DC Motor PI Speed Control System

By:
Martti Muzyka (0697280),
Obatosin Obat-Olowu (0680475),
Devin Hutchinson (0645439)
&
Alexander Bukowicki (0883774)

Circuit Theory & Design II Project (EELE-3316-WA)
Department of Electrical Engineering
Lakehead University
Thunder Bay, Ontario, Canada

Abstract

The purpose of this paper is to outline the design and implementation of a DC motor speed control system. This design requires a 48V DC motor to run at 40 rpm, and maintain this speed up to the maximum applied load of the motors magnetic brake. Since the DC motor used for this project is not intended to be ran at 40 rpm, the standard procedure of exciting the terminals of the DC motors armature with a DC voltage would not be valid for this design. In order to operate the DC motor at this low of a speed, an h-bridge circuit was required. The h-bridge circuit works in conjunction with an ESP8266 microcontroller, which outputs both inverted and non-inverted PWM signals to the inputs of the h-bridge. These PWM signals saturate and cut-off certain transistors which constantly alternates the direction of the current through the motor, thereby changing the direction of rotation. The duty cycle (DT) of these output PWM signals can be changed to favour one direction of rotation, and create a net motion in that direction. A duty cycle of 76.6% was found to have the motor rotate at the desired speed of 40 rpm, under no load conditions. In order to keep the motor running at 40 rpm while the load is increased, a control system is required. An ST188 infrared sensor was placed on the motor, which provides an analog signal whose frequency can be used to determine the speed of the DC motor. The program uploaded onto the ESP8266 constantly monitors this speed, and if the speed deviates from 40 rpm, the ESP8266 microcontroller incorporates a PI control algorithm to change the duty cycle of the output PWM signals, and thereby the speed of the motor. Through the implementation of the aforementioned code and circuitry, the PI control system was able to effectively maintain the speed of the DC motor at 40 rpm for all loads applied by the brake.

Table of Contents

Title Page 1

Abstract 2

Table of Contents 3

Introduction (1) 5
- Objective (1.1) 5
- Background Knowledge and Theory (1.2) 5

Design (2) 8
- Design and Operation (2.1) 8
- LTspice Simulations (2.2) 14
- Potential Improvements (2.3) 15
- Post-Demonstration Revisions (2.4) 16

Testing (3) 17
- Initial Testing (3.1) 18
- Failure Analysis (3.2) 21
- Redesigns (3.3) 23
- Experimental Testing (3.4) 25
- Results and Calculations (3.5) 27

Conclusion (4) 29

Appendix A (5) - Parts List 30

Appendix B (6) - PI Control Program 31

Appendix C (7) - LTspice Schematics 34

Appendix D (8) - LTspice Simulation Results 35

Appendix E (9) - ESP8266 Circuit Diagram 37

Appendix F (10) - Experimental Results 37

Appendix G (11) - References 40

Introduction (1)

- Objective (1.1)
- Background Knowledge and Theory (1.2)

Objective (1.1)

The main objective of this project is to design a DC motor control system that can maintain a 48V DC motor running at 40 rpm when varying degrees of load are applied. The control system must incorporate a 24V power supply, as well as feedback and speed sensing controls.

Background Knowledge and Theory (1.2)

Proportional integral (PI) control is a control algorithm that utilizes both proportional and integral control techniques. Before introducing this concept, however; it is important that the general concepts that pertain to all control systems are fully understood. In most control systems there is a value known as the set point (x(t)), this is the desired value of the input variable that the system will try to maintain. With respect to this project, the set point is the DC motor speed of 40 revolutions per minute (rpm). Furthermore, control systems that utilize PI control, are always closed-loop systems. A closed-loop system is a type of control system that incorporates feedback loops to help reduce errors and improve system stability. Feedback loops are often used to compare the output of the system to the set point, and by doing so generate an error signal (e(t)). This error signal is essentially the difference between the output signal (y(t)) and the set point, it can either be negative or positive. Once an error signal is calculated, a control algorithm (i.e. P, PI, PID, etc.) can use it to obtain a correction factor that will then be applied to the system control variable (u(t)). Regarding this project, the output signal is the measured speed of the DC motor in rpm, and the control variable is the duty cycle (DT) of the PWM signal that's supplying the DC motor. Additionally, a PI control algorithm will be employed to maintain the DC motor running at 40 rpm. Proportional (P) control makes adjustments to the system based proportionally to the error signal. In a proportional controller, the error signal is multiplied by a proportional gain (Kp), to provide a correction factor. This correction factor is then either added or subtracted with the control variable, and the final result is then fed to the system plant. Proportional control is useful in systems where the output must be maintained at a steady value and where fast responses are necessary. There are drawbacks to systems that solely employ proportional control though, as they may experience a phenomenon known as steady state error. Steady state error is the control systems error during steady state operation ($t = \infty$), and it can occur due to system imperfections, aging and deterioration. To eliminate steady state error, integral control can be added to the system. Integral control retains the history of the error signal over time, it knows how long an error signal has been at a certain value, and compensates for that by calculating the cumulative error. This cumulative error is essentially a constant summation of

the current error signal with respect to time, which is another way to conceptualize an integral. By obtaining this constant summation of the current error and multiplying it by a predetermined integral gain (Ki), the correction factor can be obtained. Then by applying this correction factor to the control variable, either through addition or subtraction, the system should progress towards its set point. By combining both integral control and proportional control, the PI control algorithm is obtained. PI control systems retain the responsiveness of proportional-only controllers, while additionally eliminating steady state error. Refer to figure (1) below for a block diagram representation of a PI controller.

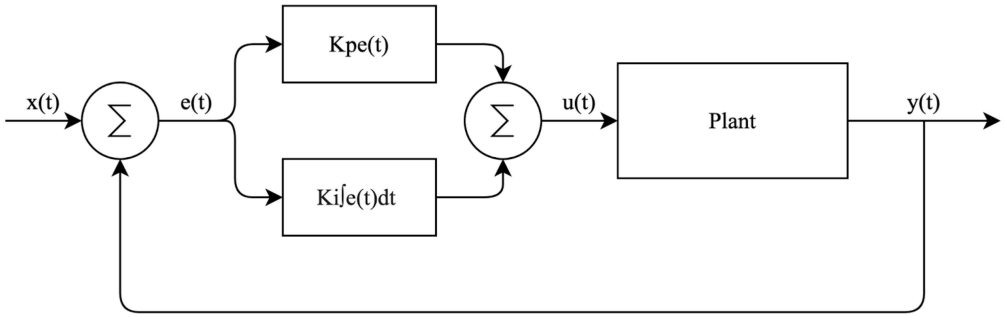


Figure (1) - PI controller block diagram.

The speed of a DC motor is proportional to the voltage applied to the terminals of its armature, the higher the applied voltage, the faster the motor will spin, and vice versa. For DC motor speed control, this concept can be advanced by utilizing a PWM (pulse width modulation) voltage signal as opposed to DC. PWM refers to an analog voltage signal that alternates between a low and high voltage at a certain frequency. By varying the duty cycle (DT) of a PWM signal, the average voltage received by the connected components can be adjusted. For instance, a 5V PWM signal with a DT of 50% will provide an average voltage of 2.5V. This concept can be used in DC motor speed control by supplying one DC motor armature terminal with a non-inverted PWM signal, and the other by the same PWM signal, but inverted. If the PWM signal has a DT of 60%, then the inverted signal will possess a DT of 40%. This means that one terminal will receive a higher average voltage than the other, causing the DC motor to ultimately rotate in a single direction. This technique also incorporates a “jitter” effect through the constant switching. This is important at low DC motor operating speeds as it minimizes the effects of friction on the shaft of the DC motor.

An h-bridge is a circuit that primarily consists of four switching components (i.e. BJT transistors, MOSFETs, etc.), connected as pairs with a centralized load. For instance, an h-bridge constructed from BJT transistors would consist of two sides, each with a pnp and an npn transistor connected together via their collectors to a load (i.e. a DC motor). Furthermore, the emitters of the npn transistors would be grounded, and the emitters of the pnp transistors would be connected to a high potential voltage source. The bases of the transistors on one side of the

h-bridge would be supplied by a non-inverted PWM signal, whereas the bases of the transistor on the opposite side would be supplied by the same PWM signal, but inverted. If the non-inverted PWM signal is high (i.e. 3.3V), and the circuit is effectively biased, the npn transistor supplied by the non-inverted PWM signal will saturate, and its pnp counterpart will enter cut-off. On the opposite side, because the PWM signal is low (i.e. 0V), the pnp transistor will saturate, and the npn transistor will enter cut-off. The two saturated transistors will then create a pathway for current to flow from the DC voltage power supply, through the load, and then to ground. The opposite will occur when the PWM signals change state, and the polarity across the load will switch. If this load is a DC motor, the constant polarity switching will cause it to rotate in different directions. However, if the frequencies of the supplied PWM signals are high enough, and the duty cycle (DT) of either the non-inverted or inverted signal is above 50%, then the DC motor will maintain a single direction of rotation. By varying the DT of the supplied PWM signals, the average voltage seen by the DC motor from one side of the h-bridge will either increase or decrease, and likewise, so will its speed. This relationship reveals how adjustments made to the DT of the PWM signals supplying the h-bridge can be used to control the speed of the DC motor.

The DC motor chosen for this project possesses a disk attached at the end of the shaft with 90 black tick marks along its perimeter, as well as a brake that will be used as the load for testing purposes. Moreover, there is an ST188 infrared optical sensor attached closely to the disk that will be used to measure the speed of the DC motor during operation. The ST188 consists of an infrared LED and a photo-transistor housed within a plastic case. When the DC motor is running and the disk with the tick marks is spinning, the infrared light will only be reflected when it strikes the metal of the disk, whereas it will be absorbed by the black tick marks. Therefore, the photo-transistor will only activate when the light is reflected, and will then shut-off once the light strikes the black tick marks. This constant switching of the photo-transistor creates an analog signal that resembles a sine wave at low speeds. The frequency of this analog signal output from the emitter of the photo-transistor can be used to determine the speed of the DC motor in revolutions per minute (rpm). To elaborate, the frequency of this analog signal is essentially the number of tick marks (including both black and metallic) that will pass in front of the ST188 sensor per second. By multiplying this frequency by 60 and dividing it by 90, it will be converted to rpm. Refer to the derivations below for reasoning behind this declaration. Furthermore, equation (1) below depicts the formula used to determine the speed of the motor in rpm.

$$\text{Derivation}: f = \frac{cycles}{seconds},\ rpm = \frac{cycles}{seconds} \times \frac{60\ seconds}{minute} \times \frac{1\ revolution}{90\ cycles} = \frac{revolutions}{minute}$$

$$\text{Equation (1)}: N(rpm) = f \times \frac{60}{90}$$

Design (2)

- Design and Operation (2.1)
- LTspice Simulations (2.2)
- Potential Improvements (2.3)
- Post-Demonstration Revisions (2.4)

Design and Operation (2.1)

The design and operation details for each of the different core components of the DC motor PI controller can be found below:

1. PI Control Program:
    - The program designed to perform the PI control that regulates the speed of the DC motor, was composed using the Arduino IDE. There are three main features of this program that operate simultaneously to provide the PI control that maintains the motor rotating at 40 revolutions per minute (rpm), regardless of the applied load. These features include, a PI control algorithm, a speed sensor, and a PWM output with an adjustable duty cycle (DT). The designed PI control algorithm is implemented within the “void loop” function, and is triggered every time the speed of the motor is measured. If the speed of the motor is measured to be either above or below 40 rpm, the program enters an if statement that initially determines the current time, the time at which the previous iteration occured, the current error, and the cumulative error. The current time and previous time are used to calculate the time elapsed since the previous iteration. Furthermore, the current error is determined by subtracting the measured speed (y(t)), by the set point speed of 40 rpm (x(t)), and the cumulative error is determined by computing the Riemann sum of the current error. The Riemann sum of the current error as time tends to infinite provides an accurate approximation for the integral of the current error as time tends to infinite. The cumulative error and current error are required to compute the correction factor that is applied to the duty cycle of the output PWM signals supplying the motor. If the measured speed of the motor was greater than 40 rpm, then the correction factor will be added onto the current PWM duty cycle, whereas if the measured speed is below 40 rpm, the correction factor will be subtracted. In theory the duty cycle of the output PWM wave should be reduced to slow down the motor, and likewise, increased to speed it up, however; due to the DC motors

magnetized field winding, the opposite is true for this project. Nevertheless, this PI control algorithm will continue to repeat and make adjustments to the duty cycle every time the speed of the motor is measured. Equation (2) below shows the formula used to determine the current error of the system, and equation (3) shows the formula used to compute the correction factor (C) that would either be subtracted or added onto the current duty cycles of the PWM signals.

$$\text{Equation (2)} : e(t) = 40 - N(rpm)$$

$$\text{Equation (3)} : C = Kp \times e(t) + Ki \times \sum_{i=1}^{\infty} (e(t) \cdot \Delta t)$$

- To measure the speed of the motor in revolutions per minute (rpm), the emitter of the ST188 infrared optical sensor attached to the DC motor is connected to the ESP8266s analog pin (A0). The input value read on this analog pin is stored in a variable declared as “temp”. This variable is then divided by 1024 to normalize it as a value between 0 and 1 and it is then digitally output from the pin D2 to the pin D4 via a connected wire. The pin D4 is initialized as an input, and an interrupt service routine (ISR) is setup as to be triggered when its state changes (i.e. low-high, or high-low). Therefore, when the value of “temp” changes, indicating that half a period has passed on the incoming analog signal, the interrupt is triggered. This interrupt triggers a function that increments a variable by 1 every time it is called upon. This variable is used to count the number of pulse changes. Within the void loop portion of the program, there is an if statement that compares the value of this variable to 2, and if satisfied, it will determine the current time, and the time at which the previous iteration occurred. The ISR is disabled when this if statement is satisfied, as to prevent an interrupt from occurring while performing speed and PI control calculations. It is re-enabled at the end of the void loop function. The variable that is incremented within the ISR, is also set to 0 at the end of the void loop function. Furthermore, by utilizing equation (1) outlined in section 1.2 (background and theory) of this report the speed of the DC motor in rpm is calculated.
- The output inverted and non-inverted PWM signals used to supply the h-bridge connected to the armature of the DC motor are controlled via the PI control program. The ESP8266 microcontroller possesses the ability to adjust the PWM range and PWM frequency of a PWM signal output using the “analogwrite” command. The frequency of this output PWM signal is

set to 500Hz, whereas its range is set to 1024. The range corresponds to the duty cycle of the PWM signal, if the duty cycle is set to 1024, this refers to a duty cycle of 100% (DC). Nevertheless, the “analogwrite” command is used to output a PWM signal with a set frequency and adjustable duty cycle from the pin D5 which is connected to the pin D3 via a wire. A variable declared as “hold” is implemented within the void loop function to store the digital value read at pin D3. The pin D0 is then used to digitally output the value of “hold” and the pin D1 is used to output the inverse of the “hold” value. Therefore, the non-inverted PWM signal is output from the pin D0, whereas the inverted signal is output from the pin D1. This setup reduces the phase shift that occurs if both pins were initialized to output the PWM signals separately, due to their different on-board timers. Refer to Appendix B (2) for the PI control program code with comments explaining the individual lines.

2. H-bridge Circuit:
    - For this project an h-bridge was designed to alternate the polarity of the 24V DC supply powering the armature of the DC motor. For an outline of the operation and function of a h-bridge circuit refer to section 1.2 (background knowledge and theory) of this report. Regarding this project, the h-bridge consists of two TIP32C pnp power transistors each with their emitters connected to a 24V DC power supply. Furthermore, the collectors of both TIP32C transistors are each connected to the collectors of two TIP31C npn power transistors and the emitters of both these TIP31C transistors are grounded. There are also four 1N4148 diodes located within the h-bridge circuit, two of the 1N4148 diodes are each connected in parallel with the two TIP31C transistors, their cathodes are each connected to the collectors of the TIP31C transistors, and their anodes are grounded. Similarly, the remaining two diodes are each connected in parallel with the TIP32C transistors, their cathodes are connected to the 24V DC power supply, and their anodes are each connected to the collectors of the TIP32C transistors. The DC motor itself is connected between the joined collector terminals on either side of the h-bridge via its armature. The bases of each of the transistors are connected in series with four 1.8kΩ resistors. An h-bridge can be conceptualized as having two sides, each side possessing an npn and a pnp transistor pair connected together via their collectors. On each side of this h-bridge there are two parallel 1.8kΩ resistors each connected to the bases of the TIP32C and TIP31C transistors. These two 1.8kΩ resistors connect the bases of the transistors on one side to the collector of the corresponding 4N33 opto-coupler, the

same is done on the opposite side of the h-bridge. These resistors are used to limit the current entering and/or leaving the bases of the transistor pairs on either side of the h-bridge. When the TIP32C transistor is saturated there will be a voltage drop of 0.7V between the emitter, which is at 24V, and the base. Therefore, the base voltage will be at approximately 23.3V, and if the TIP32C pnp transistor is conducting, this means that the internal 4N33 photo-transistor is saturated, acting as a short to ground. The base current from the TIP32C transistor to ground, must then be less than both the maximum rated base current for the TIP32C transistor, as well as less than the maximum rated collector current of the 4N33s photo-transistor. With a 1.8kΩ resistor, the current from the TIP32C transistor base to ground is approximately 10.6mA, which is less than the 100mA maximum collector current rating for the 4N33 photo-transistor, and the 1A maximum base current rating for the TIP32C transistor. The current drawn from the 24V DC power supply must also be taken into account, this current will also be flowing through the photo-transistor of the 4N33 with the base current from the TIP32C transistor. This current is approximately 20mA, which means that there is a current total of 30mA flowing through the 4N33 photo-transistor, this is less than the 100mA rating. Similarly, when the 4N33 photo-transistor is in cut-off, the TIP31C npn transistor will be saturated, and seeing as its emitter is grounded, there will be a voltage of 0.7V at its base. Because the 4N33 photo-transistor is in cut-off, the 24V power supply connected to its collector through a 1.2kΩ resistor will supply current to the base of the saturated TIP31C transistor. Due to the series 1.2kΩ and 1.8kΩ resistors, this current will be approximately 8mA, significantly less than the 1A maximum base current rating for the TIP31C transistor.

- The 4N33 possesses a 5µs turn-on time and a 100µs turn-off time. The frequency of the PWM signal that the ESP8266 microcontroller is outputting is 500Hz. A frequency of 500Hz, means that the period of the signal is 2ms, and if the switching characteristics of the 4N33 outlined above are correct, then there will be instances where both the photo-transistors in the 4N33 chips will be saturated, and thus both the TIP32C pnp transistors will be saturated as well. Simultaneously, the TIP31C npn transistors will also be in cut-off, blocking all current from flowing through the motor, from the 24V DC power supply, to the ground. If this were to occur without the 1N4148 diodes in place, because there is no where for the current to flow, a high reverse voltage would be generated across the armature winding as it tries to push the current out of

the circuit. Because there is no where for the current to flow, this reverse voltage would continue to build up, reaching a point where the transistors used in the h-bridge would be damaged. This situation can be prevented, however; by adding four diodes in parallel with the TIP31C and TIP32C transistors. With the addition of the four 1N4148 diodes, there is now a path for the current to flow through when both TIP31C npn transistors are in cut-off simultaneously. The high reverse voltage generated across the armature winding will now forward bias one of the 1N4148 diode connected to ground as well as one of the 1N4148 diodes connected to the 24V DC power supply. Thus, providing a path for current to flow out of the armature winding, from ground, into the DC power supply where its internal capacitors will dissipate it.

- Each of the npn and pnp transistor pairs used to construct the h-bridge are connected to two aluminium profile heat sinks as to help disperse the heat caused by the power dissipation in the transistors. Due to a lack of resources, the provided heat sinks were randomly assigned and not specifically chosen. However, located below in table (1) are calculations that determine a theoretical heat sink that could be used with this circuit, based off of measurements performed during its operation. For these calculations R_{cs} is neglected as it is assumed that there is good thermal contact between the heat sink and the casing. Furthermore, this value would be difficult to estimate and determine as there are no specifications available for the thermal compound utilized. The junction temperature was chosen to be $60°C$ as its advisable that the transistors do not exceed this temperature. From these calculations a heat sink of $6.93°C/W$ should theoretically be capable of limiting the temperatures of the transistors below the desired value.

Heat Sink Calculations for the TIP31C npn and TIP32C pnp Transistors	
TIP31C npn Transistor	TIP32C pnp Transistor
$P_d = V_{CE} \times I_C = (19.76)(0.176) = 3.477W$ $T_j = 60°C, T_a = 25°C, R_{jc} = 3.13°C/W$ $R_{sa} = \frac{T_j - T_a - R_{jc} \times P_d}{P_d} = 6.93°C/W$	$P_d = V_{CE} \times I_C = (4.24)(0.176) = 0.746W$ $T_j = 60°C, T_a = 25°C, R_{jc} = 3.13°C/W$ $R_{sa} = \frac{T_j - T_a - R_{jc} \times P_d}{P_d} = 43.79°C/W$

Table (1) - Heat sink calculations.

3. ESP8266 Microcontroller Circuit:

- The ESP8266 microcontroller used in this design serves as the PI controller, the DC motor speed sensor, and as an output for the PWM signals powering the h-bridge. However, to determine the speed of the DC motor and to output the inverted and non-inverted PWM signals to the h-bridge there is also various external circuitry involved. For instance, an ST188 infrared optical sensor is attached to the DC motor, and setup in such a way so that the infrared light emitted from its LED, is effectively absorbed and/or reflected by the different coloured “ticks” (black or metallic) on the disk connected to the shaft of the motor. When reflected, the complementary photo-transistor will conduct and provide a voltage high at the emitter, otherwise, when the photo-transistor is off, the voltage at the emitter will be low. The emitter is then fed into the analog pin (A0) of the ESP8266 microcontroller, where its frequency is used to determine the speed of the DC motor. Both the anode and collector of the ST188 are connected to a 3.3V DC power supply, and a 620Ω resistor is connected in series with the cathode, and a 20kΩ resistor is connected in series with the emitter. Both resistors are grounded to the microcontroller GND pin. The resistors are employed to limit the current flowing through both components, their respective values follow those shown on the ST188s specifications sheet. The ST188 is grounded to the microcontroller so that the analog waveform is properly received from the emitter, and 3.3V DC is used to power the device as that is the operating voltage of the ESP8266 microcontroller. Supplying the microcontroller with a voltage above 3.3V might cause damage and improper readings.
- The h-bridge powering the armature of the DC motor is connected to a 24V DC power supply, during operation there is a large amount of current flowing through this circuit as to maintain the motor running at 40 revolutions per minute (rpm). To protect the ESP8266 microcontroller that is required to supply the inverted and non-inverted PWM signals to this h-bridge, two 4N33 opto-couplers were incorporated into the circuit. One for the non-inverted 3.3V PWM signal output from the ESP8266 microcontroller, and the other for the inverted signal. By utilizing these 4N33 chips, the ESP8266 microcontroller won’t be affected by the high voltages and currents of the h-bridge circuit. Each of the 4N33s were setup with either the inverted or non-inverted PWM signals output from the ESP8266 microcontroller (pins D1 and D0), connected to their anodes. Furthermore, two 140Ω resistors were each placed in series between the 4N33 cathodes and the ESP8266 micrcontrollers GND pin. These resistors limit the current flowing through the 4N33s LED to approximately 20mA,

which guarantees that the light emitted from the LED should be sufficient to turn on the transistor, but 20mA is also well below the 60mA maximum rating. Regarding the outputs of the 4N33 chips, each of their collectors are connected in series with a 1.2kΩ resistor to a 24V DC power supply, and each of their emitters are grounded (separately from the ESP8266 microcontroller). Finally, from each collector there is also a wire that connects to either of the two inputs to the h-bridge. If the non-inverted PWM signal is high, the LED of the 4N33 chip connected to the pin D0 will activate the corresponding photo-transistor, essentially creating a short that will saturate the pnp transistor in the h-bridge. The opposite will occur with the inverted signal, and the npn transistor on the opposite side of the h-bridge will saturate creating a path for current to flow through the motor. Refer to Appendix E (9) for an image depicting the circuit diagram of the ESP8266 microcontroller and the various connected components.

LTspice Simulations (2.2)

LTspice was used to model and simulate the behaviour of the designed h-bridge circuit. The circuit used for these simulations can be seen in Appendix C (7) and the corresponding simulation results can be found in Appendix D (8). By performing a transient simulation with a stop time of 80 milliseconds, the currents and voltages at different points on the circuit were observed to ensure that the circuit was theoretically behaving as desired. The armature of the DC motor was modelled using an inductor of 0.0995H in series with a 17.85Ω resistance and a voltage source mimicking the back EMF (E_A) of the DC motor at an operating speed of 40 revolutions per minute (rpm). Due to the limits of LTspice, there was no way to have the voltage source representing the back EMF switch polarity depending on the input PWM signals, which it would do in reality. Furthermore, the voltage sources providing pulses of 3.3V to the opto-coupler models are used to represent the inverted and non-inverted 3.3V PWM signals output by the ESP8266 microcontroller, to the inputs of the 4N33 chips. For these simulations, the duty cycle (DT) of the non-inverted PWM signal was set to 76.6%, while the inverted PWM signal was set to 23.4%. Figure (6) depicts the resultant voltage waveforms across the DC motor armature after performing a transient simulation with a stop time of 80 milliseconds. The simulation results show that the polarity across the DC motor armature terminals fluctuates from +24V to -24V at a frequency of 500Hz and a duty cycle (DT) of 76.6%. The upper portion of the signal is not symmetric and forms a “peak” due to the fact that the DC motor is an inductive load. By referring to figure (5), which depicts the simulated waveforms at each of the terminals, this anomaly is more easily viewed. This occurs only when the duty cycles of the input PWM signals are not equal, when both are set to 50%, the simulated waveforms across the DC motor armature terminals are symmetric and don’t possess this undesired rise in voltage. Furthermore,

figure (4) depicts the simulated voltage waveforms at the 4N33 photo-transistor collectors, this signal peaks at 17V, indicating that there is a 7V across these resistors. These LTspice simulations results aided in the design of the circuit and provided valuable insight that helped while troubleshooting various complications. Due to the limitations of LTspice, MATLAB and the lack of documentation for the motor, the speed control part of this project could not be simulated, instead experimentally obtained data was utilized.

Potential Improvements (2.3)

The following list provides potential improvements that could be made to the DC motor PI controller given more time and resources:

1. Larger heat sinks: The heat sinks currently used to absorb and dissipate the heat radiated from both the TIP31C npn and TIP32C pnp transistors reach temperatures too high for sustainable use. The incorporation of larger heat sinks would increase the effectiveness of the h-bridge with prolonged use and increase the lifespan of the circuit. The lack of heat dissipation by the heat sinks corresponds to higher transistor junction temperatures, which can cause the operating parameters of the transistors to fluctuate. For instance, the transistors collector current will increase due to higher junction temperatures, and likewise when the collector current increases there is an increase in power dissipated as heat in the transistor. This relationship will cause the collector current to keep increasing, as will the junction temperature, and the transistor will eventually die, this is known as thermal runaway.
2. Improved speed control: The current DC motor PI speed controller is capable of maintaining the DC motor running at 40 revolutions per minute (rpm), regardless of the applied load. However, the speed does fluctuate enough as to suggest that improvements could be made. This fluctuation in the speed is especially noticeable when the motor is operating without a load applied. This occurs as the shaft of the motor is bent which causes it to slow down and speed up at varying points within its rotation. The PI controller helps minimize the effects of this characteristic of the motor, but it is more pronounced when there is no load applied. However, when a load is applied, the torque and armature current increase helping the motor rotate more smoothly. To help the motor maintain a more constant speed of 40 rpm during no load situations a faster method of control would be required. Currently the PI controller cannot effectively adjust for the motors speed changes caused by the bent shaft fast enough, thus providing the fluctuating speed readings. A control method that takes into account the regions where the DC motor shaft slows down could be implemented. When these regions are reached, an increased proportional gain could be applied to the error signal, as to keep the DC motor rotating at a more constant speed, despite the bent shaft.

3. Motor maintenance: The motor currently used for the project presented multiple challenging issues throughout the design of the DC motor PI speed controller. Firstly, the field coil is magnetized in such a way that the motor can only be operated in alternating directions, if it is separately excited. Additionally, this magnetization causes the motor to react to duty cycle (DT) adjustments of the input PWM signals differently than anticipated. For optimal usage of this motor, it should be demagnetized by supplying it with a high AC voltage. Furthermore, the shaft of the motor is bent, this causes the shaft to lag in certain spots as its rotating. This lag causes the DC motor to slow down and makes maintaining its speed at 40 rpm more challenging. Either the purchase of a new motor, or an attempt to straighten the shaft of the current DC motor would be beneficial.
4. Multithreaded processing: The ESP8266 microcontroller used for this project is responsible for multiple tasks that need to occur simultaneously. Due to the limits of the ESP8266, as well as most other microcontrollers for that matter, it can only perform one task at a time as it possesses minimal processing power. This issue is most noticeable when observing the inverted and non-inverted output PWM waveforms from the ESP8266 microcontroller on an oscilloscope. Due to the other operations in the PI controller program that occur irregularly, such as the ISR, the timing of the output PWM signal can be altered. For instance, the ISR might be triggered moments before the output PWM signal is supposed to change states, the time it takes for this ISR to complete will delay the PWM signal from changing states, thus affecting the power supplied to the motor. Now, if the PI control program were implemented onto a multithreaded processor capable of completing different tasks simultaneously, this issue would be resolved. The part of the program responsible for outputting the inverted and non-inverted PWM signals could run un-obstructed by the programs other operations.
5. PWM signal delay: Due to the rise and fall times of the TIP31C and TIP32C transistors used to construct the h-bridge circuit, there are instances where both will be saturated simultaneously on one side. When both these transistors are saturated, this essentially creates a short from the 24V DC power supply, straight to ground which can damage the transistors. The period for which this “shoot-through” event occurs is very brief, but with prolonged usage could eventually cause real damage to the transistors. By implementing a delay in the output PWM signals supplying the bases of the h-bridge transistors, either through software or hardware, this would resolve this issue and increase the longevity of the circuit. This delay would need to be long enough so that the transistor entering cut-off has sufficient time to do so, before the transistor its paired with saturates.

Post-Demonstration Revisions (2.4)

Upon presenting this project and its functionality to colleagues and professors at Lakehead University, some important concerns were brought up regarding the temperature of the heat sinks cooling the TIP31C npn and TIP32C pnp transistors. The fact that these sizeable heat sinks were reaching such high temperatures, suggested that the transistors were dissipating excessive power and not fully saturating. To test this theory, the collector-emitter voltage drops (V_{CE}) across the TIP32C and TIP31C transistors during operation were measured using a digital multimeter. A V_{CE} of -4.24V was measured across the TIP32C transistor and a V_{CE} of 19.76V was measured across the TIP31C transistor. Furthermore, by utilizing LTspice to observe the currents entering/exiting the bases of the transistors, it was noted that the base currents were considerably less than the collector currents of the corresponding transistors. For a transistor to fully saturate, the base current should be at least 1/10th the collector current, providing a beta (β=Ic/Ib) of 10 or less. The LTspice simulations showed that the betas of the transistors used in the h-bridge circuit were closer to 20, thus indicating that the base current was not high enough to fully saturate the transistor. Circuit modifications that would correct this issue include; using smaller resistors in series with the bases of the transistors, and increasing the amplitude of the PWM voltage signals supplying the bases. By lowering the base resistors this will increase the base current and saturate the transistors, and by increasing the amplitude of the PWM signals applied to the bases, this will ensure that the p-n junctions within the transistors are effectively biased for saturation.

Further changes to the designed h-bridge that would minimize power loss and heat dissipation involve switching out the BJT transistors for MOSFETs. The LTspice schematic depicted by figure (3) in Appendix C (7) presents such a circuit. This circuit replaces the previously used TIP31C npn and TIP32C pnp transistors with IRF7343 PMOS and NMOS transistors. Now, when a 24V high is supplied to the gates of the MOSFETs on one side of the h-bridge circuit, the gate-source voltage (V_{GS}) of the NMOS will be above the rated threshold of 1V and it will saturate. Likewise, its PMOS pair will possess a V_{GS} of 0V and it will be cut-off. On the other side of the the h-bridge circuit, because the gates of these MOSFETs are receiving a 0V low, the V_{GS} of the PMOS will satisfy the -1V rating and enter saturation, whereas its NMOS pair will be in cut-off. The two saturated MOSFETs would then create a path for current to flow from the DC power supply, through the load to ground. This circuit would be more efficient than the one incorporating BJT transistors as the MOSFETs would dissipate less power and less resistors are needed. Refer to the last figure in Appendix D (8) for an image of the simulated voltage waveforms across the DC motors armature terminals.

Testing (3)

- Initial Testing (3.1)

- Failure Analysis (3.2)
- Redesigns (3.3)
- Experimental Testing (3.4)
- Results and Calculations (3.5)

Initial Testing (3.1)

Initial tests were performed separately on each of the DC motor PI controllers different components, these tests are outlined below:

1. PI Control Program:
    - To generate and output the inverted and non-inverted PWM signals used to supply the inputs of the 4N33 chips, the originally designed PI control program incorporated two separate “analogWrite()” commands. The “analogWrite()” command outputs an analog PWM signal with a set duty cycle (DT) from a set pin. Two of these commands were originally used to output an inverted and non-inverted PWM signal from two of the ESP8266 microcontrollers pins. When testing this program, an oscilloscope was used to monitor the PWM signals output from the ESP8266 microcontroller. The resulting waveforms displayed on the oscilloscope showed two 3.3V PWM signals out of phase with one another. If these signals were to be used to power the 4N33s connected to the h-bridge, it would cause extensive shoot-throughs, by saturating both npn and pnp transistors on one side of the h-bridge.
    - To measure the speed at which the DC motor was operating, the originally designed PI control program implemented two “pulseIn()” functions. These functions were used to determine the length of time at which the input from the ST188 sensor was either low or high, and if both these quantities were known, the period of the incoming signal could then be calculated. By taking the inverse of this period, the frequency could be determined, and then by using equation (1) detailed in section 1.2 (background and theory) of the report, the speed of the motor in revolutions per minute (rpm) could be computed. This operation made sense in theory, however; when this program was combined with the operation that outputs the inverted and non-inverted PWM signals, it began to cause issues. When an oscilloscope was connected to the pins D0 and D1, the two pins outputting the PWM signals, the waveforms that appeared on the oscilloscope did not match those that the ESP8266 microcontroller should be outputting. The oscilloscope displayed two PWM waveforms with frequencies of 0.5Hz, whereas the ESP8266

microcontroller should have been outputting two PWM signals with frequencies of 500Hz.

2. H-bridge Circuit:
    - To test the h-bridge circuit once it had been fully assembled, a 1.2kΩ resistor was placed between the output terminals of the network to act as the load. A program was then devised to output two 3.3V PWM signals from two of the pins of the ESP8266 microcontroller at frequency of 500Hz with a 50% duty cycle (DT). These pins were then connected to the input LEDs of the 4N33 opto-couplers as to separate the ESP8266 microcontroller from the high currents and voltages of the h-bridge circuit. Furthermore, an oscilloscope probe was connected to the 1.2kΩ resistor as to display the voltage waveform across the load. Upon powering up the DC power supplies powering the circuit, it was noted that the waveform across the resistive load did not match with what was expected, and that the resistors connecting the 24V DC power supply to the output of the 4N33 chips were overheating. These resistors are referenced as Rc1 and Rc2 in the LTspice schematics shown in Appendix C (7). It also became apparent that the TIP31C npn and TIP32C pnp transistors were overheating as well.
3. DC Motor:
    - To determine the characteristics of the DC motor used for this project, the following measurements were performed (refer to table (2) below for the values obtained from these measurements):
        - To obtain the armature resistance (R_A) a digital multimeter (DMM) was connected across the armature terminals, the displayed resistance was then recorded. The same was done to obtain the field resistance (R_F), by connecting the DMM across the field terminals. This resistance does not include the admittance of the armature or field coil as it acts like a short circuit under DC operating conditions.
        - To determine the inductance of the field winding (L_F), an AC voltage of 50V (rms) at 60Hz was supplied to the field terminals and the corresponding current was measured. With the voltage and current known, by applying Ohm's law the total impedance of the field winding was determined. Furthermore, by subtracting the field resistance measured previously from this total impedance, the admittance of the field winding was calculated. With the frequency and admittance known, the inductance of the field winding was

easily found. A similar process was used to determine the inductance of the armature winding (L_A), instead of supplying 50V (rms), however; the supplied voltage was varied until the measured current was approximately 0.97A (rms). Ideally, the rated voltage of 1.5A should be used for this measurement, but due to the limits of the DMM a value under 1A was used instead. The equations used for these calculations can be found in equations (4-5) below.

$$\text{Equation (4)} : X_{L(A,F)} = Z_{L(A,F)} - R_{(A,F)}$$

$$\text{Equation (5)} : L_{(A,F)} = \frac{X_{L(A,F)}}{2\times\pi\times f}$$

- To determine the back EMF (E_A) generated by the DC motor while operating at 40 revolutions per minute (rpm), the DC motor was setup following the separately excited model and the emitter of the ST188 sensor was connected to an oscilloscope. A constant 24V DC was supplied to the field terminals and the DC voltage across the armature terminals was adjusted until the ST188 emitter waveforms displayed on the oscilloscope provided a frequency indicating a speed of 40 rpm. The voltage applied to the armature terminals as well as the current drawn from the DC power supply were both recorded. By utilizing the equations outlined below by equations (6-7), the generated back EMF (E_A) and back EMF constant (K_E) were determined.

$$\text{Equation (6)} : E_A = V_A - R_A I_A$$

$$\text{Equation (7)} : K_E = \frac{E_A * 60}{2\pi * n}$$

Parameters	Values Obtained
Armature Resistance, R_A	17.85 Ω
Armature Inductance, L_A	0.0995 H
Field Resistance, R_F	279.45 Ω
Field Inductance, L_F	11.314 H

Back EMF constant, K_E	-0.157 V/rad/s

Table (2) - DC motor characteristics.

- While initially testing the DC motor it was set up in a shunt configuration, with the armature in parallel with the field. To ensure that the method of controlling the speed of the DC motor through duty cycle (DT) adjustments would work for this design, the armature terminals were switched as to observe that the DC motors direction of rotation would reverse. Instead of reversing directions, however; the DC motor maintained the same direction of rotation despite the armature polarity reversal.

4. Infrared Optical Sensor (encoder):
    - The DC motor used for this project originally possessed a different optical sensor than the ST188, the previous one had been used by past groups attempting this project. To test the original sensor, the anode and collector of the device were connected to a 5V DC power supply. Furthermore, a 120Ω resistor was connected in series from the cathode of the infrared LED to ground, and a 12kΩ resistor was connected in series from the emitter of the devices photo-transistor to ground. Additionally, an oscilloscope probe was connected to the emitter of the device as to observe the output waveform. The shaft of the DC motor was then rotated by hand to simulate the motor operation. Theoretically, when the DC motor is running and the optical sensor is functioning correctly, a voltage waveform should be visible at the emitter of the device. However, there was no waveform displayed on the oscilloscope, and the DC power supply outputting the 5V showed that no current was being drawn.

Failure Analysis (3.2)

1. PI Control Program:
    - The initial tests performed using an oscilloscope on the part of the PI control program responsible for outputting the inverted and non-inverted PWM signals revealed a significant phase shift between the two signals. This phase shift was concluded to be a result of the difference in timers associated with the respective pins used to output the two PWM signals. The “analogWrite()” command utilizes the ESP8266 microcontrollers on-board timers to operate, and the timers of the pins used to output the inverted and non-inverted PWM signals were not synchronized, thus leading to the phase shift in the output waveforms.

- When the pins outputting the inverted and non-inverted PWM signals from the ESP8266 microcontroller (D0 and D1) were connected to an oscilloscope, the oscilloscope displayed two PWM waveforms with frequencies of 0.5Hz. Whereas the oscilloscope should have displayed two 3.3V PWM signals each with a frequency of 500Hz, inverted from one another. To troubleshoot this issue, different parts of the PI control program that had recently been added, were removed and the output PWM signals displayed on the oscilloscope were monitored. When the "pulseIn()" functions were removed from the PI control program, the oscilloscope displayed the desired waveforms, indicating that these functions were causing the PI control program to malfunction. The "pulseIn()" function essentially acts as a while loop that measures the time at which an input signal is either high or low, and it possesses a 1 second timeout. If this function does not receive an input signal, it will wait 1 second before allowing the program to continue with its other operations. Furthermore, there was no input signal being supplied to the "pulseIn()" function. Therefore, because this function was implemented within the void loop, with the "analogWrite()" command used to output the PWM signals, the "pulseIn()" function was delaying the program from repeating. By delaying the loop by 1 second, the periods of the output PWM signals were increased to 2 seconds, and correspondingly their frequencies were changed to 0.5Hz.

2. H-bridge Circuit:
    - The resistors Rc1 and Rc2 were overheating as they were dissipating far more power than what they were rated for. To elaborate, when the 4N33 chip receives a high (3.3V) signal from the ESP8266 microcontroller, the output photo-transistor saturates essentially creating a 24V drop across these resistors. By performing a simple power calculation ((24V)^2/140Ω), the result indicates that these 140Ω resistors are dissipating approximately 4.11W of power. Since these resistors were only rated for 1W, they couldn't handle the excessive power dissipation and were burning out.
    - To help dissipate the heat generated by the TIP31C npn and TIP32C pnp transistors in the h-bridge circuit it was decided that the transistors should be connected to a heat sink to ensure that they remain cool. It was noted during the presentation, however; that the transistors in an h-bridge should not be dissipating enough power to overheat, and therefore a design error has caused the transistors to require heatsinks. Most likely this is caused by the transistors not fully saturating due to minimal base current. To enter

saturation, the base current of a transistor should be at least be 1/10th the collector current. If this is not the case, then the transistor is most likely not fully saturating and the voltage drop between its collector and emitter will never decreases to the value of 1.2V specified in the data sheet. This leads to the transistor dissipating more power than it should.

3. DC Motor:
    - The inability of the DC motor to change its direction of rotation when the armature polarity was reversed was an immediate cause for concern. The designed PI control system revolved around this key behaviour of the DC motor, and without it would not function. By convening with the lab technologist responsible for overseeing this project, it was concluded that the DC motors field winding had become magnetized. This problem arose because the DC motor had only been run in one direction for most of its lifetime, and thus the field winding had become magnetized to favour that direction of rotation. An attempt was made to mediate this behavior by applying a high voltage AC sine wave through the field terminals of the DC motor to demagnetize the field winding. The final results of this test are unclear as a more immediate solution was determined. Instead of setting up the DC motor in the shunt configuration, the separately excited model would be used instead.
4. Infrared Optical Sensor (encoder):
    - The fact that the sensor was not drawing any current from the DC power supply during operation, indicated that either the LED or photo-transistor was malfunctioning. By connecting the terminals of both the LED and the photo-transistor to an Arduino based component tester it showed that neither component was recognized as a transistor or diode. Past groups attempting this project had most likely connected the infrared sensor straight to a DC power supply, instead of adding resistors in series to limit the drawn current. Thus, the on-board LED and photo-transistor had been permanently damaged as they were exposed to currents exceeding their maximum ratings.

Redesigns (3.3)

1. PI Control Program:
    - To resolve the issues pertaining to the part of the PI control program responsible for outputting the inverted and non-inverted PWM signals, the design described in section 2.1 (design and operation) was implemented. Instead of using two separate “analogWrite()” commands, this design utilizes a single “analogWrite()” command outputting an analog PWM

signal with an adjustable duty cycle (DT). The pin from which this signal is output, is connected via a wire to a pin initialized as a digital input that stores the value it receives in a variable declared as “hold”. Two “digitalWrite()” commands are then used to output the value of “hold” and its inverse, from two separate pins set as digital outputs. The digital signals output from these pins are the inverted and non-inverted PWM signals that are supplied to the 4N33 opto-coupler inputs.

- To overcome the issues faced while testing the part of the PI control program responsible for measuring the speed of the DC motor, the design outlined in section 2.1 (design and operation) of the report was utilized. For this design, the “pulseIn()” functions are no longer used to measure the time for which the input analog signal from the ST188 is either high or low. Instead, this analog signal is fed into the analog pin (A0) of the ESP8266 and its value is stored in a variable declared as “temp”. The value of “temp” is then output from a pin set as a digital output, which is connected to a different pin via a wire. This pin possesses an interrupt that is set to trigger whenever the pin changes states, and whenever this interrupt is triggered a variable declared as “pulses” in incremented by 1. Furthermore, within the void loop portion of the program there is an if statement that determines whether or not “pulses” is equal to 2, meaning an entire period has passed. If true, the proceeding commands determine the period of the analog signal output from the ST188 sensor. This period is then used to determine the frequency of the signal, which is then converted to revolutions per minute (rpm) to determine the speed of the DC motor. The incorporation of an interrupt to read the speed of the DC motor makes the PI control program more robust and effective overall.

2. H-bridge Circuit:
    - The issue of Rc1 and Rc2 overheating was very easily overcome by increasing the values of the resistors. By increasing the values of the resistors, their power dissipation is significantly reduced. The new resistor value was determined to be 1.2kΩ with a power rating of 1W. This results in a power dissipation of ((24V)^2/1.2KΩ) 0.48W. This value is much lower than the rated value of the resistors, indicating that there should be no more overheating.
    - By bonding the TIP31C npn and TIP32C pnp transistor pairs to aluminium profile heat sinks, the heat generated by the transistors was effectively dissipated. However, when in operation for long periods of time, the heat sinks themselves begin to heat up significantly, further indicating that the transistors themselves are not fully entering saturation. In order to fully

saturate the transistors, the ratio between the base current and collector current (**β**) would need to be changed such that the base current is at least one tenth of the collector current. This could be achieved by adding resistors in series with the collectors of the transistors. The exact value of these resistors would need to be determined experimentally, but the addition of the resistors should decrease the collector current, and thereby allow the transistors to fully enter saturation.

3. DC Motor:
    - By setting up the DC motor following the separately excited model, the DC motor was capable of changing its direction of rotation when the polarity of the armature was reversed. Due to the requirements of this design to incorporate a 24V DC power supply, it was decided that a 24V DC power supply would be used to separately excite the field winding, while the armature winding would be powered via 24V DC from the h-bridge circuit. This setup ensured the designed PI control method of varying the duty cycles (DT) of the PWM signals input to the h-bridge circuit, would prove both functional and effective.
4. Infrared Optical Sensor:
    - To replace the old, damaged infrared sensor a new ST188 infrared sensor was purchased. To replicate the placement and accuracy of the previous sensor, casing identical to that of the old sensor was 3D printed, and the LED and photo-transistor from the ST188 were removed from their enclosure and placed within the newly printed casing. Wires were then soldered onto the LED and photo-transistor pins, and the device itself was secured to the motor. The LED and photo-transistor had to be adjusted within the casing as to ensure that the infrared light was effectively being reflected and received by the photo-transistor.

Experimental Testing (3.4)

Once the errors negatively impacting the DC motor PI control system had been effectively troubleshooted and resolved, experimental tests were run. These tests were used to determine the output PWM duty cycle (DT) at which the DC motor would run at 40 revolutions per minute (rpm), and the optimal control gains (integral and proportional). To determine the duty cycle of the output PWM signal at which the DC motor would run at 40 rpm, the PI control program outlined in Appendix B (6) was uploaded onto the ESP8266 microcontroller. This program was modified slightly as the proportional and integral gains were set to 0 to eliminate the PI control. PI control was not needed for this test, as the duty cycle was adjusted manually, and intended to remain constant as the DC motor speed was observed. Furthermore, the DC motor was setup following the separately excited DC motor model, with a separate 24V DC

power supply connected to its field winding. The armature of the DC motor was then connected to the terminals of the h-bridge and the h-bridge itself was connected to a 24V DC power supply. Moreover, a 3.3V DC power supply was used to power the ST188 sensor and the ESP8266 microcontroller was powered via a 5V USB port on a laptop. This USB connection from the laptop to the ESP8266 microcontroller was necessary so that the ESP8266 microcontroller could display the DC motor speed measurements on the Arduino IDE serial monitor. Refer to Appendix C (7) and E (9) for circuit schematics visually depicting the setup outlined above. With the circuit setup, to test for the duty cycles of the output PWM signal that would achieve a DC motor speed of 40 rpm at varying respective loads, the following steps were followed:

1. The DC motor load (brake) was initially removed from the DC motor allowing it to operate under no load (0°) conditions. The variable that corresponds to the duty cycle of the output PWM signal, referred to as “PWMDT” in the PI control program, was initially set to a value of 70%.
2. This PI control program was uploaded onto the ESP8266 microcontroller and the DC power supplies powering the field and armature windings were turned on. As the DC motor began operating, its speed in rpm was displayed on the serial monitor of the Arduino IDE.
3. With a duty cycle of 70% the DC motor ran too quickly, well above 40 rpm, and thus the DC power supplies were turned off, the duty cycle variable was increased, and the PI control program was re-uploaded onto the ESP8266 microcontroller.
4. Step 3 above was repeated until a duty cycle was reached that allowed the DC motor to run at 40 rpm under no load conditions. The measured duty cycle for the output PWM signals that allowed the motor to operate at 40 rpm under no load, was approximately 76.6%.
5. This 76.6% duty cycle was then used as a starting point for the same tests trying to find the duty cycle that would allow the motor to run at 40 rpm at various applied loads. For instance, when the applied load was set to 2.5°, at a duty cycle of 76.6% the motor ran too slowly. The duty cycle must instead be decreased to 76% for it to operate at a speed of 40 rpm with this applied load.
6. The load was applied to the DC motor in increments of 2.5°, up until full load at 10°. For each variation in load, the duty cycle of the output PWM signal was decreased from its previous value until the DC motor began to operate at 40 rpm, at the respective load. The duty cycles for each of the different applied loads, that allowed the DC motor to run at 40 rpm, were each recorded. These duty cycles, along with their respective load, can be seen tabulated in Appendix F (10).

The following procedure was followed as to tune the proportional (Kp) and integral (Ki) gains of the PI control program. Suitable proportional and integral gains would provide a fast and

stable response with minimal overshoot as to maintain the motor operating at 40 rpm when the load was applied and/or removed. The same setup outlined above for the experimental test used to determine the output PWM signal duty cycle at which the DC motor would run at 40 rpm was also used for this portion of the testing.

1. Both the integral and proportional gains were initially set to 0.001 by initializing both the "Kp" and "Ki" variables on the PI control program to 0.001. Furthermore, the duty cycles of the inverted and non-inverted PWM signals output from the ESP8266 microcontroller were set to 76.6% (inverted is 23.4%). This was done by initializing the "PWMDT" variable in the PI control program to 0.766*1024. This program was then uploaded onto the ESP8266 microcontroller and the DC power supplies powering the armature and field windings were turned on.
2. The proportional gain was then incremented in steps of 0.01 (i.e. 0.001, 0.01, 0.02, etc.), while the integral gain was held constant at 0.001. For each increment, the speed of the DC motor in rpm was monitored using the Arduino IDE serial monitor. Furthermore, for each increment the DC motor was operated under both full load and no load conditions by applying and removing the brake. While operating under no load conditions the variation of the DC motors speed from 40 rpm was observed. The same was done as the DC motor was operating under full load conditions. Additionally, by using the Arduino IDE serial plotter to graph the speed of the DC motor with respect to time, the speed of the PI control response to the addition and/or removal of the load was observed.
3. Once a proportional gain of 0.09 was reached, the DC motor PI controller was able to effectively maintain the DC motor running at 40 rpm, under both no load and full load operating conditions. Furthermore, the reaction speed of the PI controller to adjust for changes in the applied was significantly improved.
4. The integral gain was then incremented in steps of 0.001 while the proportional gain was held constant at 0.09. The conditions and measurements outlined in step 2 above were repeated similarly for each increment of the integral gain. However, it was determined that the original integral gain of 0.001 worked best for this design.

Results and Calculations (3.5)

Through the first set of experimental tests outlined above in section 3.4 (experimental tests) it was determined that by supplying the h-bridge circuit with PWM signals with a duty cycle (DT) of 76.6%, the DC motor would operate at 40 revolutions per minute (rpm) under no load conditions. This result is significant, as the PWM signals output from the ESP8266 microcontroller will be initialized to this value when the PI control program is run. This was done assuming that when testing the DC motor speed control, the DC motor would be initially run under no load conditions so that even without any control adjustments it should still maintain a speed of 40 rpm. Furthermore, the other measured duty cycles that allowed the DC motor to

operate at 40 rpm, under varying loads, provided a range of values that were taken into consideration while tuning the PI control gains. These duty cycles and their corresponding applied loads can be seen tabulated in Appendix F (10).

The second set of experimental tests were performed as to obtain reasonable proportional (Kp) and integral (Ki) gains that would allow for fast and stable responses to changes made to the system (i.e. load applied/removed). Through these tests, a Kp of 0.09 and a Ki of 0.001 were obtained. With the proportional and integral gains set to these values on the PI control program, the control system responded quickly, with minimal overshoot, to changes made to the applied load. Furthermore, by incorporating these control gains and utilizing the Arduino IDE serial plotter, graphs of the DC motor speed under varying load conditions were obtained. From these graphs it was visible that there is some variance in the speed of the DC motor, and that the PI control program does not maintain it running at exactly 40 rpm. While operating under no load conditions, the speed of the DC motor fluctuates between 60-30 rpm. Whereas, while operating under full load conditions, this variance is significantly less, but still visible, with the DC motor speed varying between 50-35 rpm. These graphs can be seen in Appendix F (10). Moreover, by monitoring the speed of the DC motor under both no load and full load conditions and recording the measured speed displayed on the Arduino IDE serial monitor, statistical calculations were performed to summarize this data. By inputting this data into Excel, the mean, standard deviation, and variance of the DC motor speed under both no load and full load conditions were determined. These results are shown in equations (8-13) and table (3) below.

$$\text{Equation (8)} : \overline{N}\ (no\ load) = \frac{\sum_{i=1}^{n} N_i}{n} = \frac{1537.5}{41} = 37.5\ rpm$$

$$\text{Equation (9)} : s_N\ (no\ load) = \sqrt{\frac{\sum_{i=1}^{n} (N_i - \overline{N})^2}{n-1}} = 13.43503\ rpm$$

$$\text{Equation (10)} : s_N^2\ (no\ load) = 180.50003\ rpm^2$$

$$\text{Equation (11)} : \overline{N}\ (full\ load) = \frac{\sum_{i=1}^{n} N_i}{n} = \frac{1757.5}{37} = 47.5\ rpm$$

$$\text{Equation (12)} : s_N\ (full\ load) = \sqrt{\frac{\sum_{i=1}^{n} (N_i - \overline{N})^2}{n-1}} = 4.949747\ rpm$$

Equation (13) : $s_N^2\ (full\ load)\ =\ 24.49999\ rpm^2$

Parameters	No Load	Full Load
$\overline{N}$ (rpm)	37.50000	47.50000
s_N(rpm)	13.43503	4.949747
s_N^2(rpm*rpm)	180.50003	24.49999

Table (3) - Statistical calculations pertaining to the measured speed of the DC motor.

Conclusion (4)

To conclude, the main objective of this project was successfully achieved. A PI control system was designed that could maintain a DC motor operating at 40 revolutions per minute (rpm), under varying load conditions. The designed PI control system features an ESP8266 microcontroller with a PI control program uploaded onto it, an h-bridge circuit, and an ST188 infrared sensor. The PI control program uploaded onto the ESP8266 microcontroller is responsible for the following; outputting an inverted and non-inverted PWM signal to the h-bridge circuit, calculating the speed of the DC motor through the analog signal generated by the ST188 sensor, and implementing the PI control algorithm that adjusts the duty cycle (DT) of the output PWM signals based on the speed of the DC motor. Through experimental testing an initial duty cycle of 76.6% was determined, as were proportional (Kp) and integral (Ki) gains of 0.09 and 0.001 respectively. When initialized to these values, the PI control system demonstrates fast and stable responses to variations in the applied load (i.e. magnetic brake). Under no load conditions, the average speed of the DC motor is 37.5 rpm, with a standard deviation of 13.43 rpm, and a variance of 180.5 rpm*rpm. Whereas, under full load conditions, the average speed of the DC motor is 47.5 rpm, with a standard deviation of 4.94 rpm, and a variance of 24.9 rpm*rpm. From the results outlined previously, it is clear that although the PI speed controller does not maintain the DC motor operating at exactly 40 rpm, it maintains an average of approximately 40 rpm. To account for this variation in the DC motor speed, future iterations of this design could consider implementing derivative control as well as a larger incorporation of integral control as well (i.e. Ki > 0.001). Furthermore, adjustments to the h-bridge circuit itself would be beneficial as to ensure that the transistors are fully entering saturation and not dissipating excessive power. Finally, to prevent both transistors on one side of the h-bridge from saturating simultaneously and creating a short to ground, a delay should be incorporated into the

system, either through software or hardware, this will increase the lifespan and effectiveness of the h-bridge circuit.

Appendices

- Parts List (5)
- PI Control Program (6)
- LTspice Schematics (7)
- LTspice Simulation Results (8)
- Experimental Results (9)
- References (10)

Appendix A (5) - Parts List

Refer to table (4) below for a complete list of the components used to construct the DC motor PI controller and their corresponding role in the project.

Quantity	Components Used (Name and Manufacturer Part/Model Number)	Role in DC Motor PI Controller
2	TIP31C npn power transistor Central Semiconductor Corp. TO-220 CASE	Alternates the polarity of the DC power supply powering the DC motor.
2	TIP32C pnp power transistor Central Semiconductor Corp. TO-220 CASE	Alternates the polarity of the DC power supply powering the DC motor.
2	Opto-coupler 4N33 Vishay semiconductors	Protects the ESP8266 from the high voltages and currents in the h-bridge network.
4	Diode 1N4148 Vishay semiconductors	Protects the h-bridge transistors from the high reverse voltage generated across the armature coil during switching.
2	Heat sink (Aluminium profile)	Dissipates the heat generated by the TIP31C npn and TIP32C pnp transistors.

2	140Ω resistor	Limits the current through the 4N33 input LED.
1	620Ω resistor	Limits the current through the ST188 LED.
2	1.2kΩ resistor	Limits the current through the 4N33 output photo-transistor.
4	1.8kΩ resistor	Limits the current entering/exiting the bases of both the TIP31C and TIP32C transistors.
1	20kΩ resistor	Limits the current through the ST188 photo-transistor.
1	ST188 infrared optical sensor	Generates an analog signal which possesses a frequency proportional to the speed of the DC motor.
1	50V class B DC motor FRACMO	The DC motor that the developed PI DC motor speed controller will attempt to maintain running at 40 rpm.
1	ESP8266 microcontroller	Responsible for measuring the speed of the DC motor, outputting the inverted and non-inverted PWM signals to the 4N33 chips, and performing the PI control algorithm.

Table (4) - Complete list of the components used to construct the DC motor PI controller and their corresponding role.

Appendix B (6) - PI Control Program

```
//SPEED SENSOR
volatile int pulses = 0; //variable shared between ISR and main program,
initialized to 0
int InterruptPin = 2; //interrupt pin (GPIO2), D4
unsigned long CurrentTime = 0; //the current time is initialized to 0
unsigned long LastTime;
unsigned long rpm;
int EncoderDigital = 4; //the digital encoder signal output (GPIO4), D2
int temp; //temporary storage variable for the analog encoder input
```

```
//PWM SIGNAL
int PWMfrequency = 500; //the frequency of the PWM signal
int PWMrange = 1024; //the range of the PWM signal (0-1023 bits), 1023 = 100% DT
int PWMsource = 14; //the analog PWM signal output pin (GPIO14), D5
int PWMDT = 0.766*1024; //the duty cycle of the PWM signal, maintains 40 rpm at
no-load
int PWM = 16; //the non-inverted PWM signal output pin (GPIO16), D0
int PWMI = 5; //the inverted PWM signal output pin (GPIO5), D1
int hold; //variable to hold the value of the analog PWM signal
int PWMinput = 0; //the analog PWM input pin (GPIO0), D3

//PI CONTROL
unsigned long TimeDifference,PreviousTime; //variable declaration
unsigned long NowTime = 0; //the current time is initialized to 0
double CurrentError = 0; //the current error is initialized to 0
double ErrorSum = 0; //the cumulative error is initialized to 0
float Kp = 0.09; //proportional gain is initialized to 0.09
float Ki = 0.001; //integral gain is initialized to 0.001

void setup() {

  //SPEED SENSOR
  pinMode(EncoderDigital,OUTPUT); //sets the pin D2 as an output
  pinMode(InterruptPin,INPUT); //sets the pin D4 as an input
  Serial.begin(115200); //sets the serial data transmission rate to 115200 bps
  attachInterrupt(InterruptPin,PulseChange,CHANGE); //sets the interrupt to trigger
the PulseChange function when the state of pin D4 changes

  //PWM SIGNAL
  pinMode(PWMsource,OUTPUT); //sets the pin D5 as an output
  pinMode(PWMinput,INPUT); //sets the pin D3 as an input
  analogWriteRange(PWMrange); //sets the range of the PWM signal from 0-1023 bits
  analogWriteFreq(PWMfrequency); //sets the frequency of the PWM signal to 500Hz
  pinMode(PWM,OUTPUT); //sets the pin D1 as an output
  pinMode(PWMI,OUTPUT); //sets the pin D0 as an output
}

void loop() {

  //PWM SIGNAL
  analogWrite(PWMsource, PWMDT); //outputs a PWM signal with a set duty cycle
  hold = digitalRead(PWMinput); //stores analog PWM signal
  digitalWrite(PWM,hold); //outputs the non-inverted PWM signal
  digitalWrite(PWMI,!hold); //outputs the inverted PWM from

  //SPEED SENSOR
```

```
  temp = (analogRead(A0))/1024; //the encoder output is stored in the temp variable
and normalized from 0-1
  digitalWrite(EncoderDigital,temp); //the analog encoder output is then digitally
output from the pin D2
  if(pulses == 2){ //there are 2 pulses (changes in logic) per period
    noInterrupts(); //prevents the interrupt from occurring during this loop
    LastTime = CurrentTime; //sets the last time that the loop was run
    CurrentTime = millis(); //sets the current time
    rpm = ((1000.0 / (CurrentTime - LastTime)) / 90.0) * 60.0; //calculates the
motor speed in rpm

      //PI CONTROL
      if(rpm > 40.0){ //when the speed of the motor is determined to be greater
than 40 rpm, this if statement will be satisfied
        PreviousTime = NowTime; //sets the previous time as the current time of the
previous loop, will be 0 initially
        NowTime = millis(); //sets the NowTime as the current time
        CurrentError = abs((40.0 - rpm)); //computes the current error, which is
the difference between 40 rpm and the current speed
        ErrorSum = abs(ErrorSum + (CurrentError * ((NowTime - PreviousTime) /
1000.0))); //computes the cumulative error
        PWMDT = PWMDT + (Kp * CurrentError + Ki * ErrorSum); //addition because the
speed decreases when the DT increases
      }
      if(rpm < 40.0){ //when the speed of the motor is determined to be less than
40 rpm, this if statement will be satisfied
        PreviousTime = NowTime; //sets the previous time as the current time of the
previous loop, will be 0 initially
        NowTime = millis(); //sets the NowTime as the current time
        CurrentError = abs((40.0 - rpm)); //computes the current error, which is
the difference between 40 rpm and the current speed
        ErrorSum = abs(ErrorSum + (CurrentError * ((NowTime - PreviousTime) /
1000.0))); //computes the cumulative error
        PWMDT = PWMDT - (Kp * CurrentError + Ki * ErrorSum); //subtraction because
the speed increases when the DT decreases
      }
    pulses = 0; //returns the variable to 0, it can now determine when another
period passes
    interrupts(); //enables interrupts now that the loop has completed
    Serial.println(rpm); //displays the measured speed of the motor in rpm on the
serial monitor
  }
}

//SPEED SENSOR
void PulseChange() //interrupt service routine
{
```

```
  pulses++; //everytime the state of the interrupt pin changes this value is
incremented by 1
}
```

Appendix C (7) - LTspice Schematics

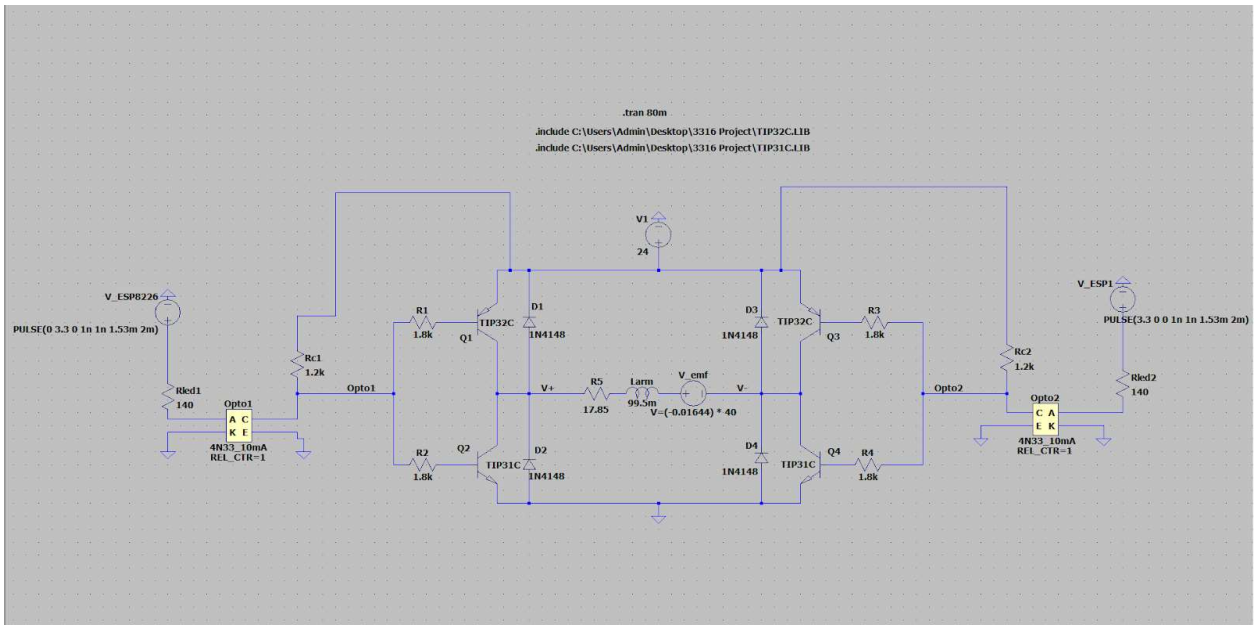


Figure (2) - H-bridge LTspice schematics used for simulations.

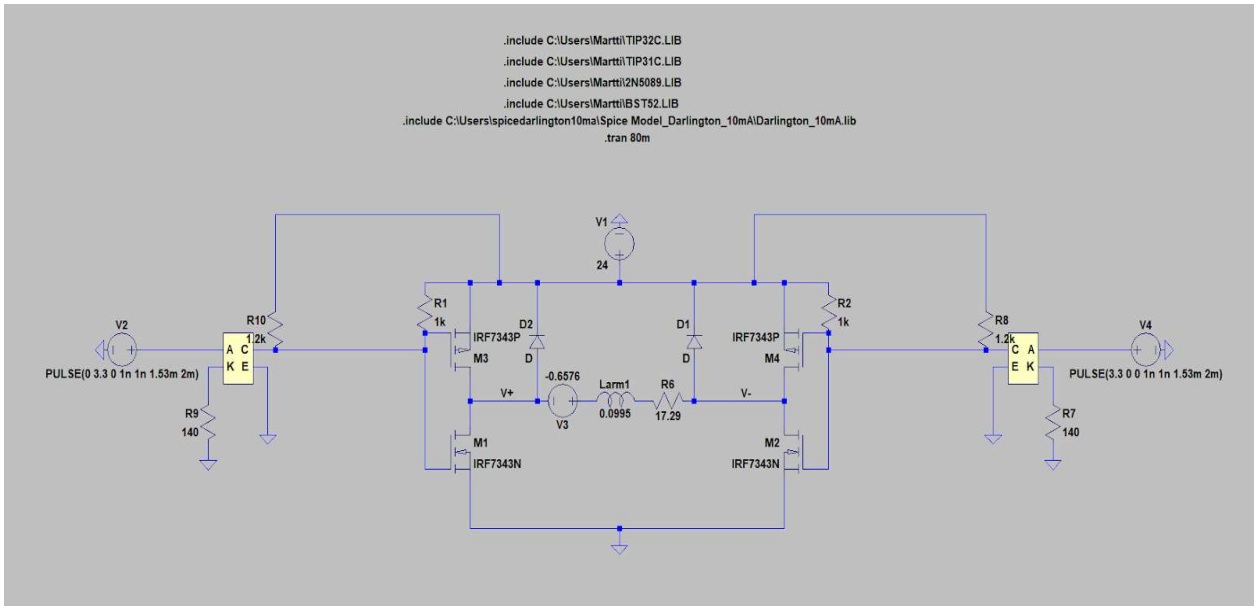


Figure (3) - Revised h-bridge LTspice schematics.

Appendix D (8) - LTspice Simulation Results



Figure (4) - Transient simulation results showing the voltage output from the 4N33 collector (inverted side 23.4% DT), and the base currents at Q1 and Q2.

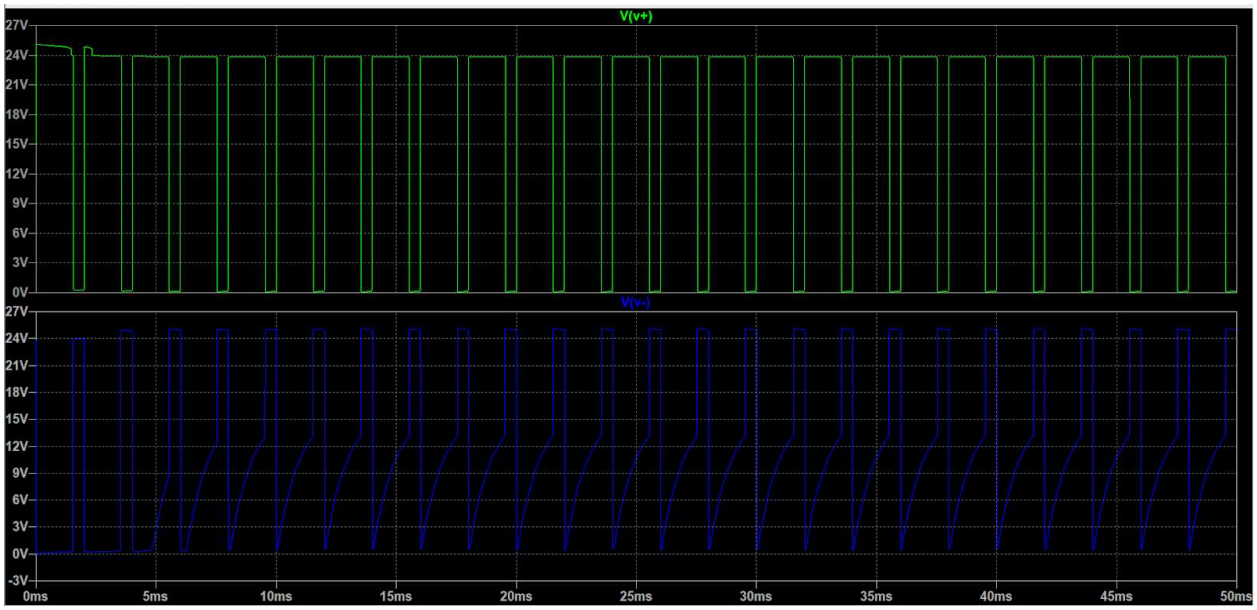


Figure (5) - Transient simulation results showing the voltages at the DC motor armature terminals.

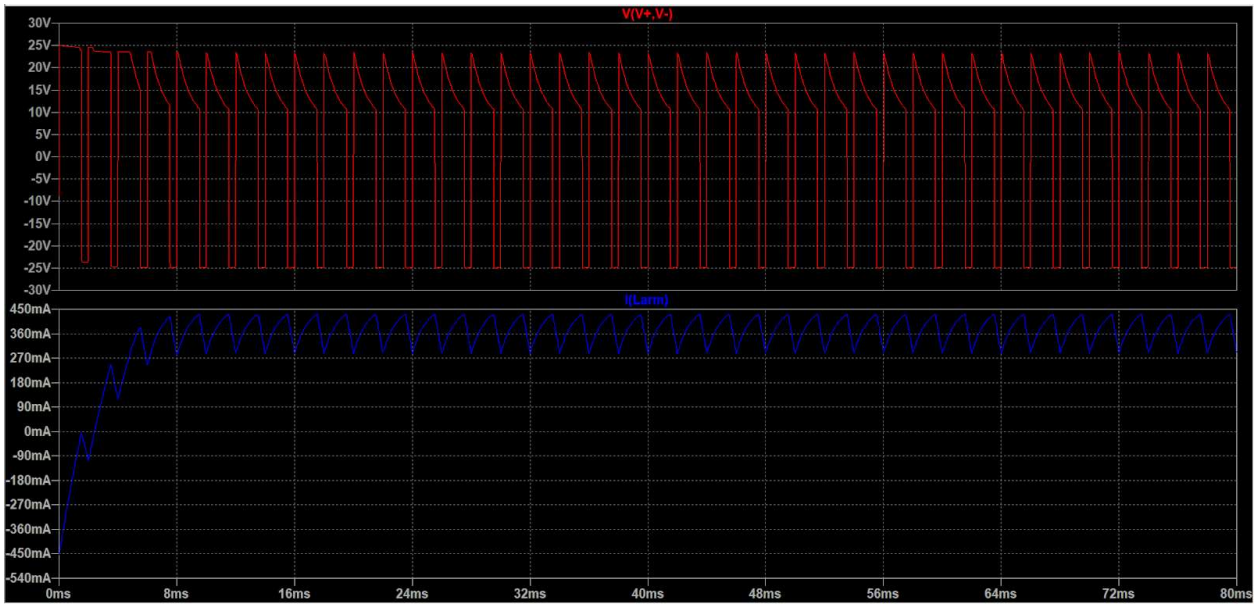


Figure (6) - Transient simulation results showing the voltage across the DC motor armature, as well as the current through the armature winding.

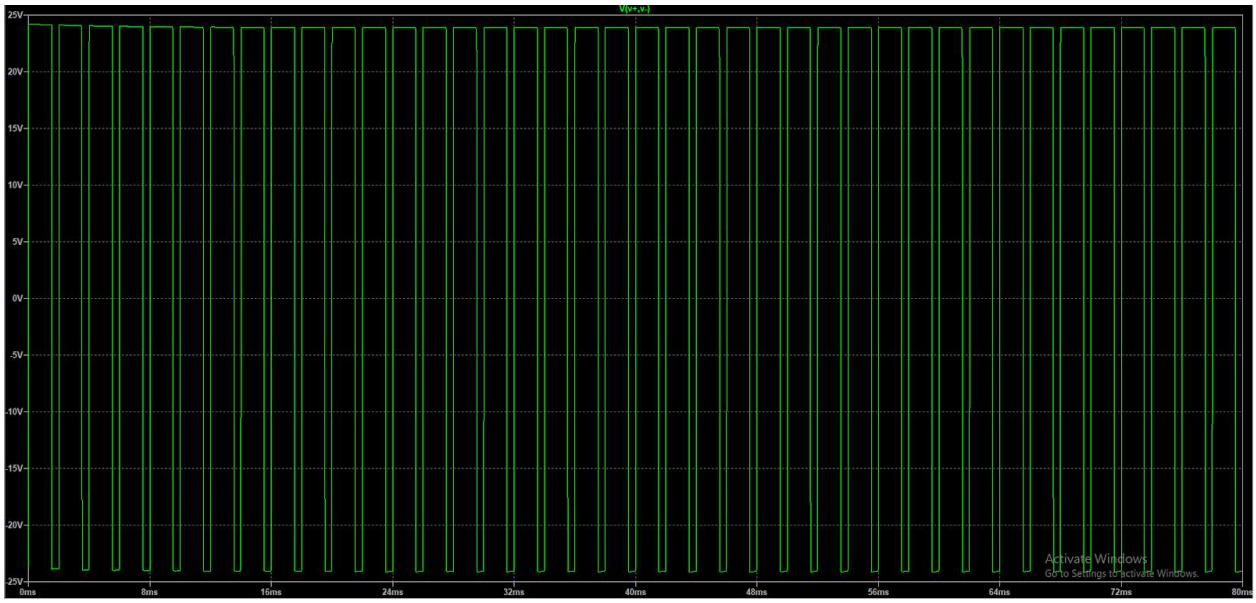


Figure (7) - Transient simulation results for the revised h-bridge circuit, shows the voltage across the DC motor armature.

Appendix E (9) - ESP8266 Circuit Diagram

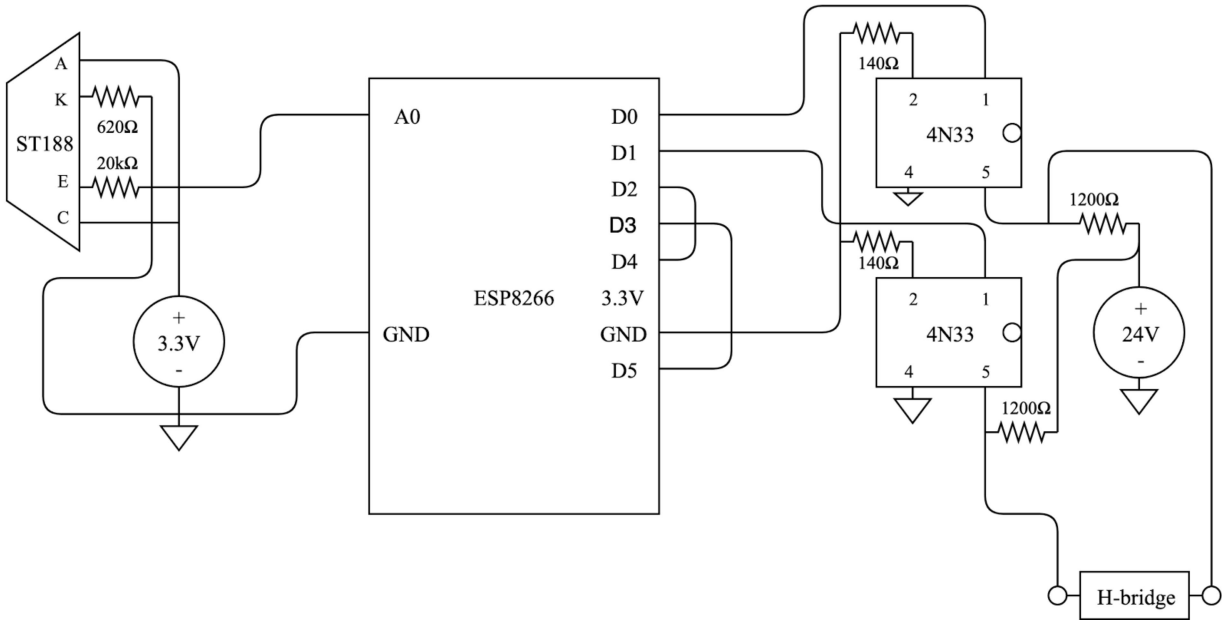


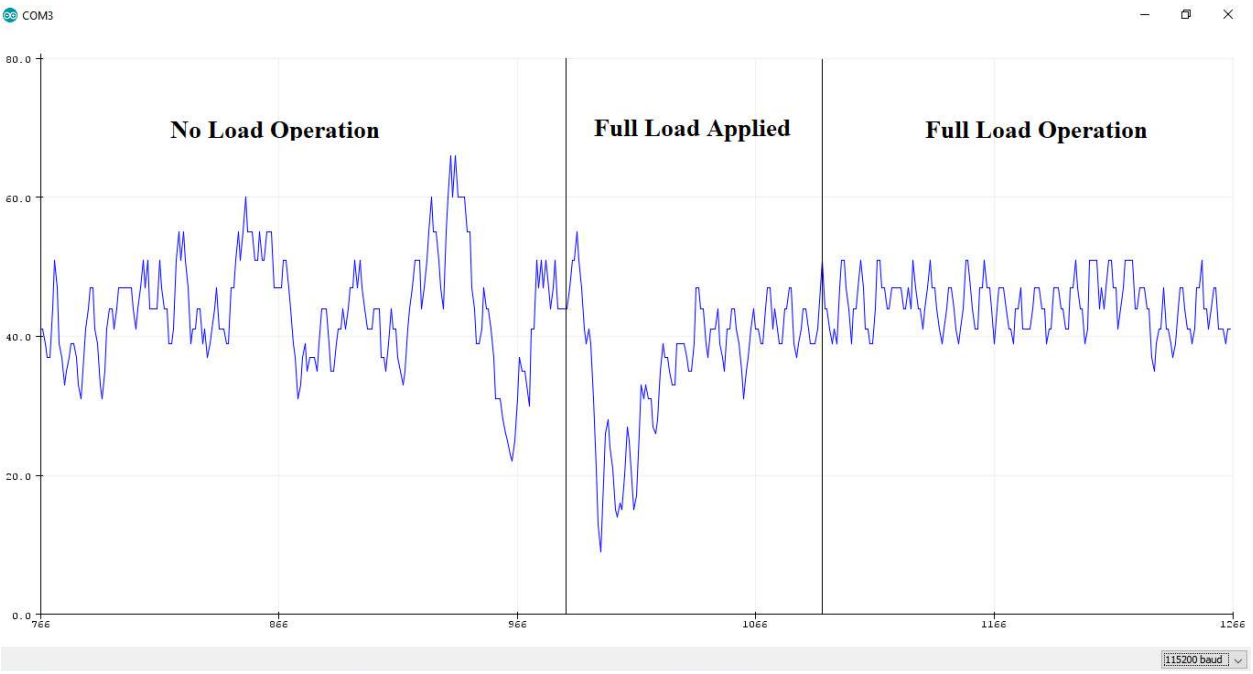
Figure (8) - ESP8266 Circuit Diagram.

Appendix F (10) - Experimental Results

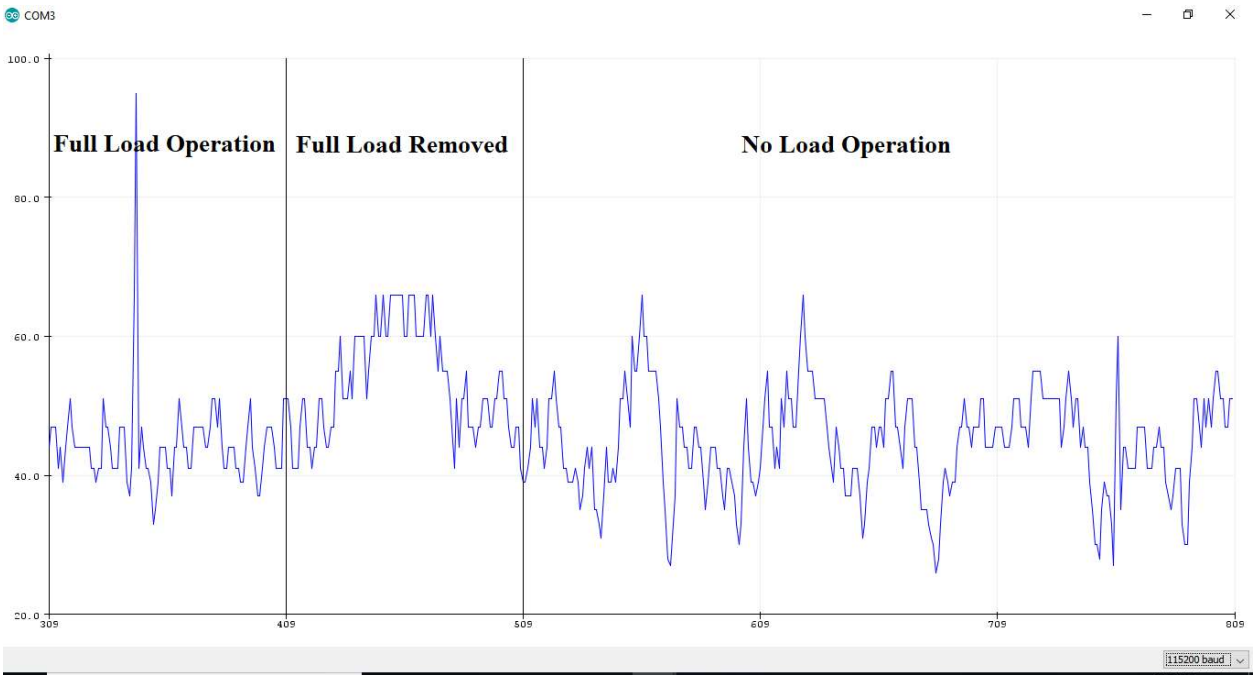
Refer to table (5) below for the experimental results gathered while testing for the duty cycle of the output PWM signal that would allow the motor to run at 40 revolutions per minute (rpm). For this test the applied load was varied from 0° (no load) to 10° (full load), and the duty cycle of the output PWM signal was adjusted until the motor ran at 40 rpm with the respective load applied. Furthermore, refer to graphs (1) and (2), and figures (9) and (10) below for the experimental results gathered once the most effective integral and proportional control gains had been determined. The graphs show the speed of the DC motor in rpm with respect to time, time is shown in the x-axis, whereas the speed of the DC motor is shown on the y-axis. The figures present the speed of the DC motor in rpm as it was displayed on the serial monitor of the Arduino IDE.

Applied Load	Output PWM Signal Duty Cycle
0° (No Load)	76.6%
2.5°	76%
5°	74%
7.5°	72.5%
10° (Full Load)	69.5%

Table (5) - Tabulated experimental results showing the effects of the applied load, and how the duty cycle of the output PWM signal must change to maintain the DC motor operating at 40 rpm.



Graph (1) - DC motor speed in revolutions per minute (rpm) when the brake is applied (full load).



Graph (2) - DC motor speed in revolutions per minute (rpm) when the brake is removed (no load).



Figure (9) - DC motor speed in revolutions per minute (rpm) during no load operation.



Figure (10) - DC motor speed in revolutions per minute (rpm) during full load operation.

Appendix G (11) - References

[1] I. G. A. P. R. Agung, S. Huda and I. W. A. Wijaya, "Speed control for DC motor with pulse width modulation (PWM) method using infrared remote control based on ATmega16 microcontroller," *2014 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS)*, Kuta, 2014, pp. 108-112.doi: 10.1109/ICSGTEIS.2014.7038740

[2] J. Barua and M. A. Abedin, "Design and Implementation of a FPGA Based Closed Loop Speed Controller for DC Motor using PWM Technique," *2018 International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)*, Gazipur, Bangladesh, 2018, pp. 1-4.doi: 10.1109/ICAEEE.2018.8642965

[3] V. D. Yurkevich and N. A. Stepanov, "PWM speed control of DC motor based on singular perturbation technique," *2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, St. Petersburg, 2014, pp. 434-440.doi: 10.1109/ICUMT.2014.7002140

[4] T. Hemanand and T. Rajesh, "Speed control of brushless DC motor drive employing hard chopping PWM technique using DSP," *2006 India International Conference on Power Electronics*, Chennai, 2006, pp. 393-396.doi: 10.1109/IICPE.2006.4685404

[5] R. K. Munje, P. P. Shinde and S. S. Kale, "Performance comparison of PI/PID controllers for DC motor," *2014 Annual IEEE India Conference (INDICON)*, Pune, 2014, pp. 1-3.doi: 10.1109/INDICON.2014.7030359

[6] Faroqi, Adam & Ramdhani, Muhammad & Frasetyio, F & Fadhil, A. (2018). DC Motor Speed Controller Design using Pulse Width Modulation. IOP Conference Series: Materials Science and Engineering. 434. 012205. 10.1088/1757-899X/434/1/012205.

[7] “Libraries,” *Arduino*. [Online]. Available: https://www.arduino.cc/en/reference/libraries. [Accessed: 11-Dec-2019].