DC MOTOR SPEED CONTROL SYSTEM

PROJECT REPORT

6/14/2017

SOLARILLION FOUNDATION
SIDDHARTH DIVI

DESIGN OF A CLOSED LOOP SPEED CONTROL SYSTEM FOR A 12V DC MOTOR

EXPERIMENTAL SETUP:

BRIEF DESCRIPTIONS OF COMPONENTS USED:

1. IR Obstacle Detector Sensor:

- The IR sensor's output is high when the IR rays emitted by the transmitter are reflected back by the blade of the motor, and detected by the receiver. This is used to measure the speed of the motor in terms of RPM, by counting the number of times the blade has cut the transmitted rays.

2. DC Motor:

- The **DC motor** practically represents the system, in this case.

3. Transistor (TIP122):

- The transistor acts as a switch, enabling the Arduino to control the DC Motor, which requires a higher current requirement than the Arduino is able to provide.

4. Voltage Regulator (7812):

- 7812 - 12V Voltage Regulator converts regulated or **unregulated input voltages to 12V output** voltage.

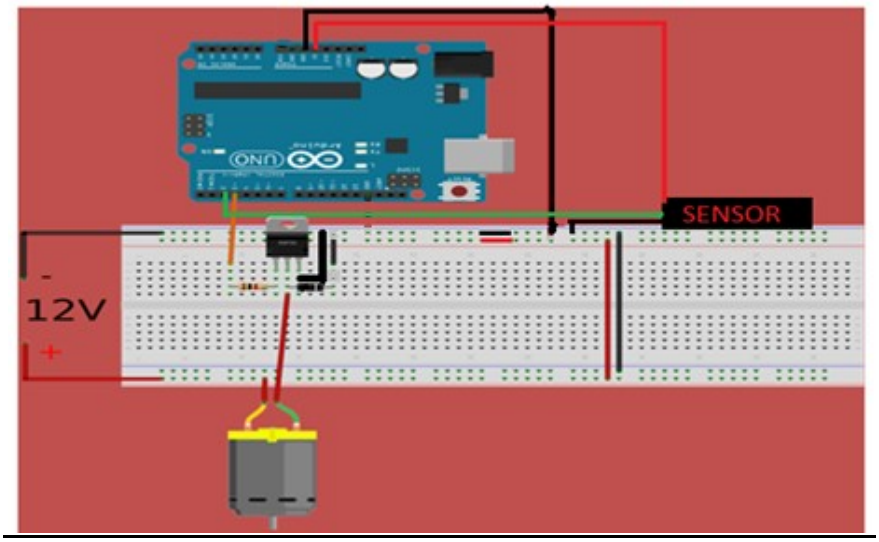
5. Fly-back Diode:

- The **flyback diode** is used to eliminate **flyback**, which is the sudden voltage spike seen across an inductive load when its supply current is suddenly reduced or interrupted.

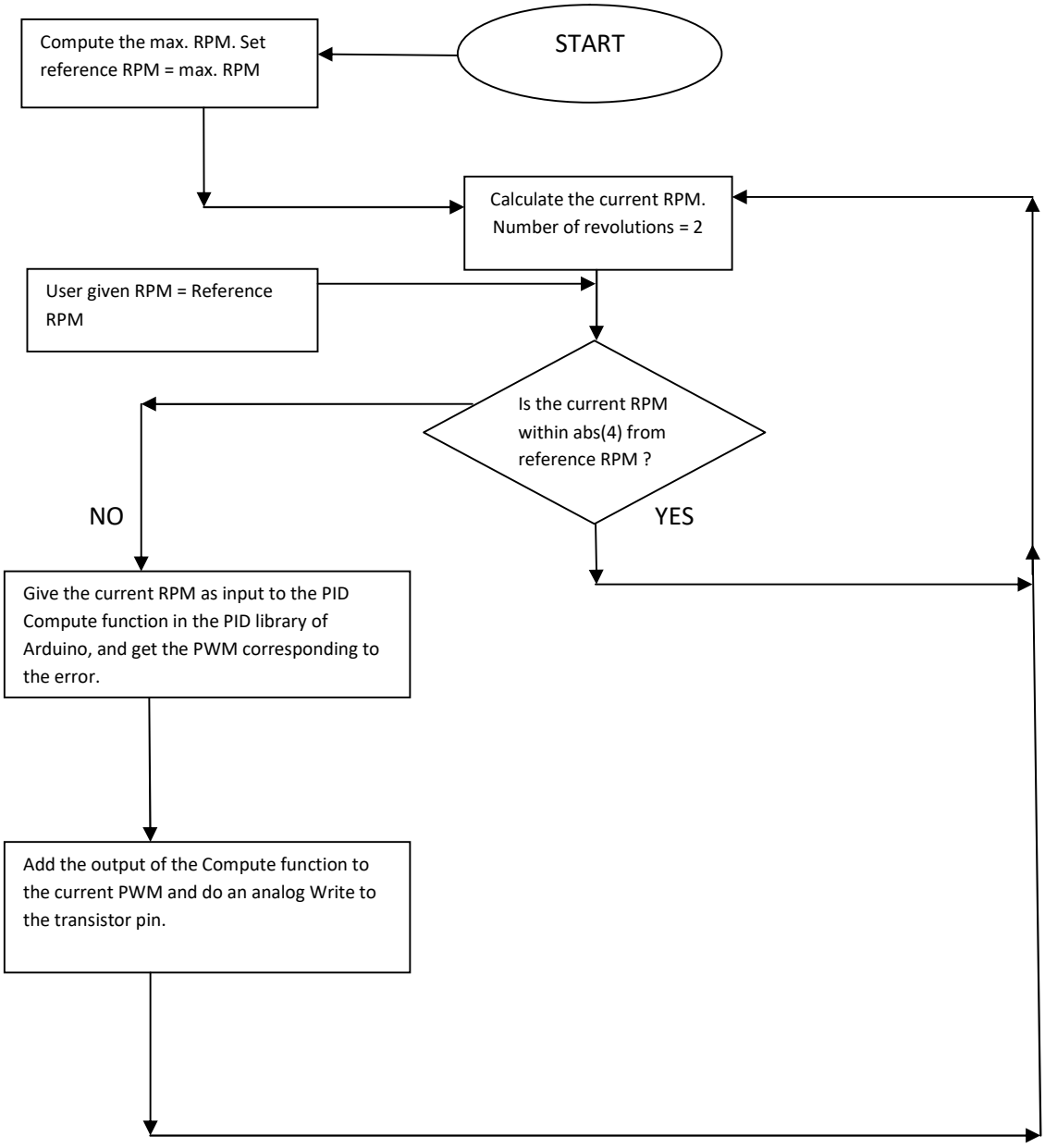
6. Arduino Uno:

- An Arduino Uno R3 ATmega328P is used.

CIRCUIT DIAGRAM:



FLOW CHART:



METHODOLOGY USED:

1. RPM CALCULATION:

- The method used to calculate the RPM of the DC Motor is the **Interrupt Service Routine** of Arduino.
- The interrupt continuously monitors a low to high transition in the IR Sensor, based on which a counter value, rev, is incremented.
- Based on this counter the RPM is calculated by the formula given below:
- **RPM = (temp * rev)/time**

Where,

- temp = 60000,
- rev = 2,
- time = (Current Time – Time when the previous RPM was calculated), in **ms**.

DC Motor Characteristic Curve (PWM vs RPM):

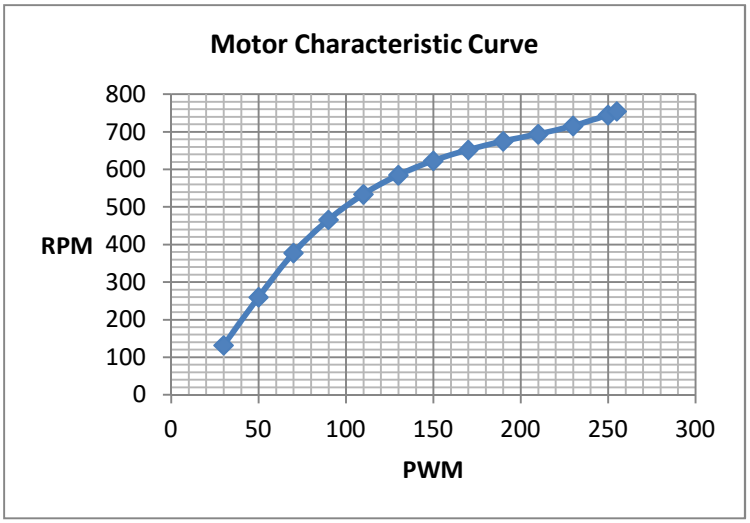


Fig. 1 : DC Motor Characteristic Curve.

2. MAKING THE MOTOR ROTATE AT A USER GIVEN RPM:

- For this module, the method used is that of the well known PID Controller, with some slight modifications, to suit the system.
- This system is built using only a P - controller, and to reduce the oscillations caused due to the P-term, the PWM is linearly incremented/decremented , when the current RPM is within a certain range of the reference RPM, i.e. when the abs(current RPM – Reference RPM) is within 20.
- The P – term is the proportional term, which gives the PWM value corresponding to the error difference between the current RPM and the reference RPM.
- The current RPM is given as input to the PID Compute function, which returns a PWM output corresponding to the error.

➢ **Output** = **Kp*error**, where **Kp = 0.2** : Proportional Term Constant and
error = abs(Current RPM – Reference RPM)
where **Output**, is the output of the PID Compute function.
➢ **PrevOutput** = **PrevOutput** + **Output**, where **PrevOutput = Current PWM.**

- **PrevOutput**, is then written to the transistor pin.
- **Reference RPM = User given RPM.**

(1)DC Motor Settling Times:

RPM	150	200	250	300	350	400	450	500	550	600	650	700	750	AVG Time (sec)
150	-	4.1	5.3	6.9	4.6	4.611	4.56	4.7	3.3	3.4	2.1	2.32	3.28	4.0975
200	4.67	-	5.8	3.19	5.9	3	4.3	3.2	3.4	3.9	1.6	2.2	2.3	3.5263
250	4.41	2	-	3.6	3.3	1.7	4.27	3.2	3.6	3.5	1.6	4.23	2.5	3.0454
300	4.07	6	2.8	-	3.7	3.7	3.3	3	3.5	3.9	2.9	4.7	2.5	3.6363
350	3.74	4	5.6	4.2	-	2	3.27	1.6	3.4	3.8	1.7	4.1	2.5	3.2881
400	5.5	5.6	3.4	3	4.17	-	1.2	3.6	3.1	1.4	1.6	2.65	2.5	2.929
450	4.137	4.6	5.3	6	4	4	-	4	3	1.3	1.6	5.05	2.7	3.7772
500	4.12	4.9	5.8	5.8	2.6	4.4	3.7	-	1.7	1.6	2.1	5.22	2.7	3.6836
550	4.7	4	3.5	3.2	6.6	4.42	3.9	1	-	1.9	1.8	2.76	2.6	3.2436
600	4.7	5.3	3.5	4.4	6	4.7	4.8	3.1	1.2	-	2.3	4.81	2.5	3.9425
650	4.71	4.6	3.5	3	6.4	4.2	4.2	4.5	2.52	2.3	-	3.1	2.2	3.7691
700	4.67	4.3	5.2	5.7	5.9	6.1	4.14	2.6	2.7	2.5	1.3	-	1.4	3.8758
750	4.48	5.7	3.7	4.6	4.4	4.3	4.2	3.1	1.8	2.4	2.8	2.1	-	3.6316
AVG Time (Sec)	4.4922	4.6363	4.3727	4.2445	4.8154	3.8654	3.7527	2.9909	2.72	2.5909	1.9363	3.72	2.4	**3.5727**

Table 1 : Tabulation of DC Motor Settling Times.

- The average motor response time is calculated to be **~3.5727 seconds**.
- The AVG column for each row, denotes that on an average, it takes that much amount of time (in seconds) to reach from that particular row to any of the columns.

	Ascent Time (sec)	Descent Time (sec)
Delta 50	2.9	2.7867
Delta 100	3.2518	3.839
Delta 150	3.116	3.889
Delta 200	3.2322	4.482
Delta 250	3.5951	4.5675
Delta 300	3.1871	4.8052
Delta 350	3.5166	4.1366
Delta 400	3.2	4.72
Delta 450	2.9325	4.775
Delta 500	2.2667	4.2367
Delta 550	2.31	5.185
Delta 600	3.28	4.48
AVG	**3.0656**	**4.3252**
Combined	**AVG**	**3.6954**

Table 2 : Table of Delta Values.

- The term Delta, in the above table, means that the difference between the current RPM and the Reference RPM is equal to that delta value.

(2) Selection of optimal Kp:

- For the determination of an optimal Kp for the system, it is imperative to take both the ascent as well as descent readings for whatever path is chosen.
- Here, the path chosen is in terms of delta 100, for both ascent as well as descent.
- **ASCENT:**

Ascent						
Start : 150						
delta 100	**Kp**	**0.17**	**0.19**	**0.2**	**0.21**	**0.3**
	RPM					
	150->250	4.933	4.13	4.7	5.1	4.93
	250->350	3.777	4.826	4.79	5.404	4.4
	350->450	3.331	3.329	3.31	1.786	4.89
	450->550	3.109	3.073	1.77	3.362	2.69
	550->650	3.156	3.21	3.2	1.784	2.038
	650->750	2.537	0.9	0.7	2.523	0.7
	AVG	**3.4738**	**3.2446**	**3.0783**	**3.3265**	**3.2746**

Table 3 : Ascent Values of delta 100.

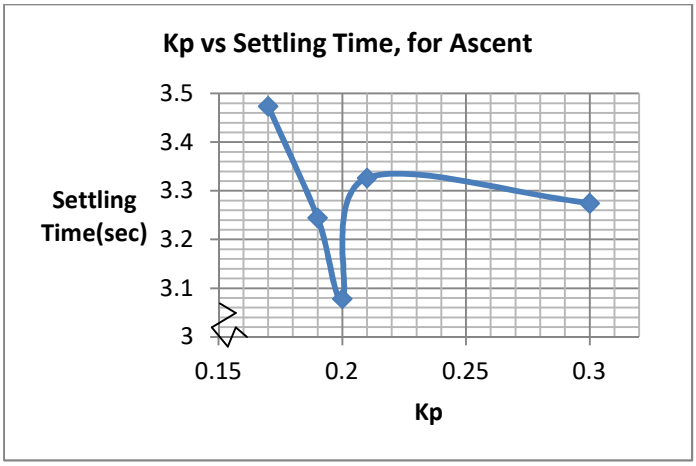


Fig. 2 : Plot of Kp versus Settling Time for Ascent Values.

- From the above graph, it is evident that as far as ascent is concerned, a **Kp** value of **0.2**, is the most optimal.
- **DESCENT :**

Descent						
Start : 750						
delta 100	Kp	0.17	0.19	0.2	0.21	0.3
	RPM					
	750->650	3.67	3.858	3.67	3.096	2.85
	650->550	2.643	2.306	2.01	1.711	1.39
	550->450	3.087	3.929	4.13	2.96	3.53
	450->350	4.33	6.64	5.58	5.45	7.46
	350->250	5.244	5.67	4.95	7.36	11.3
	250->150	9.297	4.54	4.61	6.212	5.6
	AVG	4.7118	4.4905	4.1583	4.4648	5.355

Table 4 : Descent Values of delta 100.

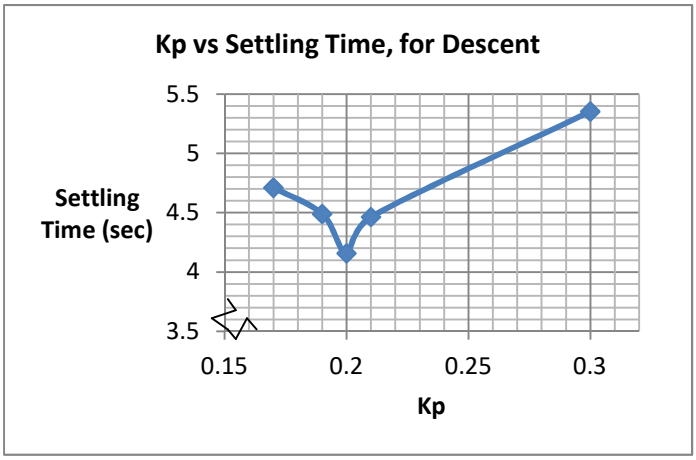


Fig. 3 : Plot of Kp versus Settling Time for Descent.

- For this system, it is observed that, in terms of descent as well, a **Kp** value of **0.2** is again found to be optimal.
- The following table is the **Kp** value versus the **average** of the corresponding ascent and descent times.

Kp	Average Time (Ascent + Descent) in (sec)
0.17	4.0928
0.19	3.8675
0.2	3.6183
0.21	3.8956
0.3	4.3148

Table 5 : Average Times for various values of Kp.

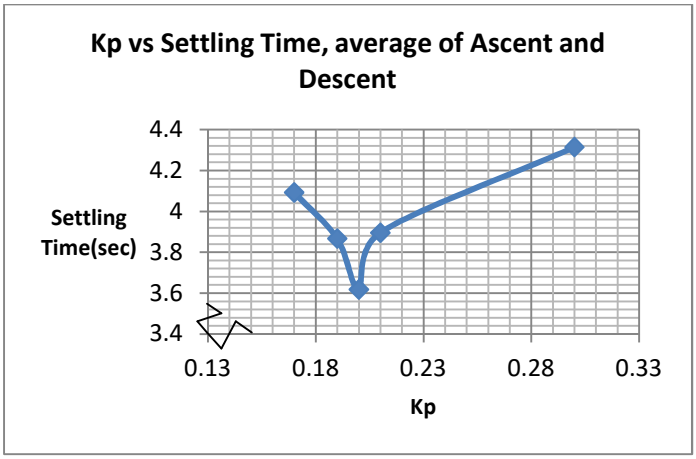


Fig. 4 : Plot of Kp versus Settling Time, for averaged Ascent and Descent values.

- The **Kp** value of **0.2** is used for this system as is evident from the above graph.

(3) Choosing the window size, for the linear Increment/Decrement part:

Start : 150							
Window Size	**150->250**	**250->350**	**350->450**	**450->550**	**550->650**	**650->750**	**Time taken for the path**
10	5.35	4.28	2.89	3.05	1.78	2.81	3.36
15	5.4	3.21	2.89	2.71	1.78	2.81	3.1333
20	3.806	3.2	2.9	1.43	2.05	2.81	2.6993
25	3.91	1.73	4.53	3.39	2.35	2.8	3.1183
30	4.52	4.25	3.29	1.44	1.78	2.81	3.015
35	3.81	4.25	3.32	2.71	2.063	6	3.6804

Table 6 : Time Taken for the path, for various values of Window Size.

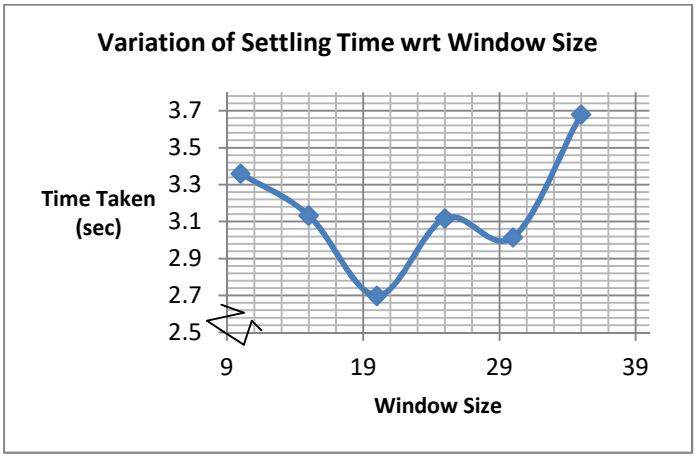


Fig. 5 : Variation of Settling Time with Window Size.

- From the graph, for the **window size** of **20**, the time taken for the specified path traversal, takes 2.858 seconds, and hence the same is chosen.

(4) Determining the number of revolutions for the calculation of the RPM:

- This data has been taken to justify the choice of the number of revolutions for the calculation of RPM.
- It is generally observed that greater the number of revolutions considered, greater is the time takes for the computation of a single RPM value, and hence greater the time for settling, but the accuracy also increases.
- Hence depending upon the type of application, the trade-off between accuracy and the time can be chosen.
- The **starting point** is the **maximum RPM** and the path taken is given below :
- **Max.RPM->750->650->550->450->350->250->150.**

No. of Revolutions(greater than)	Max.rpm ->750	750->650	650->550	550->450	450->350	350->250	250->150	Average Time	% ACCURACY
1	1.08	2.51	1.73	2.74	4.27	8.46	2.6	**3.3414**	**96.93**
2	2.38	1.27	2.23	3.92	5.11	4.87	4.3	**3.44**	**97.889**
3	1.57	3.45	1.57	4.71	6.81	5.8	5.76	**4.2385**	**98.1542**
4	2.16	3.41	3.01	5.66	6.79	3.46	5.4	**4.27**	**98.703**

Table 7 : Average Time Taken and Accuracy for various values of Revolutions Count.

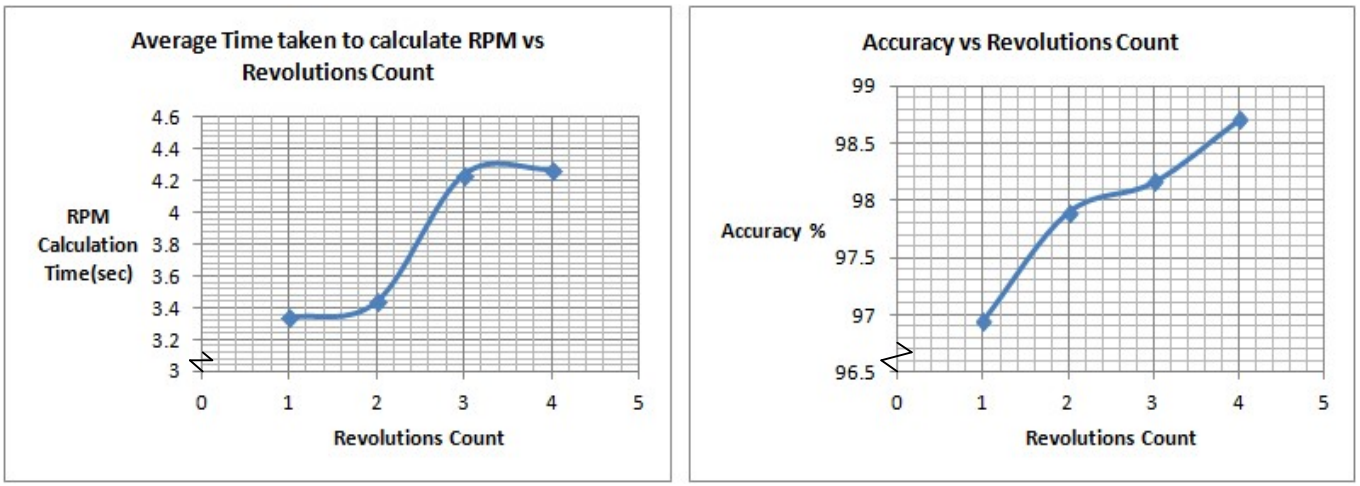


Fig. 6 : Plot of average time taken and the accuracy obtained for various values of revolutions count.

- Neglecting the extreme points, the interest is only in the two interior points, 2 and 3.
- The **choice** made here is to **go in** for a value of **2**, because, when compared to 3, the accuracy offered by the point three is at a marginal cost of accuracy, of about 0.4%.

(5) Tolerance (Closeness of Speed):

- The tolerance of the system is actually dependent on the type of application that it is going to be used in.
- For most systems the tolerance is calculated as follows :
  - **floor([Max. RPM / Max. PWM]) + 1 .**
- For this system, the **tolerance = 4.**

3. LOAD STABILIZATION:

- For this module, the methodology used is the same as that in the previous module, i.e. that of the PID controller.
- When the DC Motor is rotating at any RPM, and an external load is applied, then the system will have to increase the voltage supply to the DC Motor, by ramping up the PWM, to maintain the system at the reference RPM.
- Once the load is removed, the current RPM will overshoot the reference RPM, and hence the system needs to be brought back to the reference RPM, by decreasing the voltage supplied to the motor.

DC Motor Response Curve:

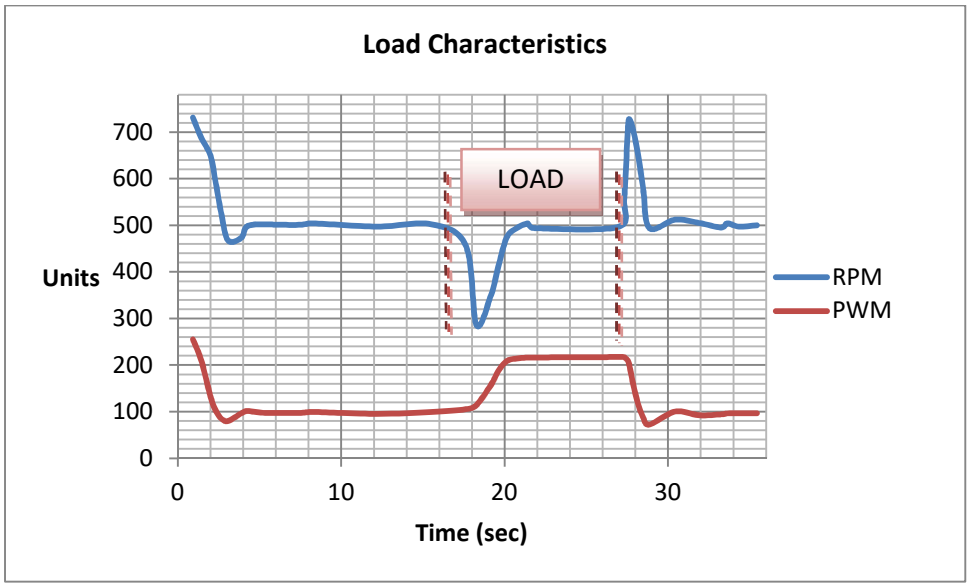


Fig. 7 : Motor Response Curve of RPM and PWM versus Time under the application of load.

4. MAX RPM. Calculation:

- In the setup function, first a PWM of 255 is given to the transistor so as to make the DC Motor run at its max RPM. And then a delay of **1.7 seconds** is given. The reason being, that the motor requires a certain amount of time, to reach its max. RPM.
- Then 5 values of the RPM are taken, with the revolution count being 4, for greater accuracy. And then these values are stored in an array of 5 elements, and the average of the 3rd, 4th and the 5th elements, leaving out the first two values, is taken.
- The reason behind leaving out the first two values is that the first two values of RPM are always off, most of the time, which has been observed on different motor.
- The average is taken to better capture the variance of the values.

4. MIN RPM. Calculation:

- The lower RPM of this DC Motor is found to be **130**.

EARLIER METHODS:

- Quite a few unsuccessful attempts made at using other strategies, some of which are listed below:

- Firstly, the thought of using a look-up table sprouted up, wherein the PWM-RPM mappings are computed beforehand, considered an overhead. Then, when the user given RPM is taken in, it is compared with the current RPM, and the PWM corresponding to the difference in the RPM values, is obtained from the look-up table. **But the problem with this approach is that, the PWM-RPM mapping is based on no load, and hence won't work when load is applied.**

- Next the thought of Machine Learning surfaced, of using a Supervised Learning Algorithm, in this case Linear Regression, because it was just PWM vs RPM, i.e. with two features, which is a pretty straightforward task. **But then, the problem with this approach is that this method will not be generic as the model would be quite specific to the motor on which it is built.**

- As the parameters of the motor might vary, this system's performance would degrade considerably. The alternative to this, would of course be to build a model on the go, in Arduino or link it to Matlab/Octave/R/Python where the model could be built, and then the hypothesis could be extracted to be used.
- All this could be unnecessary as ultimately our aim in this approach, is to fit the curve of PWM vs RPM, which could be done mathematically.

MY LEARNINGS FROM THIS PROJECT: (KEY TAKEAWAYS)

- The most important takeaway is that the hardware plays truant often, and it is as important to test hardware as testing is done for software.

- The ability to debug hardware issues comes with practice.

- More importantly, I learned about Control Systems (Open and Closed Loop System) and also about the PID Controller and the existence of the Arduino libraries.(to a basic extent)

- Prior to this, I didn't have any experience of working with Arduino, so I'll have to add that to my kit too.

- There is always a solution no matter what the circumstances, only time will reveal. This fact was accentuated, especially early on in my project phase.

- I also learnt that the solution is always discernible from the data, and most importantly that data acquisition is actually a very difficult task, learnt it the hard way.

- During the course of this project, I felt that working with hardware was not my cup of tea.

CONCLUSION:

A closed loop speed control system for a 12V DC Motor has been designed using a form of P-controller, by using the Arduino PID library, in which the average settling time of the motor is 3.57 seconds.