POLITECNICO DI TORINO

Department of Control and Computer Engineering

MASTER DEGREE

IN

MECHATRONICS ENGINEERING

Didactic Test-Bench of a Car's Engine

Tutors:

Prof. Raffaella Sesana

Prof. Daniela Maffiodo

Candidate

HAN YING

December 2018

Abstract

The car's engine bench test is an important part of the engine development phase. It is not only used to verify the reliability of the whole engine and related components, but also to verify whether the engine performance has reached the initial design specifications. The test-bench is mainly composed of a measurement control system, a test object(engine) and related auxiliary systems. When the bench test of the whole machine is carried out, the engine is not equipped with a radiator or a fuel tank, so an auxiliary system is needed instead of these components, so that the engine can perform complete water circulation, supply of fuel system. For a beginner, the best way to understand how the engine works and status is intuitive observation of an engine's work. This is the main purpose of the research project, developing a test-bench for didactic and giving the chance to the students for hands-on the engine and the testing procedure. The entire project consists of three phases: the development of the simulation test-bench with an embedded system, the development of the real test-bench, and the testing of the engine and data analysis. The project was jointly developed by a three-person team, and this thesis mainly reports the development of the simulation part and the GUI. The results of the research are available for the students of daily didactic. This research project collaborated by the institution Filos and Politecnico di Torino.

Keywords: test-bench, didactic, car's engine, embedded system, GUI, data measure, data analysis

Table of contents

List of Figures

List of Tables

List of Symbols

$\dot{m}_{ai}$	Mass flow rate into manifold	g/s
θ	Throttle angle	deg
P_m	Manifold pressure	bar
P_{amb}	Ambient (atomosperic) pressure	bar
R	Specific gas constant	
T	Temperature	K
V_m	Manifold Volume	m^3
$\dot{m}_{ao}$	Mass flow rate of air out of the manifold	g/s
$\dot{P_m}$	Rate of change of manifold pressure	bar/s
N	Engine angular speed	rad/s
P_m	Manifold pressure	bar
m_a	Mass of air in cylinder for combustion	g
A/F	Air to fuel ratio	
σ	Spark advance	degree
$Torque_{eng}$	Torque produce by engine	Nm
J	Engine rotational moment of inertia	$\text{kg} \cdot m^2$
$\dot{N}$	Engine angular acceleration	rad/s^2

CHAPTER 1: Introduction

The engine is used in a wide range of applications. Various transportation vehicles (cars, agricultural vehicles, diesel locomotives) on the ground, power stations, agricultural machinery, forestry machinery, mining, petroleum, construction and engineering machinery all use the internal combustion engine as the prime mover. Water transportation can be used as the main engine and auxiliary machine for inland and marine vessels. In aviation, some small civilian aircraft are still powered by internal combustion engines.

The engine is the source of power for the vehicle. Due to its complicated structure and poor working conditions, the failure rate is very high, which often becomes the target of key detection. The comprehensive performance test of the engine can provide a basis for scientific diagnosis of engine technical conditions.

Whether the performance of the engine is excellent or not directly affects the performance of the vehicle. For test various performance indexes of the engine and various performance curves, it is usually carried out on an engine test-bench.

1.1 Background

Usually, when teaching the engine structure and working principle, the teacher can only use textbooks or pictures to display, which is difficult for students, especially high school students, to understand.

The test-bench of car's engine is very common at the engine manufacturers. But in general it is very difficult to use, its construction is very complicated, and it is very large, Such as SIMPRO Engine Test Stand, DTC Engine Test Benches, etc. They cannot be used for the teaching work.

1.2 Purpose

Under such circumstance, this project is dedicated to the development of an engine test-bench for didactic. This test-bench must have a clear structure, easy to operate, cost effective and not very big. Another most important feature is to ensure the accuracy of the test bench measurement.

For achieve this purpose, we can consider that design a simulation test bench before building the real test bench. Building the simulation test bench only need an Arduino board, some sensors and electronic components, even could a set of each group to learn engine. It can simulate most of the functions of the real test bench abstractly.

Also, for the real didactic test bench, considering that it is used at classroom, we have to design a suitable stand which can mount all the components, the engine, the electrical parts, the mechanical parts, the cooling system, fuel tank, etc.

1.3 Components

The project mainly consists three parts:

- Simulation of the test-bench

1. the Arduino embedded systems as an ECU
2. the DC motor, servo motor, rpm sensor, temperature sensor and the like, to simulate engine test-bench
3. test-bench GUI with LabView

- The real test-bench

1. the engine test-stand
2. the mechanical part of the engine
3. the electrical part of the engine

- measure and analysis

1. Bosch Diagnostics electronic board
2. Multiecuscan Diagnostics electronic board

1.4 Methods

Considering that the test bench is developed aimed at high school students, before build the real test-bench, it is better design a simulation test-bench. Using the Arduino board to develop embedder system as an ECU, then using such as a DC motor, servo motor, rpm sensor to building the simulation engine electrical circuit. Designing a GUI with LabView to measure and storage data.

For the real engine test-bench, we designed the engine test-stand at first. Then mounting the engine and all the mechanical part include the cooling system, fuel tank etc. Then mapping all the electrical part according to the experience sheet, include the fuses and relays protect system.

Using the Multiecuscan Diagnostics electronic board to measure and storage data when engine is running.

1.5 Individual contribution

This research project was developed by a three-person team, and everyone involved in every aspect of development, but the focus is different. Therefore, this paper will mainly introduce the part that I am mainly responsible for, briefly introduce other parts, as follows:

- Development of embedded systems with Arduino
- Design of simulation electronic circuit
- Implementation of electronic circuits
- Development of GUI with LabView
- Development and maintenance all the program.

CHAPTER 2: Bibliographic review

After a deep understanding of the purpose of this project, it is very important to do some research to make a master plan before start developing the project. The selection of the engine, the selection of the embedded system, even what kind of software to be used.

2.1 Embedded system

As the control core of the engine working, Engine Control Unit(ECU) is the most important part. So for the simulation part, considering use an embedded system to implement the functions of ECU.

An embedded system is a programmed controlling and operating system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. The properties of a typical embedded system are low power consumption, small size, rugged operating ranges, and low per-unit cost.

For this project, the embedded systems are combination of hardware and software.

2.1.1 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. There are many advantages with this board for our project:

- low cost
- small size
- fast processing speed
- enough available resources
- open source
- Easy to program

2.1.2 Arduino Software (IDE)

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the Arduino board.

It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and others open-source softwares. This software can be used with any Arduino board.

- free to use
- open source
- multi-programming language support
- many libraries and functions

2.2 Simulation of the didactic test bench

The construction and operation of the engine can be simulated with circuits made up of simple electronic components.

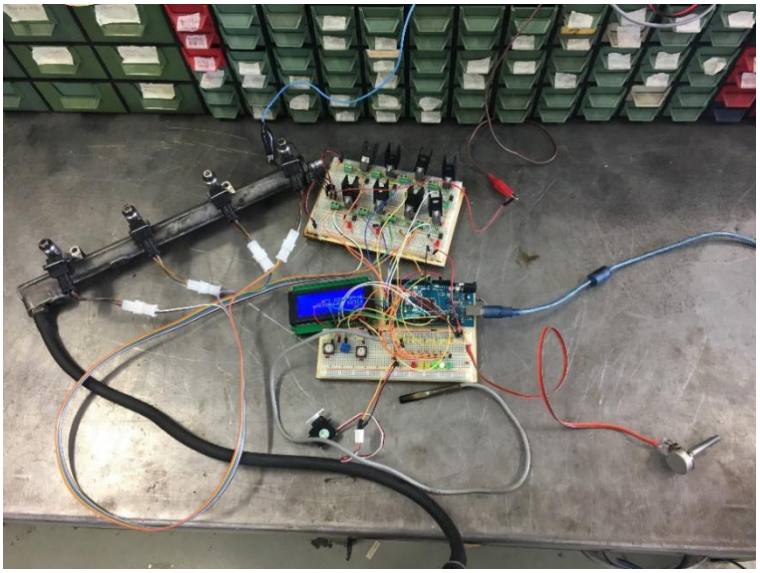


Figure 2-1 Simulation of the didactic test bench

Comparing the real engine test bench, the simulation one has the following advantages:

- very small and even could put on the desk
- very low cost
- only need 12V power supply

- good security

2.3 Graphical User Interface

For giving students a clearer understanding of the working state of the engine, we need sample more details data and display them with a physical curve. Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment for a visual programming language from National Instruments.

With this software we can develop a graphical user interface to sample and display all the data from the simulation didactic test bench easily. There are a lot of advantages with LabView:

- low cost(provide a free educational version)
- graphical programming language
- extensive support for interfacing to devices
- many libraries and functions

2.4 Engine test-stand

The engine test stand is a stand where the engine set, include the starting circuit, protect circuit, cooling system, fuel tank, etc. Obviously, have a good stability is an important parameter for the test stand. Also, it is a didactic test bench, so you must try to control the volume is not too big.

Consequently, for designing a test stand that will be the suitable solution for this specific project, it has to define some parameters. The parameters are listed as follows:

- low cost
- common materials
- reasonable structure
- strong stability
- strong security

2.5 Engine

The engine used in the test bench is taken from the vehicle Fiat, model punto. The volume of cylinders is 1.2L, with four cylinders and four strokes of piston per cylinder, and the combustion is described as internal, spark ignition. The specifications of this engine are:

- low cost but running well
- a classical four-stroke internal combustion engine
- structural integrity

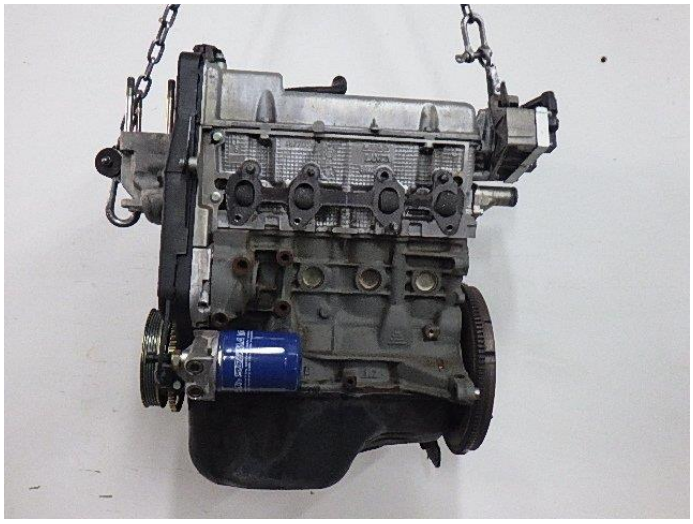


Figure 2-2 Fiat punto 1.2 8V

2.6 Sensors, actuators and auxiliary system

For the engine to work properly, all the sensors, actuators and auxiliary system must work well. The ECU obtains the working state of the engine through all the sensors, and then control the engine by the actuators. The auxiliary system provides fuel and protection for the engine.

The sensors list for this project is as follows:

- The Lambda sensor
- The RPM sensor

- The throttle
- The ambient temperature sensor
- The temperature sensor of the engine
- The knock sensor

The actuators list for this project is as follows:

- The injectors
- The canister
- The transformers
- The stepper motor
- The spark plugs

The auxiliary system list for this project is as follows:

- Cooling system
- Fuel supply system
- Ignition system
- Lubrication system

2.7 Sampling and analysis

When testing the engine on a car, using an electronic control unit diagnostic board which model is Bosch KTS-570. It is a professional test equipment for modern vehicle electronics.

For the test bench measurement, using a Multiecuscan diagnostic electronic board. And both two boards need combined with the aligned software: ESI and Multiecuscan.

CHAPTER 3: Modeling and simulation

It is well known that modeling and simulation of system is an important method and prerequisite for researching systems. It is a substitute for physical experimentation. In this way actual experimentation can be avoided which is costly and time consuming instead of using mathematical knowledge and computer's computation power to solve real world problems cheaply and in time efficient manner. For this research project, we can design a simulation of the didactic test bench at first.

3.1 Simulation of the ECU

Humans have brains, computers have CPUs, and cars have ECUs. Obviously, if you compare a car to a person, then the four wheels are the human limbs, the body and the chassis are human bones, the various circuits and circuits are the feedback nerves of the human, and the ECU is the control person. a brain of thinking and behavior. Many times when you look at a person who is smart, it depends on whether his brain is good or not. Looking at the performance of a car, most people look at its engine performance. In fact, the most important part of determining the performance of the vehicle is its ECU.

The ECU is the abbreviation of the Engine Control Unit. It can also be called the “driving computer”. As one of the core components of modern automotive electronics, ECU electronic control units may be several in the car, each managing different functions; and there is information exchange between each ECU system. Although the control system on the whole vehicle is more and more complicated, it still has to have the most basic structure—microprocessor (CPU), memory (ROM, RAM), input/output interface (I/O), analog-to-digital conversion. (A/D) and large-scale integrated circuits such as shaping and driving. It is also well recognized in appearance - there is a control element shaped like a square box in the engine electronic fuel injection system, which is the ECU. There are many fine slots around the components to connect the numerous input and output circuits. Together with other electronic control components, they form the brain's nerve center system, which monitors the input data (such as brakes, shifts, etc.). And various states of operation of the car (acceleration, slip, fuel consumption, etc.), and calculate the information sent by various sensors according to a pre-designed program. After processing, each parameter is sent to each relevant actuator to execute various control function.

The operating voltage range of the ECU is generally 6.5~16V, the working current is 0.015~0.1A, the working temperature is -40~ +80 degrees, and it can withstand vibration below 1000Hz. Therefore, the probability of ECU damage is very small. In addition to processing digital signals, it also has features of fault self-diagnosis, protection and learning. When the system fails, it can automatically record the fault in the memory RAM and take protective measures to keep the engine running by reading the program

in the memory so that the car can drive to the repair shop. Under normal circumstances, RAM will keep track of the data you are driving, become the learning program of the ECU, and provide the best control state to adapt to your driving experience. This program is also called adaptive program. However, once the RAM is powered off, all data will be lost.

Therefore, we can say that the ECU is a small embedded system, which have the most basic structure-microprocessor (CPU), memory (ROM, RAM), in-put/output interface (I/O), analog-to-digital conversion. (A/D) and large-scale inter-grated circuits such as shaping and driving.

For simulating the ECU, the microcontroller board must have sufficient resources and processing speed. After comparison, the Arduino MEGA 2560 was chosen.

3.1.1 About Arduino

Arduino is an easy-to-use, open source electronic prototyping platform. Includes hardware (various models of Arduino board) and software (Arduino IDE). Developed in the winter of 2005 by a European development team. Its members include Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis and Nicholas Zambetti, etc.

It is built on the open source simple I/O interface and has a Processing/Wiring development environment that is similar with Java and C. There are two main parts: the hardware part is the Arduino board that can be used for circuit connection; the other is the Arduino IDE, the program development environment in your computer. As long as you write the program code in the IDE and upload the program to the Arduino board, the program will tell the Arduino board what to do.

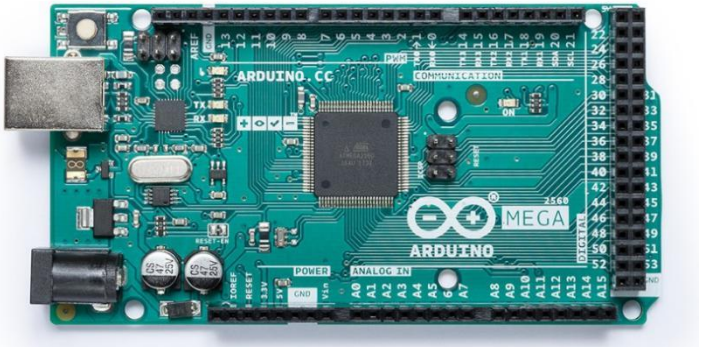


Figure 3-1 Arduino board

Arduino can sense the environment through a variety of sensors, feedback, influence the environment by controlling lights, motors and other devices. The microcontroller on the board can be programmed in Arduino's programming language, compiled into a binary file, and burned into the microcontroller. Programming for Arduino is done through the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino-based projects can include only Arduino, or Arduino and other software running on a PC, communicated between them (such as Flash, Processing, MaxMSP).

3.1.2 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 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

Table 3-1 Arduino Mega 2560 resources

The Arduino Mega2560 is also compatible with expansion boards designed for the Arduino UNO. The Arduino Mega2560 has been released to the third edition and has the following new features compared to the previous two editions:

- Two pins SDA and SCL are added at AREF to support the I2C interface; IOREF and a reserved pin are added, and the expansion board will be compatible with 5V and 3.3V core boards in the future
- Improved reset circuit design

- The USB interface chip replaces the ATmega8U2 with the ATmega16U2

The Arduino Mega2560 can be powered in 3 ways and automatically selects the power supply:

- External DC power is supplied through the outlet
- Connect the battery to the GND and VIN pins of the power connector
- The USB interface is powered directly

There are 54 digital input and output ports: The working voltage is 5V, and each channel can output and connect with a maximum current of 40mA. Each channel is equipped with a 20-50K ohm internal pull-up resistor (not connected by default).

In addition, some pins have specific functions:

- 4 serial port signals: serial port 0---0 (RX) and 1 (TX); serial port 1---19 (RX) and 18 (TX); serial port 2---17 (RX) and 16 (TX); Serial port 3---15 (RX) and 14 (TX). Serial port 0 is connected to the internal ATmega8U2 USB-to-TTL chip to provide serial port receiving signals with TTL voltage level
- 6 external interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). The trigger interrupt pin can be set to rising edge, falling edge or simultaneous trigger
- 14 pulse width modulation PWM (0--13): Provides 14 8-bit PWM outputs
- SPI (53 (SS), 51 (MOSI), 50 (MISO), 52 (SCK)): SPI communication interface
- LED (pin13): Arduino is specially used to test the reserved interface of the LED. When the output is high, the LED is lit. When the output is low, the LED is off.

There are16 analog inputs: Each channel has a resolution of 10 bits (that is, the input has 1024 different values), the default input signal range is 0 to 5V, and the input upper limit can be adjusted by AREF.

In addition, some pins have specific functions:

- TWI interface (20 (SDA) and 21 (SCL)): Supports communication interface (compatible with I2C bus).

For this project, the board resources are used as follows:

- Using analog inputs to sample the data of sensors, such as temperature, rpm, throttle angle, etc.
- Using pulse width modulation to control the motor driver.
- Using serial port to communication with upper computer.
- Using others digital input and output to control others driver, such as injectors, LED, LCD, Fan, etc.

3.1.3 Arduino software (IDE)

Arduino software(IDE) is a professional Arduino development tool, mainly used for the writing and development of Arduino programs. It has open source circuit diagram design, supports ISP online burning, and supports Flash, Max/Msp, VVVV, PD, C, Processing, etc. A variety of program compatibility features.

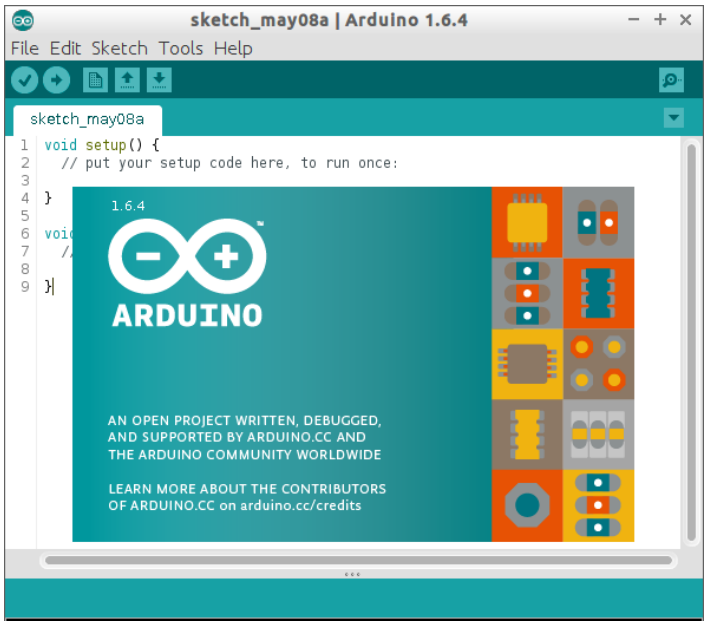


Figure 3-2 Arduino software(IDE)

Simply put, it is used to write code and download code. Any Arduino product needs to be downloaded to run. The hardware circuits we build are done with auxiliary code, and both are indispensable. Just as a person controls the physical activity through the brain

is a truth. If the code is the brain, the peripheral hardware is the limb, and the activity of the limb depends on the brain, so the hardware implementation depends on the code.

Therefore, The Arduino Mega 2560 board can be programmed with the Arduino Software (IDE) in C language.

3.2 Modeling of the engine

A car's engine is a device that powers a car. It is the heart of a car and determines the power, economy, stability, and environmental friendliness of the car. Depending on the source of power, automotive engines can be classified into diesel engines, gasoline engines, electric vehicle motors, and hybrids.

Common gasoline engines and diesel engines are reciprocating piston internal combustion engines, which convert the chemical energy of fuel into mechanical energy of piston movement and output power externally. The gasoline engine has high speed, low quality, low noise, easy starting and low manufacturing cost. The diesel engine has a large compression ratio, high thermal efficiency, and better economic performance and emission performance than the gasoline engine.

To build a Modeling of the engine, it is necessary to understand the structure and working principle of the engine.

3.2.1 Engine structure

A car's engine is composed of two major mechanisms: the crank-link mechanism and the air distribution mechanism, as well as cooling, lubrication, ignition, fuel supply, and starting system.

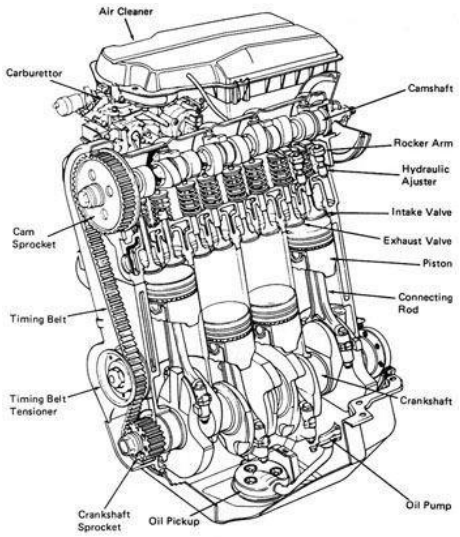


Figure 3-3 Engine structures

The crank-link mechanism consists of a cylinder block crankcase, a piston connecting rod and a crankshaft flywheel. The function of the air distribution mechanism is to open and close the inlet and exhaust valves in time according to the working sequence of the engine and the working cycle of each cylinder, so that the combustible mixture enters the cylinder and discharges the exhaust gas into the atmosphere.

The auxiliary system of the engine mainly includes an ignition system, a cooling system, a lubrication system, a starting system.

3.2.2 Working principle

The engine used in the test bench is taken from the vehicle Fiat, model punto. The volume of cylinders is 1.2L, with four cylinders and four strokes of piston per cylinder.

Four-stroke engine requires four strokes (L) of the piston to complete a cycle and produce one power stroke. The piston moves between a top dead center position (TDC) and a bottom dead center position (BDC), that correspond respectively to the minimum cylinder volume, and to the maximum cylinder volume.

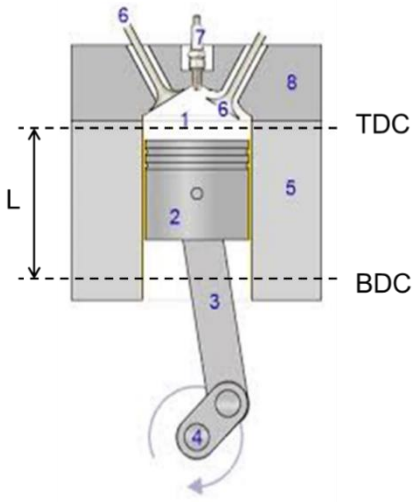


Figure 3-4 Engine working principle

The cycle is completed in 4 strokes:

- Intake stroke. The piston moves from TDC to BDC and draws fresh mixture into the cylinder. To increase the mass inducted, the inlet valve opens shortly before the stroke starts and closes after it ends.
- Compression stroke. Both valves are closed and the mixture inside the cylinder is compressed to a small fraction of its initial volume.

- Power stroke. The high-temperature, high pressure gases push the piston toward its BDC position and force the crank to rotate.
- Exhaust stroke. The piston moves from BDC to TDC and the burned gas are expelled by the piston as it moves towards TDC. As the piston approaches the TDC position the inlet valve opens. The exhaust valve closes just after TDC.

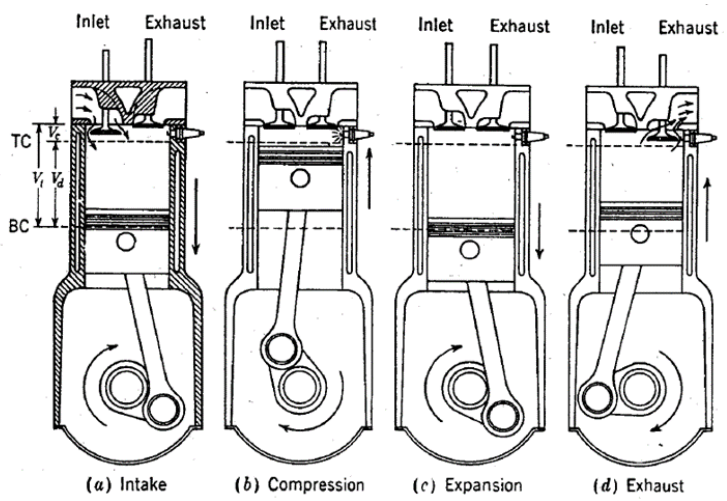


Figure 3-5 Four-stroke engine

3.2.3 Mathematical model

An example is provided in the reference of MathWorks, this example shows how to model a four-cylinder spark ignition internal combustion engine from the throttle to the crankshaft output. This model is based on published results by Crossley and Cook[22].

In this simulation, the triggered subsystem simulates the transfer of the air-fuel mixture from the intake manifold to the cylinder through discrete valve events. This occurs simultaneously with the continuous flow of intake air flow, torque generation and acceleration.

- Throttle

  This is the first element of the model. When the manifold pressure is lower, the flow rate through the throttle is sonic and is only a function of the throttle angle.
  Equation I:
  $\dot{m}_{ai} = f(\theta) \times g(P_m)$
  $\mathrm{f}(\theta) = 2.821 - 0.05231 \times \theta + 0.10299 \times \theta^2 - 0.00063 \times \theta^3$
  $\mathrm{g}(P_m) = 1$; if $P_{amb}/2 \leq P_m \leq P_{amb}$

$$g(P_m) = \frac{2}{P_{amb}}\sqrt{P_m P_{amb} - {P_{amb}}^2} \text{ ; if } P_{amb/2} \le P_m \le 2P_{amb}$$

$$g(P_m) = -\frac{2}{P_m}\sqrt{P_m P_{amb} - {P_{amb}}^2} \text{ ; if } P_{amb} \le P_m \le 2P_{amb}$$

$$g(P_m) = -1; \text{ if } P_m \ge 2P_{amb}$$

$\dot{\boldsymbol{m}}_{\boldsymbol{ai}}$ Mass flow rate into manifold(g/s)
θ Throttle angle(deg)
$\boldsymbol{P_m}$ Manifold pressure(bar)
$\boldsymbol{P_{amb}}$ Ambient (atomosperic) pressure(bar)

- Intake Manifold

This is the second element. The intake manifold is modelled as a differential equation for the manifold pressure.
Equation II:

$$\dot{P_m} = \frac{RT}{V_m}(\dot{m_{ai}} - \dot{m_{ao}})$$

$\boldsymbol{R}$ Specific gas constant
$\boldsymbol{T}$ Temperature(K)
$\boldsymbol{V_m}$ Manifold Volume(m^3)
$\dot{\boldsymbol{m}}_{\boldsymbol{ao}}$ Mass flow rate of air out of the manifold(g/s)
$\dot{\boldsymbol{P}}_{\boldsymbol{m}}$ Rate of change of manifold pressure(bar/s)

- Intake Mass Flow Rate

The third element is intake mass flow rate. This rate is a function of the manifold pressure and the engine speed.
Equation III:

$$\dot{m_{ao}} = 0.366 + 0.08979 \times N \times {P_m}^2 + 0.0001 \times N^2 \times P_m$$

$\boldsymbol{N}$ Engine angular speed(rad/s)
$\boldsymbol{P_m}$ Manifold pressure(bar)

- Torque Generation and Acceleration

The fourth element is the torque developed by the engine.
Equation IV:

$$Torque_{eng} = -181.3 + 379.36 \times m_a + 21.91 \times \left(\frac{A}{F}\right) - 0.85 \times \left(\frac{A}{F}\right)^2 + 0.26 \times \sigma - 0.0028 \times \sigma^2 + 0.027 \times N - 0.000107 \times N^2 + 0.00048 \times N \times \sigma + 2.55 \times \sigma \times m_a - 0.05 \times \sigma^2 \times m_a$$

$\frac{\boldsymbol{A}}{\boldsymbol{F}}$ Air to fuel ratio

$\boldsymbol{\sigma}$ Spark advance(degree)

$\boldsymbol{Torque_{eng}}$ Torque produce by engine(Nm)

- Finally, using the Equation 5 to calculate the engine angular acceleration.

  Equation V:

  $$J\dot{N} = Torque_{eng} - Torque_{load}$$

  $\boldsymbol{J}$ Engine rotational moment of inertia($\mathrm{kg} \cdot m^2$)
  $\dot{\boldsymbol{N}}$ Engine angular acceleration(rad/s^2)

According to all the equations, we can model the engine modeling in Simulink as follow:

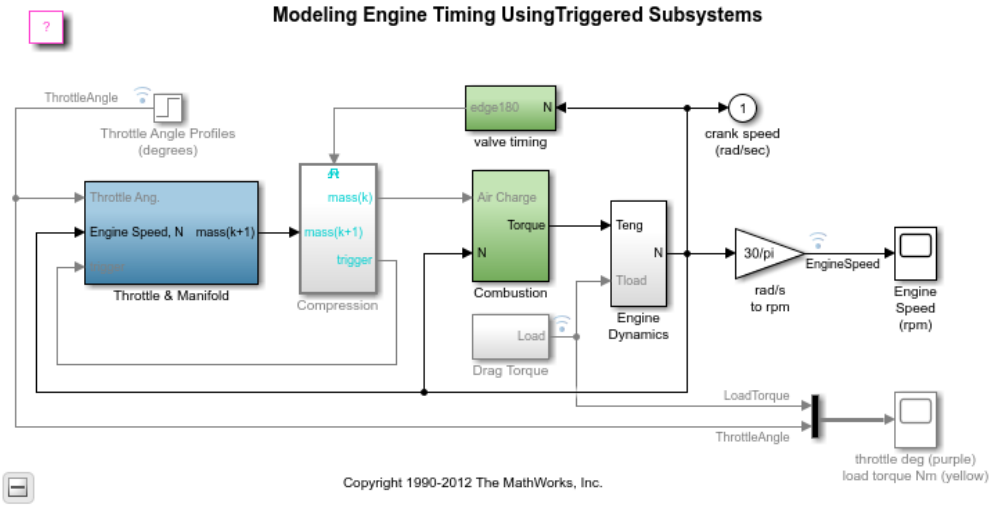


Figure 3-6 Modeling engine timing

Running this simulation, we can get the top level of the engine model and simulation results as below Figure 3-7:

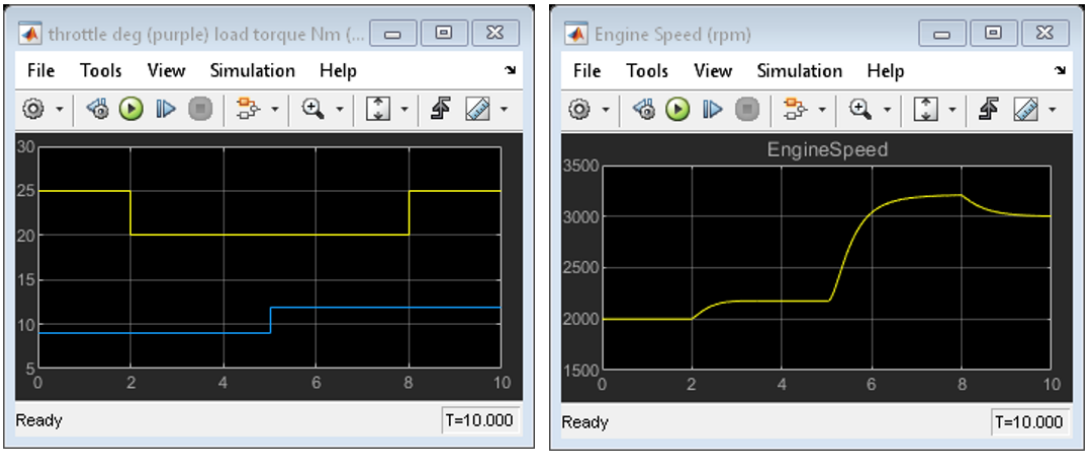


Figure 3-7 The top level results

If we want to verify the accuracy of this modeling, we can give the following data as the default input:

$$\text{Throttle} = 8.97(\text{deg}) \;\; \text{if} \;\; t < 5$$
$$\text{Throttle} = 11.93(\text{deg}) \;\; \text{if} \;\; t \geq 5$$
$$\text{Load} = 25(\text{Nm}) \;\; \text{if} \;\; t \leq 2 \text{ or } t \geq 8$$
$$\text{Load} = 25(\text{Nm}) \;\; \text{if} \;\; 2 < t < 8$$

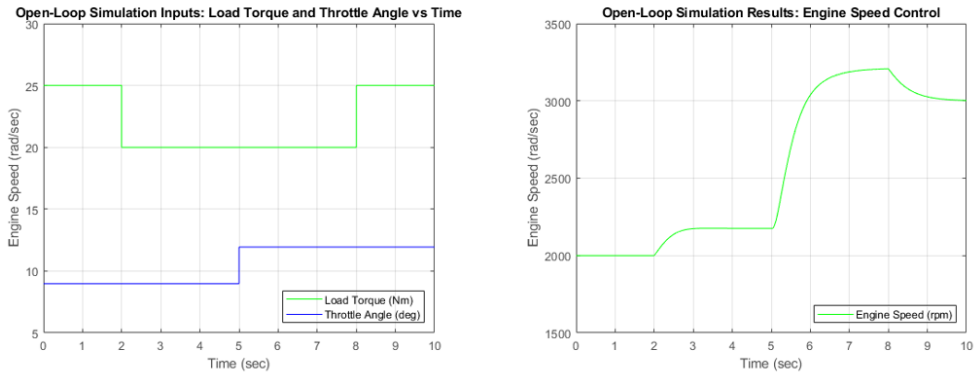


Figure 3-8 The results of the open-loop simulation

Try adjusting the throttle to compensate for the load torque. Figure 3-8 shows the simulated engine speed, the throttle commands which drive the simulation, and the load torque which disturbs it.

Finally, we can get the results of the open-loop simulation. It is the same for comparing with the results of top level simulation.

3.3 Building the simulation engine

Have a deeper understanding of engine construction and working principles through engine modeling, now it is time to build a simulation engine with some electronic components.

3.3.1 Throttle body

In four-stroke fuel injected engines, Throttle body belong to the air intake system. In many cars, the accelerator pedal movement is transmitted through the throttle cable, the throttle cable is mechanically connected to the throttle link, and the throttle link drives the throttle plate to rotate.

Figure 3-9 Throttle body

When the driver steps on the accelerator pedal, the throttle plate rotates within the throttle body, opening the throttle passage to allow more air to enter the intake manifold. The greater the angle of rotation of the throttle plate, the more air is drawn in, and the faster of the engine speed.

The range of rotation angle of the throttle plate is 0°~90°. The Arduino Mega 2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By defaulting they measure from ground to 5 volts.

Therefore, a 10KΩ potentiometer is used to simulate the throttle body. The Arduino can read the analog values from the potentiometer voltages from 0 to 1023. Then using the map() function re-maps the number to another range (0, 90), as the angle of the throttle.

```
1.  throttle_angle = (float)map(value, 0, 1023, 0, 90);
```

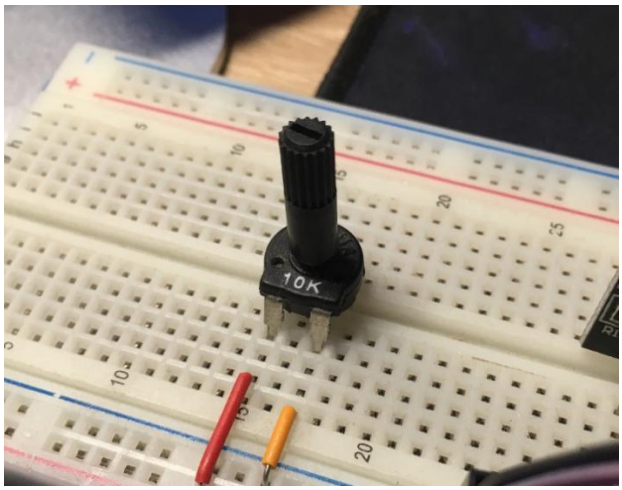


Figure 3-10 10K potentiometer

When regulating the potentiometer from 0 to 10K, the Arduino could read the value from 0 to 90 and save it to the variable ‘throttle_angle’.

Figure 3-11 shows the schematic of the potentiometer.

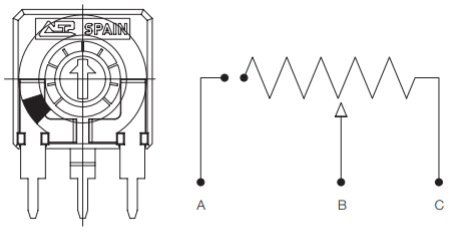


Figure 3-11 Potentiometers with cut track

Table 3-2 shows the electric specifications of the potentiometer.

CA14. Electric Specifications

These are standard features; other specifications can be studied on request.

Range of resistance values Lin (A) Log (B) Antilog (C)	 100Ω ≤ Rn ≤ 5MΩ 1 KΩ … 2,2 MΩ
Tolerance Special tolerances available on request	100Ω … 1MΩ ±20% >1MΩ … 5MΩ ±30% Out of range: Rn> 5MΩ: +50%, -30%
Variation laws	Lin (A), Log (B), Antilog (C) Other tapers available on request
Residual resistance	Lin (A), Log (B), Antilog (C) ≤ $5*10^{-3}$*Rn Minimum value 2Ω
CRV - Contact Resistance Variation (dynamic)	≤3%Rn
CRV - Contact Resistance Variation (static)	≤5%Rn
Maximum power dissipation at 40º C. Lin (A) No Lin (B, C)	 0,25W 0,13W
Maximum voltage at 40ºC Lin (A) No Lin (B, C)	 250VDC 200VDC
Operating temperature	-25°C … +70°C
Temperature coefficient	100Ω - 10KΩ → +200/ -300 ppm. >10KΩ - 5MΩ → +200/ -500 ppm

Table 3-2 Potentiometers electric specification

3.3.2 Crankshaft rotation

The crankshaft is a sequence of cranks (and crank pin journal), up to 5/6 in-line cylinder engines, each crank carrying one connecting rod. It rotates on the main journals and between two successive main journals. At one end there is the nose for the distribution pulley, at the other end there is the flange for the flywheel mounting

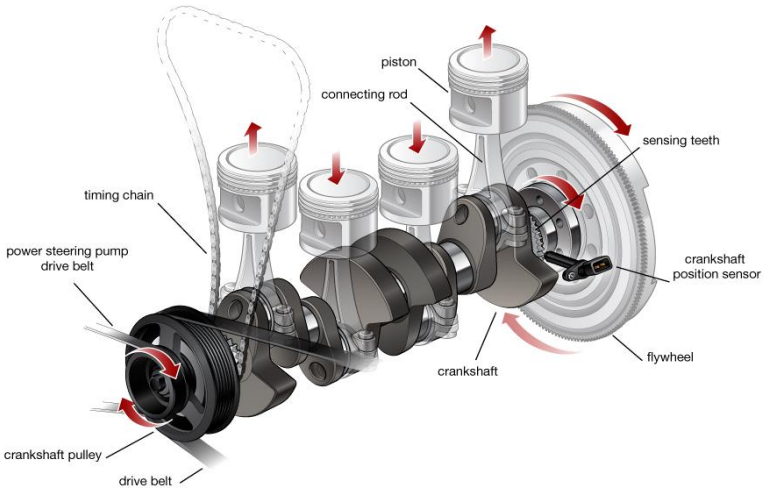


Figure 3-12 Crankshaft

One DC motor is used to simulate the rotation of the crankshaft. This DC motor come with a wide operating range: from 9 to 12VDC. This range makes it perfect for controlling with the motor shield. Figure 3-13 shows an 12V DC motor.

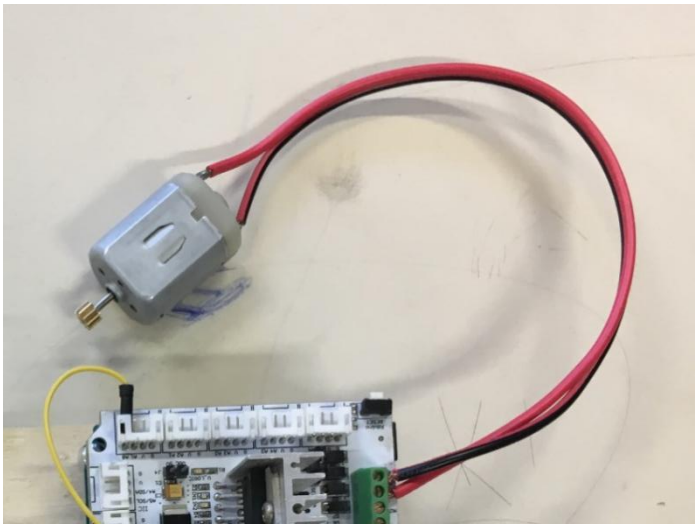


Figure 3-13 12V DC motor

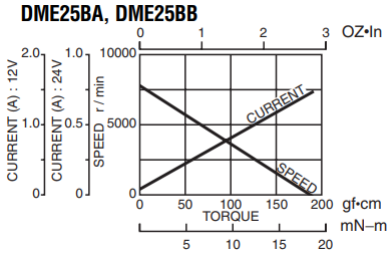


Figure 3-14 Current speed-torque curve

Model	Rated						No load		Stall torque		Weight	
	Output	Voltage	Torque		Current	Speed	Current	Speed	mN-m	oz-in		
	W	V	mN-m	oz·in	A	r/min	A	r/min			g	lb
DME25BA	3	12	4.9	0.69	0.47	5800	0.07	8000	17.7	2.50	55	0.12
DME25BB	3	24	4.9	0.69	0.23	5800	0.04	8000	17.7	2.50	55	0.12

Table 3-3 12V DC motor standard specifications

For controlling the rotation of the DC motor, the method is relatively simple. It is only necessary to add appropriate voltage to the two control lines of the motor to make the motor rotate. The higher the voltage, the higher the motor speed.

But when you want to control the rotation speed and the direction of the DC motor, the DC motor driver must be used, and it designed by Fritzing, which is an open source electronic design automation software. It has a rich library of components, everyone can use it design circuit schematics for free.

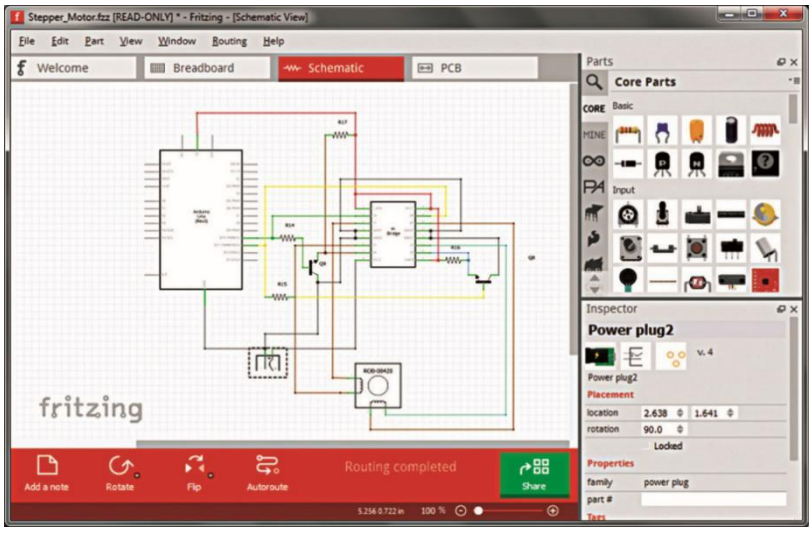


Figure 3-15 Fritzing

The L298N is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are

connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

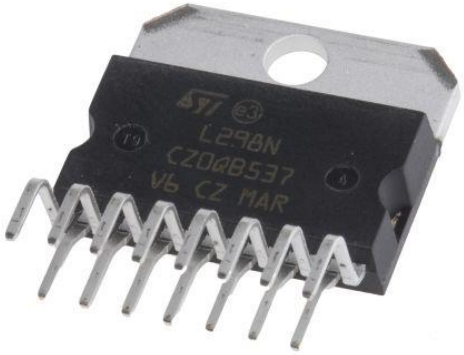


Figure 3-16 L298N

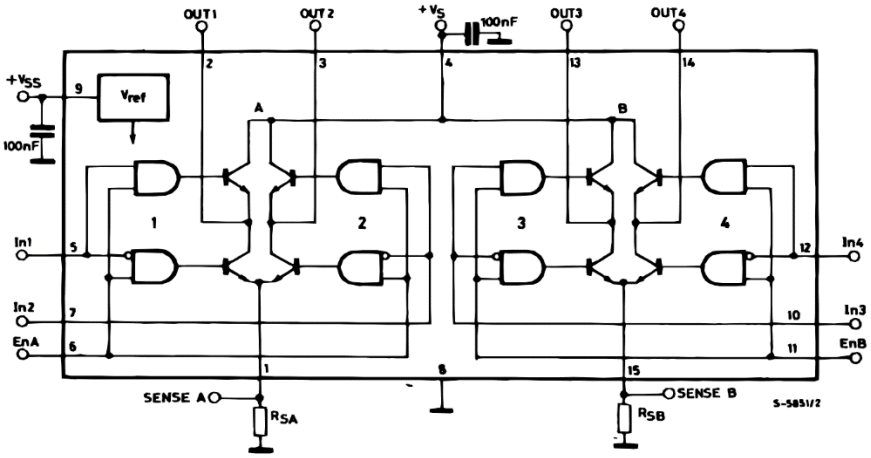


Figure 3-17 L298N block diagram

From the truth table of L298N, when the pin of ENA is high, the input level of IN1 and IN2 is high and low, the DC motor is rotating either clockwise or counterclockwise, otherwise, the motor brake.

ENA	IN1	IN2	The State of DC Motor A
0	X	X	Stop
1	0	0	Brake
1	0	1	Rotate Clockwise
1	1	0	Rotate Counterclockwise
1	1	1	Brake

Table 3-4 L298N truth table

The coil of the motor will generate a back EMF on both sides during the rotation process, which will cause an impact on the L298N even break it. Therefore, two diodes are added to each line to protect the L298N. These two diodes are used as clamps, so that the voltage (or potential) on the output line is clamped between -0.7V~+Vcc~+0.7V.

Figure 3-18 1N4007

As a diode, 1N4007 has two features:

- Low forward voltage drop
- High surge current capability

The complete circuit schematic of the DC motor driver is as follows:

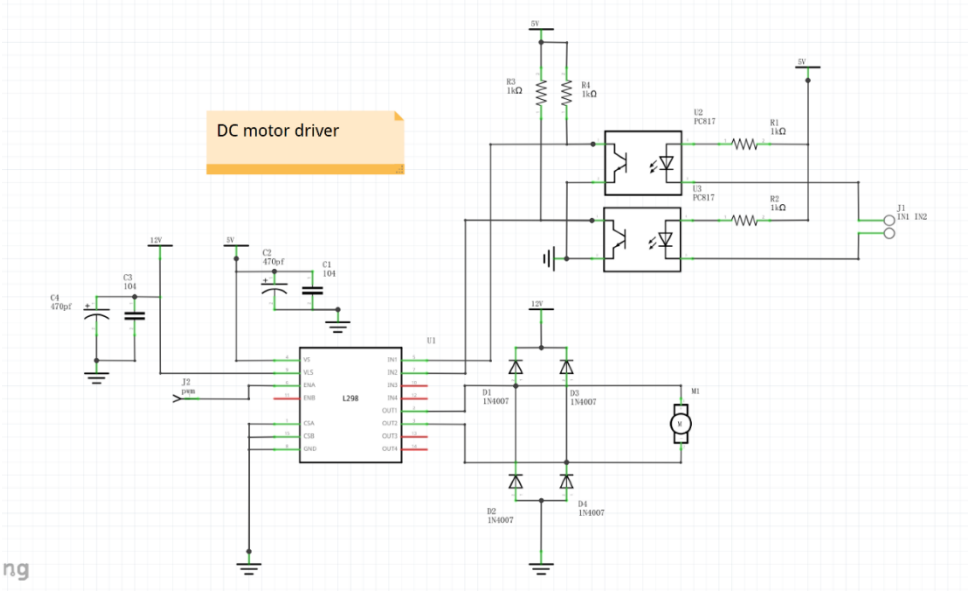


Figure 3-19 DC motor driver circuit schematic

And the physical circuit is as follows:

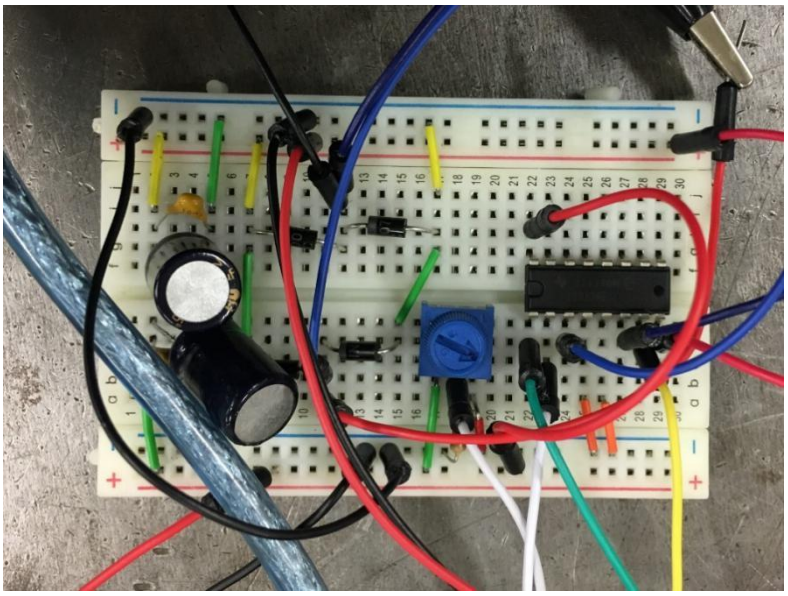


Figure 3-20 DC motor driver physical circuit

The DC motor can be controlled simply through a PWM signal. PWM does not regulate the current. PWM means pulse width modulation, that is, adjusting the time ratio of a square wave high level and low level. For example, a 20% duty cycle waveform, there will be 20% high time and 80% low time, while a 60% duty cycle waveform has 60% high time and 40% low time. The larger the duty cycle, the longer the high time, the more the pulse amplitude of the output, the higher of the voltage. If the duty cycle is 0%, then the high time is 0, then there is no voltage output. If the duty cycle is 100%, then the full voltage is output.

Therefore, by adjusting the duty cycle, the purpose of adjusting the output voltage can be achieved, and the output voltage can be adjusted continuously and steplessly.

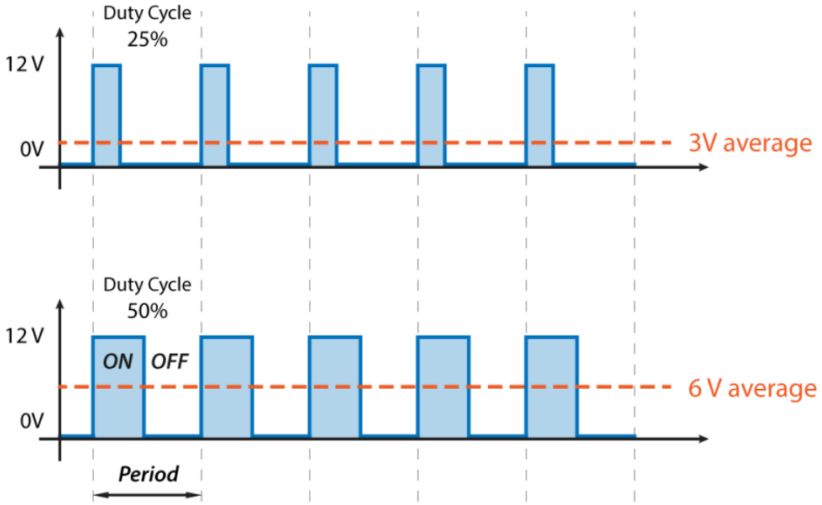


Figure 3-21 Pulse Width Modulation

Using the Arduino board to generate PWM waves is very simple. Just need to call the analogWrite(pin, value) function. This function have two parameters, pin, which means the pin to wire to, and value, which means the duty cycle: between 0 (always off) and 255 (always on). Here is a simple example:

```
1.  void loop()
2.  {
3.    int sensorValue = analogRead(A8); //read the value of potentiometer
4.    val = map(sensorValue, 0, 1023, 0, 255); //change the value range
5.    analogWrite(pwm, val); //use the value as a duty cycle
6.    digitalWrite(pin1, LOW); //make motor rotate clockwise
7.    digitalWrite(pin2, HIGH);
8.    Serial.println(val, DEC); //display the value to serial monitor
9.    delay(100);
10. }
```

3.3.3 RPM sensor

The engine rpm sensor is also called crankshaft position sensor, its function is: detecting the engine speed; detecting the top dead center position of the piston, so it is also called top dead center sensor, including detecting the top dead center signal of each cylinder for controlling ignition, for controlling The first cylinder top dead center signal of the sequential injection.

Figure 3-22 RPM sensor

There are three different types of rpm sensor commonly: the Hall Effect sensor, optical sensor, and the inductive sensor.

The ECU must receive the real speed values of the engine to control the engine. So it is very important to use the RPM sensor to getting the engine speed.

An Opto Interrupter also known as photo interrupter is a transmission-type photo sensor that integrates optical receiving and transmitting elements in a single package. This device will turn a circuit on and off optically. It consists of a molded plastic housing with an IR LED facing a photo transistor across a gap. Any object in the gap will interrupt the IR beam and consequently switch the photo transistor on and off. The device is very fast and ideal for counting, timing or sensing.

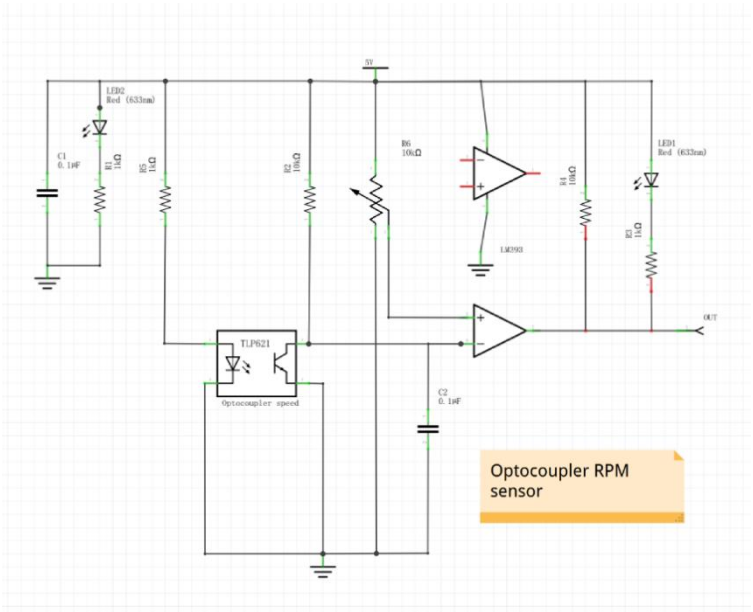


Figure 3-23 Optocoupler RPM sensor

Installing a missing tooth plate on the DC motor, let the missing tooth can pass through the U groove of the opto interrupter. The photo-interrupter employs transmissive technology to sense obstacles existence, acts as on / off switchers or even to sense low resolution rotary or linear motions. The principle states that objects opaque to infrared will interrupt the transmission of light between an infrared emitting diode and a photo sensor switching the output from an "ON" state to an "OFF" state.

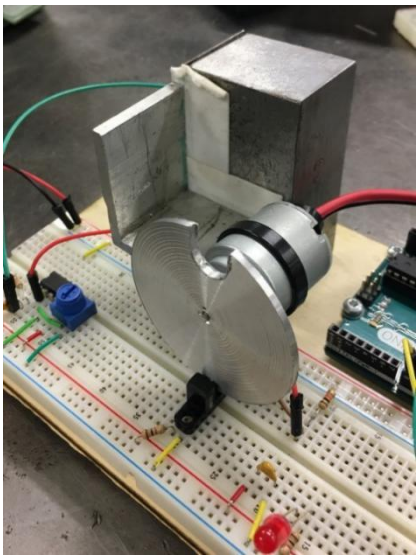


Figure 3-24 Count rpm with missing tooth plate

The opto interrupt gives one signal to the Arduino every rotation of the DC motor. Arduino can count all the signals within 100ms, then calculate the engine speed every 100ms. Arduino provides a timer library, which named FlexiTimer2, to use the timer 2 with a configurable resolution. You can enable a timer interrupt just call one function: FlexiTimer2::start(). For example.

```
#include <FlexiTimer2.h>

attachInterrupt(0,RCount, FALLING);
attachInterrupt(1,LCount, FALLING);

FlexiTimer2::set(1000, flash); // timer interrupt 1s
FlexiTimer2::start(); //enable interrupt

void LCount() //Interrupt service function
{
  l_wheel++;
}
void RCount() //Interrupt service function
{
  r_wheel++;
}
```

From the results of the measurement by Bosch electronic diagnostic board, describe the relationship between the throttle angle and the rpm as following figure:

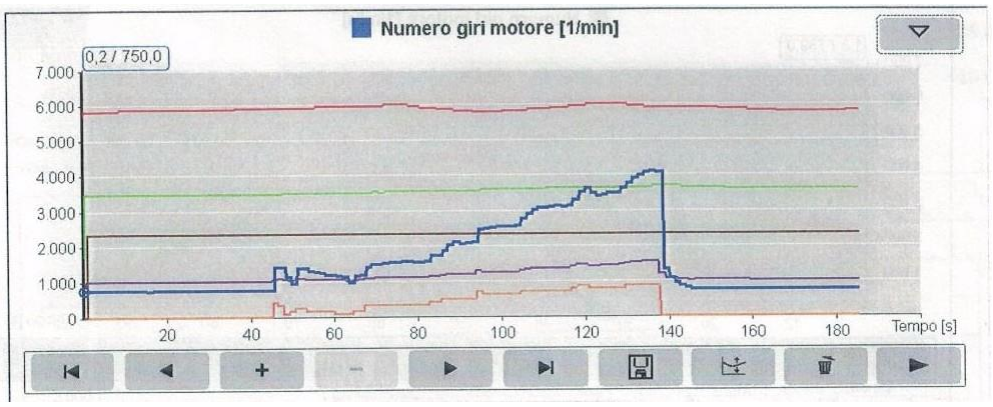


Figure 3-25 Representation of Bosch results

Color of lines	parameters
blue	engine speed
orange	throttle angle

Table 3-5 Association between line color & parameter

Analyzing the data combine with the previous modeling results, there is relationship between the throttle angle and engine speed. When the throttle plate is opened, more air intake to the engine through the carburettor. The mass of air entering is proportional to the 'RPM * engine capacity * density of the air leaving the carburettor for the inlet manifold'. Energy conversion rate is:

$$\begin{aligned} &\text{RPM} \cdot \text{capacity} \cdot \text{throttle\_setting} \\ &\boldsymbol{KE} = 0.5 \cdot \text{vehicle\_mass} \cdot \text{velocity}^2 \end{aligned}$$

3.3.4Temperature sensor

In automotive electronic systems, temperature sensors are used to monitor the actual temperature of various components, liquids, and gases, such as engine coolant temperature sensors, intake air temperature sensors, transmission fluid temperature sensors, evaporator temperature sensors, and the like.

The temperature sensor uses a negative temperature coefficient thermistor (NTC) as the sensing element. The higher the temperature, the smaller the resistance value of the component. Conversely, the lower the temperature, the higher the resistance value.

Figure 3-26 Coolant temperature sensor

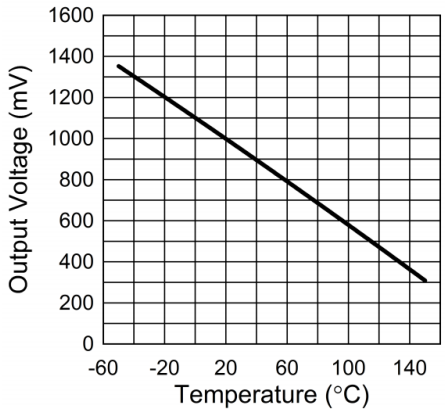


Figure 3-27 Temperature sensor feature description

It is better to simulate the temperature sensor with a same type analog temperature sensor. The Arduino Mega 2560 can get the analog value from the temperature sensor then convert to the real temperature.

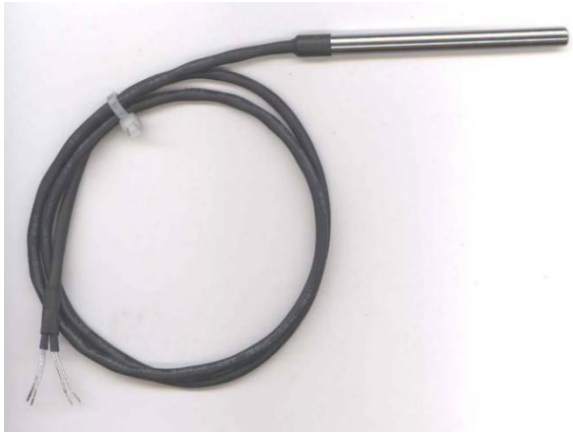


Figure 3-28 NTC analog temperature sensor

3.3.5 Cooling fan

The function of the car's cooling system is to dissipate part of the heat absorbed by the heated parts in time to ensure that the engine works at the optimum temperature. The cooling system of the automobile engine is a forced circulation water cooling system, that is, the pump is used to increase the pressure of the coolant, and the forced coolant is circulated in the engine.

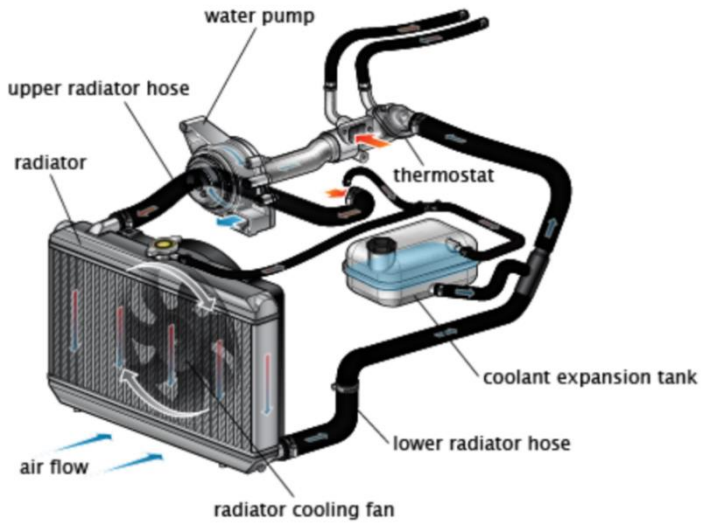


Figure 3-29 Cooling system

When the engine is working, the coolant flows in the radiator, the air passes outside the radiator, and the hot coolant cools due to heat dissipation to the air. During normal driving, the high-speed airflow is sufficient to dissipate heat, and the fan generally does

not work at this time; however, when running at slow speed or in situ, the fan may rotate to assist the radiator to dissipate heat. The start of the fan is controlled by the water temperature sensor. When the engine inlet temperature exceeds 90 °C, the water temperature sensor will turn on the fan.

The temperature when the fan start running could be looked as the limiting temperature of the coolant. Thus, we can use a fan blade and a small DC motor to simulate the fan of cooling system. When the temperature of the engine exceeds 90℃, the fan is running. And the rotation speed of the fan blade will change with the temperature.

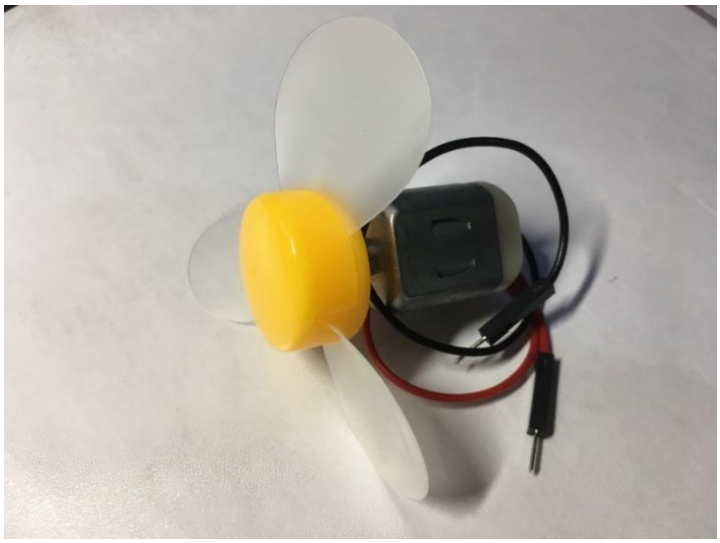


Figure 3-30 Cooling fan

```
1.  double Thermister(int rawADC) //cal temperature
2.  {
3.    double tmp;
4.    tmp = log(((10240000 / rawADC) - 10000));
5.    tmp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tmp * tmp)) * t
    mp);
6.    tmp = tmp - 273.15;
7.    return tmp;
8.  }
9.
10. void loop()
11. {
12.   float raw = analogRead(A8); //read the value of sensor
13.   waterTemp = Thermister(raw)
14.   if(waterTemp >= 90)
15.   {
16.     val = map(waterTemp, 0, 1023, 0, 255); //change the value range
17.     analogWrite(pwm, val); //use the value as a pwm duty cycle
18.     digitalWrite(pin1, LOW); //make motor rotate clockwise
19.     digitalWrite(pin2, HIGH);
```

```
20.   }
21.   Serial.println(waterTemp, DEC); //display the value to serial monitor
22.   delay(100);
23. }
```

3.3.6 Connecting rod

The connecting rod is the component which links the piston and the crankshaft and transforms the reciprocating in the rotating motion. It is a crucial element of the engine not only for the working aspect of the engine itself but also for its overall performance: it is fundamental for converting the chemical energy in mechanical energy with the slider crank mechanism, but its design also deeply affects the overall engine dynamic behavior.

Figure 3-31 Connecting rod

With the connecting rod position, it can calculate about the piston position by KIVA4 method.

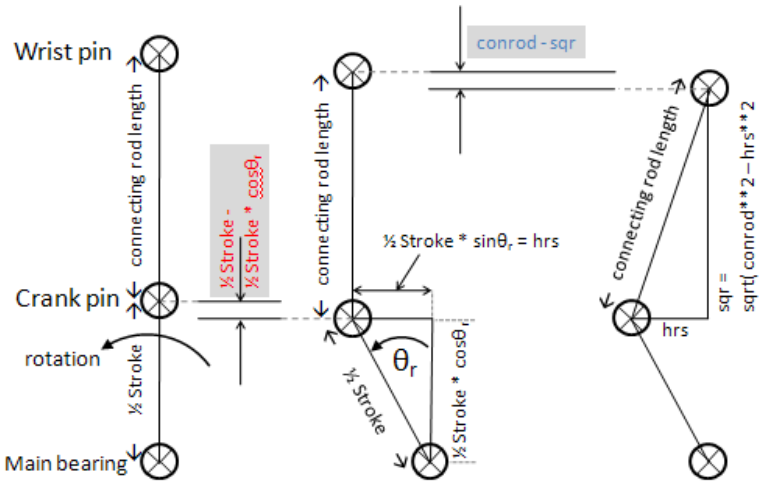


Figure 3-32 Calculate piston position

The servo motor SG90 could be used to simulate the connecting rod. It can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control the servo. The motor move to a rotating position, this position is determined by the signal sent on the signal line. Once the motor reaches the position specified by the signal, it will maintain its position and resist any external forces attempting to move it from that position. It comes with an arm just like the connecting rod. Thus, it can be used to simulate the working process of the connecting rod.

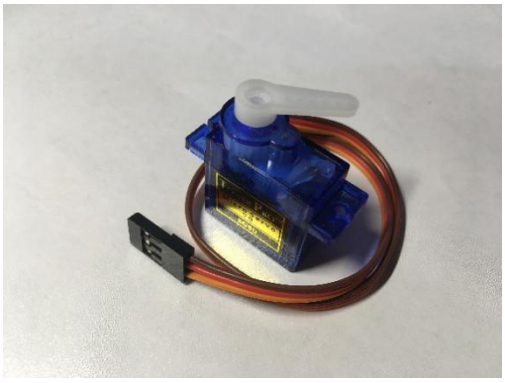


Figure 3-33 Servo motor SG90

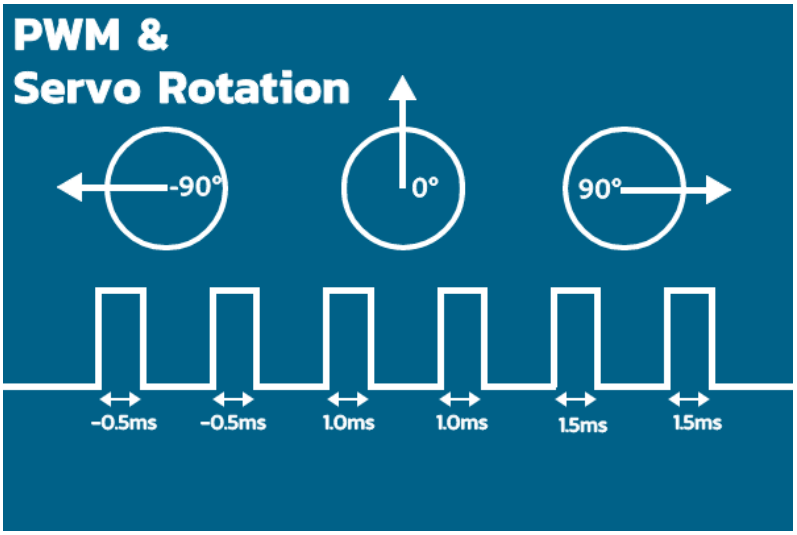


Figure 3-34 Servo motor work principle

In Arduino IDE there is a perfect library of servo motor. The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega 2560. With this library could control the servo motor very simply.

```
1.  Servo coreservo; //Name the Servo
2.
3.  void setup() {
4.    coreservo.attach(10);
```

```
}

void loop() {
  coreservo.write(0);
  delay(200);
  coreservo.write(90);
  delay(200);
  coreservo.write(180);
  delay(200);
}
```

3.3.7 Fuel injection

A fuel injection system is a fuel supply device that uses a fuel injector to inject a certain amount of fuel directly into a cylinder or an intake port under a certain pressure. Depending on the type of fuel injected, it can be classified into a gasoline injection system, a diesel injection system, a gas fuel injection system, and the like. According to the different control methods, it can be divided into mechanical control, electronic control and electromechanical hybrid control.

There is no good solution to simulate the process of the fuel injection system, so the physical fuel injection system which amount from the engine was chosen finally.

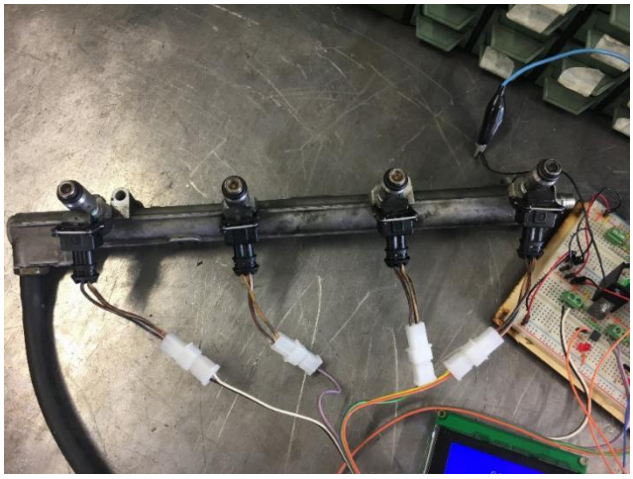


Figure 3-35 Fuel injection system

The injection system needs a giant transistor driver circuit to work. When the engine is working, the ECU sends a control signal according to the relevant signal to control the giant transistor turn on and off. The drive mode of the injector is divided into two types: current drive and voltage drive. Since the type of the injector we using is a low resistance injector, it must chose the current drive mode.

There is no additional resistance in the current drive circuit. The low resistance injector is directly connected to the battery, and the current flowing through the injector solenoid is controlled by the transistor in the ECU. Since there is no additional resistance, the loop impedance is small. When the conduction starts, the large current causes the needle valve to open quickly, and the injector has good responsiveness. When the needle valve is opened, the required holding current is small, which can prevent the injector coil from heating and reduce power consumption. The schematic diagram shows as below Figure 3-36 and the physical connection shows as Figure 3-37:

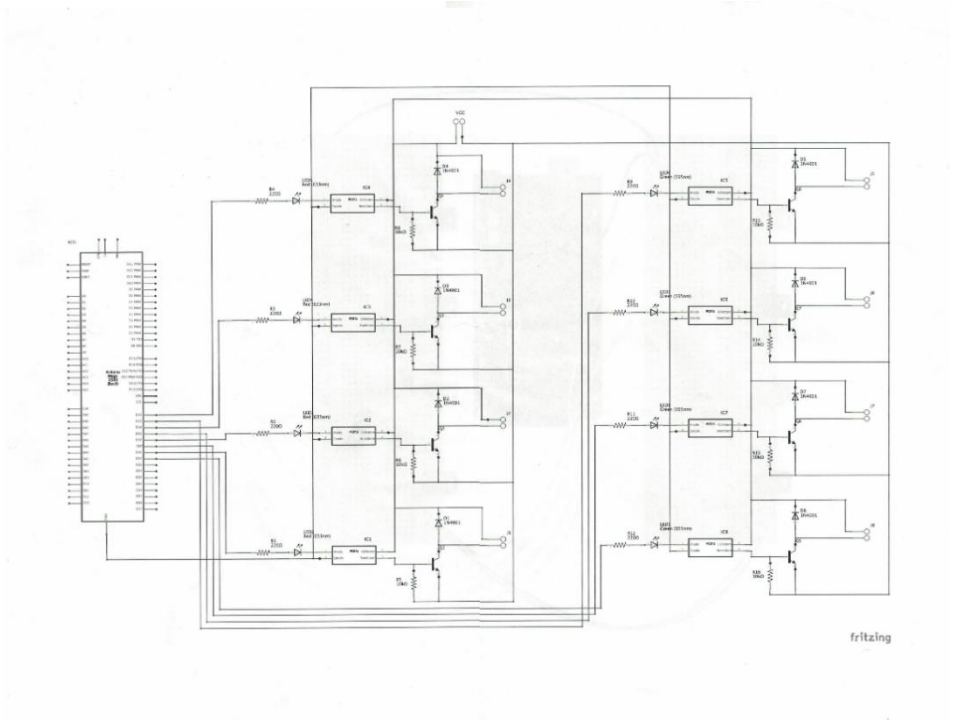


Figure 3-36 Injector driver circuit

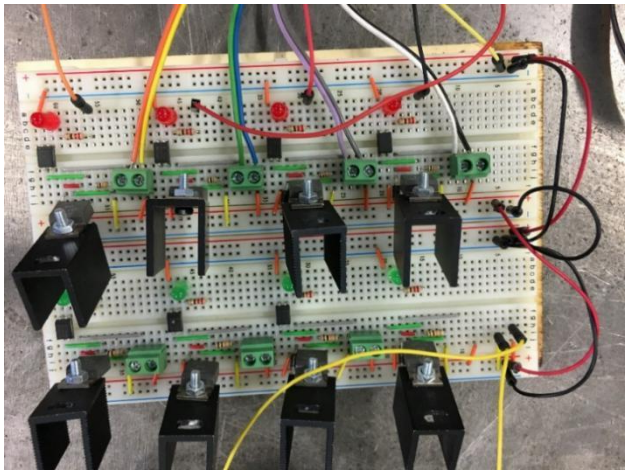


Figure 3-37 Physical connection diagram of injector driver

3.3.8 Engine mapping

Open-loop digital systems store ignition-timing and fuel mixture data in an ECU memory. Data stored in individual computer memory cells can be graphically represented as a characteristic map. The information for this map is detected by running a series of tests on the engine and the program for these tests is called engine mapping.

Through a series of sensors, such as engine rpm sensor, intake manifold vacuum sensor, throttle position sensor, crankshaft position sensor, etc. to determine the working state of the engine. Finding the ignition advance angle of the engine in the mapping and ignition it according to the working state requirement. The ignition timing is then corrected based on the knock sensor signal to operate the engine at the optimum ignition timing.

There is a typical engine mapping of the engine which is simulated in the FIAT reference datasheet. Select 5 sets of data with different working state from the mapping and use them to build new mapping for the simulation test bench engine.

	Cicli al minuto	Cicli al secondo	Durata ciclo in "	Durata ciclo in msec			
Rpm	3125	52.08333333	0.0192	19.2			
6250	CALCOLO MAPPE						
Tempo ciclo msec	Mappa 01					Da	A
19.2	TI	TM1	TM2	TS		1000	2142
Tempo fase msec		0					
4.8	0	0	0	0			
Tempo iniezione msec	131.25	-41.25	-1.25	50		180	181.25
3.466666667							
Tempo scintilla	Mappa 02					Da	A
0.033333333	TI	TM1	TM2	TS		2143	4285
Tempo morto 1		0					
0	0	0	0	0			
Tempo morto 2	131.25	-41.25	-1.25	50		180	181.25
1.3							
Anticipo scintilla °	Mappa 03					Da	A
50	TI	TM1	TM2	TS		4286	5624
Controllo		0					
4.8	0	0	0	0			
	131.25	-41.25	-1.25	50		180	181.25
Formule:							
TCM	Mappa 04					Da	A
(60*1000)/(Rpm/2)	TI	TM1	TM2	TS		5625	6206
msec/°		0					
TCM/720	0	0	0	0			
	131.25	-41.25	-1.25	50		180	181.25
	Mappa 05					Da	A
	TI	TM1	TM2	TS		6207	6700
		3.5	50	1.333333333			
	3.466666667	0	1.3	0.033333333		1.25	
	130		48.75	1.25		180	180
msec/°							
0.026666667							
	Calcolo anticipo scintilla						
	RPM Rif	Anticipo Base	RPM effettivi	Anticipo Effettivo		Rapporto	Controllo
	1000	8	6250	50		0.008	50
	Calcolo durata iniezione						
	RPM Rif	Anticipo Base	RPM effettivi	Anticipo Effettivo			
	1000	21	6250	131.25		0.021	131.25

Table 3-6 Mapping calculate

Downloading this mapping to the simulation ECU(Arduino board) by programming. Using a button to select different map, and six LEDs to show which map is using.

Figure 3-38 shows the mapping programming at the appendix.

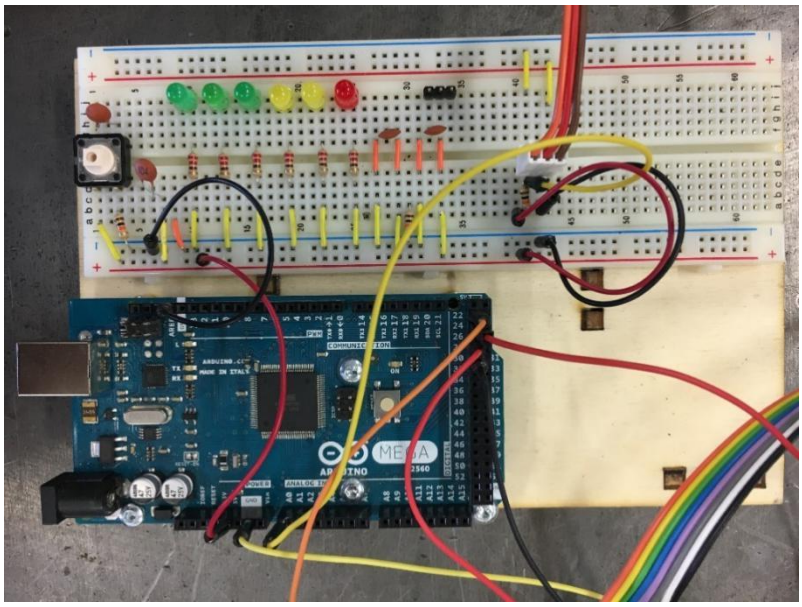


Figure 3-38 Mapping selection board

3.4 Voltage regulation

The power supply is another one important part for the simulation test bench. Because the Arduino Mega 2560 works at 5 Volts, but the working voltage of the DC motor and the injector is 12 Volts, and the working current varies widely. The maximum working current of the Arduino Mega 2560 is 50 mA, but when the DC motor working with load, it need more than 300 mA of the current. In order to protect all the components and prevent the interference, the voltage regulation must be used.

LM2596HV is a step-down switching regulator, capable of driving a 3A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3V, 5V, 12V, and an adjustable output version.

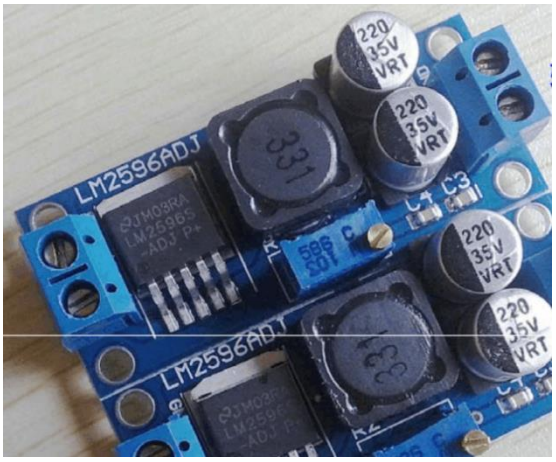


Figure 3-39 LM2596 voltage regulation module

The input voltages range up to 60V, and the output voltages range 1.23V ~ 57V. Adjusting the output voltage through change the value of the potentiometers. Figure 3-40 shows the schematic of LM2596 module.

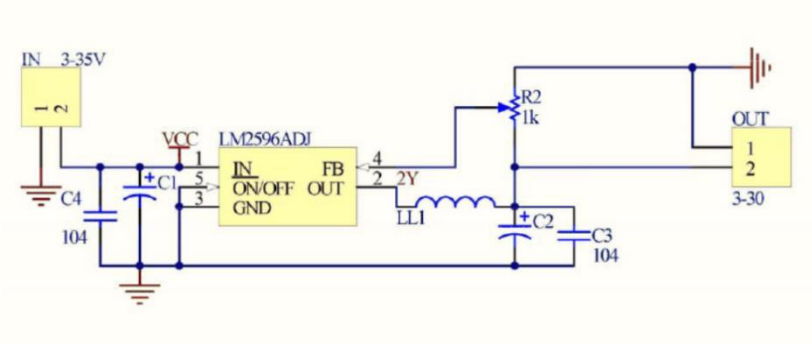


Figure 3-40 Schematic of LM2596 module

3.5 LCD display

When the simulation test bench working, the status of every part must be shown to the students at real time. Such as the values of temperature, RPM, throttle angles and errors. The LCD 2004 module is a display module which have 20 characters and 4 lines. It can be used to display all the values.

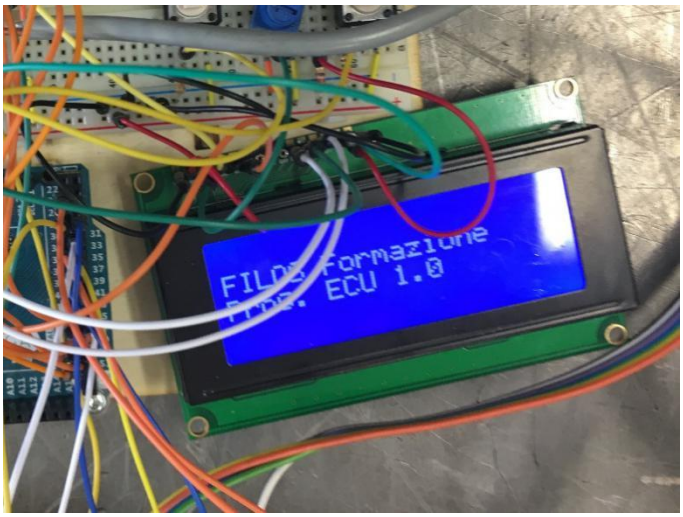


Figure 3-41 LCD 2004 module

It is easy for control the LCD with the library of Arduino IDE, which named LiquidCrystal. You just need define the pin of the LCD used, then write what you want to display directly.

```
1.  #include <LiquidCrystal.h>
```

```
2.
3.  LiquidCrystal lcd (12,11,5,4,3,2); //define pins
4.
5.  LCDsetup() //setup LCD
6.
7.  void loop
8.  {
9.    lcd.clear();
10.   lcd.setCursor (0,1);
11.   lcd.print ("  FILOS ECU 2.0");
12.   lcd.setCursor (0,2);
13.   lcd.print ("Err: Rpm Elevato");
14. }
```

3.6 Communication with lower and upper computer

For displaying all the data with graphical and teaching students how to analyze the data, all the data must be uploaded to the computer. ZigBee is an open global standard for wireless technology designed to use low-power digital radio signals for personal area networks. ZigBee operates on the IEEE 802.15.4 specification and is used to create networks that require a low data transfer rate, energy efficiency and secure networking. It is employed in a lot of applications such as building automation systems, heating and cooling control and in medical devices.

DL-20 TTL ZigBee wireless serial communication module is a full-duplex wireless transparent transmission module with UART port, it operates within public frequency band 2400MHz~2450MHz. The module uses CC2530 chip produced by TI, supports IEEE-802.15.4 protocol, enables traditional serial device and MCU-controlled equipment to realize serial wireless transmission, and avoids complex wiring efforts. Being available in point-to-point communication and broadcast communication, it features in plug and play, development-free and flexible usage.

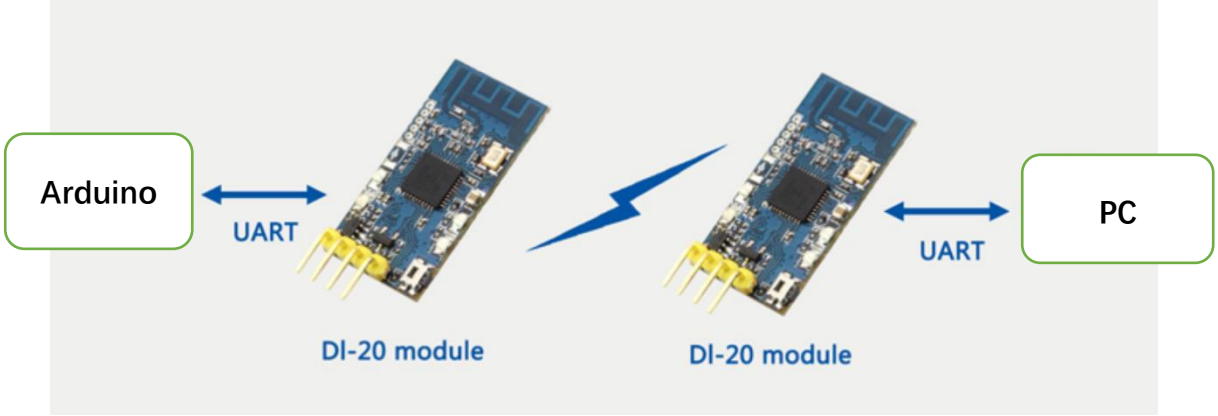


Figure 3-42 Principle of ZigBee communication module

In point-to-point working mode, DL-20 can be divided into terminal A and terminal B. Data sent to terminal A by the serial port is sent to terminal B by virtue of wireless transmission, terminal B sends the received data to the serial port, vice versa. Communication between two nodes in the same channel is allowed in this mode.

For the simulation test bench, one DL-20 is connected to the serial port on the Arduino Mega 2560, as the terminal A. Another one is connected to the computer by an USB cable, as the terminal B. Setting two modules at a same channel and same baud rate. Then the data can be transformed between Arduino Mega 2560 and the computer.

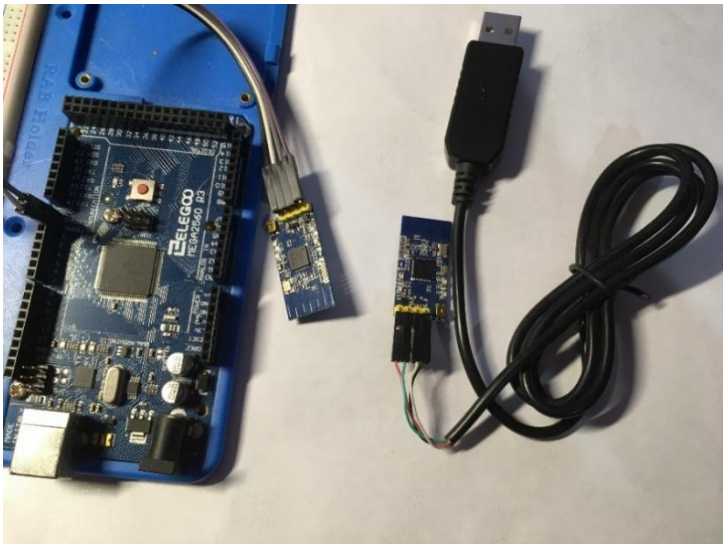


Figure 3-43 Terminal A and B

3.7 The operation flow of the simulation test bench.

Turning on the power, the ECU(Arduino Mega 2560) starts to work, at first check if the key(button) is turned. When the key is turned, ECU read the value of the throttle angle(potentiometer) and give PWM let the crankshaft(DC motor) starts to turn, the connecting rod(servo motor) also turn an angle with the crankshaft. According to the RPM, ECU choose a suitable map to control injector and sparking. Then the ECU read the temperature of the engine, if the temperature exceeds 90℃, the cooling fan will be turned on.

For the communication part, the ECU displays all the data on the LCD in real-time, and send them to the computer with serial wirelessly at the same time.

The operation flow shows as figure 3-44:

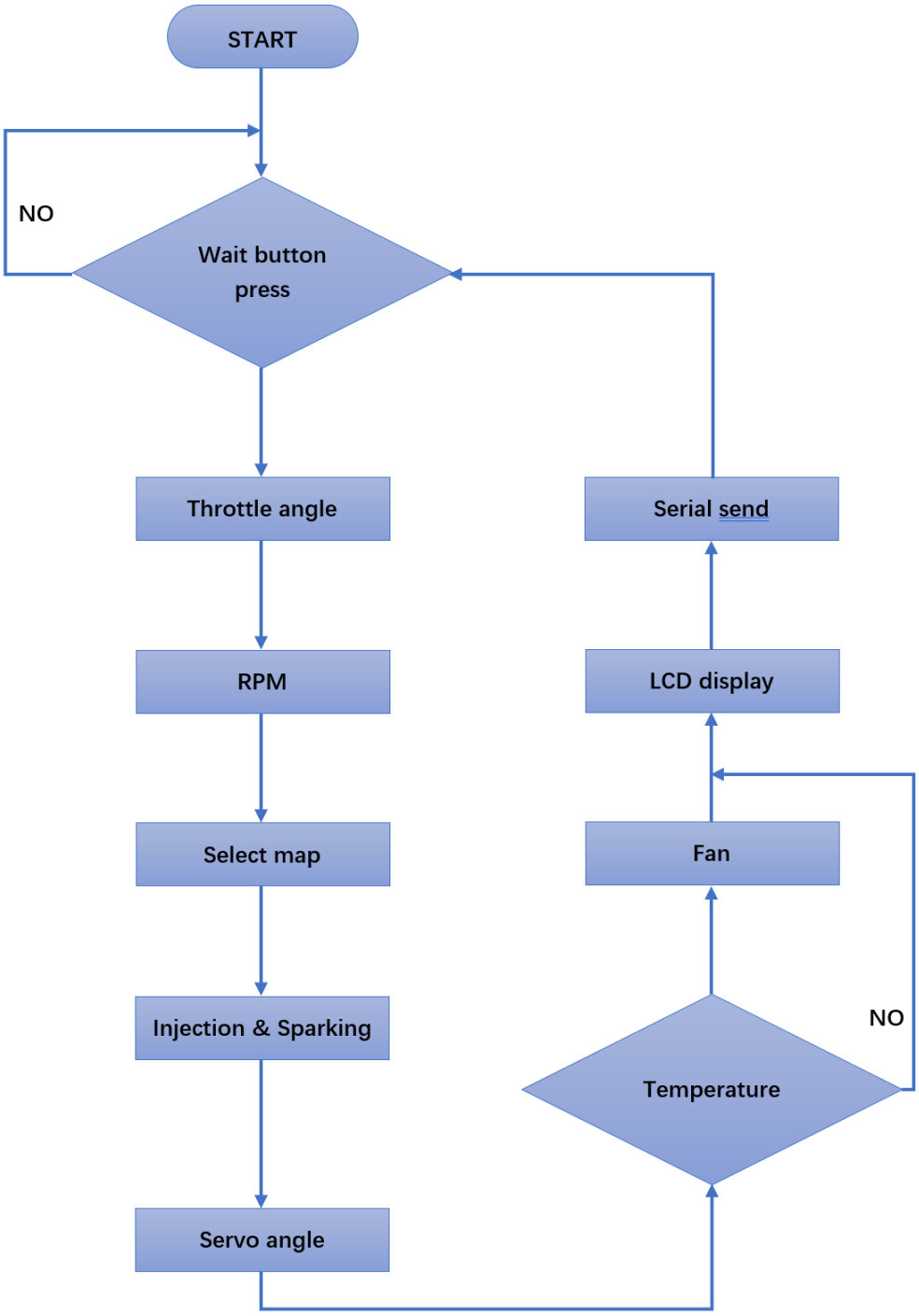


Figure 3-44 The operation flow of the simulation test bench

CHAPTER 4: Developing of GUI with LabVIEW

The emergence of computers has completely changed people's work and lifestyle. Computers are now ubiquitous and enter everyone's life. From the perspective of engineers, computers are not only common PCs, but also various microprocessors. From this perspective, we use computers all the time. For example, TVs, washing machines, cash dispensers, etc. rely on computers to implement various convenient functions.

The same computer can do different things, because they use different programs, and the programs are created by computer programming languages. In just a few decades, a large number of programming languages have emerged. The common feature of these programming languages is the use of text to create programs. Text mode programming is very demanding for programmers, which makes computer programming a profession that only a few people can do.

4.1 About LabVIEW

LabVIEW, an innovative software product from National Instruments (NI), allows graphical programming to eliminate obscure text code, making computer programming no longer a minority patent. The earliest version of LabVIEW was born in 1986 and is almost synchronized with the earliest versions of Windows, which makes LabVIEW a multi-platform programming language for different operating systems.

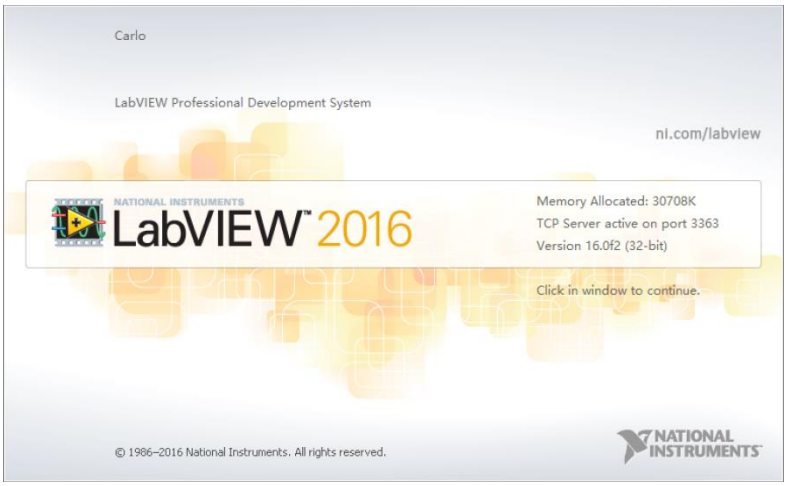


Figure 4-1 LabView

LabVIEW is a specialized programming language developed by test engineers. Therefore, LabVIEW has distinct industry characteristics and was originally used primarily in test and measurement.

As LabVIEW continues to evolve, new versions are available almost every other year or two. LabVIEW's applications cover a wide range of industries including industrial automation, test and measurement, embedded applications, motion control, image processing, computer simulation, and FPGA. With LabVIEW as the core, different specialized toolkits and unified graphical programming methods can be used to meet the needs of different technical fields.

4.2 Test Bench Surveillance System

As these reasons, LabVIEW become a better choice to show the situation of the engine running. The test graphical user interface, which was developed based on the G programming language, named Test Bench Surveillance System. It can run on the upper computer, and communicate with the lower computer through the serial port.

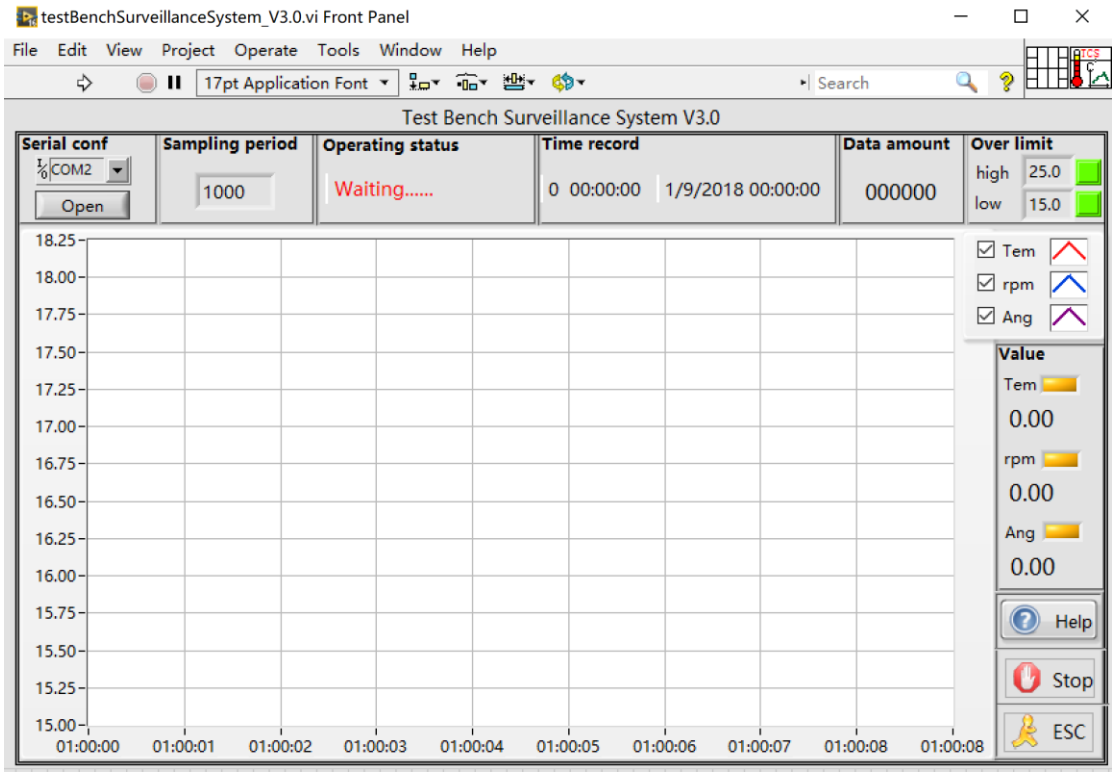


Figure 4-2 Front panel of test bench surveillance system

The main function of the GUI is display the status in graphic at real-time when the engine is working. Thus, the students could understand how is the engine working and what happened in different status.

There are 3 channels to display 3 different data at the same time. The channel can be defined to display the temperature, the rpm, the angle of the throttle or the angle of the connecting rod.

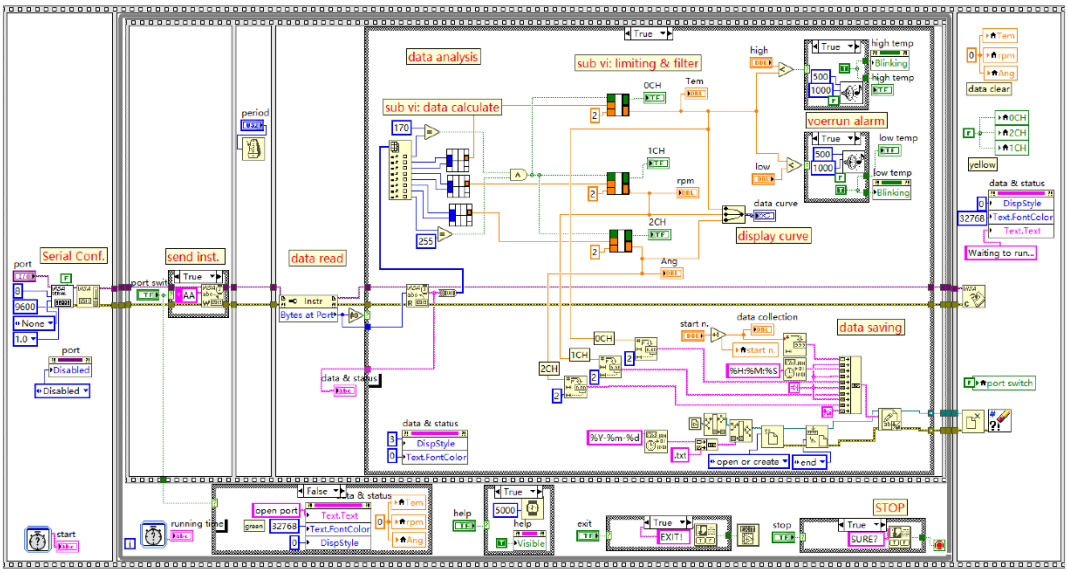


Figure 4-3 Block diagram of test bench surveillance system

4.3 Program structures

LabVIEW's program execution structure contains graphical code and controls how and when internal code runs. The most common execution structures are while loops, for loops, and conditional structures.

4.3.1 While loop structure

Similar to the do loop or repeat-until loop of a text programming language, a while loop executes the code it contains until the conditional terminal ends up to a particular Boolean value.

The mechanism tunnel is used to input and output data in the structure. The solid small square on the border of the while loop is the tunnel, and the color of the small square is the same as the color of the data type connected to the tunnel. Data can be output after the loop is terminated. When the data is input to the loop, the loop starts executing only after the data reaches the tunnel.

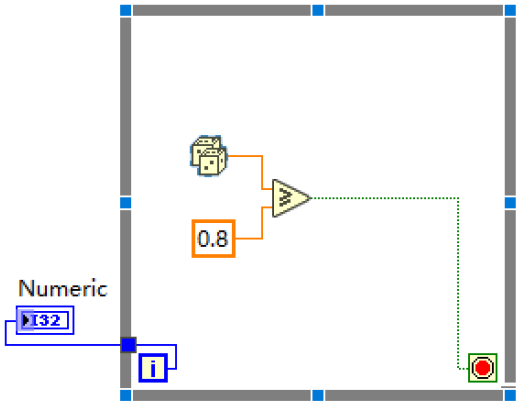


Figure 4-4 While loop structure

4.3.2 For loop structure

The For loop executes the subroutine block according to the set number of times. When the loop structure executes a loop, it immediately starts executing the next loop unless the stop condition is met. The for loop iteration count always starts from zero. The difference between a for loop and a while loop is that the for loop is only executed a specified number of times, while the while loop is executed until the conditional terminal ends to a certain value.

The total terminal N is an input terminal whose value indicates the number of times the subroutine is repeatedly executed. The count terminal i is an output terminal that indicates the number of cycles that have been completed.

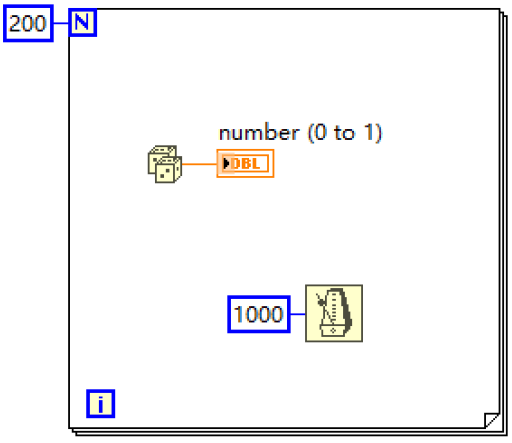


Figure 4-5 For loop structure

4.3.3 Case structure

A case structure consists of two or more case branches, one can be displayed at a time, and only one case branch is executed at a time. The input value will determine which case branch to execute. The selection of a case branch can be achieved by entering the corresponding value in the branch selector identifier or by editing the value using the label tool. Multiple input and output tunnels can be created for a case structure, all inputs are available for case branching, and case branches do not have to use all inputs.

As following figure, the VI executes different code depending on the input control Selection.

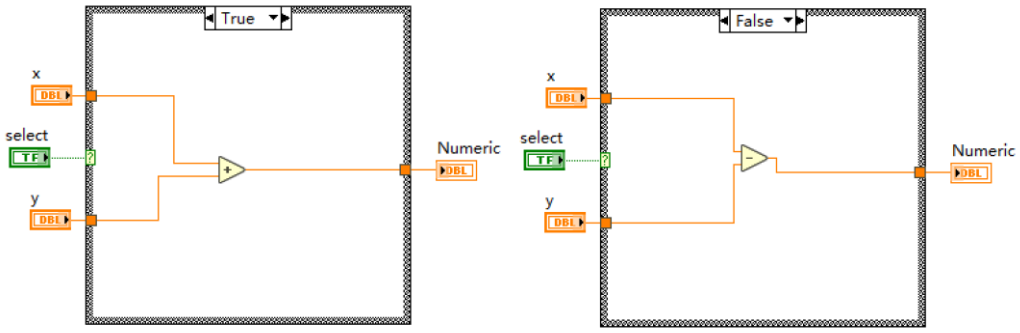


Figure 4-6 Case structure and case branch

4.3.4 Sequence structure

Usually, LabVIEW's program is executed according to the way of data flow, according to the flow direction of the data on the connection wire, the whole process is a process from left to right.

If you want to execute programs that are not connected to each other in a certain order, you can use the sequence structure to achieve. When the program is running, it will be executed one by one according to the frame structure of the sequence structure.

There are two sequential structures in LabVIEW:

- Flat sequence structure:

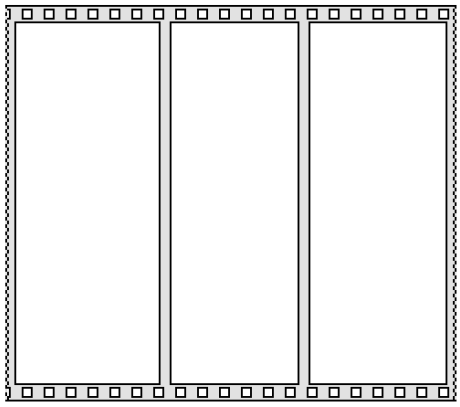


Figure 4-7 Flat sequence structure

- Stacked sequence structure:

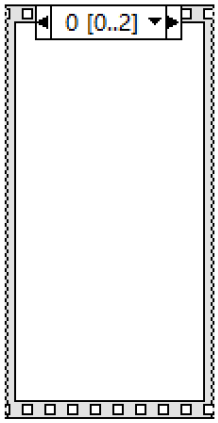


Figure 4-8 Stacked sequence structure

4.4 Communication

The serial port is the most widely used interface for computers and the oldest communication interface. The serial port is usually called COM port or RS232 port. The COM port is described from a hardware perspective. The configuration of the serial port can be seen from both the BIOS of the computer and the device manager of Windows.

RS232 (ANSI/ EIA-232 standard) is a serial connection standard on IBM-PC and its compatible machines. Based on RS232, the serial communication standard has undergone many changes, and it has gradually evolved into two new standards, RS422 and RS485.

NI provide a free driver at LabVIEW to communicate with devices connected to a serial port, which named VISA. This driver is very powerful and useful, but not complicated. There are only need four basic functions to realize all serial communication function.

4.4.1 VISA Configure Serial Port

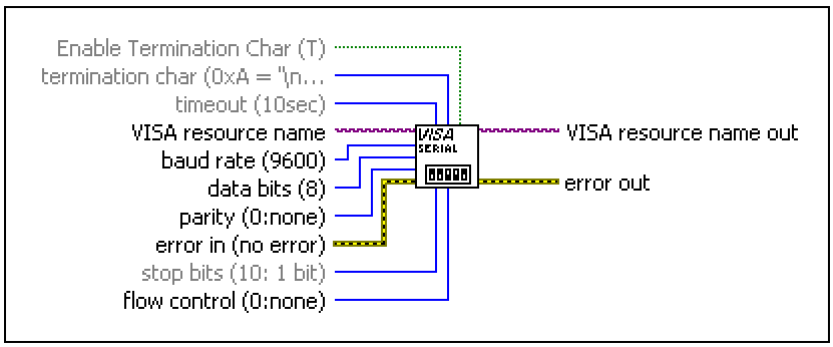


Figure 4-9 VISA configure serial port

The serial communication needs to be configured with 5 parameters before communicating. Also, the settings on both sides of the communication must be the same, otherwise communication will not be possible.

The five parameters are:

1. Baud rate:
   It represents the number of bits transmitted per second, such as the more commonly used 9600, which means 9600 bits per second.
2. Data bits:
   It is the number of bits in the incoming data. The value of data bits is between five and eight. The default value is 8.
3. Stop bits:
   Serial port communication transfers data by frame. The last stop bit of each frame is automatically added. You can select 1, 1.5 or 2. Data is transmitted on the transmission line at regular intervals, and each device has its own clock. A small out-of-synchronization is likely to occur between two devices in communication, so the stop bit not only indicates the end of the transfer, but also provides the opportunity for the computer to correct the clock synchronization. The more bits used to stop a bit, the greater the tolerance for different clock synchronizations, but the slower the data transfer rate.
4. Parity:
   It is used to check if the received data is correct. There are five ways to perform parity, even, odd, none, mark, and space. The most common ones are odd parity and even parity. They determine whether the parity bit is 0 or 1 depending on the number of bits in the data bit.
5. Flow control:

Flow control refers to the control of data flow in serial communication, which is what we often call "handshake". In fact, it is the coordination between sending and receiving data.

The same baud rate is necessary for serial port communication, but the speed at which the sender continuously transmits frames depends on the sender, and the speed at which the receiver accepts frames depends on the receiver. If the data is sent faster than the receiver, it will cause the send buffer to overflow. Therefore, the sender and receiver should coordinate with each other. The sender and the receiver agree that the sender's instruction is used to initiate the sender's transmission or to suspend the transmission. This is called flow control. Since serial port communication is generally two-way communication, both sides of the communication have transmission and reception, so flow control is a problem that must be considered. There are three ways to control flow, namely XON/XOFF flow control, hardware flow control, and no flow control.

All configurations should configure at the block diagram, according to the communication protocol between the upper computer and the lower computer. And it only needs configure one time, so it should be configured before the loop.

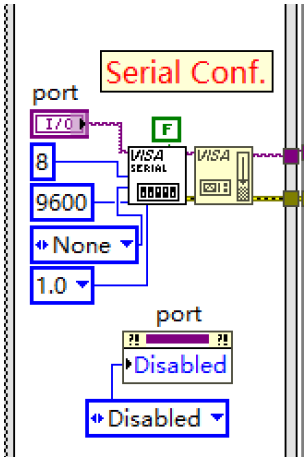


Figure 4-10 Block diagram of VISA configure serial port

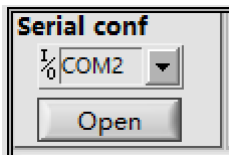


Figure 4-11 Front panel of VISA configure serial port

The VISA resource name control is the only control which display at the front panel. It is to specify the resource to which a VISA session will be opened and to maintain the session and class.

In addition, special attention should be paid to the settings for the "Timeout", "Termination char" and "Enable Termination char".

Timeout is used to set the timeout for read and write operations. The default is 10000ms. When the timeout period is reached and the read and write operations are not completed, a timeout error is returned. Timeout errors usually occur in read operations.

Termination char, when the read operation reads the termination char, the read operation ends immediately. It is especially suitable for sending string information. The default termination char is 0xA, which is a newline character. In addition, the carriage return character is also commonly used as the terminator 0xD, for example, \r.

Enable Termination char is used to decide whether to use a termination char. It is enabled by default.

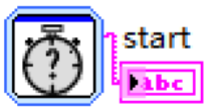


Figure 4-12 Start timer

When the system starting run, there is an Elapsed Time Express before the loop, to record what the system starts running. Then the program goes into the loop.

4.4.2 VISA Write Function

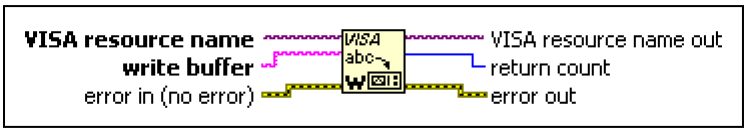


Figure 4-13 VISA write function

The VISA Write Function is relatively simple, just enter the string to be sent. The general programming language supports two kinds of communication methods: byte transmission and ASCII character transmission. LabVIEW serial communication is no exception. If the string being sent is a string that is displayed in the normal way, the ASCII of the string is sent, and the serial port communication is an ASCII string. If the transmitted string is a byte displayed in HEX mode, the byte is sent, and the byte communication mode is mostly used to directly transfer the hexadecimal number.

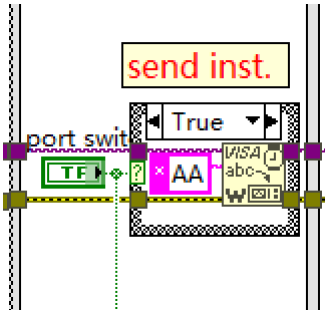


Figure 4-14 Block diagram of VISA write function

The mechanical state of the serial port write switch is conversion when clicked. Writing instruction when serial port is open, serial port is closed
when no instructions are written. Where the string instruction AA is Hexadecimal.

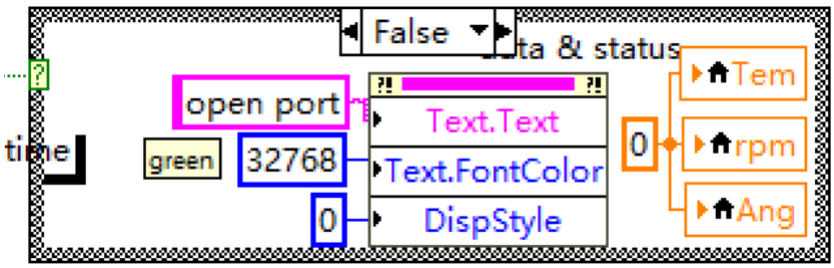


Figure 4-15Block diagram of data & status

Figure 4-16 Front panel of data & status

One case structure is used to display operating status and data value. When the open button is not clicked, the dialog at the front panel displaying “Please open the port…”. If there are some data received, it will display the value of the data in Hexadecimal.

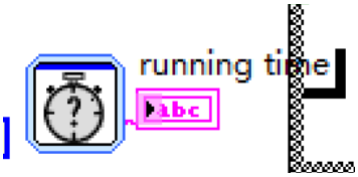


Figure 4-17 Block diagram of running time record

Figure 4-18 Front panel of running time record

There is also an Elapsed Time Express in the loop, to recording the time of the program have run.

Since the lower computer is also made by itself with Arduino, the communication protocol between the upper computer and the lower computer can be controlled. The answering communication method is a better choice of the communication protocol, that is, sending an instruction returns data once. This makes it easier to ensure the correctness of the read data than the sequential upload of the data by the lower computer.

Of course, this is not absolute. The specifics still need to be based on the function of the lower computer. In many cases, how to communicate with the lower computer is not something you can control. You can only use the existing protocol.

Similarly, if the data protocol is designed by itself, it is best to include flag characters such as frame headers and frame trailers (and possibly checksums) instead of just transmitting useful data.

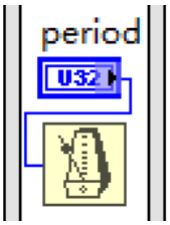


Figure 4-19 Block diagram of sampling period

Figure 4-20 Front panel of sampling period

After sending an instruction, it needs to wait for a certain time, and give the lower computer enough time (how long it takes to determine according to the situation of the lower computer) to return the data. Thus, the speed of the whole cycle can be roughly controlled. For example, if the period is 1000ms, the whole system frequency is about 1 Hz (sampled once per second).

4.4.3 VISA Read Function

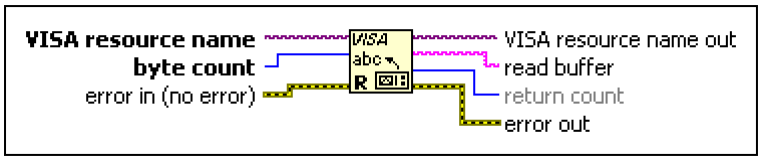


Figure 4-21 VISA read function

VISA read function reads the specified number of bytes from the device or interface specified by VISA resource name and returns the data in read buffer.

The read operation ends in the following three cases:

1. When the termination char is enabled, the read function ends the read operation when the read function reads the termination char.
2. The input parameter of the read function sets the number of bytes that need to be read. When the required number of bytes is read from the input buffer, the read operation is ended.
3. When a timeout error occurs, the read operation ends.

In the absence of a termination char or a termination char is not enabled, if the number of bytes to be read is greater than the number of bytes received in the receive buffer, the read function remains in a wait state until a timeout occurs. If the read function waits forever, it will block the read thread, causing other operations in the thread to fail. Therefore, it is usually necessary to determine the number of bytes in the receive buffer before performing a read operation.

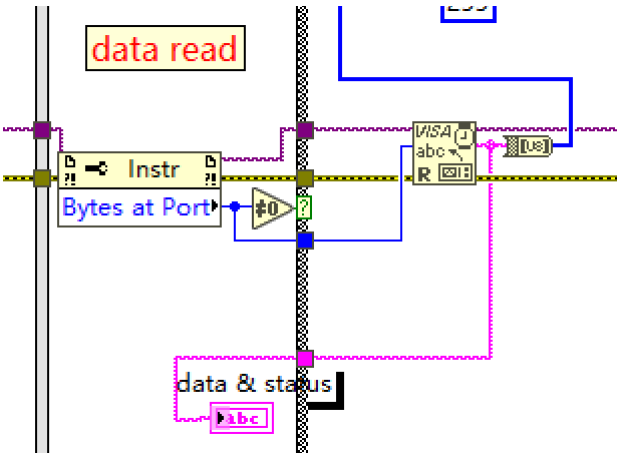


Figure 4-22 Block diagram of VISA read function

The Byte at Port property node is used to return the number of bytes that already exist in the input buffer. When the number of bytes in the serial receive buffer is 0, it means that the lower computer does not return data, and the display is “communication exception”; when the number of bytes in the receive buffer is not 0, the entire contents of the buffer will be read and the original data will be displayed in hexadecimal.

The data which read from the buffer directly are very complicated. Because all the 3 date send at the same time. Therefore, the data must be parsed according to the communication protocol before it can be used.

The string read from the serial port can be converted into decimal array data by a string-to-byte array conversion function, and then the data on the corresponding byte is indexed and parsed according to the protocol.

When the serial data read contains more bytes (longer message) or more useless characters, you can use the intercept string function to intercept useful characters and then perform subsequent processing. It depends on the form of the string content read by the serial port.

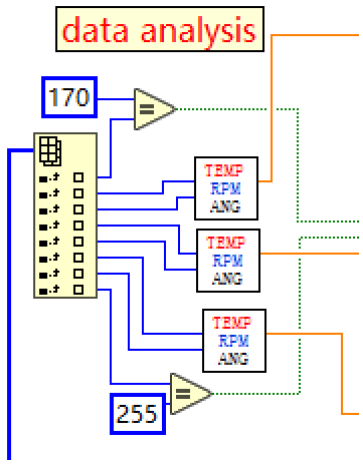


Figure 4-23 Block diagram of data analysis

When collecting data according to specific landmark characters such as frame headers, it can generally use the search/intercept string function to search for iconic characters, and then intercept the valid content for data processing.

For the Test Bench Surveillance System V3.0 and the lower computer Arduino, the serial communication protocol is below:

```
1.  unsigned char TestBench_OutPut_Buffer[14] = {0};
2.
3.  //Convert the single-precision float data to 4 byte data and store it in the specified address
4.  void Float2Byte(float *target,unsigned char *buf,unsigned char beg)
```

```
{
    unsigned char *point;
    point = (unsigned char*)target;
    buf[beg]   = point[0];
    buf[beg+1] = point[1];
    buf[beg+2] = point[2];
    buf[beg+3] = point[3];
}

//Write the single precision float data of the channel to the transmit buffer
void TestBench_Get_Channel_Data(float Data,unsigned char Channel)
{
   if ( (Channel > 3) || (Channel == 0) ) return;
   else
   {
      switch (Channel)
         {
          case 1:  Float2Byte(&Data,TestBench_OutPut_Buffer,1); break;
          case 2:  Float2Byte(&Data,TestBench_OutPut_Buffer,5); break;
          case 3:  Float2Byte(&Data,TestBench_OutPut_Buffer,9); break;
         }
   }
}

//Generate a frame format that TestBenchV3.0 can correctly recognize
unsigned char TestBench_Data_Generate(unsigned char Channel_Number)
{
   if ( (Channel_Number > 3) || (Channel_Number == 0) )
   { return 0; }
   else
   {
    TestBench_OutPut_Buffer[0] = '$';  //frame header

    switch(Channel_Number)
       {
        case 1:  TestBench_OutPut_Buffer[5]  =  5; return  6; break;
        case 2:  TestBench_OutPut_Buffer[9]  =  9; return 10; break;
        case 3:  TestBench_OutPut_Buffer[13] = 13; return 14; break;
       }
   }
   return 0;
}
```

For example, when test the throttle angle of the engine with Arduino, display the curve at the channel 1,

```
void loop()
{
  value = analogRead(throttlePin);
  throttle_angle = (float)map(value, 0, 1023, 0, 45);

  TestBench_Get_Channel_Data(throttle_angle, 1);//choose channel 1

  Send_Count = TestBench_Data_Generate(1);

  for(i = 0; i < Send_Count; i++)  //Send the formatted frame to the buffer
  Serial1.write(TestBench_OutPut_Buffer[i]);

  delay(100); //send period
}
```

4.4.4 VISA Close Function

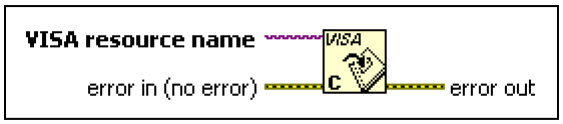


Figure 4-24 VISA close function

When ending the serial communication, it must close the device dialog handle and release the serial port resources. It can use the VISA close function.

4.5 Sub Vi Limiting Filter

In order to ensure data reliability and prevent interference, it is necessary to filter the sampled data using a suitable filtering algorithm.

For a large inertia like the temperature, it does not change much over time. Thus it is suitable for using the limiting filter. At first, according to experience, determine the maximum deviation allowed for two samplings (set to A). Determine each time when a new value is detected: If the difference between the new value and the previous value is less than or equal to A, the current value is valid; If the difference between the new value and the previous value is greater than A, the current value is invalid, and the current value is replaced by the previous value.

The filter could be used at the lower computer before send to the upper computer. But that will take up too many resources to reduce the system stability. So it is better use it at upper computer with a Sub Vi.

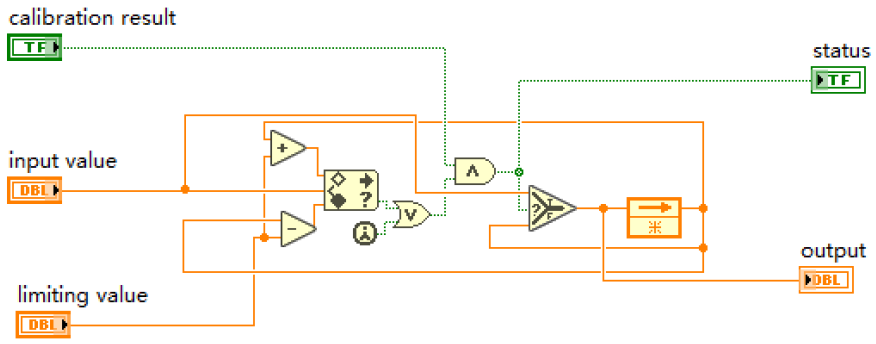


Figure 4-25 Sub Vi limiting filter

Determine whether the temperature difference between the two samples before and after is more than 2 °C. If it is exceeded, the data is considered abnormal. Replace the current sampling temperature value with the previous normal sampling temperature value; otherwise, the data is considered reasonable, sent it, and continue to be compared with the next value.

4.6 Curve display

After analysis all the data, display them in graphic curve.

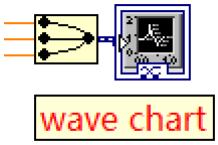


Figure 4-26 Block diagram of wave chart

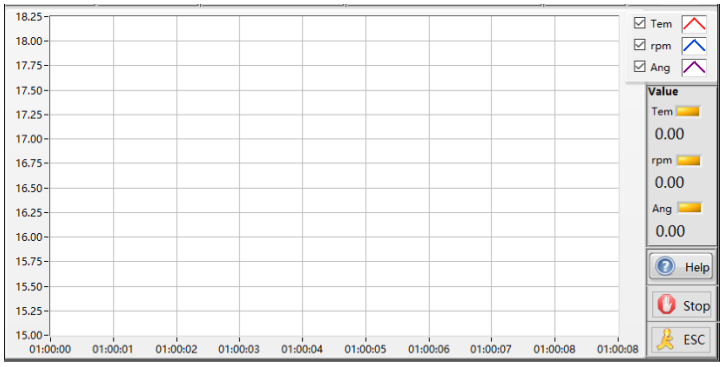


Figure 4-27 Front panel of wave chart

Waveform chart is the most commonly used data display controls in LabVIEW. The biggest feature of the waveform table is that the control contains a first in first out buffer (First In First Out, FIFO). This is a very important concept. The waveform's default first-in, first-out buffer size is 1024 data. In the editing environment, the buffer size can be set and cannot be changed at runtime. When new data arrives, the data is automatically added to the end of the buffer. When the buffer is full, the data that first entered the buffer will be automatically moved out of the buffer to keep the size of the buffer set. Since the waveform chart contains a FIFO, it is especially suitable for displaying real-time data. Waveform diagrams, waveform diagrams, XY waveform diagrams, intensity diagrams, etc. are collectively referred to as graphics controls. Graphical controls are the most complex controls in LabVIEW. A basic graphical control is a combination of many requirements. These elements are functionally equivalent to one control, but they cannot exist independently and must be attached to the graphic controls.

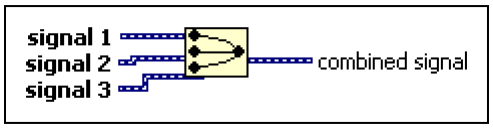


Figure 4-28 Merge signal function

Merges two or more supported signals, such as scalar numeric, 1D or 2D arrays of numeric, scalar Booleans, 1D or 2D arrays of Booleans, waveforms, or 1D arrays of waveforms, into a single output. Resize the function to add inputs. This function automatically appears on the block diagram when you wire a signal output to the wire branch of another signal.

4.7 Overrun Alarm

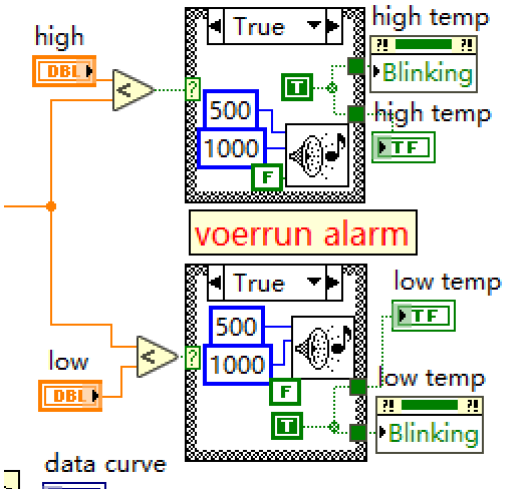


Figure 4-29 Block diagram of overrun alarm

When the temperature exceeds the threshold, the cooling system starts running. There is also need a buzzer and indicator to alarm if the temperature is too high.

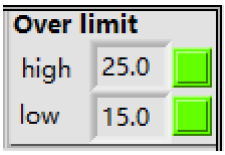


Figure 4-30 Front panel of overrun alarm

When the temperature exceeds the threshold, the buzzer will emit a buzzer with a duration of 1000ms at a frequency of 500Hz, and use the attribute node of the alarm indicator to control the indicator blinking.

4.8 Data storage

Files normally supported by programming languages, including text files and binary files. Essentially, a text file is simply a binary file with a special format. When data is stored in both the internal memory and the magnetic disk, the data is recorded in binary code using the binary code as the basic unit. The computer itself cannot store characters or numbers such as A, B, C, 1, 2, 3, and 4. Within the computer, they are codes that consist of 0 and 1 and are in bytes.

Text files store characters in ASCII mode. Naturally, when reading this file, the contents of the file should also be displayed in ASCII mode. Display text in ASCII mode, in addition to displaying characters (letters, numbers, punctuation), including undisplayable characters. After opening the VI file with Notepad, garbled characters appear because they contain many undisplayable characters.

The suffix of the file name usually indicates the type of the file. For example, TXT stands for text file and JPG stands for graphic file. Once we know the file type, we can interpret the data. It is also a text file, and if it is specially specified in the format, it can be changed to another file type. For example, an INI file is a text file that can be easily read through Notepad, but it has a specific format. Only by knowing its format and regulations, do you really know what it stands for.

Data is mostly binary when stored and communicated. The biggest feature of this approach is space and security. For example, to store 12345, if you need 5 bytes for text access, it is 31 32 33 34 35 for ASCII code. In binary mode, you only need two bytes 3039. For very large data, using binary files can save a lot of space and increase the speed of read/write. Another important feature of binary files is security. If you don't understand the format of the file, it is impossible to interpret the binary.

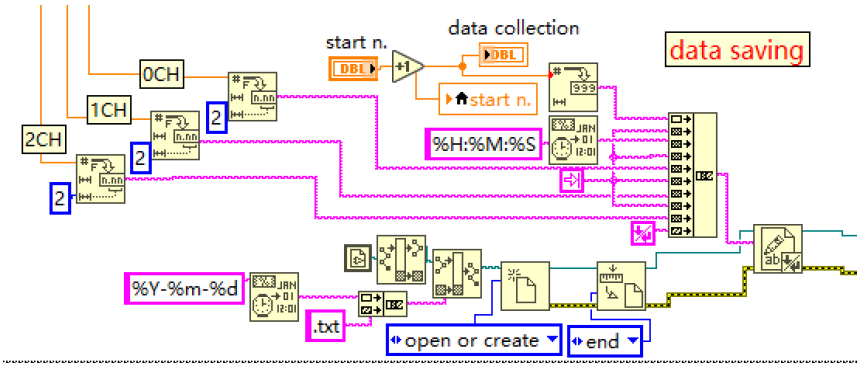


Figure 4-31 Block diagram of data saving

Saving the collected data in the path where the program is located, and create a .txt file with the date name (change .txt to .xls and save it as an excel file), and write the data after the last data each time.

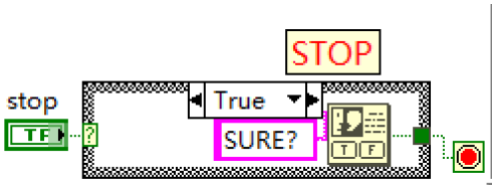


Figure 4-32 Block diagram of STOP button

Figure 4-33 Front panel of STOP button

When the stop button is pressed and released, a dialog box will pop up to indicate whether to stop the running program. If you select OK, the system will automatically stop after the current cycle (while loop detects its own conditional terminal at the end of each cycle). If you choose Cancel, the program will continue to run.

4.9 Demonstrate

Figure 4-34 shows an example of running the simulation test bench. Choosing three parameters to display: Temperature, engine speed and throttle angle. We can see that the curves display and the values.

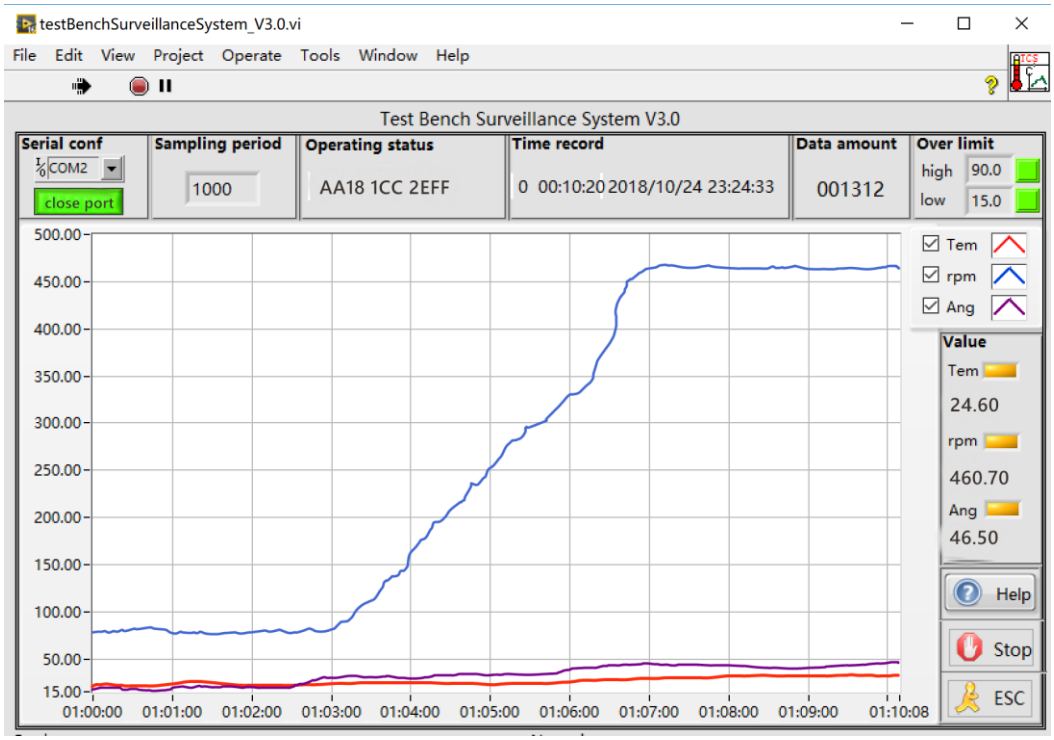


Figure 4-34 Running example

4.10 Debugging and optimization

After a continuous debugging, the Test Bench Surveillance System has updated to version 3.0. Now it is running stable, and all pre-planned features are implemented. It has a very good effect of didactic when using combine with the simulation of the test bench.

However, there are also many practical function can be added when update in the future. For example, it only could test 3 channels data at the same time, in the future, we can increase the number of the channels to 6, even 10, to test more data together.

Moreover, LabVIEW also have many 3D controls to build the 3D modeling. It can design a 3D engine modeling, which movement related to the value of the sensors.

CHAPTER 5: Design and installation

After complete the simulated engine test bench and the GUI system, in this chapter briefly introduces the design and installation of the real didactic test bench.

5.1 Engine test stand

The engine stand is the basis of the didactic test-bench. It is used to mounted not only the engine, but also the power supply system, the ignition system, the cooling system and all the electrical parts.

Moreover, it also requires high robustness but easy to disassemble and move. Comprehensive consideration of various situations, the final design of the engine test stand shows as follow:

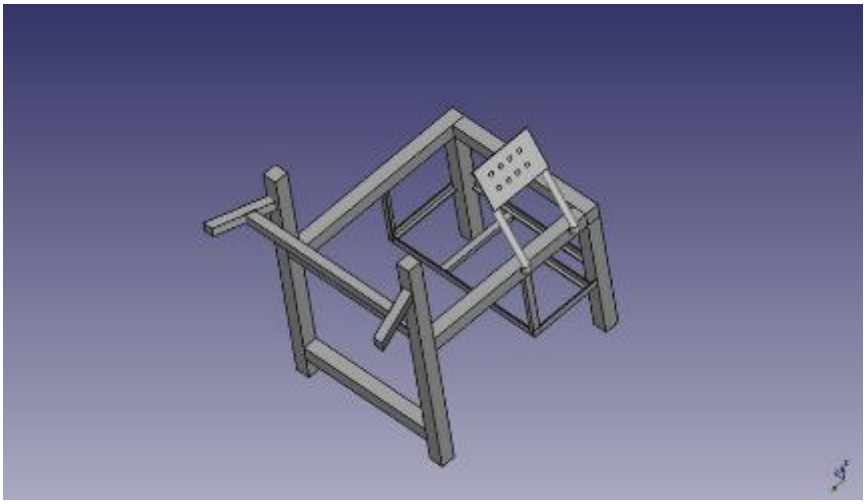


Figure 5-1 Engine test stand

5.2 Engine preparation

The engine Fiat punto 1.2 8V, which was produced in 2000. It means that it has been used for 18 years. Although it can still work normally, there are many damaged and aged parts. Dismantling every component, cleaning, checking, replacing and remounting.

5.3 Electrical part

The electrical part of the engine mainly includes connect sensors to ECU in correct pins shows in Figure 5-2, a relay protection system, starting system and power supply system.

For the electrical part, we need remap all the connectors of ECU, remap the relays and fuses according to the reference datasheet.

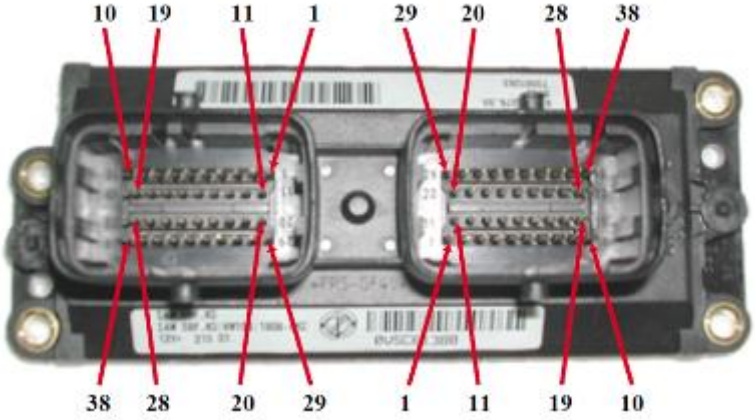


Figure 5-2 Pins of ECU

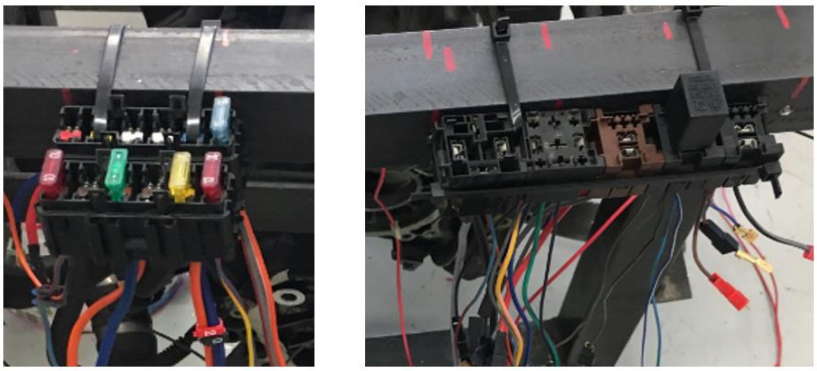


Figure 5-3 Relays and fuses

5.4 Mechanical part

A lot of auxiliary systems are required for an engine operation, such as cooling system and fuel supply system. Cooling system includes coolant, radiator, cooling fan, water pump and the like. The fuel supply system includes fuel tank, fuel pump and the like.

Figure 5-4 Fuel tank

Figure 5-5 shows the overview of the didactic engine test bench as follow:

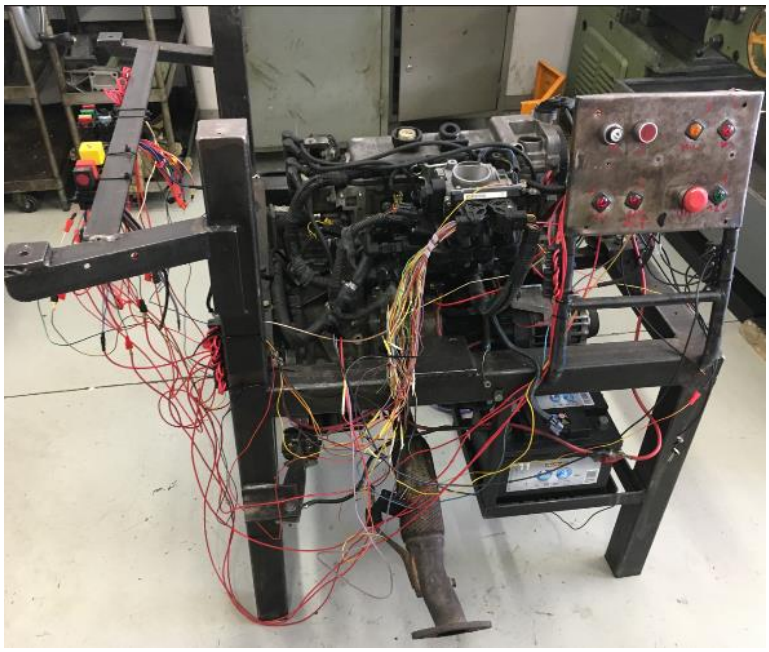


Figure 5-5 Overview of the didactic test bench

CHAPTER 6: Data collection and analysis

For the simulation engine test bench, collecting the data by the Test Bench Surveillance System and storage them on the upper computer, through serial communication. But it is different for the real engine. In this chapter briefly introduces the data acquisition strategy for the real engine and the method of the data analysis.

6.1 Data collection

The diagnostic tool can read data stored in the ECU, and it also can read the working state of the engine at real time through EOBD interface. There are two diagnostic tools used in the project, Bosch KTS-570 diagnostic electronic board and Multiecuscan diagnostic electronic board.

Figure 6-1 Bosch KTS 570

Figure 6-2 Multiecuscan diagnostic electronic board

To observe the data in the chart and save them, the diagnostic board must work with surveillance software on the upper computer. The diagnostic boards use the serial port for communication. The software for the Bosch KTS-570 diagnostic electronic board named ESI software, for the Multiecuscan diagnostic electronic board is Multiecuscan.

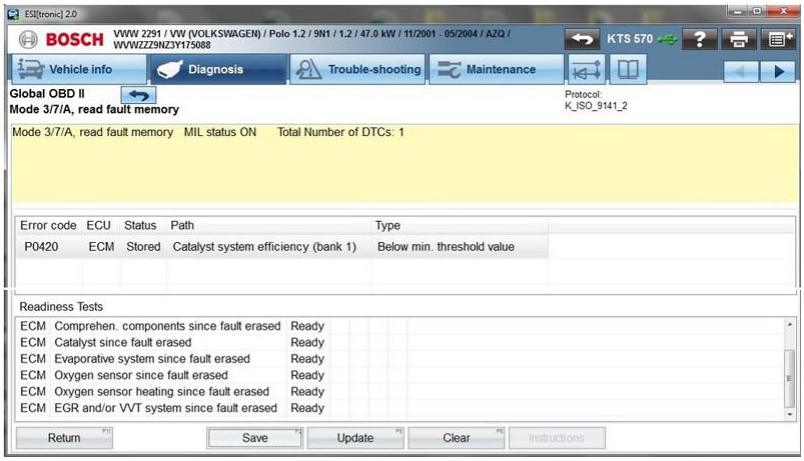


Figure 6-3 ESI software

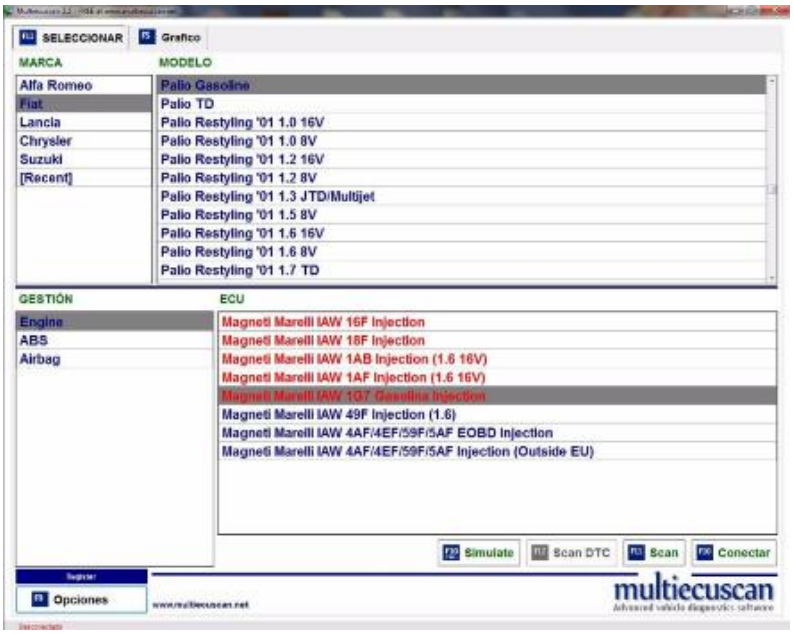


Figure 6-4 Multiecuscan software

6.2 Data analysis

Both software ESI and Multiecuscan provide a curve display, so it can do screenshot from the software directly, then compared with the reference data. And, they also save the data as an excel file. It gives a way to analysis the data in a professional data analysis software, Matlab.

Table 6-1 shows an example of an excel file of the storage data.

Time	Battery voltage	Engine speed	Engine temperature	Injection time	Throttle position	Throttle position	Idle actuator
sec	V	rpm	°C	ms	deg.	%	steps
0,00	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
1,15	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
2,30	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
3,46	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
4,63	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
5,78	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
6,94	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
8,10	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
9,24	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
10,40	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
11,56	122,000	0,0000	470,000	0,0000	0,0000	0,0000	1,150,000
12,71	96,000	0,0000	470,000	188,800	0,0000	0,0000	1,150,000
13,97	126,000	22,160,000	470,000	67,580	0,0000	0,0000	1,010,000
15,12	126,000	26,990,000	470,000	54,180	0,0000	0,0000	790,000
16,27	127,000	26,490,000	470,000	52,400	0,0000	0,0000	800,000
17,42	126,000	27,000,000	470,000	49,660	0,1000	0,0000	870,000
19,25	125,000	16,320,000	470,000	73,460	0,1000	0,0000	760,000
20,57	124,000	11,730,000	470,000	74,100	0,1000	0,0000	790,000

Table 6-1 Save the data as an excel file

Figure 6-5 shows the curve create of the storage data by Matlab. From the curve we can see the relationship of every parameter clearly.

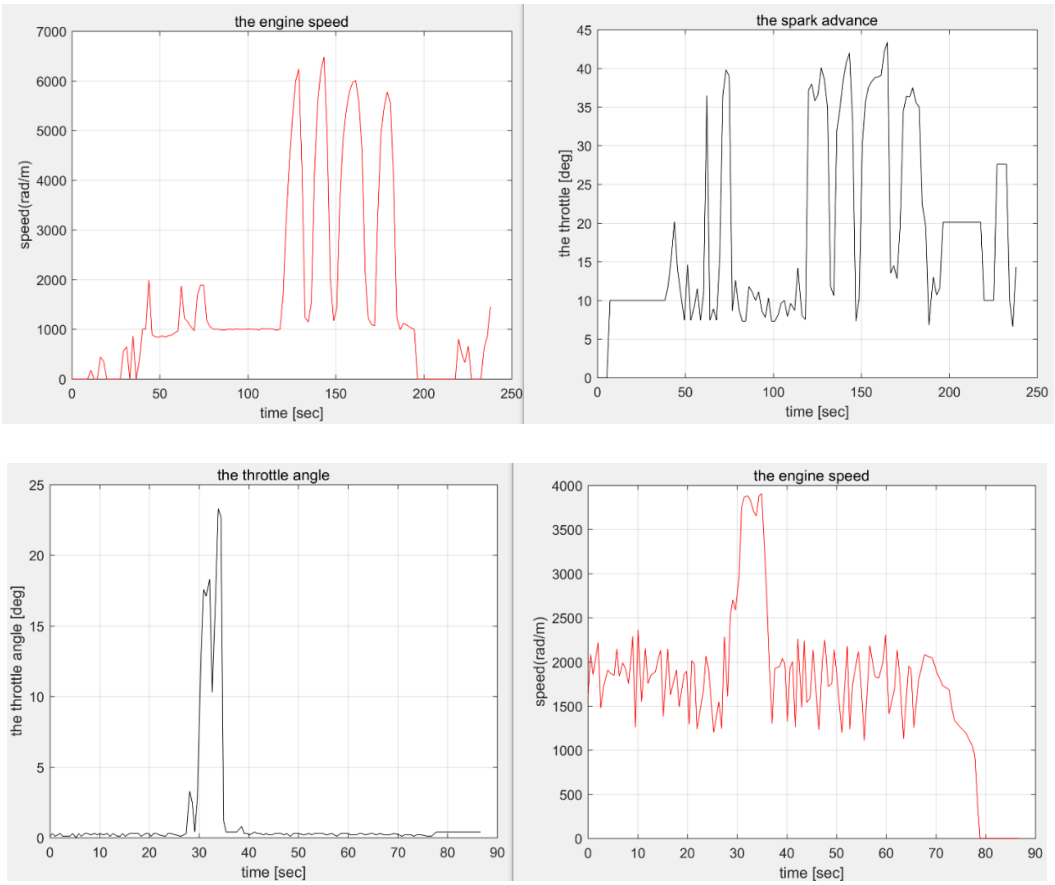


Figure 6-5 Comparison of curve changes in Matlab

CHAPTER 7: Conclusion

7.1 Conclusion

From the beginning, learn the purpose and content of the project, established theoretical basis, determined research method, established models, purchased and installed materials, analyzed and verified data, and finally put it into use, step by step. All the plans and goals of the project have been completed so far. The simulation engine test bench can perfectly restore the working principle of the engine, Test Bench Surveillance System runs stably and real test bench working well.

The didactic test bench of car's engine makes a good change about teaching of engine for the high school students. Compared to teaching only with books, on one hand, it allows students to see the structure and working principle of the engine more intuitively, and it is the best show of dynamic and static. On the other hand, it is also an innovation in the teaching model, which simplifies the complex structure and working process of the engine, making it easier for students to understand and remember, then to use it freely.

7.2 Future works

The didactic test bench of car's engine has been used in teaching at Filos formazione. The teacher and students give some feedback and suggestions. At first the engine test bench is designed for high school teaching. For students, they do not need to care about the parts which not related to the course, such as the circuit principle, the pin connection, etc. Therefore, every part of the simulation engine test bench can be integrated and hidden in a plastic box, leaving only the operation buttons, display and sensors. This not only protects the test bench, but also looks more beautiful. For the real engine test bench, there are too many wires bare outside, and the engine is too noisy when running. The engine stand also need to do insulation treatment

Appendix A: Mapping datasheet

MAPPA1						
Cilindro	Fase	TI	TM1	TM2	TS	
1	e	L	L	L	L	Formula Mappa
2	s	L	L	L	L	TCM=(60*1000)/(Rpm/2)
3	c	L	L	L	HS	MSG=TCM/720
4	a	HI	L	L	L	TI=MSG*(Rpm*0,021)
						TS=MSG*(Rpm*0,008)
1	s	L	L	L	L	TM1=90-TI
2	a	HI	L	L	L	TM2=90-TS
3	e	L	L	L	L	
4	c	L	L	L	HS	
1	a	HI	L	L	L	
2	c	L	L	L	HS	
3	s	L	L	L	L	
4	e	L	L	L	L	
1	c	L	L	L	HS	
2	e	L	L	L	L	
3	a	HI	L	L	L	
4	s	L	L	L	L	

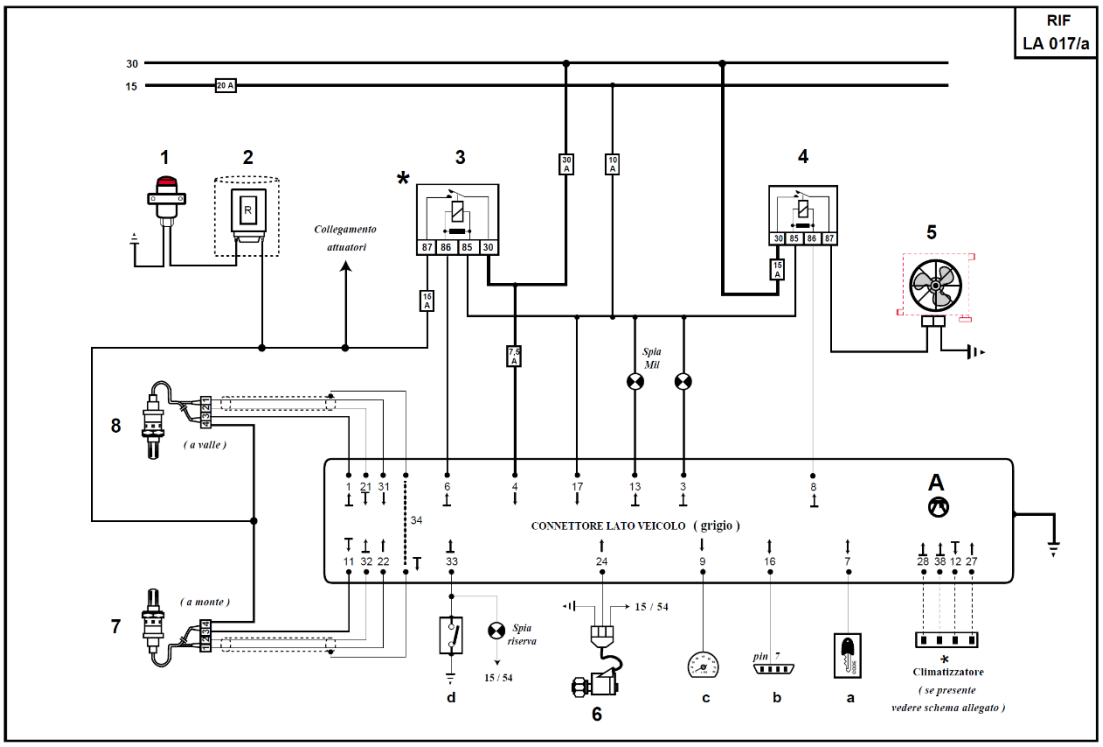
MAPPA2						
Cilindro	Fase	TI	TM1	TM2	TS	
1	e	L	L	L	L	Formula Mappa
2	s	L	L	L	L	TCM=(60*1000)/(Rpm/2)
3	c	L	L	L	HS	MSG=TCM/720
4	a	HI	HI	L	L	TI=MSG*(Rpm*0,021)
						TS=MSG*(Rpm*0,008)
1	s	L	L	L	L	TM1=90-TI
2	a	HI	HI	L	L	TM2=90-TS
3	e	L	L	L	L	
4	c	L	L	L	HS	
1	a	HI	HI	L	L	
2	c	L	L	L	HS	
3	s	L	L	L	L	
4	e	L	L	L	L	
1	c	L	L	L	HS	
2	e	L	L	L	L	
3	a	HI	HI	L	L	
4	s	L	L	L	L	

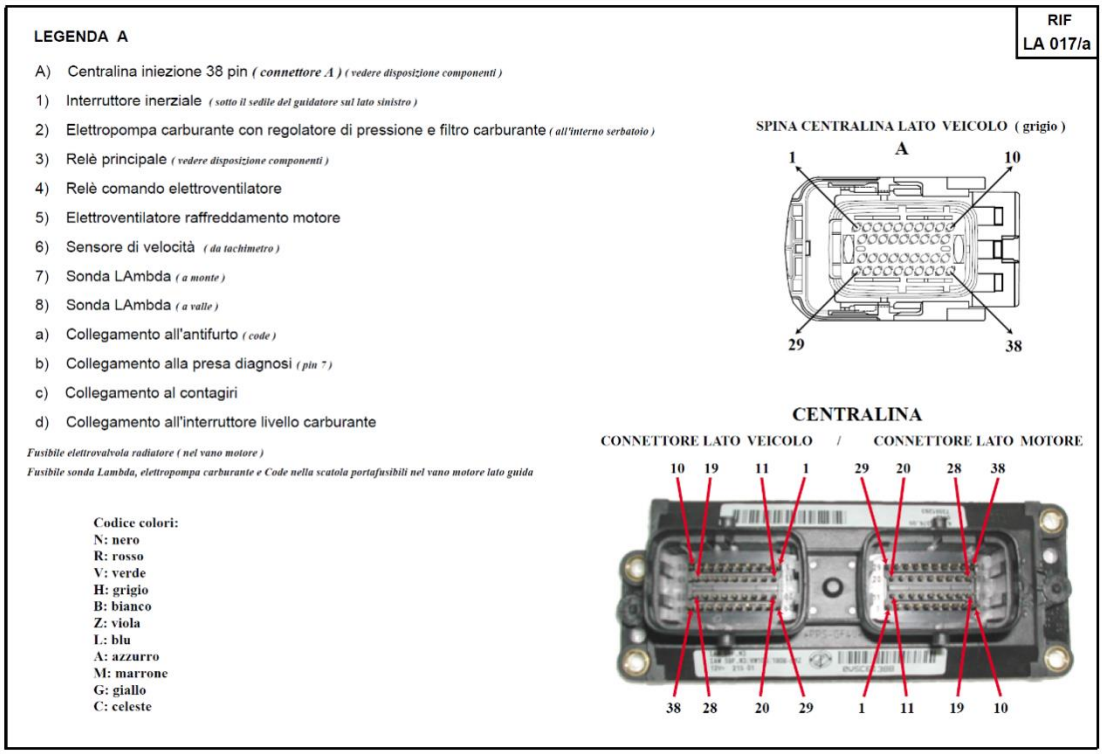
MAPPA3						
Cilindro	Fase	TI	TM1	TM2	TS	
1	e	L	L	L	L	Formula Mappa
2	s	L	L	L	L	TCM=(60*1000)/(Rpm/2)
3	c	L	L	L	HS	MSG=TCM/720
4	a	HI	HI	HI	L	TI=MSG*(Rpm*0,021)
						TS=MSG*(Rpm*0,008)
1	s	L	L	L	L	TM1=0
2	a	HI	HI	HI	L	TM2=(90+(90-TI))-TS
3	e	L	L	L	L	
4	c	L	L	L	HS	
1	a	HI	HI	HI	L	
2	c	L	L	L	HS	
3	s	L	L	L	L	
4	e	L	L	L	L	
1	c	L	L	L	HS	
2	e	L	L	L	L	
3	a	HI	HI	HI	L	
4	s	L	L	L	L	

MAPPA4						
Cilindro	Fase	TI	TM1	TM2	TS	
1	e	L	L	L	L	Formula Mappa
2	s	L	L	L	L	TCM=(60*1000)/(Rpm/2)
3	c	L	L	HS	HS	MSG=TCM/720
4	a	HI	HI	HI	L	TI=MSG*(Rpm*0,021)
						TS=MSG*(Rpm*0,008)
1	s	L	L	L	L	TM1=0
2	a	HI	HI	HI	L	TM2=(90+(90-TI))-TS
3	e	L	L	L	L	
4	c	L	L	HS	HS	
1	a	HI	HI	HI	L	
2	c	L	L	HS	HS	
3	s	L	L	L	L	
4	e	L	L	L	L	
1	c	L	L	HS	HS	
2	e	L	L	L	L	
3	a	HI	HI	HI	L	
4	s	L	L	L	L	

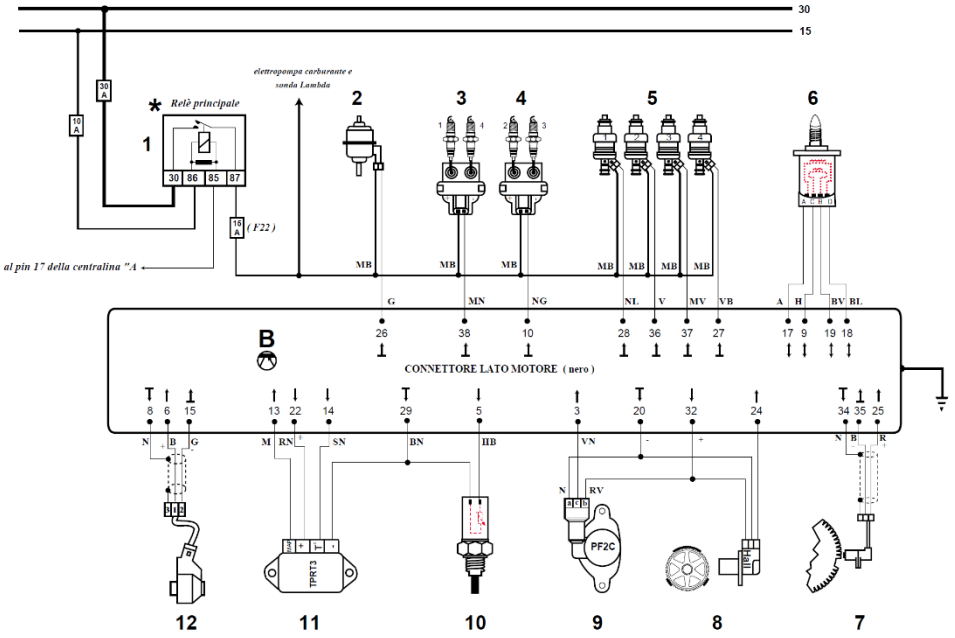
MAPPA5						
Cilindro	Fase	TI	TM1	TM2	TS	
1	e	L	L	L	L	Formula Mappa
2	s	L	L	L	HI	TCM=(60*1000)/(Rpm/2)
3	c	L	L	HS	HS	MSG=TCM/720
4	a	HI	HI	HI	L	TI=MSG*(Rpm*0,021)-TS
						TS=(MSG*(Rpm*0,021)+MSG*(Rpm*0,008))-1
1	s	L	L	L	HI	TM1=0
2	a	HI	HI	HI	L	TM2=MSG*(Rpm*0,008)-TS
3	e	L	L	L	L	
4	c	L	L	HS	HS	
1	a	HI	HI	HI	L	
2	c	L	L	HS	HS	
3	s	L	L	L	HI	
4	e	L	L	L	L	
1	c	L	L	HS	HS	
2	e	L	L	L	L	
3	a	HI	HI	HI	L	
4	s	L	L	L	HI	

Appendix B: Electric circuit of ECU





RIF
LA 017/b



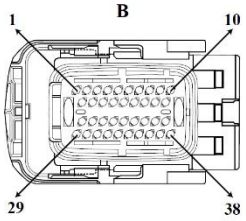
RIF
LA 017/b

LEGENDA B

B) Centralina iniezione 38 pin *(connettore B)*
1) Relè principale *(vedere disposizione componenti)*
2) Elettrovalvola vapori benzina *(canister)*
3) Bobina d'accensione *(cil 1 - 4)*
4) Bobina d'accensione *(cil 2 - 3)*
5) Elettroiniettori
6) Motorino del minimo
7) Sensore di giri / P.M.S.
8) Sensore di fase *(hall)*
9) Potenziometro acceleratore
10) Sensore temperatura motore
11) Sensore pressione assoluta / temperatura aria
12) Sensore di battito

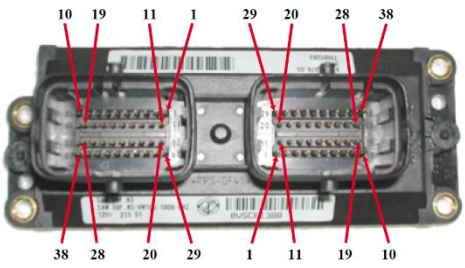
Codice colori:
N: nero
R: rosso
V: verde
H: grigio
B: bianco
Z: viola
L: blu
A: azzurro
M: marrone
G: giallo
C: celeste

SPINA CENTRALINA LATO MOTORE (nero)



CENTRALINA

CONNETTORE LATO VEICOLO / CONNETTORE LATO MOTORE



Appendix C: Reference of sensors and actuators

OPERAZIONI DA ESEGUIRE SUL SISTEMA IN ESAME					Pag 1 / RIF LA 017/a
MISURA	*CHIAVE*	PIN CONNETT. CENTRALINA		**OPERAZIONI DI CONTROLLO (su spina scollegata)**	VALORI e/o OSCILLOGRAMMA DA RILEVARE
(N.B. CHIAVE DISINSERITA/ASPETTARE 5 MINUTI E SCOLLEGARE SPINE CENTRALINA)					
SPINA LATO VETTURA A (grigio)					
Volt	Stop	⏚	**4**	**Tensione** permanente	**10 ÷ 15 Volt**
Volt	Mar	⏚	**17**	**Tensione** a chiave inserita	**10 ÷ 15 Volt**
Volt	Mar		**6 a** ⏚	**Controllo** comando relé principale alimentazione: elettropompa e attuatori: elettroiniettori, bobine A.T., riscaldamento sonda lambda, elettrovalvola intercettatore vapori benzina *(canister)* *(collegare pin 6 a massa)*	*(rotazione elettropompa)* / *(tensione su attuatori)* **10 ÷ 15 Volt**
N.B. **Considerando la difficoltà di operare su spina connettore centralina** **_ per controllo pressione carburante** *(Palio: contenitore vano motore)* **_ per controllo tensione su attuatori, ponticellare i terminali del relè con fusibile intermedio i term 30 con 87** *(contenitore vano motore)*					
	Mar		**8 a** ⏚	**Controllo** comando relé elettroventilatore (97°C) *(1° velocità)* *(collegare pin 8 a massa)*	*(rotazione elettroventilatore)*
	Mar		**18 a** ⏚	**Controllo** comando relé elettroventilatore *(se presente)* *(2° velocità)* *(102°C)* *(collegare pin 18 a massa)*	*(rotazione elettroventilatore)*
Volt	Mar	⏚	**27**	**Segnale** inserimento condizionatore *(se presente)*	**10 ÷ 15 Volt**
	Mar		**12 ad** ⏚	**Controllo** comando relé compressore condiz. *(se presente)* *(collegare pin 12 ad intervalli a massa)*	*(prova uditiva)* *(chiusura frizione compressore)*
	Mar		**3 a** ⏚	**Controllo** spia max temperatura acqua *(collegare pin 3 a massa)*	*(illuminazione spia)*
	Mar		**13 a** ⏚	**Controllo** spia avaria *(Mil)* *(collegare pin 13 a massa)*	*(illuminazione spia)*

Stop = Commutatore non inserito; Mar = Commutatore inserito

OPERAZIONI DA ESEGUIRE SUL SISTEMA IN ESAME					Pag 1	RIF LA 017/b
MISURA	CHIAVE	PIN CONNETT. CENTRALINA		OPERAZIONI DI CONTROLLO (su spina scollegata)	VALORI e/o OSCILLOGRAMMA DA RILEVARE	
SPINA LATO MOTORE B (nero)						
	Stop	*Per le 3 prove successive, collegare il pin 6 della spina A a massa (o in maniera descritta su spina A)*				
Volt	Mar	⏚	38	*Cil 1 - 4* **Controllo** continuità primario bobine	**10 ÷ 15 Volt**	
			10	*Cil 2 - 3*		
	Mar		28	*Cil 1* **Controllo** comando elettroiniettori *(collegare i pin indicati ad intervalli a massa)* *(scollegare elettropompa)*	*(prova uditiva)*	
			36	*Cil 2*		
			37	*Cil 3*		
			27	*Cil 4*		
	Mar		26 ad ⏚	**Controllo** elettrovalvola vapori benzina *(collegare pin 26 ad intervalli a massa)*	*(prova uditiva)*	
Ohm	Stop	+ 25	- 35	**Resistenza** sensore di giri / P.M.S.	**1100 ÷ 1400 Ω**	
Ohm	Stop	+ 6	- 15	**Resistenza** sensore di battito	**530 ÷ 580 Ω**	
Ohm	Stop	+ 32	- 20	**Resistenza** potenziometro acceleratore *(term a - b)*	**1200 ÷ 1250 Ω**	
		S 3	- 20	**Variazione** pista potenziometro *(term a - c)*	**0 → 1200 Ω**	
Ohm	Stop	18	17	*(tra i term A - D)* **Attuatore del minimo** *(tra i term B - C)*	**50 ÷ 60 Ω**	
		9	19			
Ohm	Stop	+ 5	- 29	*(motore)* **Sensori temperatura** *(aria) (nel sensore pressione assoluta)*	*0 °C* **9.5 KΩ** *10 °C* **3.8 KΩ** *20 °C* **2.5 KΩ** *60 °C* **500 Ω** *80 °C* **300 Ω** *90 °C* **200 Ω**	
		+ 14	- 29			

Stop = Commutatore non inserito; Mar = Commutatore inserito

Bibliography

[1] G. Ferrari. Internal Combustion Engines. Esculapio. 2014.

[2] J.B. Heywood. Internal Combustion Engines Fundamentals. McGraw-Hill, N.Y.. 1988.

[3] E.F. Obert. Internal Combustion Engines and Air Pollution. Harper & Row, Pub lishers, N.Y.. 1973.

[4] R. Basshuysen, F. Schäfer. Internal Combustion Engine Handbook. SAE International, Warrendale, Pa..2004.

[5] Chen Lvzhou. Arduino programming basis. Beijing. Beijing University of Aeronautics and Astronautics Press.2013

[6] Shen Jinxin. Arduino and labview development. Beijing. China machine press. 2014.3

[7] Arduino.< https://www.arduino.cc>. 2018

[8] Gary W. Jojnson, Richard Jennings. Labview Graphcial programming, 4th edition. United States. McGraw-Hill Companies. 2006

[9] Matlab and Simulink. The mathworks. Hill Drive Natick, Ma.2014

[10] Bosch diagnostic. <http://es.bosch-automotive.com>. 2018

[11] Multiecuscan. <https://www.multiecuscan.net>. 2018

[12] LabVIEW™ Real-TimeApplication DevelopmentCourse Manual. National Instruments Corporation. 2006

[13] Steven F. Barrett. Arduino Microcontroller processing for everyone, 2nd edition.

[14] How does the throttle affect the RPM of an engine. <www.physicsforums.com>. 2018

[15] Yu Chongzi. Arduino and LabVIEW development guide. 2014.7

[16] Chen Shuxue, Liu Xuan. LabVIEW treasured book. 2017.9

[17] Engine Mapping (Automobile). <www.what-when-how.com>. 2018
[18] How does KIVA4 calculate piston location, with and without wrist pin offsets. <www.homepages.cae.wisc.edu>. 2018

[19] LM2596HV, Datasheet. <www.htckorea.co.kr>. 2018

[20] Marc Gregoire. Professional C, 3rd edition. 2015.5

[21] David Johns, Ken Martin. Analog integrated circuits and design. 2016

[22] Modeling Engine Timing Using Triggered Subsystems. <www.mathworks.com>. 2018

[23] P.R. Crossley and J.A. Cook, IEEE® International Conference 'Control 91', Con ference Publication 332, vol. 2, pp. 921-925, 25-28 March, 1991, Edinburgh, U.K.

[24] Mark M.Horestein, Microelectronic circuits and devices, 2nd edition. 2009

[25] L298N, Datasheet. <www.sparkfun.com>. 2018

[26] Transmissive Optical Sensor with Phototransistor Output, Datasheet. <www.vishay.com>. 2018

[27] Cai Ruiyan. The principle and application of Arduino. Tsinghua University press. 2012

[28] Song ming. Detailed LabVIEW programming. 2015.5. Beijing