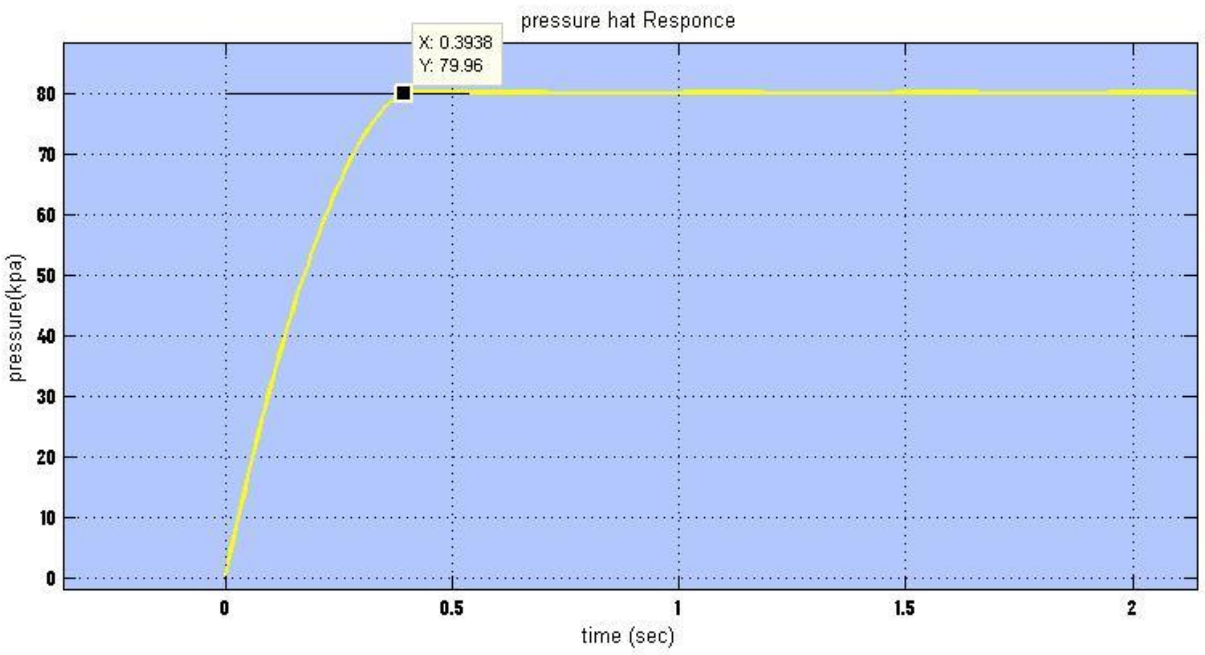


Figure 5.8: pressure hat responce at test road pressure

In pressure 80 kpa the settling time is greater from last pressure 0.39 second and the exist a small error value that show in last figure.

Figure 5.9 and Figure 5.10 that show the error curve at two point , ideal speed pressure and test road pressure, We would like to remember the estimation technique which depends on the error value, the value of injector that make compensation of error value addition of estimation state to reach the actual state   by super twisting algorithm, parameter of SOSM observer governing the converge to zero ,speed of error and value of final percentage error.

the curve show the value of initial error, large value at starting estimation but by twisting Swinging becomes even up to a value equal to zero to get the correct prediction in estimation.

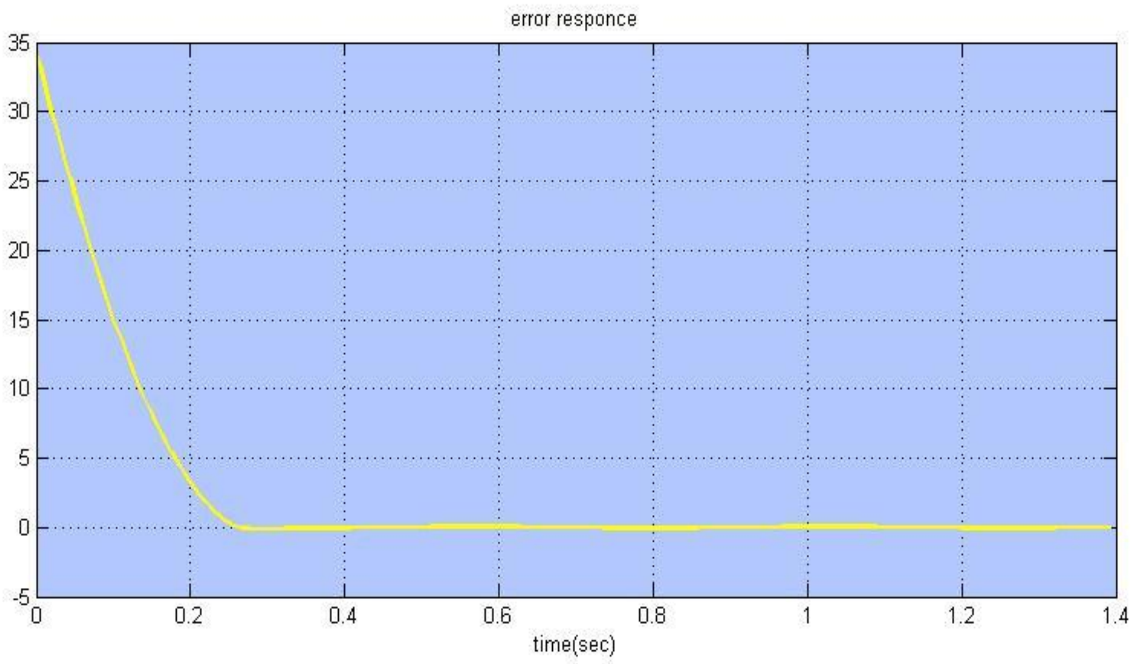


Figure 5.9: Error responce at pressure 35Kpa

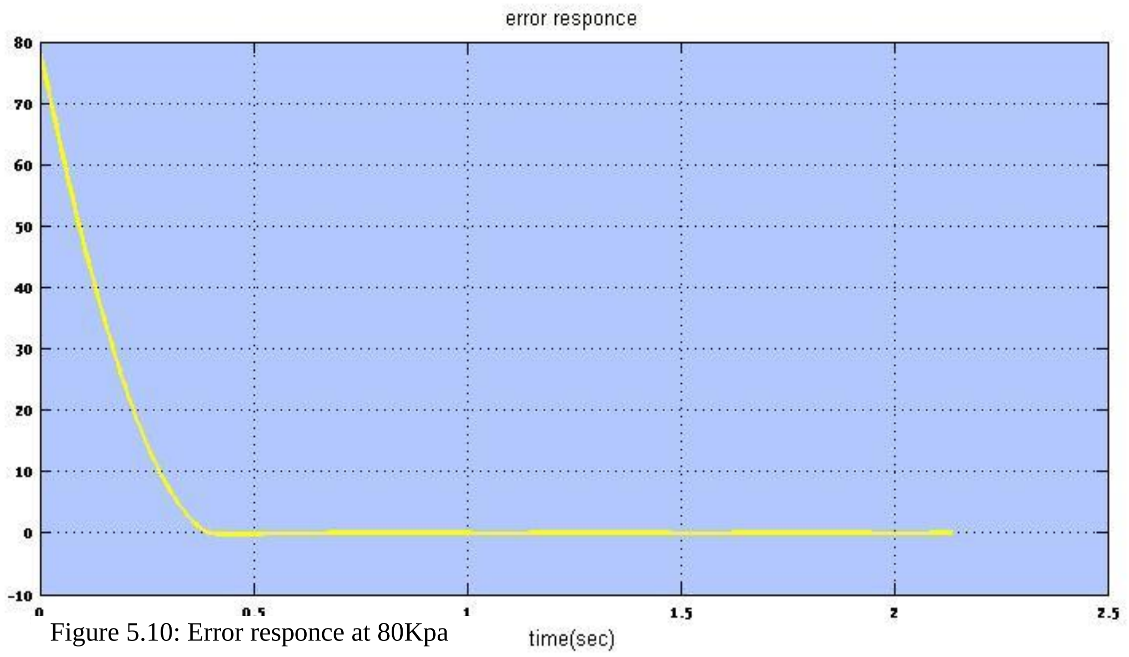


Figure 5.10: Error responce at 80Kpa

conclustion

- After trail and appropriateness requirmente of the project virtual sensor for speed and pressure sensor with the car of opel corsa engine 1.4L ,and take the reading of each sensor we found that reading are identical with the real reading theory.

- The second order sliding mode observer used in the vechile sensor to give value of speed sensor from pressure sensor or pressure from speed , which save the time and effort in finding the value of pressure , and improve the efficent of engine in all conditions from the value of speed acucuratly and exllent skill.

- where looking secand sliding mode observer in fault handling where ther is adefect in the sensor.
- In the final step in the project the problem that we face , the setling time and value of error but by Tuning parameters we have in the last ,the value of parameter that make the better condition.
- After the implementation of the project by using this method ,sliding mode can we using anather method like kalman filter.

APPENDIX A

Matlab Code

```
                    %%First Embedded Matlab function block


function
[x1hatdot,x2hatdot]=vertual_sensor(alphai,x1hat,z1,z2,x2hat)
%#eml

pa=101325;               %%% Atmospheric pressure Pa
D=0.054;                 %%% Throttle diameter  m
alphacl=0.1709;          %%% Throttle angle at closed postion
R=0.287;                 %%% Specific gas constant
Tm=325;                  %%% Manifold Temperature K
Vm=0.1127 ;              %%% Manifold Volume m^3
Cd=0.004;                %%% Throttle Disharge cofficient
Vd=0.0172409362;         %%% Displaced volume

Ta=298;                  %%%  Ambient pressure pa
Y=1.4;                   %%% Ratio of heat capacities
AE=((pi*D*D)/4)*(1-cos((alphai+alphacl)/alphacl));

Yc=sqrt(1/(R*Ta))*sqrt(Y*(((2/(Y+1))^((Y+1)/(Y-1)))));
A1=(R*Tm/Vm)*AE*pa*Cd*Yc;
            %%X1hat and X2hat calculated
f_x1hat=(1-exp(9*(x1hat/pa)-1));
A3hat=(9*A1/pa)*(exp((x1hat/pa)-1));
A4hat=A1*f_x1hat*A3hat;
x1hatdot=x2hat+z1;
x2hatdot=A4hat+z2;

                         %%Second Embedded Matlab function block
function
[z1,z2,v1dot,v2dot,e1]=p_sensor(x1,x1hat,v1,v2,alphai,e1dot)
%#eml

alpha1=0.9;
alpha2=1.1;
lamda1=5;
lamda2=3000;
e1=x1-x1hat;
z1=lamda1*((abs(e1))^0.5)*sign(e1)+v1;
v1dot=alpha1*sign(e1);
v2dot=alpha2*sign(z1);

if abs(e1)<0.01  && abs(e1dot)<0.01
        z2=lamda2*((abs(e1))^0.5)*sign(e1)+v2;

else
```

```
    z2=0;

-end;

         %%Third Embedded Matlab function block

                        Volumetric efficiency
function VolEff = volumetric_efficiency(z2_f,x1,x3,x4,alphai)
%#emll
Yc=sqrt(1/(R*Ta))*sqrt(Y*(((2/(Y+1))^((Y+1)/(Y-1)))));
A1=(R*Tm/Vm)*AE*pa*Cd*Yc;
A2=Vd/(Vm*4*pi);
f_x1=(1-exp((x1/pa)-1));
A3=(9*A1/pa)*(-exp((x1/pa)-1));
A5=A3*A2*x1*x3;
A6=A1*A2*x3*f_x1;
A7=x1*((A2*x3)^2);
A8=A2*x1*x4;
A4=A1*A3*f_x1;

VolEff=((A5+A6+A8)+(sqrt((A5+A6+A8)^2+(4*A7*z2_f))))/(A7);

           %%x3hat Embedded Matlab function block

function x3hat = w_sensor(alphai,x1,z1_f,VolEff,x2hat)
%#eml


Yc=sqrt(1/(R*Ta))*sqrt(Y*(((2/(Y+1))^((Y+1)/(Y-1)))));
A1=(R*Tm/Vm)*AE*pa*Cd*Yc;
f_x1=1-exp(9*(x1/pa)-1);
A9=A1*f_x1;
A2=Vd/(Vm*4*pi);
x3hat=(1/(VolEff*A2*x1))*(A9-(z1_f+x2hat));
```

APPENDIX B

Aurdino Code

```
                        //Constant

const float pi=3.14;

const float pa=101325;                  // Atmospheric pressure

const float D=0.054;                    // Throttle diameter

const float alphacl=0.1709;            //Throttle angle at closed

const float R=0.287;                    //Specific gas constant

const float Tm=325;                   //Manifold Temperature

const float Vm=0.1127 ;            // Manifold Volume

const float Cd=0.004;                 //Throttle Disharge t

const float Ta=298;                // Ambient pressure

const float Vd=0.0172409362;    // Displaced volume

const float Y=1.4;               // Ratio of heat capacities


                //definition of variables


float AE=0.0;

float Yc=0.0;

float v1dot=0.0;

float v1=0.0;

float v2dot=0.0;

float v2=0.0;
float e1=0.0;
float e1dot=0.0;
float f_x1hat=0.0;
float x1hat=0.0;
float x1hatdot=0.0;
float x2hatdot=0.0;
float A3hat=0.0;
float A4hat=0.0;
float x2hat=0.0;
float z1=0.0;
float z2=0.0;
float alphai=0.0;
float Thro=0.0;
float P=0.0;
float x1=0.0;
const float alphaisensor = A0;       //Throtle sensor Analog Pin
const float pressure_sensor=A2;      //pressure sensor Analog pin

void setup() {
Serial.begin(9600);
}
void loop()
{
                                // EXTERNAL INPUT
Thro=analogRead(alphaisensor); // Analog read input data
float alphai=(float)((Thro/1024)*5)*22.5;//convert volt to deg
P=analogRead(pressure_sensor);    //Analog pressure read
float X1=(19.94*((sensorValue1)*(P/1024)*5))+4.134

                                // convert volt to pressur

                           //Main fanction

const float alpha1=0.9;

const float alpha2=1.1;

const float lamda1=5;

const float lamda=3000;

            e1=x1-x1hat;

            z1=lamda1*(pow(abs(e1),0.5))*sin(e1)+v1;

            v1dot=alpha1*sin(e1);

            v2dot=alpha2*sin(z1);

            if ((abs(e1)<0.01) && (abs(e1dot)<0.01))

            z2=lamda2*(pow(abs(e1),0.5))*sin(e1)+v2;

            else

            z2=0;

AE=((pi*D*D)/4)*(1-cos((alphai+alphacl)/alphacl));


Yc=sqrt(1/(R*Ta))*sqrt(Y*pow((2/(Y+1)),((Y+1)/(Y-1))));

               f_x1hat=(1-exp(9*(x1hat/pa)-1));

               A3hat=(9*A1/pa)*(exp((x1hat/pa)-1));

               A4hat=A1*f_x1hat*A3hat;

A1=(R*Tm/Vm)*AE*pa*Cd*Yc;
f_x1=1-exp(9*(x1/pa)-1);
A9=A1*f_x1;
A2=Vd/(Vm*4*pi);
x3hat=(1/(VolEff*A2*x1))*(A9-(z1_f+x2hat));


}
```

References

[1] R. Stone, Introduction to internal combustion engines, Gas*,* 2012 (1999),pp. 05-10.

[2] E. F. Obert, Internal combustion engines and air pollution, (1973),

[3] G. Honig, U. Kiencke, and R. Bone, "System and apparatus to determine the angular position, and speed of a rotating shaft, particularly crank shaft of an internal combustion engine," Google Patents, 1982.

[4] P. C. Young, P. McKenna, and J. Bruun, Identification of non-linear stochastic systems by state dependent parameter estimation, International Journal of Control*,* 74 (2001),pp. 1837-1857.

[5] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*, Springer Publishing Company, Incorporated, 2012.

[6] C.-M. Ong, *Dynamic simulation of electric machinery: using MATLAB/SIMULINK*, Prentice Hall PTR Upper Saddle River, NJ, 1998.

[7] U. Kiencke, A view of automotive control systems, IEEE Control Systems Magazine*,* 8 (1988),pp. 11-19.

[8] M. S. Roden, Analog and digital communication systems, Englewood Cliffs, NJ, Prentice Hall, 1991, 521 p.*,* 1 (1991),

[9] A. G. Smith, "Introduction to Arduino," September, 2011.

[10] Y. Saito, "Noise filter," Google Patents, 1987.

[11] S. Özoğuz, A. Toker, and C. Acar, Current-mode continuous-time fully-integrated universal filter using CDBAs, Electronics Letters*,* 35 (1999),pp. 97-98.