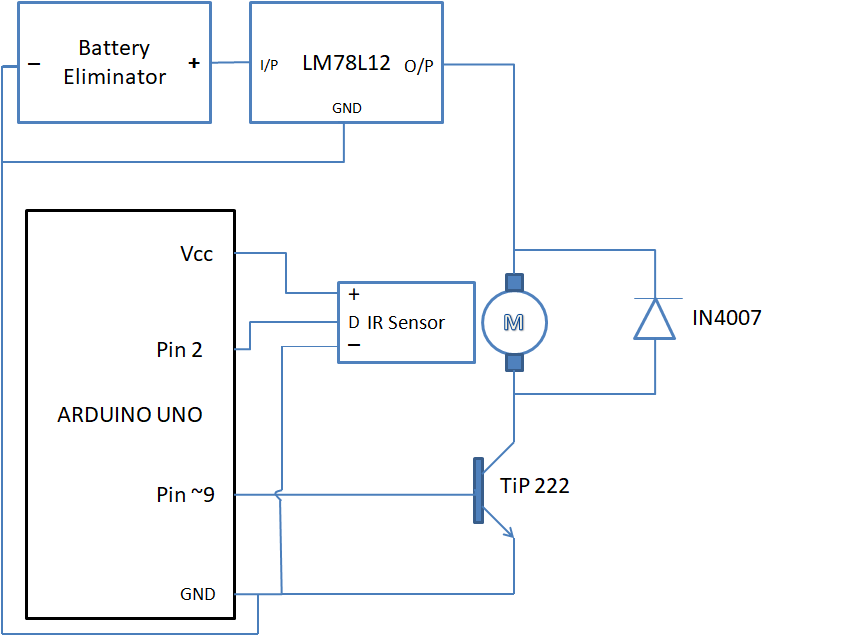
|  |
| --- |
| Motor Project |
| Year 2017-18 |
| Prithvi Suresh |

**Q:** To design a closed loop speed control system for a 12 volt DC motor.

# SETUP



* Battery eliminator: Provides DC Voltage for the motor.
* LM78L12(voltage regulator): regulates voltage from battery eliminator and provides constant voltage of 12v.
* DC motor: A motor that runs on a rated DC voltage of 12V.
* IN4007 (diode): Provides a safe path for the inductive kickback of the motor and protects the transistor.
* TiP222 (NPN transistor): Provides means of controlling voltage across the DC motor by controlling base voltage of the transistor from the arduino.
* IR SENSOR: emits IR radiation and detects the reflection of the same. Changes output signal level when the receiver detects IR radiation reflected from the motor blade.
* Arduino UNO: microcontroller board based on the ATmega328. Used to detect IR sensor output level, provide 5v for the IR sensor and provide PWM signal to control the base voltage of the transistor.

# Algorithm

1. Start
2. Calculate maxRPM, by writing max PWM and for every 3 revolutions, note the time taken, and calculate RPM. (refer section 3.1,Eqn3.1)
3. Obtain 50 such RPM values, and obtain mode of the 50 values as max RPM.
4. Get reference RPM from user.
5. Based on error between current RPM and reference RPM determine change in PWM and add to current PWM value. (refer section 3.2)
6. Write PWM value.
7. Calculate current RPM by recording the time taken between revolutions and display current RPM. (refer section 3.1,Eqn3.1)
8. Repeat steps 4 and 5 until current RPM is within tolerable range. (refer section 3.2)
9. Once motor is stable for 1 second within tolerable range of reference RPM, pause PWM correction and display current RPM values.
10. If current RPM reduces from tolerable range of reference RPM, resume pwm correction (steps 5 to 8) until current RPM is within tolerable range of reference RPM.
11. Wait for user to enter new reference RPM and repeat steps 5 to 10.

# Methodology

**3.1 RPM calculation**

* RPM is the number of revolutions the motor blade makes in a minute.
* The RPM is calculated by finding the time for a specified number of revolutions (rev count).
* This time is then used to find the RPM using the formula
  + - *  ------- EQN 3.1
      * where n is the rev count.
      * Where 60000 is the factor used to convert milliseconds to minutes.
      * And the time taken is in milliseconds.
* ***SPIKES*** are abnormal RPM values due to incorrect signals from IR receiver. A Spike is obtained when IR sensor receives IR radiation by repeated reflection from the blade of the motor. For example, if the rev count is 2 the Arduino waits for two high signals from the IR signal to calculate the RPM, by noting the time difference. Now if there is repeated reflection, the arduino receives two consecutive HIGH signals from the arduino and thus, records a time approximately equal to the time taken to complete one revolution. And hence, the calculated value would be higher than expected resulting in a spike.
* Spikes can be filtered by eliminating all RPM values above the calculated maximum RPM, because no RPM at any PWM can exceed the maximum RPM of the motor. And it is to obtain this condition that the RPM is calculated for different number of revolutions depending on the speed of the motor.
* The motor used has a maximum RPM of 800, although it is lesser in practice.
* Now the time taken for a revolution( at a particular rev count) is given by rearranging EQN 3.1

where n is the rev count. Eqn----------3.1

* So, the smallest time interval an 800 RPM motor can have is (in ms)

|  |  |  |
| --- | --- | --- |
| For rev count = 1 | For rev count = 2 | For rev count = 3 |
| 75 | 150 | 225 |

* A spike is seen when any RPM has a calculated time less than the rated time interval shown above. Therefore, for a rev count of 1 the time for one blade cut can never be less than 75 ms, and for a rev count of 2 it can never be less than 150 ms and for a rev count of 3 it can never be less than 225 ms.
* CALCULATING CUT-OFF RPM.
  + Not all spikes will be above the max RPM. There exists a cut-off RPM for a pertaining rev count, below which the spike produced will not exceed the maximum RPM, thus failing our condition.
  + For a rev count of 2, let the cutoff RPM be rc2 and its corresponding time interval be tc2.
  + rc2 and tc2 are related by the equation

https://latex.codecogs.com/gif.latex?r_%7Bc2%7D%20%3D%20%5Cfrac%7B60000%20*%201%7D%7Bt_%7Bc2%7D%7D

* + - Now if tc2 were taken for 1 blade cut but two signals, the minimum RPM produces needs to be the max RPM.
    - Thus,

https://latex.codecogs.com/gif.latex?t_%7Bc2%7D%20%3D%20%5Cfrac%7B60000*2%7D%7B800%7D

https://latex.codecogs.com/gif.latex?t_%7Bc2%7D%20%3D%20150

* + - Then,

https://latex.codecogs.com/gif.latex?r_%7Bc2%7D%3D%20%5Cfrac%7B60000%7D%7B150%7D

https://latex.codecogs.com/gif.latex?r_%7Bc2%7D%3D%20400

* + - Thus if a rev count of 2 or 3 is used for any RPM below 400, the spike will fail to be greater than max RPM. Thus a rev count of 1 is used.
  + Similarly rc3 can be found out.

` https://latex.codecogs.com/gif.latex?t_%7Bc3%7D%20%3D%20%5Cfrac%7B60000%20*%203%7D%7B800%7D

https://latex.codecogs.com/gif.latex?t_%7Bc3%7D%20%3D%20225

* + - Then,

https://latex.codecogs.com/gif.latex?r_%7Bc3%7D%3D%5Cfrac%7B60000*2%7D%7B225%7D

https://latex.codecogs.com/gif.latex?r_%7Bc3%7D%3D%20533.3333%5Capprox%20534

* + - Now, for any RPM below 534 using a rev count of 3 will not produce a spike over the maximum RPM. Although 1 or 2 can be used it is more favorable to use 2 because the fluctuations are smaller.
* As expected in the lower RPM range, spikes are observed for a rev count of 2 or 3. Also it can be seen that for 1 rev count the motor is found to be unstable at higher pm, thus a need for a dynamic rev count arises.

**3.2 Making Motor Run at Desired Speed**

* The speed control system of the motor employs a closed loop feedback system, wherein the magnitude of error (difference between reference and current RPM) is analyzed by the controller, and the PWM is altered proportionately.
* A proportional controller is used, wherein the change in PWM varied proportionally with the error. https://latex.codecogs.com/gif.latex?%5Cinline%20error%20%3D%20ref%20-%20rpm
* The change in PWM varies as https://latex.codecogs.com/gif.latex?%5Cinline%20%5CDelta%20pwm%20%3D%20K_%7Bp%7D%20*%20error , where Kp is the constant of proportionality.
* The constant of proportionality, Kp, is a constant based on motor characteristics. It is given by the expression
  + - https://latex.codecogs.com/gif.latex?%5Cinline%20K_%7Bp%7D%20%3D%20%5Cfrac%7BmaxPWM%7D%7BmaxRPM%20*%20C%7D
    - Here, the constant C is used to tune the value of Kp so as to minimize settling time and maximize accuracy.
* The value of https://latex.codecogs.com/gif.latex?%5Cinline%20%5CDelta%20pwm is then added with PWM and the new PWM is written. This results in a RPM change and the controller goes through the same steps again to determine the next value of PWM to be written.
* The motor has attained stability when the current RPM is within the tolerable range of the reference RPM. This is range is given by

https://latex.codecogs.com/gif.latex?%5Cinline%20tolerance%20%3D%20%5Cpm%20%5Cfrac%7BmaxRPM%7D%7BPWM%7D

https://latex.codecogs.com/gif.latex?%5Cinline%20tolerance%20%3D%20%5Cpm%20%5Cfrac%7B800%7D%7B255%7D

https://latex.codecogs.com/gif.latex?%5Cinline%20tolerance%20%3D%20%5Cpm%203.137254901%5Capprox%5Cpm%203

* TUNING MOTOR CONSTANT (C) :
  + Tuning the motor constant is done by observing the settling time for different values of motor constant. The value of C ranges from 1 to 2. The constant is reduced to this range because it is within this range that the motor shows maximum performance when it is made to spin from max RPM to 400 RPM. Max RPM to 400 RPM is chosen for the sake of obtaining a constant that can stabilize the motor from high RPM to low RPM with low settling time. The highest settling time is observed in any motor that spins from max RPM to low RPM.
  + 
  + From this we can eliminate values below 1.5. Our range is now in between to 1.5 to 2.
  + 
  + The settling times for C = 1.5 and 1.75 are almost similar. Thus our range is now lessened even more from 1.5 to 1.75
  + 
  + The settling time for the motor is now within the permissible 4 second limit when it is 1.62.
  + The settling time from different RPM’s is shown in the matrix below
  + 
  + The settling time is in seconds.
  + Thus the motor constant for this motor is found to be 1.62.

# 3.3 Load Stabilization

* A motor shaft is said to be loaded when an external torque is applied in a direction opposite to the direction of the motor blade.
* This results in the motor spinning at a lower RPM than it is supposed to. But the correction algorithm detects the error, thus resulting in an increase in PWM in an attempt to stabilize the motor.
* Once the shaft is unloaded, the RPM increases drastically owing to the increased PWM value. This overshoot is detected by the error correction algorithm and thus the motor stabilizes again.

The plot of the motor RPM v time is shown below:

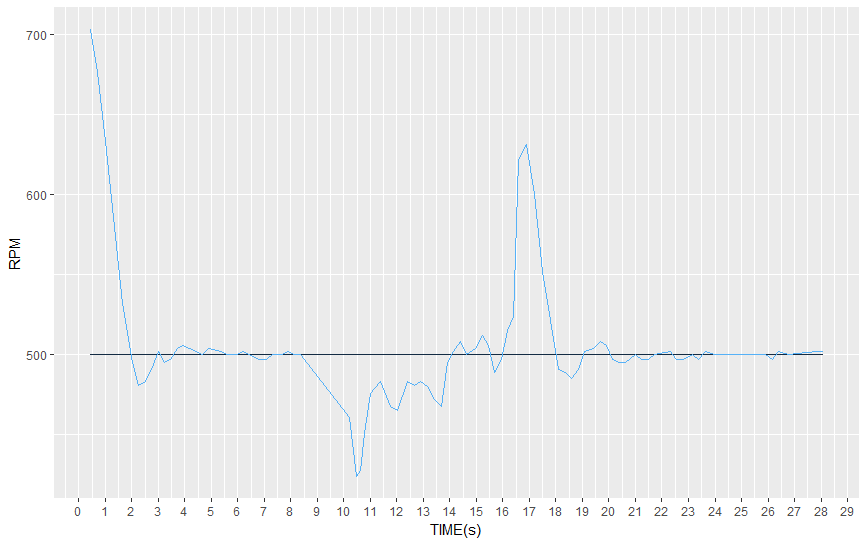


Figure 1 Load stabilization

# Learning & Experiences

* From this project, I’ve learnt how physical constraints of a hardware system can be solved using software.
* I have also learnt how obtaining the right data can support constants that might seem insignificant but are important when the hardware characteristics differ.
* Working on this project has definitely improved my reasoning skills and has taught me not to ignore even the smallest of details.

# PREVIOUS METHODS

* An attempt was made, similar to the closed loop feedback system, wherein the PWM was changed in fixed steps, regardless of the magnitude of the error between the motor RPM and the reference RPM. This led to a considerably large settling time and moreover once reference RPM was reached, the RPM began oscillating about the reference RPM and did not really stabilize.