DEVELOPMENT OF POWER ASSIST CONTROL SYSTEM OF HYDRAULIC HYBRID VEHICLE (HHV)

MUHAMMAD AL FATEH BIN MOHAMMAD ZAKI

UNIVERSITI SAINS MALAYSIA

2021

DEVELOPMENT OF POWER ASSIST CONTROL SYSTEM OF HYDRAULIC HYBRID VEHICLE (HHV)

by

MUHAMMAD AL FATEH BIN MOHAMMAD ZAKI

Thesis submitted in fulfilment of the requirements
for the degree of
Mechanical Engineering

July 2021

ACKNOWLEDGEMENT

I would like to express our gratitude to Allah SWT for giving me opportunity and help us endlessly in finishing this thesis development of power assist control system of hydraulic hybrid vehicle (HHV). Special thank you to my supervisor Dr Iftishah Ramdan for their guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project, I have taken efforts in this project to learn more about hydraulic hybrid vehicle. However, it would not have been possible without the kind support from the team. We would like to extend my sincere thanks to all of them who giving all their best. I are highly indebted to Dr.Muhammad Iftishah Ramdan. Thanks, and appreciations also go to the team in this project and people who have willingly helped with their abilities.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

TPS	Throttle position sensor
HEV	Hybrid electric vehicle
HHV	Hydraulic Hybrid Vehicle
PRF	Pulse Repetitive Frequency
USM	University Science of Malaysia

ABSTRACT

The hydraulic hybrid parallel through-the-road (TTR) design employs a regenerative braking mechanism at the rear wheels. Because it does not interfere with the traditional drive train, its architecture provides a realistic option for an efficient passenger car. As a result, unlike previous hybrid cars, this architecture enables the deployment of hybrid technology on a front-wheel-drive vehicle that already exists. This paper discusses the development of power assist control system of hydraulic hybrid vehicle (HHV) hydraulic circuit design and electric circuit design. The study shows the braking system that generates pressure to operate the vehicle during acceleration. This allows the pump and the motor to operate according to the system build and maximize the use of regenerative braking throughout the drive cycle.

ABSTRAK

Reka bentuk hidraulik selari melalui jalan (TTR) menggunakan mekanisme brek regeneratif pada roda belakang. Kerana tidak mengganggu pemanduan tradisional, seni bina memberikan pilihan yang realistik untuk kereta penumpang yang cekap. Hasilnya, tidak seperti kereta hibrid sebelumnya, seni bina ini memungkinkan penggunaan teknologi hibrid pada kenderaan pacuan roda depan yang sudah ada. Maka ini membincangkan perkembangan sistem kawalan bantuan kuasa reka bentuk litar hidraulik kenderaan hibrid hidraulik (KHH) dan reka bentuk litar elektrik. Kajian menunjukkan sistem brek yang menghasilkan tekanan untuk mengendalikan kenderaan semasa pecutan. Ini membolehkan pam dan motor beroperasi mengikut sistem yang dibina dan memaksimumkan penggunaan brek regeneratif sepanjang kitaran pemacu.

CHAPTER 1

INTRODUCTION

1.0 Development of control system

Due to worries about excessive fossil fuel use and the environmental effect of carbon emissions, the creation of efficient passenger cars has become one of the most attractive study fields. Despite the fact that regenerative braking has been demonstrated to reduce energy consumption and carbon emissions, hybrid drive trains are not widely used in developing countries. This can be seen in the increased use of hybrid electric vehicles (HEV) in developed countries such as the United States since the early 2000s compared to developing countries due to low-income consumers and limited purchasing power.

Because of its simplicity, the hydraulic system is highly durable and can be maintained by any technician, the through-the-road hydraulic hybrid parallel design, which allows for the retrofitting of regenerative braking components into an existing conventional vehicle, could be the answer to the low attainability rate of efficient hybrid cars. Low-income customers don't have to commit to purchasing a brand-new vehicle, and they can maintain it just like any other vehicle.[1]

1.1.1 Introduction

A hybrid vehicle uses a regenerative braking system to recover the kinetic energy during braking, which can be reused later to propel the vehicle. A hybrid hydraulic Vehicle (HHV) offers high power-density components for the high frequency regenerative braking, allowing the vehicle to recover large amounts of energy in a short period of time. During braking, the vehicle momentum drives the hydraulic pump to charge the hydraulic accumulator, where the kinetic energy is stored as potential energy in the form of high-pressure fluid. During acceleration, the stored energy is used by releasing the high-pressure fluid in the accumulator to drive a hydraulic motor. Hybridization is viewed as a viable option for lowering vehicle fuel consumption and emissions. One or more alternative power sources are utilised in hybrid cars to assist the engine in driving the vehicle. In recent years, mechanical hybridization, electric hybridization, and hydraulic hybridization have all been proposed, with electric hybridization being the most common. Electric drive systems, on the other hand, are confined to light-duty vehicles because of their high cost and low power density. Buses, delivery trucks, and garbage collection trucks all work in cities and make numerous stops and starts. Low engine efficiency lowers fuel economy when the vehicle is started[2]. When braking, mechanical braking wastes the vehicle's kinetic energy. If regenerative braking could be installed on these cars, it might save a significant amount of energy. Because of the high torque requirement during launching and braking as a result of the huge vehicle mass, a competent electric drive system must be a disproportionately big size, which raises the production cost.[3]

1.1.2 Acceleration

During the acceleration, the signal from the throttle signal, speed meter and state of charge (SOC) are used to give signal as input to control the valve. Then , the signal specified under specific condition to operate the motor during , which means the car will moves when it has charges between 1% to 100% and only release pressure at below 45 km/h or at very low speed.

1.2 Problem statement

Concerns about excessive fossil fuel usage and the environmental impact of carbon emissions have made the development of efficient passenger automobiles one of the most appealing research topics. Hybrid drive trains are not frequently employed in poor nations, despite the fact that regenerative braking has been shown to cut energy consumption and carbon emissions. Hybrid electric vehicle (HEV) adoption has increased in wealthy countries such as the United States since the early 2000s, compared to developing countries, due to low-income clients and limited purchasing power. For developing countries, finding innovative ways to lessen their reliance on fossil fuels is critical.

Objectives of Development of power assist control system of hydraulic hybrid vehicle (HHV).

1.1.2 Objectives 1

To develop an Arduino rule-based control strategy system for a through-the-road parallel hydraulic hybrid vehicle's regenerative braking system.

1.4.2 Objectives 2

To use rule-based control strategy on Arduino microcontroller vehicle during acceleration, which consist of a few signal from the car and produce it as an input to control solenoid valve.

.

CHAPTER 2

LITERATURE REVIEW

Introduction

Hydraulic hybrid vehicles, often known as HHVs, combine a pressurised fluid power source with a traditional internal combustion engine (ICE) to improve fuel efficiency and reduce hazardous emissions. They capture and reuse 70–80% of the vehicle's braking/decelerating energy compared to 55% for electric hybrids[4]. This can also be less expensive than electric systems for trucks and buses, due to the cost of batteries in the latter. Hydraulic hybrid vehicle systems can also weigh less than electric systems, due to the high weight of the batteries. This may have a lessening effect on cargo capacity, particularly for heavier vehicle classes.

2.2 Hydraulic Hybrid Vehicle

The working fluid, reservoir, pump/motor (in parallel hybrid systems) or in-wheel motors and pumps (in series hybrid systems), and accumulator are the four basic components of hydraulic hybrid vehicle systems. Energy is stored in the battery and supplied to the electric motor to power the vehicle in an electric hybrid system. Regenerative braking uses the vehicle's kinetic energy to charge the battery while it is braking. During braking, the pump/motor harvests kinetic energy to pump the working fluid from the reservoir to the accumulator in a hydraulic hybrid system. As a result, the working fluid is compressed, resulting in energy storage. This pressurised working fluid gives energy to the pump/motor to power the vehicle as it accelerates.

The torque produced by the parallel hybrid system reduces the mechanical stress on the internal combustion engine, resulting in increased fuel economy and lower emissions.

2.2.1 Sustainability factor

Because of worries about unsustainable fossil fuel use and the impact of carbon emissions on the environment, the production of powerful passenger vehicles has become one of the most popular fields of study. Hybrid power trains have been shown to reduce energy consumption and carbon emissions by regenerative braking but are not widely used in the construction process[3]. In poor nations, however, HEV sales have been relatively low. Because low-income consumers' purchasing power is limited, they are unable to purchase. Despite the fact that the architecture allows the regenerative braking modules to be retrofitted to an existing conventional automobile, the parallel hybrid idea might be the answer to the poor success rate of hybrid cars. Because of its simplicity, the hydraulic system is highly stable and can be maintained by any mechanic. As a result, low-income drivers do not have to buy a completely new automobile and can keep their existing vehicle.

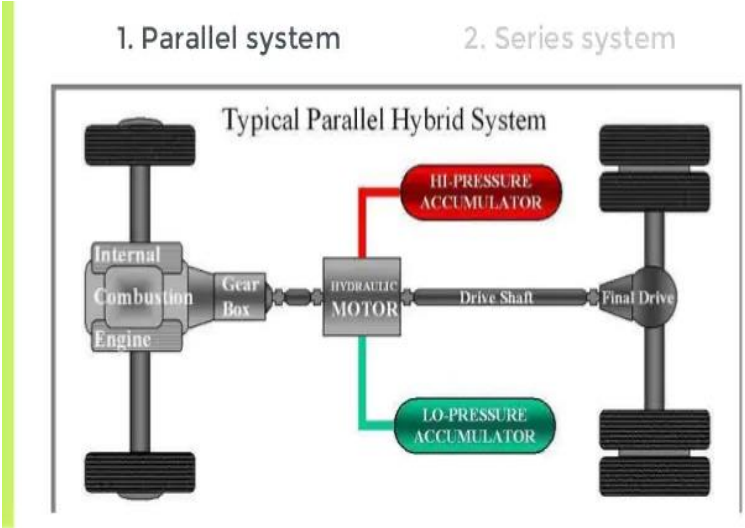

Figure 1: Parallel System

The hydraulic hybrid parallel design might be the key to overcoming an efficient hybrid vehicle's poor attainability rate. Conventional cars can be modified with the parallel hydraulic hybrid through-the-road system. Because of its simplicity, the hydraulic system is extremely robust and dependable. A regenerative braking mechanism is built into this construction design. Regenerative braking is a critical component of hybrid vehicle technology that has attracted a lot of scientific interest. One of the most significant aspects of regenerative braking for the parallel HHV through-the-road vehicle is the creation of a control plan (TTR).[2]

Dynamics programming

The usage of dynamics programming in this project ensured the best possible outcome. Which aids in determining the speed and status of charge. In this development project, dynamic programming was utilised to estimate the condition speed, state of charge, and other variables. There is some information on how to create the system there.[5]

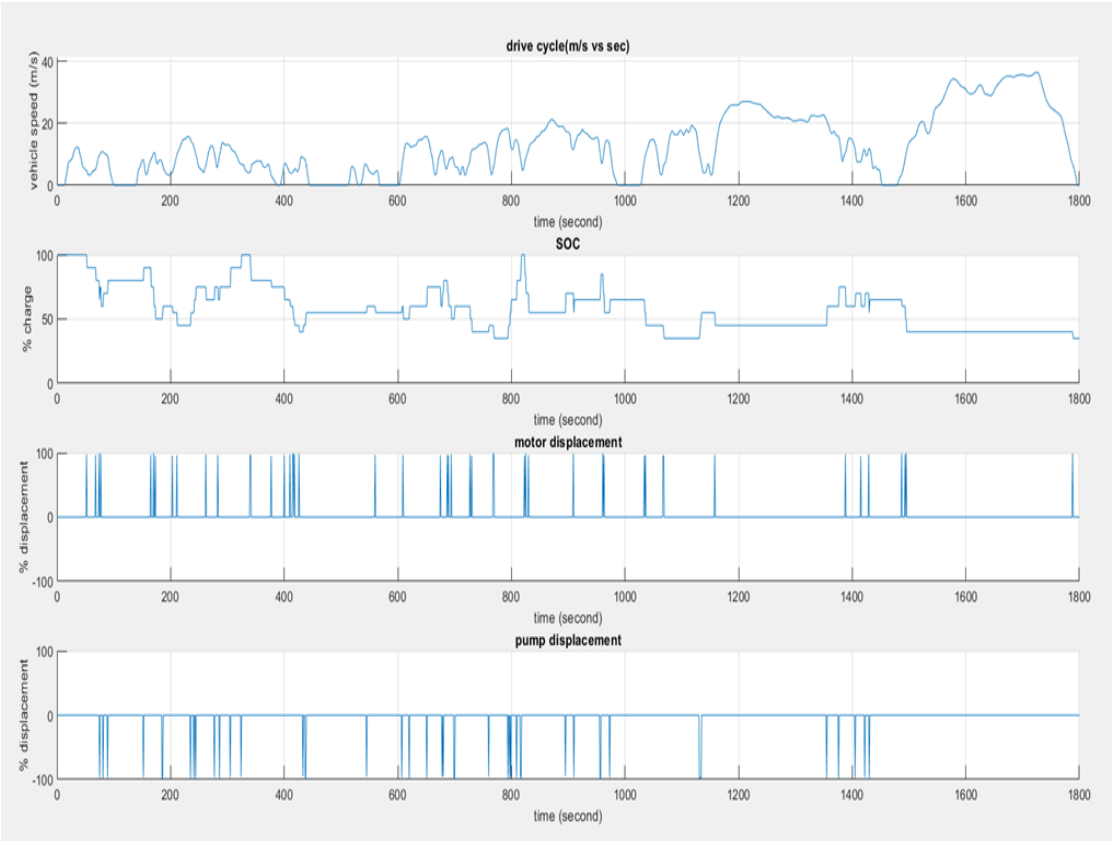

Figure 2: The result from dynamics programming

The vehicle speed, state of charge (SOC), displacement of motor and pump, engine torque and speed, gear index, hybrid on/off duration, vehicle engine fuel consumption map, and engine efficiency map graphs are the results of dynamic programming. The rule-based control approach is built using this optimal solution from dynamic programming.

CHAPTER 3

METHODOLOGY

Introduction

The through-the-road parallel HHV architecture places a regenerative braking system at the rear wheels and does not interfere with the conventional drive train. This allows the retrofitting of the hybrid system into a front-wheel-drive conventional vehicle, like a Perodua Myvi. The hydraulic hybrid system consists of a hydraulic pump, a clutch, an accumulator, a check valve, a pressure relief valve, a 3/2-way valve and a hydraulic motor .

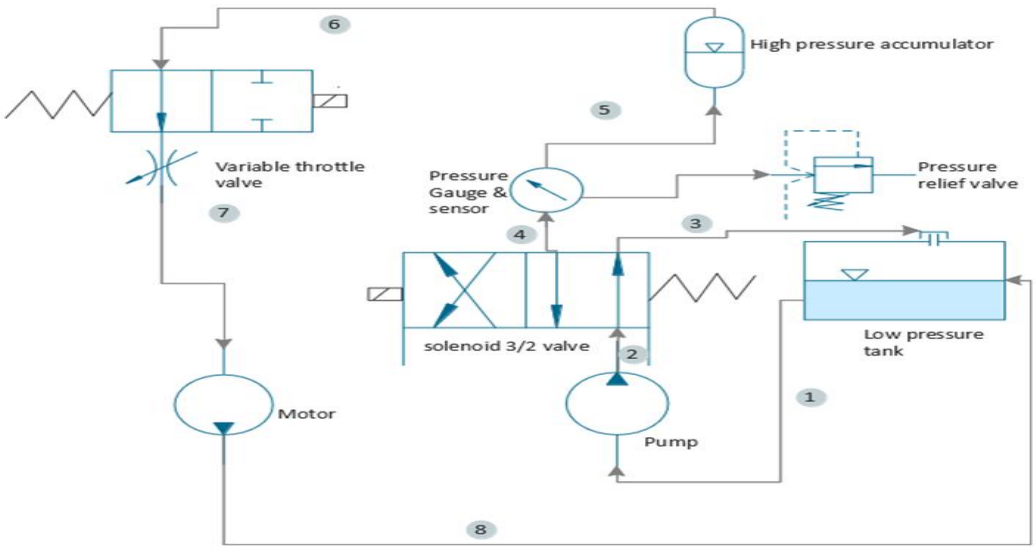

Figure 3.1: Circuit diagram

Table 3.1 List of component and their function.

LIST OF COMPONENTS	FUNCTION
High-pressure accumulator	To store the pressurized hydraulic fluid from the pump and supply high-pressure fluid to the motor
Low-pressure tank	Store the hydraulic fluid and supply it to the system through the pump.
Hydraulic Motor	To accelerate the wheel using the high-pressure fluid from the accumulator
Hydraulic Pump	To pump the fluid from the low-pressure tank when the brake is pressed.
3/2 solenoid valve	To control the flow of the fluid from the pump, to send it forward to the pressure sensor, or to send it back to the low-pressure tank
Pressure sensor and pressure gauge	To observe and monitor the pressure going into the high-pressure accumulator
Solenoid check valve	To control the hydraulic fluid going through the pressure gauge

The Perodua Myvi, a through-the-road HHV prototype automobile, was subsequently equipped with this technology. The hydraulic system now includes an Arduino microcontroller that regulates the solenoids that open and close the motor and pump valves to control the flow of hydraulic fluid through the motor and pump

3.1 During operation

During braking, the right wheel clutch is engaged to the hydraulic pump that charges the accumulator. A pressure relief valve is used to ensure the accumulator pressure does not exceed the maximum operating pressure of any of the components . A check valve is used to ensure the high-pressure fluid does not flow back to the pump.

During acceleration, the 3/2-way valve is valve is moved, to release the high-pressure fluid from the accumulator and to drive the hydraulic motor at the left wheel. Simultaneously, the proportional valve is used, to control the motor's speed and torque.

3.1.1 Throttle signal

During throttle signal gives information in form of voltage signal. From the voltage signal it can be used as an input to Arduino to tell the Arduino when the throttle is pressed.

3.2.2 Speedo meter

When the car moves, it generates speed meter digitally. Therefore, it can be used to determine the speed of the car. Arduino can use the signal as an input to determine whether the valve is open or close. Arduino can translate the signal, if the car accelerates below 45 km /h it can open the valve to release high pressure to operate the hydraulic system.

3.2.3 Pressure

Pressure is used to determine state of charge (SOC) which it can determine how much charge available in the tank which always at atmospheric pressure. The pressure is measured based on the charge remaining in the tank which means when the fluid enters the accumulator. From the current pressure, it can determine SOC as stated in the formula below.

$$\mathrm{SOC} = \frac{\mathrm{PHPA(t)} - \mathrm{Pmin}x}{\mathrm{Pmax} - \mathrm{Pmin}}$$

From the charging state, Arduino can know how much the State of charge state left in the tank. From the charging state, Arduino can only release the pressure under condition, if there is 1% to 100% state of charge (SOC) or the valve will be closed.

3.2.4. Arduino UNO

Arduino is used to program the signal from throttle position, pressure sensor and speedo meter. The program is used for opening and closing the valve, which the valve will release the pressure from the tank. The code will work is determine by the signal it received.

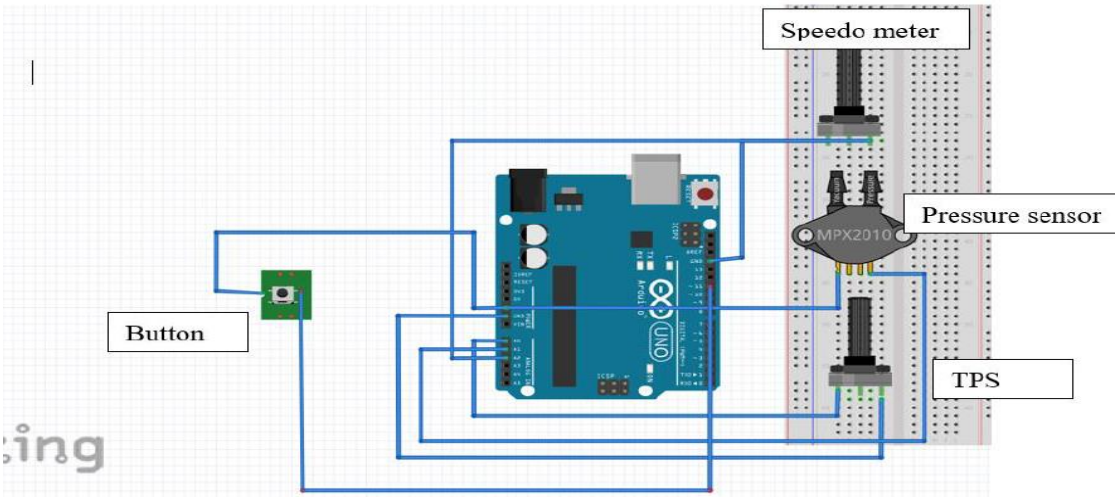

Figure 3.2 : Adruino circuit diagram

From the Figure 3.2 , the Arduino will receive the signal from speedo meter to determine the vehicle at the current time , The pressure sensor to the determine the state of charge and Throttle Position Sensor to determine the voltage inserted.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Throttle position sensor (TPS)

The signal from the throttle position sensor is in analog form which need to be convert into digital before Arduino can able to read the signal.

$$\frac{Resolution\ of\ the\ ADC}{System\ voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

Which in this project case, because the valtage supply to throttle position sensor is 5V. Therefore, the formula is as below.

$$\frac{1023}{5} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

4.1.1 Speedo meter

The speedometer signal was converted to frequency in this project. It can calculate the vehicle speed at that time by utilising pulse repetition frequency (PRF). The number of pulses of a repeating signal in a given time unit, usually measured in pulses per second, is known as the pulse repetition frequency (PRF). The word is used in a variety of technological fields, including radar. Both are measured in terms of cycle per millisecond.

$$Frequency(Hz) = 1000000 + Pulse\ High$$

To convert the frequency is as below,

$$Vehicle\ Speed\ (\frac{Km}{h}) = frequency$$

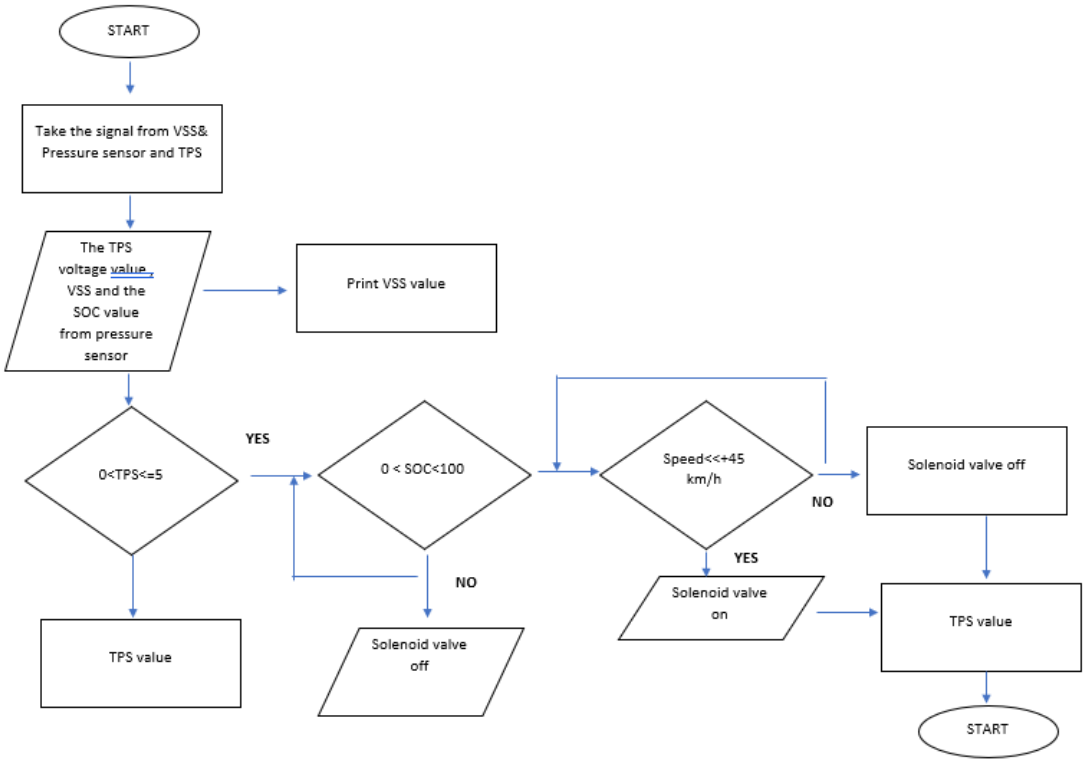

Figure 4.1: Overall flowchart for acceleration

4.1.2 Pressure Sensor

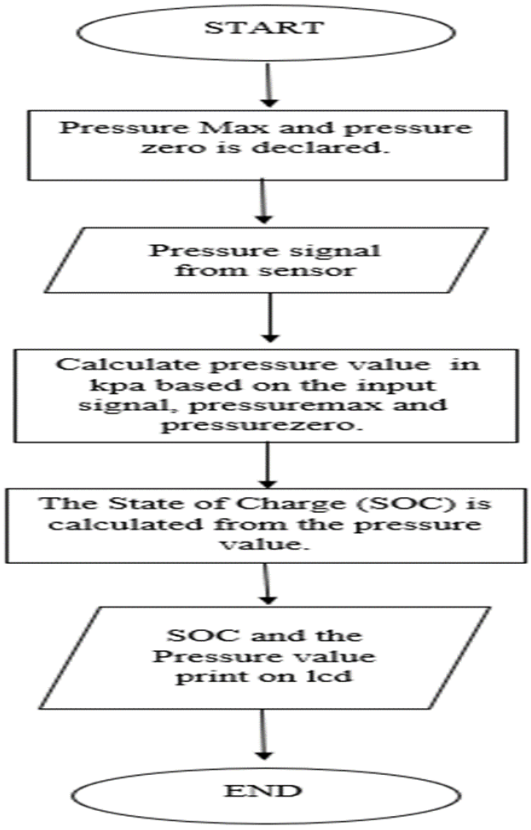

Figure 4.2: Flowchart for pressure sensor

To show the pressure of the hydraulic system, the pressure sensor is linked to the microcontroller, which is connected to the LCD screen. The EDYBT-01 Oil Pressure Transducer was utilised in this project, and it can measure up to 100bar with a 0V–5V output signal voltage.

The code begins by stating the highest and lowest pressures; using this information, the Arduino will compute the final pressure value in kilopascals using the input signal from the pressure sensor (kPa).

$$\text{pressureValue} = ((\text{pressureValue} - \text{pressureZero}) * (\frac{\text{pressuretransducermaxBAR}}{\text{pressureMax} - \text{pressureZero}})$$

$$Soc = \left(\left(\frac{pressurekpa - pressurekpamin}{pressurekpamax - pressurekpamin} \right) * 100 \right)$$

Discussion

The flowchart begins with information from the Throttle Position Sensor (TPS), Vehicle Speed Sensor (VSS), and Pressure Sensor (see Figure 4.1). The voltage from the TPS must then be determined by Arduino. It should be between 0 and 5 volts. If the solenoid valve is to be turned on, the SOC value must be between 0% and 100%. To switch on the solenoid once the TPS and SOC values have been reached, the vehicle speed must be less than 45 km/h.

Because the value from the sensor has some resistance, the code in Appendix A has been modified to reflect the current condition. As a result, it has been calibrated. Because it was modified to the real circumstances, the formula in this project differs somewhat from the theory. Aside from that, the sensor's connection may contain resistance, resulting in a different value. As a result, depending on the resistance, certain adjustments to the code are required. Because the sensor input cannot be as exact as it is in the real world, the output of the dynamics programming may deviate somewhat from the actual scenario.

The vehicle speed and state of charge have somewhat different values when compared to the dynamic programming result in Figure 2.1. It's because the value of the real result is subject to a number of resistance, resulting in a somewhat different value.

CHAPTER 5

CONCLUSION AND FUTURE RECOMMENDATIONS

Conclusion

For the through-the-road prototype car, the Arduino system was built using a rule-based method. The regenerative braking system will engage if three criteria are satisfied, according on the findings of the DP optimization. System which brings another alternative to reduce energy consumption available car in the industry. In other word, alternative that reduce fuel consumption and reduce energy pollution.

Recommendations for Future Research

On this project, the development of system needs to be placed in the right place or position. In order the signal from different source can be read and Arduino can produce the output efficiently. The signal and the sensor can be very sensitive. Therefore, if it has a good place to perform the program, it will be more efficient.

REFERENCES

[1] S. Zhou, P. Walker, and N. Zhang, "Parametric design and regenerative braking control of a parallel hydraulic hybrid vehicle," *Mech. Mach. Theory*, vol. 146, p. 103714, 2020, doi: 10.1016/j.mechmachtheory.2019.103714.

[2] S. Zhou, P. Walker, J. Wu, and N. Zhang, "Power on gear shift control strategy design for a parallel hydraulic hybrid vehicle," *Mech. Syst. Signal Process.*, vol. 159, p. 107798, 2021, doi: 10.1016/j.ymssp.2021.107798.

[3] Q. Chen, T. Lin, H. Ren, and S. Fu, "Research on the control strategy of power train systems for hybrid hydraulic excavators," *Adv. Mech. Eng.*, vol. 10, no. 7, pp. 1–10, 2018, doi: 10.1177/1687814018790666.

[4] S. Podrug, "Hydraulic hybrid vehicle configurations and comparison with hybrid electric vehicle," *Contemp. Issues Econ. Technol.*, 2019.

[5] T. J. Böhme and B. Frank, "Dynamic programming," *Adv. Ind. Control*, no. 9783319513157, pp. 199–214, 2017, doi: 10.1007/978-3-319-51317-1_6.

APPENDIX A

The Development Of Control System For Acceleration And Braking System Code

```
include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4); //sets the LCD I2C communication address; format(address, columns, rows)

#define cellpin A0

const int pulsePin = A2;              // Input signal for vehicle speed sensor(vss)
int pulseHigh;                        // Integer variable to capture High time of the incoming pulse from the vss
int pulseLow;                         // Integer variable to capture Low time of the incoming pulse from the vss
unsigned long pulseTotal;             // Float variable to capture Total time of the incoming pulse from the vss
unsigned long frequency;              // to calculated Frequency based from the pulse
float Vehiclespeed;                   // vehicle speed obtained from frequency

const int pressureInput = A1;         //select the analog input pin for the pressure sensor
const int pressureZero = 102.4;       //analog reading of pressure transducer at 0psi
const int pressureMax = 921.6;        //analog reading of pressure transducer at 100psi
const int pressuretransducermaxPSI = 100;     //psi value of transducer being used
const int baudRate = 9600;            //constant integer to set the baud rate for serial monitor
const int sensorreadDelay = 250;      //constant integer to set the sensor read delay in milliseconds

int analogPin = A3;                   // brake light anolog in
int bRake = 0;                        // variable to store the value read of break light

const int Pumpvalve = 10;             // button 1 to send the charge from pump to high pressure tank connected to pin10
const int Pumpvalve2 = 11;            // button 2 for the charge to store in the high pressure tank connected to pin 11
const int switchbrake = 13;           // alternate switch to activate brake
int statSoc;                          // status of SOC

float pressureValue = 0;              //variable to store the value coming from the pressure transducer
float pressurekpa = 0;                // fianl pressure in kilopascal
float pressurelimit = 0;              // max pressure that limited in kpa
float pressurekpamin = 0;             //minimum pressure converteed to kpa
float pressurekpamax = 0;             //maximum pressure converteed to kpa
float Valve;                          //variable for sensore
```

```
float Soc;                          //state of charge in the high pressure tank

const float mvc = 4.55;

float counts=0;
float mv=0;

int soc;


void setup() {
  Serial.begin(9600);
  pinMode(pulsePin, INPUT);                 //pulse in input

  pinMode(Pumpvalve, OUTPUT);               // engaging button 1 is output
 pinMode(Pumpvalve2, OUTPUT);               // engaging button 2 is output
  pinMode(switchbrake, INPUT);              // engaging switch as input
  pinMode(pressureInput,INPUT);             // pressure sensor input


  lcd.begin();                              //initializes the LCD screen


  delay(2000);                              // delay 2s for the setup to initiate
}

void loop() {
  pulseHigh = pulseIn(pulsePin, HIGH);

  pulseLow = pulseIn(pulsePin, LOW);

  pulseTotal = pulseHigh + pulseLow;                    // Time period of the pulse in
microseconds

  frequency = 1000000 / pulseHigh;                      // Frequency in Hertz (Hz)
Vehiclespeed = frequency*0.75;                          //converting the frequency to km/h
Serial.print("speed: ");
Serial.println(Vehiclespeed);                           // check vehicle speed in km/h

 pressureValue = analogRead(pressureInput);                                     //reads
value from input pin and assigns to variable
  pressureValue                         =                       ((pressureValue-
pressureZero)*pressuretransducermaxPSI)/(pressureMax-pressureZero);
//conversion equation to convert analog reading to psi
  pressurekpa = (pressureValue*6.89476);                                        //
convert psi to kpa
  Serial.print(pressurekpa, 1);                                                 //prints
value from previous line to serial
  Serial.println("kpa");                                                       //prints label
to serial
```

```
  lcd.setCursor(0,0);                                                    //sets cursor to
column 0, row 0
  lcd.print("Pressure:");                                                 //prints label
  lcd.print(pressurekpa, 1);                                                    //prints
pressure value to lcd screen, 1 digit on float
  lcd.print("kpa");                                                      //prints label
after value
  lcd.print("    ");                                                    //to clear the
display after large values or negatives

  delay(sensorreadDelay); //delay in milliseconds between read values

  pressurekpamax                    =                    (pressureMax*6.89476);
//converting the pmax to kpa from psi
  pressurekpamin                    =                    (pressureZero*6.89476);
//converting the pmin to kpa from psi
  bRake = analogRead(analogPin);                                                 // read
the input pin
  Soc = (((pressurekpa - pressurekpamin)/(pressurekpamax - pressurekpamin))*100);
// calculation of the SOC
 Serial.print ("Break light voltage:");                                         //break
light voltage
 Serial.println (bRake);
  lcd.setCursor(0,1);
  lcd.print("SOC: ");                                                    //monitor the
SOC value
  lcd.print(Soc);
  lcd.print("%");

   counts=analogRead(cellpin);                                             //reads the
voltage signal from TPS

  mv=(counts * mvc)/1000;


  Serial.println(String(mv)+"V");                                          //monitor
the tps voltage



 if(bRake >1000)                          //bReak light on
   { if(100>Soc>0)                          // status of charge range 0 to 100%
      {
       digitalRead(Vehiclespeed);
           if(Vehiclespeed <= 72)                    //when the vehicle speed is below
72km/h
            {
             Serial.print("SOC velue: ");              //print the Soc value
             Serial.println(Soc);
```

```
            digitalWrite(Pumpvalve,HIGH);                //the fluid flow to high pressure
for charging
            digitalWrite(Pumpvalve2,HIGH);                // the energy is stored in the
accumulator
            }
           else if(Vehiclespeed > 72)                  // the cutoff for the hybrid to work is
72km/h
            {
            Serial.print("SOC velue: ");                //print the Soc value
             Serial.println(Soc);
            digitalWrite(Pumpvalve,LOW);                // pump valve is not activated
and flow back to the low pressure tank
            }
       }
    else if(Soc >= 100)                              //when the soc is high at 100% the break
would not work
     {
      digitalRead(Vehiclespeed);
            if(Vehiclespeed <= 72)                    //the speed below 70km/h
            {
              Serial.print("SOC velue: ");             //print the Soc value
              Serial.println(Soc);
               digitalWrite(Pumpvalve,LOW);           // pump valve is not activated
and flow back to the low pressure tank
            }
            else if(Vehiclespeed > 72)
            {
             Serial.print("SOC velue: ");             //print the Soc value
              Serial.println(Soc);
             digitalWrite(Pumpvalve,LOW);             // pump valve is not activated
and flow back to the low pressure tank

            }
    }
   }

 else if(bRake <1000)                          //no break is pressed the the system does not
work
    {if(Soc<100)
            {
             digitalRead(Vehiclespeed);
             Serial.print("SOC velue: ");                //print the Soc value
             Serial.println(Soc);
             digitalWrite(Pumpvalve,LOW);                    // pump valve is not
activated and flow back to the low pressure tank
             Serial.println("Break pedal is not pressed!");          //to indicate the pedal
is not pressed
            }

    else if(Soc>=100)
```

```
            {digitalRead(Vehiclespeed);
             Serial.print("SOC velue: ");                    //print the Soc value
             Serial.println(Soc);
             digitalWrite(Pumpvalve,LOW);                    // pump valve is not
activated and flow back to the low pressure tank
             Serial.println("Break pedal is not pressed!");
             delay(200);                                  // delay 0.2s for the setup to initiate
            }
    }
 else if(switchbrake == HIGH)                    //the alternate switch is pressed
  {
    if(100>Soc>0){                              // status of charge range 0 to 100%
          if(Vehiclespeed <= 72)                    //when the vehicle speed is below
72km/h
              { Serial.print("SOC velue: ");              //print the Soc value
                Serial.println(Soc);
                digitalWrite(Pumpvalve,HIGH);              //the fluid flow to high
pressure for charging
                digitalWrite(Pumpvalve2,HIGH);          // the energy is stored in the
accumulator
              }
           }
          else {
               digitalWrite(Pumpvalve,LOW);              // other condition the flow
from the pump would not go to the accumulator
             }
  }


if ( mv  > 0 )                          //if throttle is pressed
    {if (100 <= Soc < 0 )                       //if there is charge in to be applied
       {
        digitalRead(Vehiclespeed);                  //read the vehicle speed if it below
45km/h pressure will be applied

        if(Vehiclespeed<=45){
            analogWrite(Pumpvalve2,HIGH);}                  //the button to release
the charge to motor is turned on

        else if(Vehiclespeed <45 )                  //if it below 45km/h no pressure will
be applied
             {
             Serial.print(Soc);
             analogWrite(Pumpvalve2,LOW);                  //the button to release the
charge to motor is turned off
             }
       }
    else if (Soc == 0 )
       {
```

```
          digitalRead(Vehiclespeed);                    //if there is no charge then it will
turn off

          if (Vehiclespeed < 45 )
             {
               Serial.println(Soc);
               analogWrite(Pumpvalve2,HIGH);                  //the button to release the
charge to motor is turned off
             }
         }
      }
 if (mv ==0)                                 //if throttle not pressed then valve will be close
    {
    digitalRead(Vehiclespeed);
    Serial.print("SOC velue: ");
    Serial.println(Soc);
    digitalWrite(Pumpvalve2,HIGH);                   //the button to release the charge
to motor is turned off
    Serial.println("Throttle not pressed");
    }

}
```