HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
OFFICE FOR INTERNATIONAL STUDY PROGRAMS

PROJECT: DESIGN OF THE ELECTRONIC THROTTLE VALVE POSITION CONTROL SYSTEM

INSTRUCTOR: Dr. TRAN DANG LONG

Student: (CLASS CC18OTO)

NHU QUOC HUY 1852412

HCMC, 2021

VIETNAM NATIONAL UNIVERSITY OF HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY

Faculty of Transportation Engineering

Department of Automotive Engineering

SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom – Happiness

MISSION OF THESIS

1. **Student's name:** Nhữ Quốc Huy **- Student ID:** 1852412
2. **Major:** Automotive Engineering **- Class:** CC18OTO
3. **Thesis title:** Design of the electronic throttle valve position control system
4. **Content:**
   - ❖ Introduction of the project.
   - ❖ Objective of the project.
   - ❖ Working conditions and technical requirements.
   - ❖ General layout design:
     - Design the structure diagram.
     - Design the principle diagram.
   - ❖ Technical design:
     - Select the components for the product.
     - Design the timing diagram.
     - Design algorithm flowcharts.
     - Arduino programming:
       - ➢ Timer control.
       - ➢ PWM control (using PID controller).
       - ➢ Analog to Digital Converter control.
   - ❖ Result:
     - Simulate the working process of the system using Proteus software.
5. **Product:**

✓ Presentation report.	✓ Poster.
✓ Microcontroller program.	✓ Proteus simulation.

6. **Assigned day:** April, 2021.
7. **Finished day:** June, 2021.

The content and requirements of the thesis is already approved by the Head of Department of Automotive Engineering.

HCMC, day..... month…… year 2021 *HCMC, day..... month…… year 2021*

Head of Department **Instructor**

I. Introduction:

1.1 Objective:

With the development of modern technology in the automotive field. The modernization of automobile operating processes increasingly presents a competitive advantage among company's car manufacturers in the world.

Among them, the development of the control system. Throttle valve control plays an important role in optimizing performance of the engine, helping to save fuel, optimize the amount and composition of emissions.

Therefore, this project will design an electronic throttle valve position control system.

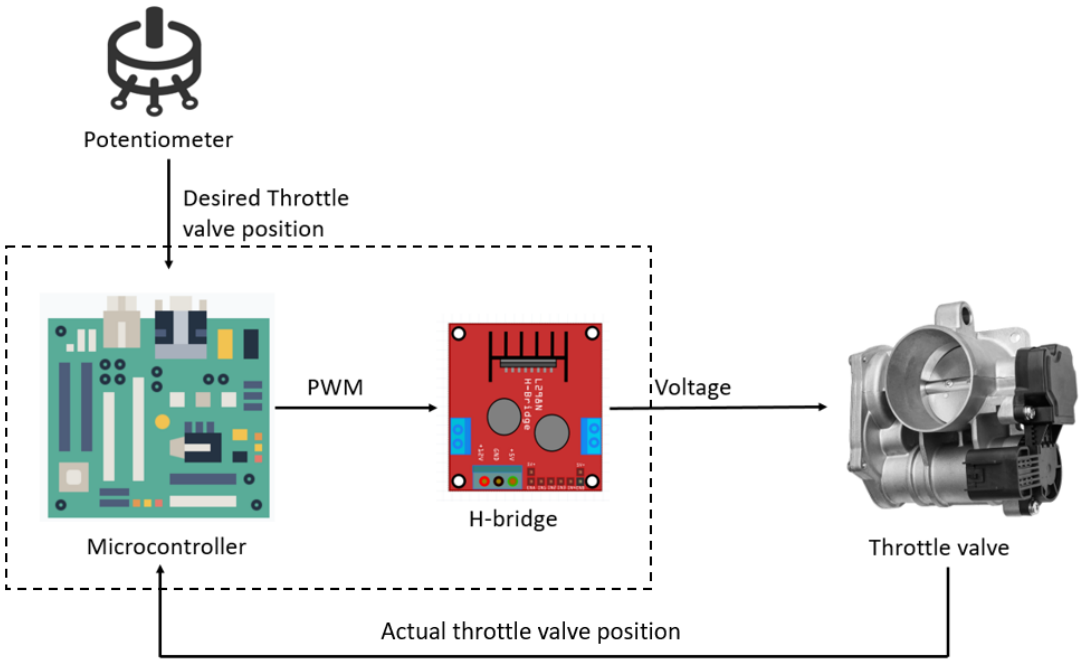

Figure 1: Pictorial diagram of the system

- This system will use throttle valve with throttle valve position sensor to generate the actual throttle valve position value and feed back to the microcontroller a voltage value.
- Using potentiometer to simulate the accelerator pedal to get desired throttle valve position value.

1.2 Working conditions and technical requirements:

1.2.1 Working conditions:

- Throttle valve is at good operating condition.
- Working under high frequency: microcontroller and executive elements working at high frequency (1000Hz), vary continuously.
- Room temperature 30^{o}C, no external shock or vibration.
- All electrical components are in good condition and tested carefully before using.

1.2.2 Technical Requirements:

- Functional requirements:
  - Regulate precisely opening and closing angle of the throttle valve.
  - Microcontroller: Read and calculate the exact input and output value of the system
- Technology requirements:
  - Using throttle valve that has throttle valve position sensor (Hall effect).
  - Choosing suitable components for the system.

II. General layout design:

2.1 General layout diagram:

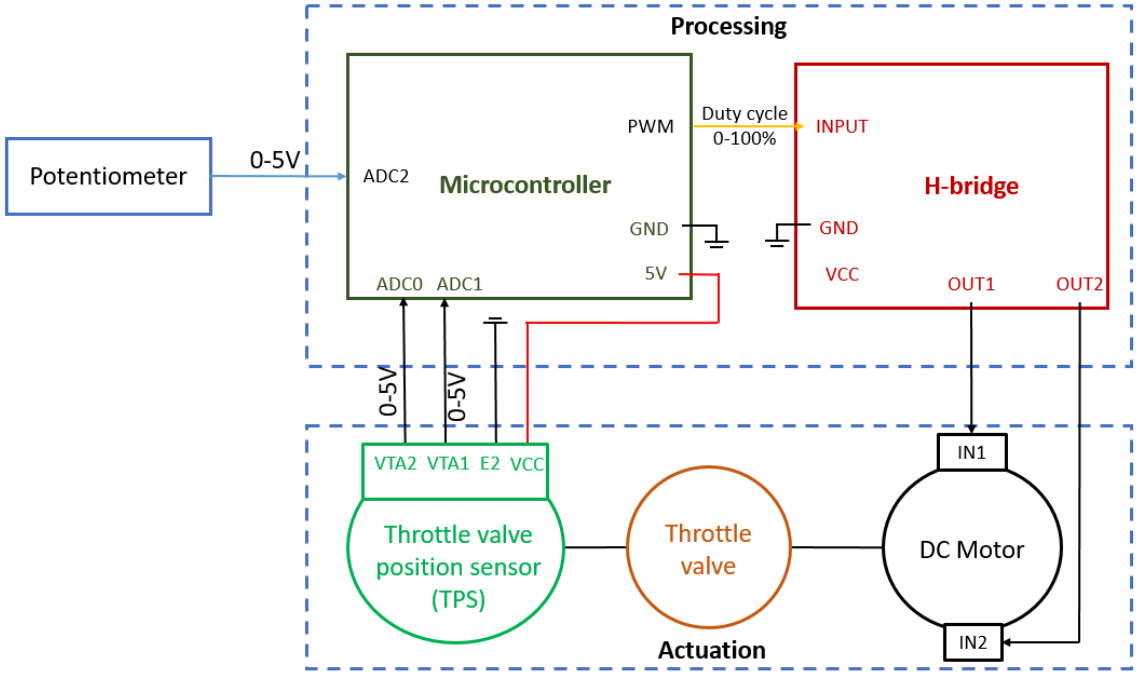

Figure 2.1: General layout diagram of electronic throttle valve system

This system includes 2 main parts:

- Controlling part: using Arduino and H-bridge
  - Microcontroller function:
    - Read and calculate the desired throttle valve position from accelerator pedal
    - Read and calculate the actual throttle valve position from the TPS
    - Calculate the width of the PWM value
    - Create the period of updating the desire and actual throttle valve position value
    - Calculate the PWM value to control the H-bridge
  - H-bride:
    - Receive PWM from Arduino
    - Generate voltage to control the motor

- Actuation part: using Throttle valve position sensor, throttle valve and the motor
    - Motor: using the voltage generated from the H-bridge to control the throttle valve
    - Throttle valve position sensor (TPS): receive the actual throttle valve position value and send to the Arduino for calculating purpose
    - Throttle valve: Rotational moment of the motor combine with the reaction moment of the reverse spring that create actual throttle valve position value which is read by the TPS

2.2 Working principles of the system:

2.2.1 Input and output signal:

	Input	**Output**
Microcontroller	- Analog signal at A0 – voltage value from the accelerator pedal - Analog signal at A1 – voltage value from throttle valve position sensor	-Digital signal at pin 9 – PWM value to control the motor
PID control	- Percentage of desire throttle valve opening position - Percentage of actual throttle valve opening position	-Digital signal at pin 9 – PWM value to control the motor

Table 2.1: Input and output signal

2.2.2 Input and output relationship:

- The relationship between 2 analog input value and digital output value (PWM value) is the PID controller
    - ➔ PID controller receives 2 analog input to calculate a Voltage value to calculate the PWM value as we will mention in PID controller

2.2.3 Processing scheme:

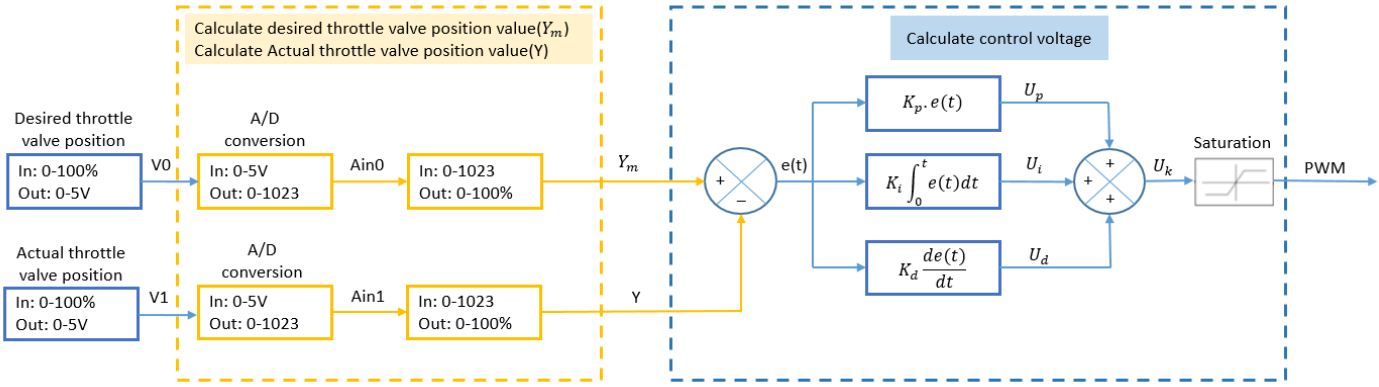

Figure 2.2: Processing scheme

In order to calculate the PWM width to control the motor, we use the PID controller. PID controller is a control loop mechanism, it continuously calculates an error value as different between desired and actual throttle valve position value and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), hence the name.

PWM value is indirectly calculated through the control variable U_k, then transfer to PWM width

With:

Ym: Desired throttle valve position value
Y: Actual throttle valve position value
e = Ym – Y: different between desired and actual throttle vale position value (Error)

The control variable U_k is calculated by 3 components:

+ $U_p = K_p.e(t)$ → Proportional with error → Proportional component
+ $U_i = K_i.\int_0^t e(t)dt$ → Proportional with cumulative value of error → Integrational component
+ $U_d = K_d.\frac{de(t)}{dt}$ → Proportional with variable function of error by time → Derivative component

Sum of 3 components we have the control variable U_k

$$U_k = U_p + U_i + U_d = K_p.e(t) + K_i.\int_0^t e(t)dt + K_d.\frac{de(t)}{dt}$$

Calculate the PWM width

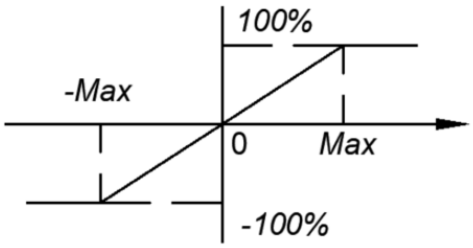

Figure 2.3: Transfer part

The calculated voltage value from PID controller will be transferred into PWM.

If U_k < -Max → PWM = -100%

If U_k > Max → PWM = 100%

If $U_k \epsilon$ (-Max;Max) → PWM = $\frac{U_k \times 100}{Max}$(%)

⇨ Depend on Kp, Ki, Kd we can regulate the calculating speed and stability of PWM value

III. Technical design:

3.1 Hardware design:

3.1.1 Electronic throttle valve:

3.1.1.1 Electronic throttle body:

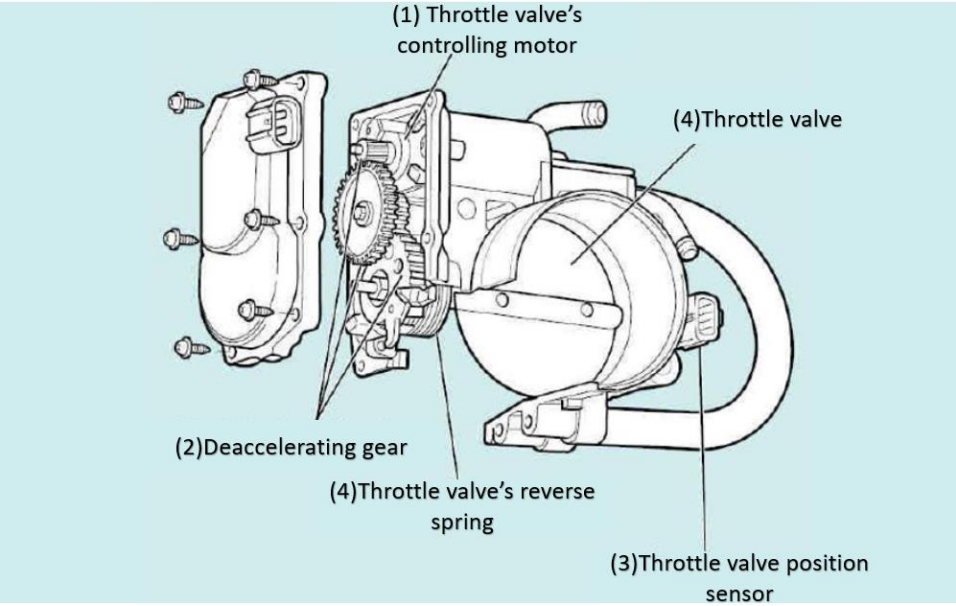

Figure 3.1: Throttle valve structure

Electronic throttle valve is a controlling module of an engine that has a fuel injection system, which has the function of changing the amount of intake air into the combustion chamber through regulating the opening and closing of the throttle valve through a DC motor.

The main parts of the electronic throttle valve:

(1) Throttle valve's controlling motor
(2) Decelerator gears
(3) Throttle valve position sensor
(4) Throttle valve and reverse spring

Working principles:
When the motor is charged, moment from the motor will transfer to the decelerator gears that make the throttle valve open at a certain angle. When the moment of the motor and the reaction moment of the spring is equal, the throttle valve position is fixed.

When the power is cut off through the motor, the spring will close to the throttle valve. The change in the throttle valve position will be read by the throttle valve position sensor (TPS) and produce a voltage signal

3.1.1.2 Throttle valve position sensor (TPS):

There are 3 types of TPS: throttle position sensors with built-in end switches also known as Closed Throttle Position Sensor, the potentiometer type and the combination of both these types.

Using an IC Hall sensor to measure the opening and closing of the throttle valve in order to transmit the signal to ECU. Then, ECU uses the signal of the TPS to calculate the load level of the engine to regulate the ignition time, fuel cutoff, ignition angle, idling and shifting gears

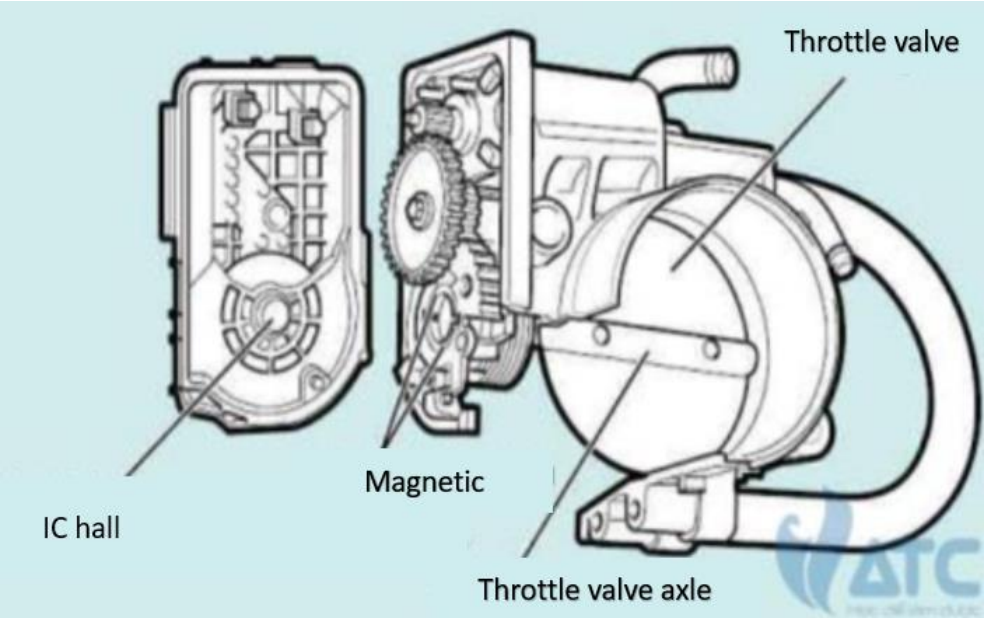

Figure 3.2: Hall throttle valve position sensor

Working principle: when the throttle valve opens/closes due to acceleration pedal, the TPS will record the position of the throttle valve by transmitting the angle into voltage.

Working principle of IC Hall sensor: when the valve is opened, the magnets rotate at the same time and change their position. The IC hall detects the change in magnetic flux due to the change in magnets position and creates voltage through VTA1 and VTA2.

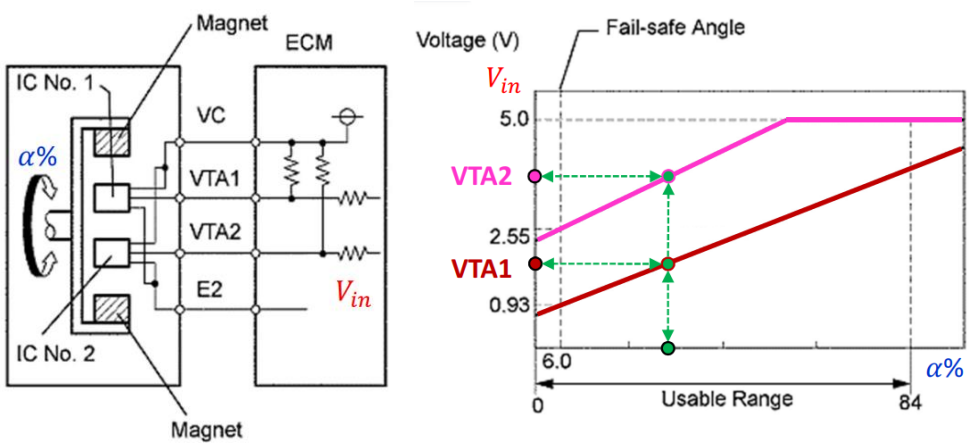

Figure 3.3: Hall sensor and throttle valve position signal characteristic diagram

2 IC Hall read the signal from the magnetics which is placed coaxial with the throttle valve

The output of the TPS is:

- VTA1: Determine the opening angle of the valve
- VTA2: Test the error of VTA1. When TPS works at normal condition, Different between VTA1 and VTA2 is 1.11V.

The throttle valve opening angle detected by the sensor terminal VTA1 is expressed as a percentage

- Between 10% and 24%: throttle valve fully close → The output voltage is 0.8V
- Between 64% and 96%: throttle valve fully open → The output voltage is 3.98V

Approximately 16%: Fail-safe angle(6^o)

3.1.2 Arduino:

Using Arduino UNO R3 in order to calculate the PWM width (Duty cycle), export PWM signal to control the H-bridge.

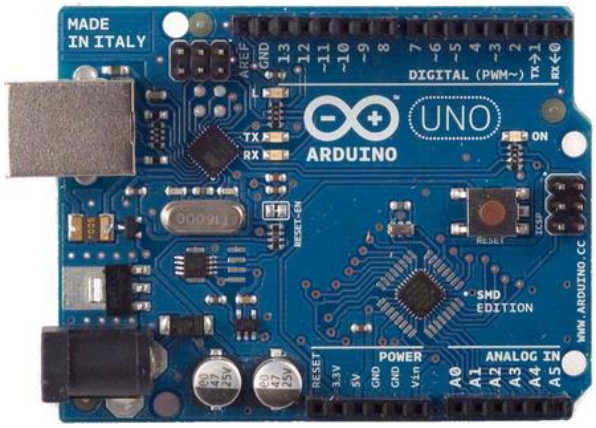

Figure 3.4: Arduino UNO R3

Arduino UNO R3:

Microcontroller	Atmega328p with 8 bits
Power	5V DC(USB only)
Clock speed	16 MHz
Digital pins	14 (6 pins for PWM)
Analog pins	6
Timer/counter	Timer0, Timer1, Timer2

Table 3.1: Arduino's parameter

Advantages:

- Simple structure, Low cost to purchase
- C programing language
- Variable library → Easy to access

By using Arduino in this project, it will create a period to read the desired throttle valve position value and the actual throttle valve position value. Then calculate the PWM value and export into H-bridge to control the motor to regulate the opening and closing position of the throttle valve.

3.1.3 H-bridge:

H-bridge L298N is a module that supports the motor to work more efficiently. The PWM value is generated from Arduino to the H-bridge so we can regulate the PWM value to control the motors under different conditions. This module has 4 output pins to connect with the motor and 4 input pins to receive signals to Arduino, ENA and ENB pins is a comparable enable input pin, at Low states disables the bridge A (enable A) and/or the bridge B (enable B).

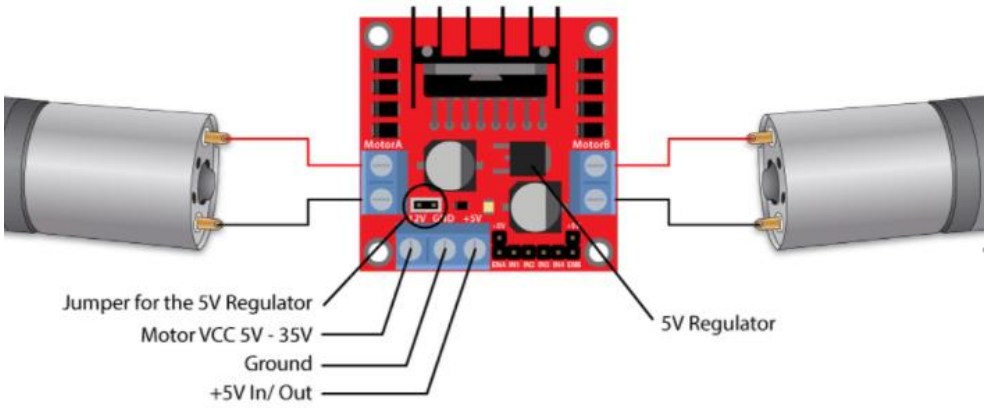

Figure 3.5: H-bridge L298N with 2 motors

- ⇨ This module can independently control 2 motors but in this project we only need to control 1 motor

Working principle of controlling the rotational direction of the motor:

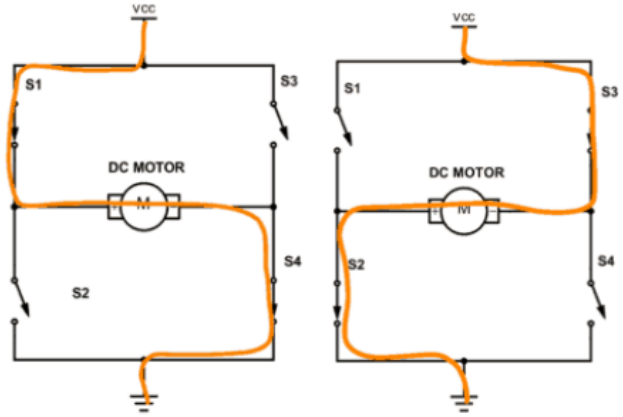

Figure 3.6: H-bridge working principles diagram

In order to control the rotational direction of the motor with the H-bridge, we control each of transistor pair or switching pair (S1, S4) and (S2, S3)

- Forward direction: close (S1, S4). The voltage will flow from VCC through S1 to motor's positive pole then from motor's negative pole through S4 to Mass
- Reverse direction: Close (S2, S3). The voltage will flow from VCC through S3 to motor's negative pole then from motor's positive pole through S2 to Mass

3.1.4 Non-inverting operational amplifier

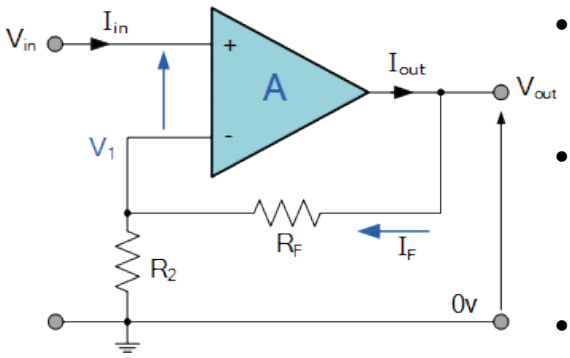

Figure 3.7: Non-inverting operational amplifier

- Vin: input signal voltage is applied directly to the non-inverting (+) input terminal which means that the output gain of the amplifier becomes "**Positive**" in value.
- Feedback control of the non-inverting operational amplifier is achieved by applying a small part of the output voltage signal back to the inverting (–) input terminal via a R3-R4 voltage divider network, again producing **negative** feedback
- Formula to calculate the output voltage of a potential divider network:

$$V_{out} = V_{in} \times \frac{R2 + Rf}{R2}$$

3.1.5 RC low pass filter

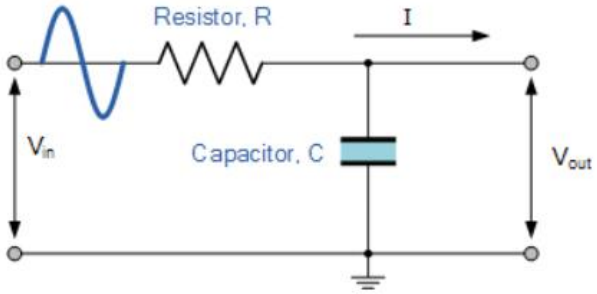

Figure 3.8: RC low pass filter

- Passive filters are generally constructed using simple RC (Resistor-Capacitor) networks. So it has no signal gain, therefore the output level is always less than the input.
- Only allows low frequency signals from 0Hz to its cut-off frequency, fc point to pass while blocking those any higher.

3.1.6 Electrical scheme:

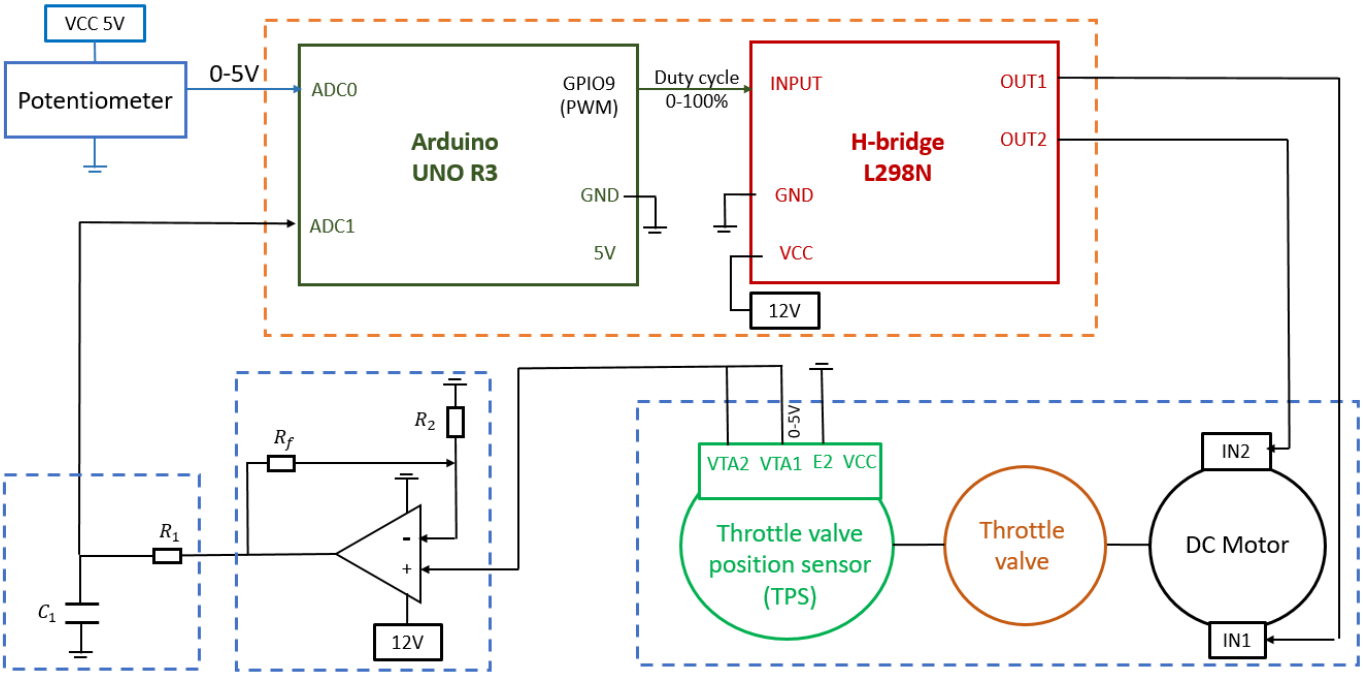

Figure 3.9: Electrical scheme of the system

After choosing components for the system, combine all the components to make an Electrical scheme.

3.2 Software design:

3.2.1 Timing diagram

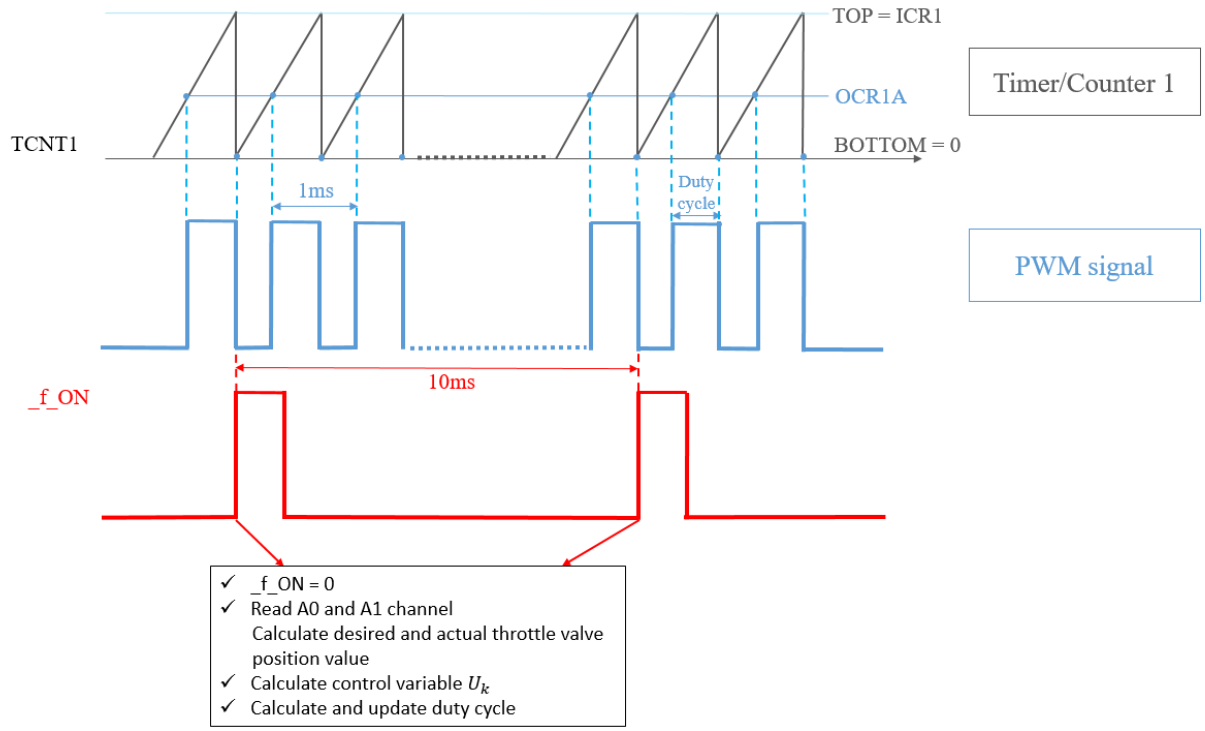

Figure 3.10: Timing diagram

- TCNT1: Timer counter 1 register
  - 16 bits' register, save the Timer/counter 1 value, allow to read and write directly
- OCR1A: Output compare register 1 A
  - Save the compare value of register A
  - When the Timer/counter 1 is running, the TCNT1 increases
  - The value of TCNT1 is continuously compared with value in OCR1A
  - When OCR1A = TCNT1 → compare match
- ICR1: Input capture register 1
  - When TCNT1 = ICR1 → Interrupt will be actuated

The Timer/Counter 1 has the following function:

- Generate PWM with frequency = 1000 Hz
- Create a calculating period with frequency = 100 Hz

After every 10ms period do the following function:

- _f_ON = 0.
- Read A0 and A1 channel.
  Calculate desired and actual throttle valve position value.

- Calculate the control variable U_k.
- Calculate and update duty cycle.

3.2.2 Algorithms:

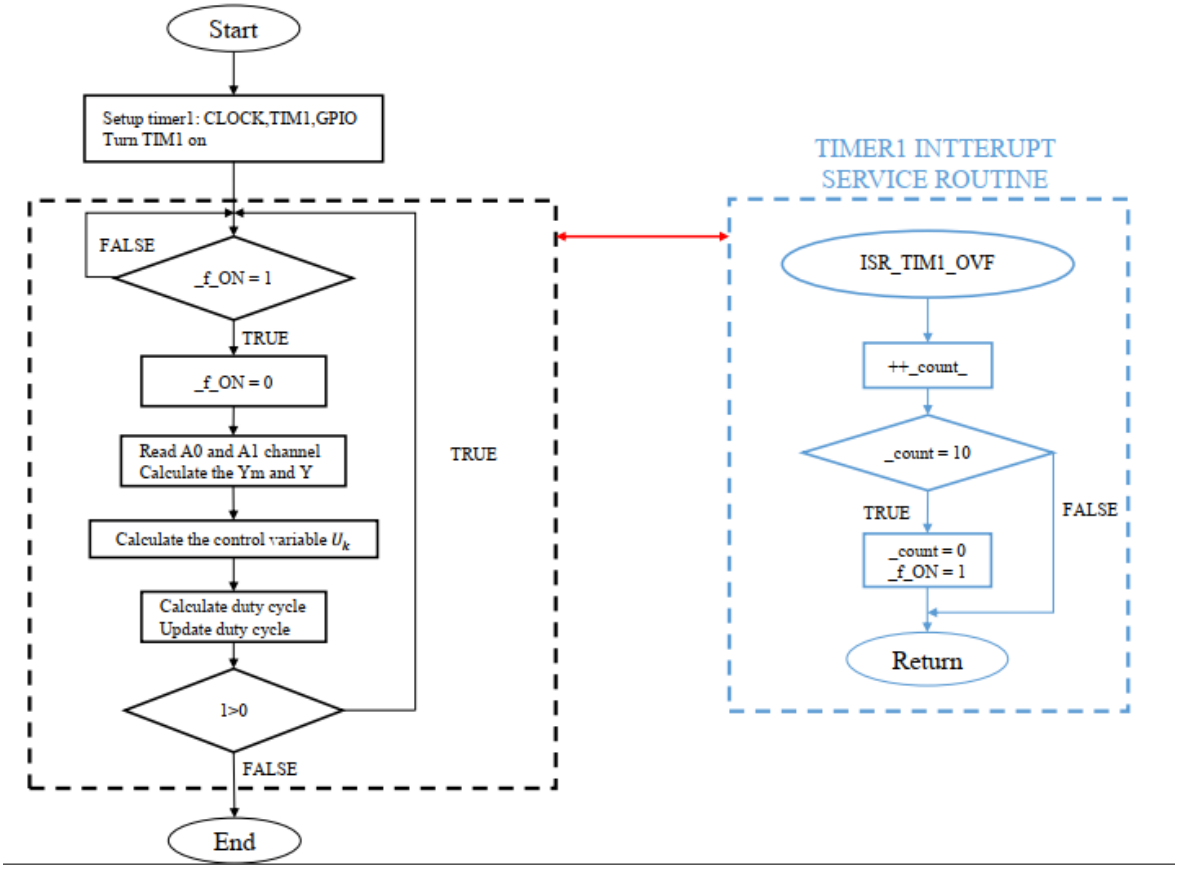

Figure 3.11: Flowchart of the system

- Start the program:
  - Declare the variable:
    - Input variables
    - PID controller's variables
  - Setup clock for TIMER1
  - Setup input and output of the microcontroller:
    - Analog input A0 and A1
    - Digital output pin 9 (OCR1A) to control the PWM
  - Turn TIMER1 on
- Calculate the desired throttle valve position value
  - Read the voltage value from the potentiometer at A0
  - $Ym = \frac{\_voltage0*100}{1023}$ %
- Calculate the actual throttle valve position value
  - Read the voltage value from pin A1 of Arduino.
  - $Y = \frac{\_voltage1*100}{1023}$ %

- Calculate PWM width
  - Using PID controller to calculate the PWM width
  - PWM = $\frac{U_k * 100}{Max}$(%)

IV.Proteus simulation:

4.1 Electrical diagram in Proteus software:

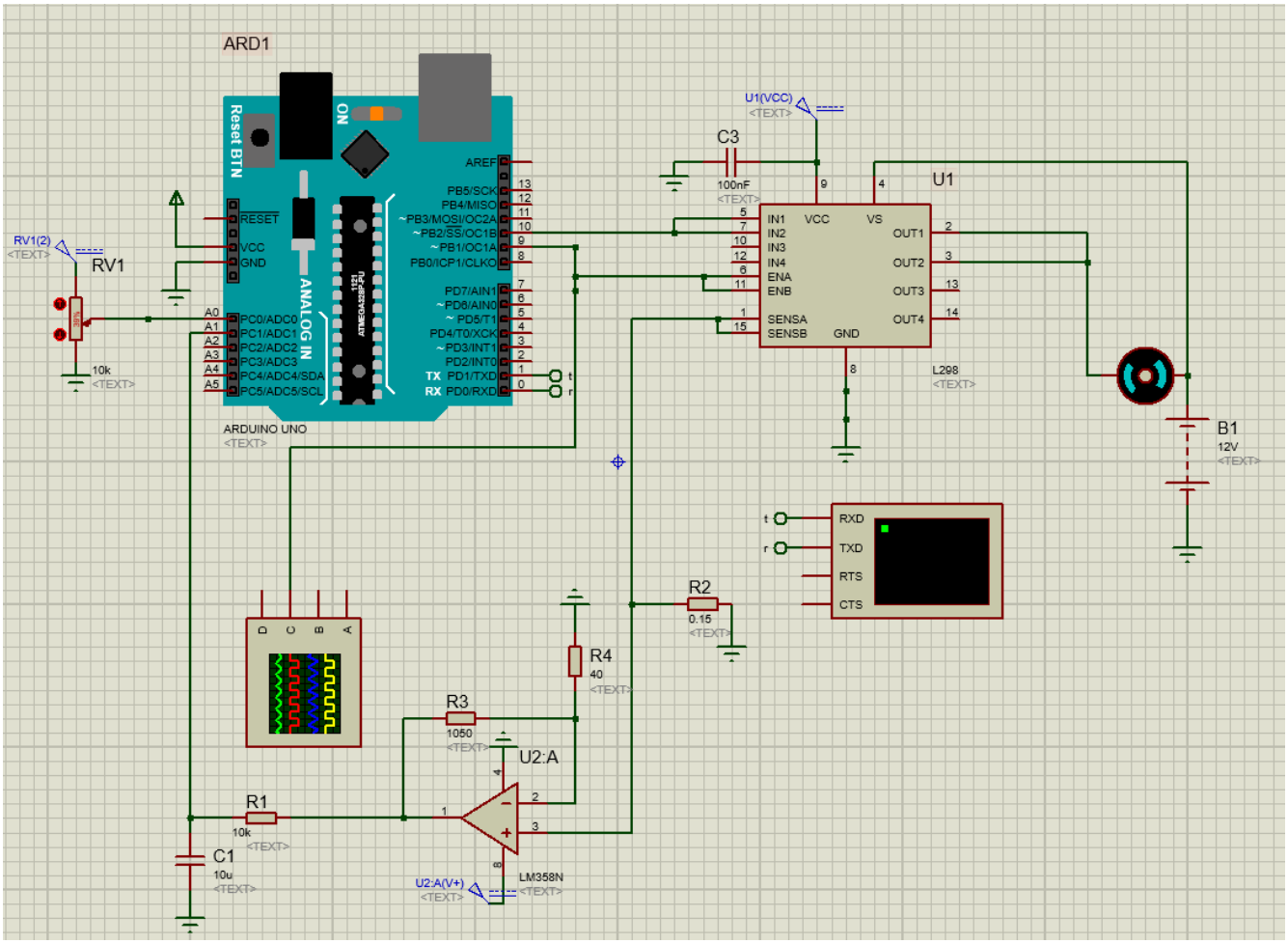

Figure 4.1: Electrical diagram in proteus

4.2 Diagram function:

Component	Connection	Function
Potentiometer	Output pin is connected to the A0	Generate voltage value to simulate desired throttle valve position value
H-bridge L298N	• 2 output pins connected parallel and connected with 1 pin of the motor • 2 input pins connected parallel and connected with pin 10 of the Arduino • Pin ENA connected with pin 9 of the Arduino	Control the DC motor to get the actual throttle valve position value
Resistance R2		Simulate as a sensor to send voltage value to A1 which is the actual throttle valve position value
LM358	Combine with R3 and R4 to work as a non-inverted operational amplifier	Work as an amplification to amplify the voltage value from A1
RC Low pass filter	Include Capacitor C1 and Resistance R1 connected directly to A1	Lower the frequency at the cut-off point that make sure the system working continuously
Digital oscilloscope	Connected to the pin 9 of the Arduino	Display the PWM
Virtual Terminal	• Pin RXD connected to pin TXD of Arduino • Pin TXD connected to pin RXD of Arduino	Display the voltage value of potentiometer and DC motor

4.3 Simulating result discussion:

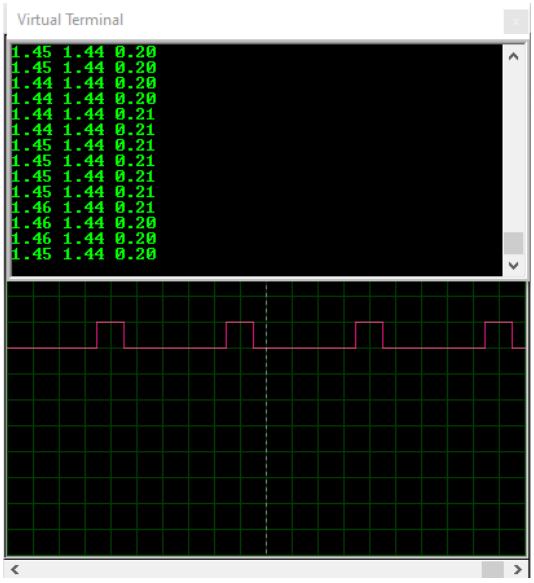

Figure 4.2: Throttle valve open 20%

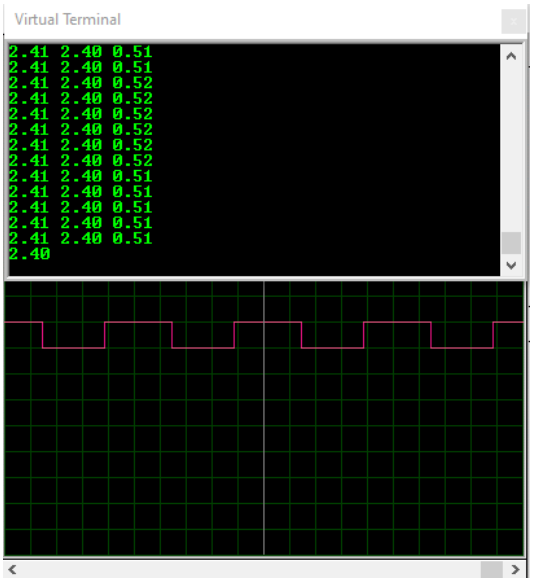

Figure 4.3: Throttle valve open 50%

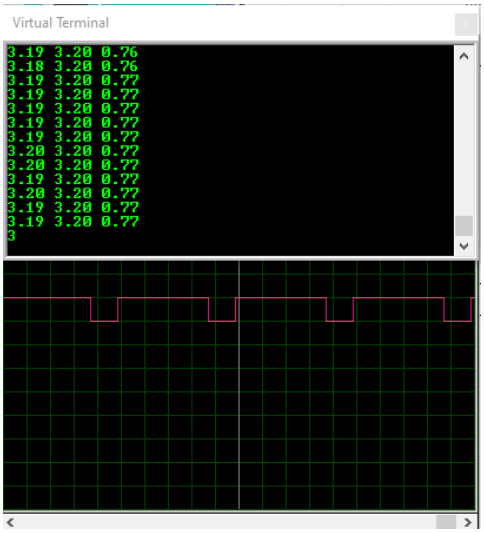

Figure 4.4: Throttle valve open 75%

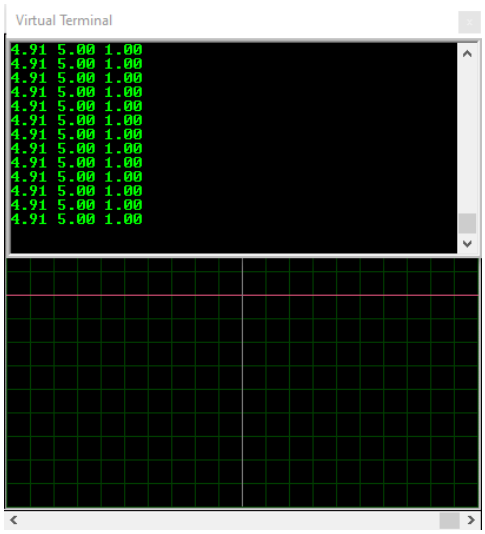

Figure 4.5: Throttle valve open 100%

Desire voltage value	Actual voltage value	PWM value	ERROR
25% ~1.44V	~1.45V	0.20	0.6%
50% ~2.40V	~2.41V	0.51	0.4%
75% ~3.20V	~3.19V	0.77	0.3%
100%~5	~4.91	1	1.8%

Table 4: Result of testing

➔ **Discussion:**

Due to PID controller, the system can calculate the ERROR between the desire and actual throttle valve position value and regulate the PWM value until the ERROR decrease.

V. Conclusion:

After working on the project of controlling throttle valve system by using Arduino is a microcontroller and proteus to simulate the system

The system has satisfied the following conditions:

- Regulate the desire throttle valve position
- Calculate the voltage value to control the valve
- Calculate and display the PWM value through PID and oscilloscope

However, the system also has some weakness:

- PID controller is not optimize in this project
- Lagging in displaying the value in proteus
- Controlling depend on the desired throttle valve position value but not link with the accelerator pedal

VI.Appendix:

6.1 CODE:

*** Program function:**

Using Timer1 with Fast PWM mode to create working cycle for calculating desired and actual throttle valve position value (set point and actual), calculating PWM width

*** Analog input pins**

-Ain0: GPIO14, 0-1023 -> 0-5V
-Ain1: GPIO15, 0-1023 -> 0-5V

*** Digital output pins**

- LPF: GPIO9, PWM

*** Timer1:**

-Prescaler=64
-Arduino frequency=16 MHz
-Timer/Counter frequency=16MHz/64=250kHz
-Working frequency= 1kHz
-Mode: 14-Fast PWM
+TOP=ICR1=249
+Update of OCR1A at BOTTOM
+TOV1 Flag Set on at TOP

-Application:
+Read ADC value to calculate the voltage value
+ADC value convert from 0-1023 to 0-5V
+Calculate _set_point and _real_vol
+Calculate PWM value

-Timer over flow:

+Set TOIE1(Timer Overflow Interrupt Enable 1): interupt when timer overflow
+Increase _count_ADC value to 10 (10ms) --> Timer1 Overflow

==> Set _f_ON=1 for communicating purpose (Serial)
Reset _count value to 0

CODE:

```
#include <avr/interrupt.h>
#include <PID_v1.h>
#define in 10
#define ena 9
bool _f_ON = 0;
unsigned int _count;
int _voltage_0, _voltage_1;
double pwm, _set_point, _real_vol, _Control_vol;
double kp = 30, ki = 20, kd = 0.01;
PID myPID(&_real_vol, &_Control_vol, &_set_point, kp, ki, kd, DIRECT);

void setup()
{
  Serial.begin(9600);
  pinMode(in, OUTPUT);
  pinMode(ena, OUTPUT);
  myPID.SetMode(AUTOMATIC);
  myPID.SetTunings(kp, ki, kd);

  cli();
  TCCR1A = 0;
  TCCR1B = 0;
  TIMSK1 = 0;

  TCCR1B |= (1 << WGM13) | (1 << WGM12) | (1 << CS11) | (1 << CS10);
  TCCR1A |= (1 << WGM11) | (1 << COM1A1) | (1 << COM1B1);
  TCNT1 = 0;
  ICR1 = ((unsigned short int) (16000000.0/64.0/1000.0)) - 1;
  OCR1A = 0;
```

```
  TIMSK1 = (1 << TOIE1);
  sei();
}


void loop()
{
  if (_f_ON)
  {
    _f_ON  = 0;
    _voltage_0 = map(analogRead(A0), 0, 1023, 0, 4350);
    _voltage_1 = map(analogRead(A1), 0, 1023, 0, 4350);
    _set_point = _voltage_0/1000.0;
    _real_vol = _voltage_1/1000.0;
    myPID.Compute();
        if (_Control_vol < 0.0) pwm = 0.0;
          else if (_Control_vol > 50.0) pwm = 1.0;
            else pwm = _Control_vol/50.0;
    OCR1A = (unsigned short int) (pwm * ICR1);
    Serial.print(_real_vol);
    Serial.print(" ");
    Serial.print(_set_point);
    Serial.print(" ");
    Serial.println(pwm);

}
}


ISR (TIMER1_OVF_vect)
```

```
{
  if (++_count == 10)
    {
    _f_ON = 1;
    _count = 0;
    }
}
```

VII. REFERENCE:

[1] *Electronicstutorials – 2021– Non-inverting Operational Amplifier*

https://www.electronics-tutorials.ws/opamp/opamp_3.html

[2] *VATC – 2021 – Cảm biến vị trí bướm ga – TPS sensor*

https://oto.edu.vn/tim-hieu-ve-cam-bien-vi-tri-buom-ga-tps-sensor/

[3] *Elprocus – 2021 – Throttle Position Sensor – Working Principle and Applications*

https://www.elprocus.com/throttle-position-sensor-working-principle-applications/

[4] *Atmel corporation – 01/2015 – 1600 Technology Drive, San Jose, CA 95110 USA - Atmega328p Datasheet*

https://bitly.com.vn/99w3u9

[5] *Electronicstutorials – 2021 – Passive low pass filter*

https://www.electronics-tutorials.ws/filter/filter_2.html