# Speed Control of DC Motor Using PID Algorithm (STM32F4)

by tahirulhaq

hello everyone,

This is tahir ul haq with another project. This time it is STM32F407 as MC. This is an end of mid semester project. Hope you like it.

It requires a lot of concepts and theory so we go into it first.

With the advent of computers and the industrialization of processes, throughout man's history, there has always been research to develop ways to refine processes and more importantly, to control them using machines autonomously. The purpose being to reduce man's involvement in these processes thereby, reducing the error in these processes. Hence, the Field of "**Control System Engineering**" was developed.

Control System Engineering can be defined as using various methods to control the working of a process or maintenance of a constant and preferred environment, be it manual or automatic. A simple example could be of controlling the temperature in a room.

**Manual Control means the presence of a person at a site who checks the present conditions (sensor), compares it with the desired value (processing) and takes appropriate action to obtain the desired value (actuator).**

The problem with this method is that it is not very reliable as a person is prone to error or negligence in his work. Also, another problem is that the rate of the process initiated by the actuator is not always uniform, meaning sometimes it may occur faster than required or sometimes it may be slow. The solution of this problem was to use a microcontroller to control the system. The microcontroller is programmed to control the process, according to given specifications, connected in a circuit (to be discussed later), fed the desired value or conditions and thereby controls the process to maintain the desired value. The advantage of this process is that no human intervention is required in this process. Also, the rate of the process is uniform.

Before we proceed further, it is essential at this point to define various terminologies:

• **Feedback Control:** In this system, the input at a certain time is dependant on one or more variables including the output of the System.

• **Negative Feedback:** In this system, the reference (input) and the error are subtracted as feedback the and input are 180 degree out of phase.

• **Positive Feedback:** In this system, the reference (input) and the error are added as feedback the and input are in phase.

• **Error Signal:** The di erence between the desired output and the actual output.

• **Sensor:** A device used to detect a certain quantity in the circuit. It is normally placed in the output or anywhere where we want to take some measurements.

• **Processor**: The part of the Control System that performs the processing based on the algorithm programmed. It takes in some inputs and produces some outputs.

• **Actuator:** In a Control System, an actuator is used to perform an event to e ect the output based on the signal produced by the microcontroller.

• **Closed Loop System:** A System in which one or more feedback loops are present.

• **Open Loop System:** A System in which no feedback loops are present.

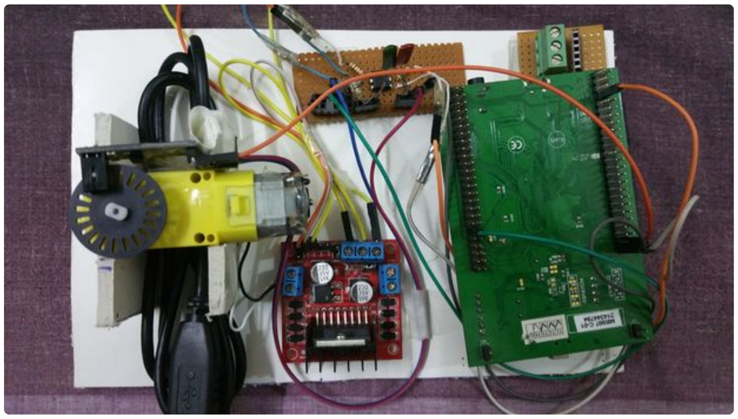
• **Rise Time:** The time taken by the output to rise from 10 percent of the maximum amplitude of the signal to 90 percent.

• **Fall Time:** The time taken by the output to fall from 90 percent to 10 percent amplitude.

• **Peak Overshoot:** Peak Overshoot is the amount by which the output exceeds its steady state value (normally during the transient response of the System).

• **Settling Time:** The time taken by the output to reach its steady state.

• **Steady State Error:** The di erence between the actual output and the desired output once the System reaches its steady state



## Step 1: Introduction to a Basic Control System

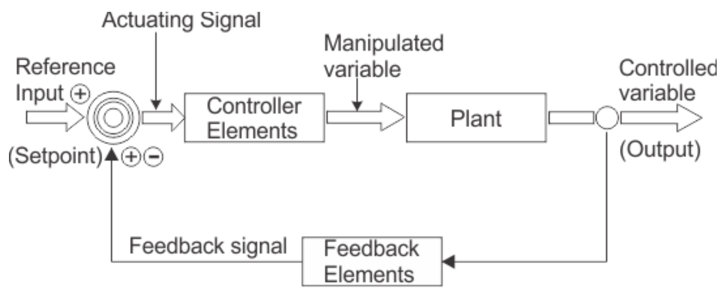
The above diagram shows a very simplified version of a Control System.

The microcontroller is at the heart of any Control System. It is a very important component therefore its choice of selection should be made carefully based on the requirements of the System.

The microcontroller receives an input from the user. This input defines the desired condition of the System. The microcontroller also receives an input from the sensor. This sensor is connected to the output, the information of which is fed back to the input. This input can also be called the negative feedback. Negative Feedback is explained earlier.

The microprocessor, based on its programming, performs various calculations and gives an output to the actuator. The actuator, based on the output, controls plant to try to maintain those conditions. An example could be a motor driver driving a motor where the motor driver is the actuator and the motor is the plant. The motor, thus rotates at a given speed.

The sensor connected reads the condition of the plant at the present time and feeds it back to the microcontroller. The microcontroller again compares, makes calculations and thus, the cycle repeats itself. This process is repetitive and endless whereby the microcontroller maintains the desired conditions.



## Step 2: Methods to Control Speed

Following are the two main methods to control the speed of dc motor

a) **Controlling the voltage manually:**

In industrial applications, it is essential to have some mechanism of speed control of DC Motor. At times, we may require a higher speed than normal or a lower speed than normal. Therefore, we need good and e cient methods of speed control. One of the crudest methods of speed control is by controlling the supply voltage. We can vary the voltage to vary the speed.

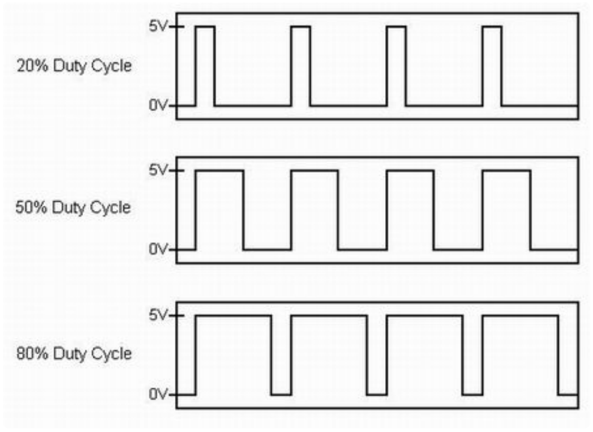
b) **Controlling the PWM using PID:**

Another more e cient method is by using a microcontroller. The DC Motor is connected to the microcontroller via a Motor Driver. The Motor Driver is an IC which receives a PWM( Pulse Width Modulation) input from the microcontroller and gives an output to the DC Motor based on the input. Figure 1.2: PWM Signal Chapter 1. Introduction 3 The working of PWM can be explained by first considering a PWM signal. It consists of continuous pulses of a certain time period. The time period is the time taken by a point to travel the distance equal to one wavelength. These pulses can only have binary values (HIGH or LOW). We also have two other quantities, namely pulse width and duty cycle. Pulse width is the time for which the output of the PWM is HIGH. The duty cycle is the percentage of the pulse width to the time period. For the rest of the time of the time period, the output is LOW. The duty cycle controls the speed of the motor directly. If in one time period, the DC Motor is provided a positive voltage for a certain time, it will move at a certain speed. If the positive voltage is provided for some more time, the

speed will be larger. So in e ect, by varying the pulse width, the duty cycle of the PWM can be varied. By varying the duty cycle of the DC Motor, the speed of the motor can be varied. Speed Control of a DC Motor

PROBLEM:

The problem with the first method of speed control was that the voltage can have variations with respect to time. These variations mean the speed is not uniform. Therefore, the first method is undesirable.



SOLUTION:

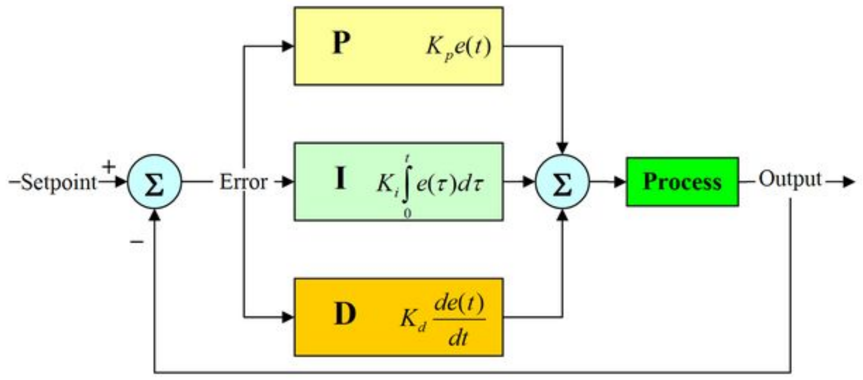
We use the second method to control the speed. We supplement the second method using a PID Algorithm.

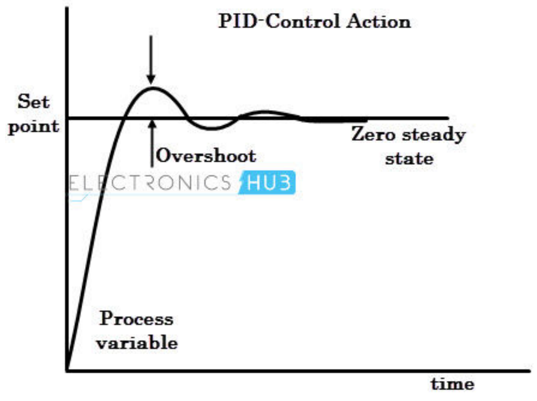
## Step 3: Intro to PID Control

PID stands for Proportional Integral Derivative.

In a PID Algorithm, the current speed of the motor is measured and compared with the desired speed. The error is used in complex calculations to vary the duty cycle of the Motor with respect to time.

This process occurs in every cycle. If the speed is more than the desired speed, the duty cycle is reduced and if the speed is less than the desired speed, the duty cycle is increased. This adjustment is made until an optimum speed is reached. This speed is continuously checked and controlled.





## Step 4: System Components

Following are the system components that were used during this project with brief details about each.

**STM 32F407:**

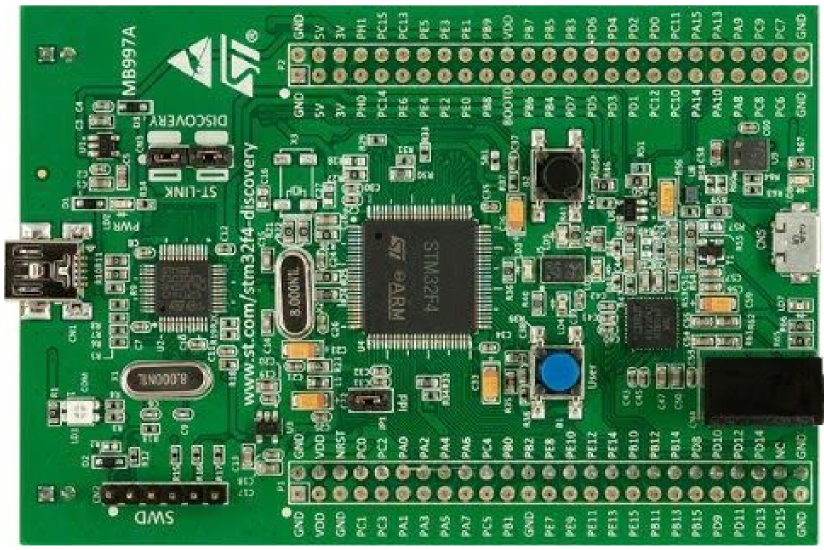
A microcontroller designed by ST Microelectronics. It works on the ARM Cortex-M Architecture. It leads its family with a high clock frequency of 168 MHz.

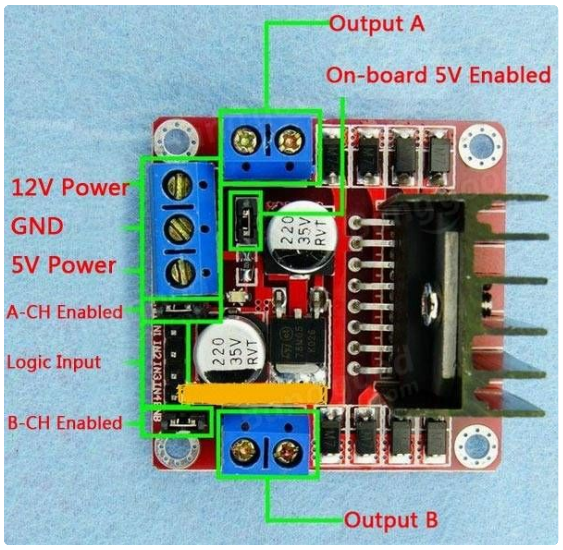
**Motor Driver L298N:** This IC is used to run the motor. It gets two external inputs. One from the microcontroller. The microcontroller supplies it a PWM signal. By adjusting the width of the pulse, the motor speed can be adjusted. Its second input is the voltage source required to drive the motor.

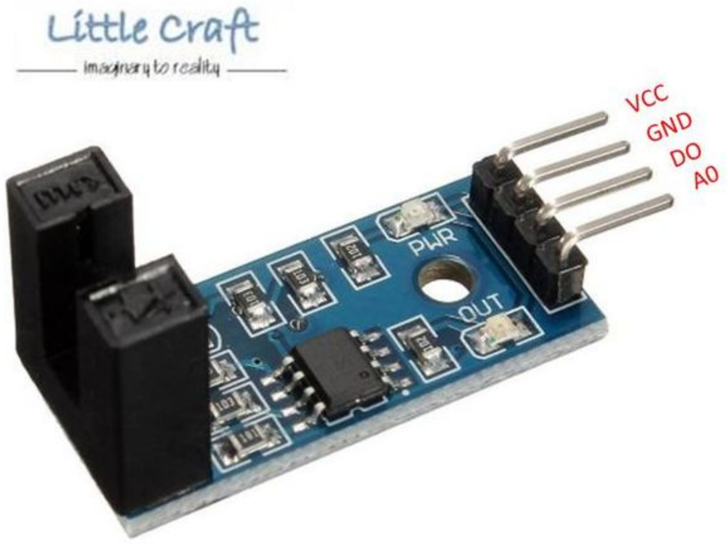
**DC Motor:** A DC Motor runs on DC supply. In this experiment, the DC Motor is run using the optocoupler connected to the motor driver.

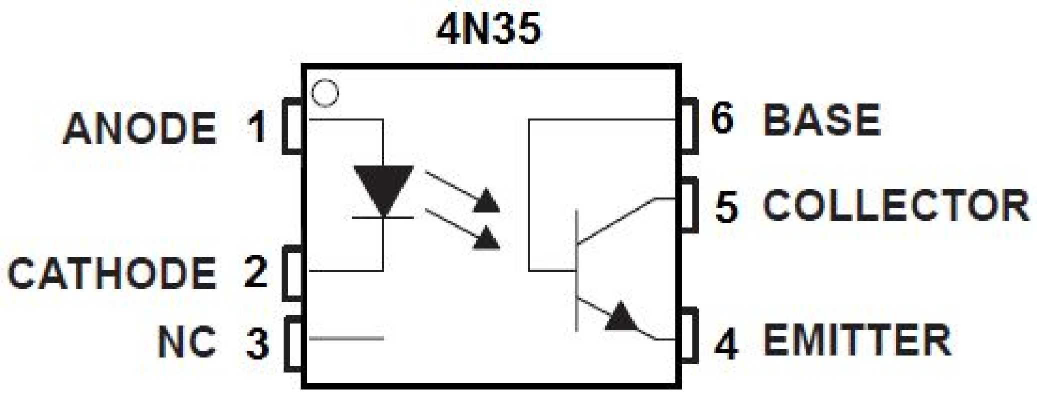
**Infrared Sensor:** An infrared sensor is actually an infrared transceiver. It sends and receives infrared waves which can be used to perform various tasks.

**IR Encoder Optocoupler 4N35:** An optocoupler is a device used to isolate the low voltage part and the high voltage part of the circuit. As the name suggests, it works on the basis of light. When the low voltage part gets a signal, current flows in the high voltage part.




Output A
On-board 5V Enabled
12V Power
GND
5V Power
A-CH Enabled
Logic Input
B-CH Enabled
Output B


Little Craft
imaginary to reality
VCC
GND
DO
A0
PWR
OUT


4N35
ANODE 1
CATHODE 2
NC 3
6 BASE
5 COLLECTOR
4 EMITTER

## Step 5: System Implementation

The system is a speed control system.The system is implemented using PID that is Proportional Integration and derivative as discussed earlier in detail.

The speed control system has the above mentioned components. The first component is the **speed sensor**. The speed sensor is an Infra red transmitter and receiver circuit.

When a solid body passes through the *u shaped slit* the sensor goes to low state. Normally it is at High state. The sensor output is connected to a ***Low pass filter to remove the debouncing***that is caused by the transients produced when the state of the sensor changes.

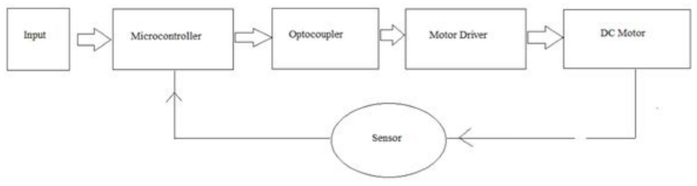
The low pass filter consists of a resistor and a capacitor. The values have been selected according to the requirements. The capacitor used is 1100nf and the resistor used is about 25 ohms. The low pass filter removes the unnecessary transients that can cause extra readings and garbage values. The low pass filter then outputs through the capacitor to the input digital pin of the stm microcontroller.

The other part is the **motor** that is controlled through the pwm supplied by the stm microcontroller. This setup has been provided an electrical isolation using an **optocoupler ic**.

The optocoupler involves a led that emits light inside the package of the ic and shorts the output terminals when given a high pulse at the input terminals. The input terminals is given pwm through a resistor that limits the current to the led that is connected to the optocoupler. At the output a pull down resistor is attached so that when the terminals short the voltage is developed at the pull down resistor and the pin attached to the upper terminal of resistor receives a HIGH state.

The output of the optocoupler is connected to the IN1 of the motor driver ic keeping the ENABLE pin HIGH. As the pwm duty cycle changes at the input of the optocoupler the motor driver IN1 pin toggles the motor and the speed of the motor is controlled.

After the pwm supplied to the motor the motor driver is supplied a voltage normally 12volts. The motor driver then enables the motor to run.



## Step 6: Coding

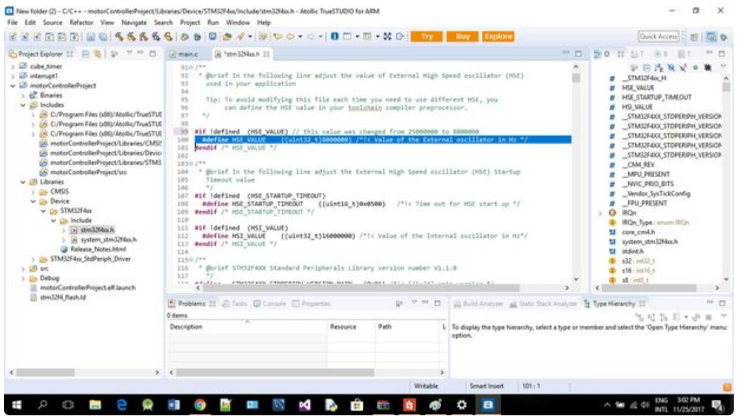
Now coming to the algorithm we used in the implementation of this motor speed control project.

The pwm to the motor is supplied by a single timer. The configurations of the timer are made and it is set to provide pwm. When the motor starts it rotates the slit attached to the shaft of the motor. The slits pass through the sensor cavity and a low pulse is generated.

At a low pulse the code starts and waits for the slit to move away. As soon as the slit goes away a high state is provided by the sensor and the timer starts to count. The timer gives us the time between two slits. Now when another low pulse comes, the **IF statement** executes again waits for the next rising edge and stops the counter. After the speed is calculated the di erence of the speed to the actual reference value is calculated and is given to the pid. The pid calculated the value of the duty to attain the reference value at the given instant. This value is provided to the **CCR(compare register)** of the timer and the speed reduces or increases depending on the error.

The code has been implemented on Atollic Truestudio. You might need to install STM studio for debugging. The project is imported in STM studio and the variables to view are imported.

A slight change is made in the stm32f4xx.h file to change the clock frequency to 168MHz exactly. The snippet has been provided above.



```
/**
 * @brief In the following line adjust the value of External High Speed oscillator (HSE)
   used in your application

   Tip: To avoid modifying this file each time you need to use different HSE, you
        can define the HSE value in your toolchain compiler preprocessor.
  */

#if !defined  (HSE_VALUE) // this value was changed from 25000000 to 8000000
  #define HSE_VALUE    ((uint32_t)8000000) /*!< Value of the External oscillator in Hz */
#endif /* HSE_VALUE */

/**
 * @brief In the following line adjust the External High Speed oscillator (HSE) Startup
   Timeout value
   */
#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    ((uint16_t)0x0500)   /*!< Time out for HSE start up */
#endif /* HSE_STARTUP_TIMEOUT */

#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the Internal oscillator in Hz*/
#endif /* HSI_VALUE */

/**
 * @brief STM32F4XX Standard Peripherals Library version number V1.1.0
   */
```

http://www.instructable…

Download (https://cdn.instructables.com/ORIG/FDW/110O/JACTVU31/FDW110OJACTVU31.txt)

(https://cdn.instructables.com/ORIG/FDW/110O/JACTVU31/FDW110OJACTVU31.txt)

## Step 7: Conclusion

As a conclusion, the speed of the motor was controlled using the PID. The curve however is not exactly a smooth straight line. There are many reasons for that:

• The sensor though connected to a low pass filter still provides certain finite debounces, these are due to the non linear resistances and some inevitable reasons of the analog electronics

• The motor does not rotate smoothly under small voltage or pwm. It provides jerks that might cause some wrong values fed to the system.

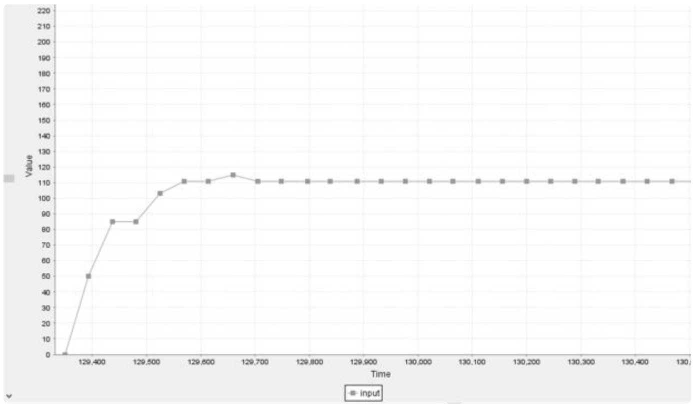
• Due to wobbling the sensor might miss some slits providing higher values

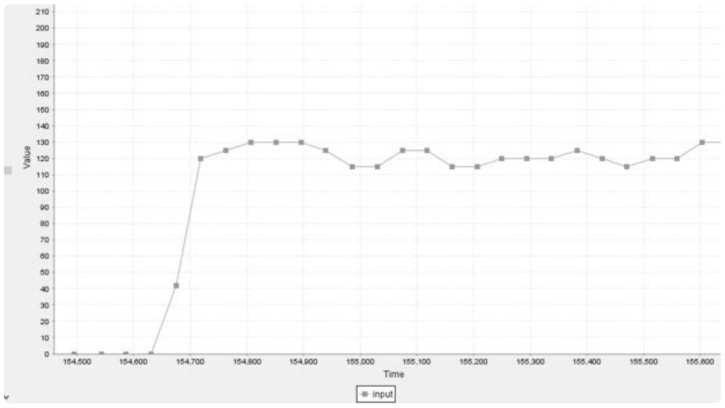
• Another major reason for errors can be the core clock frequency of the stm. This model of stm provides a core clock of 168MHz. Though dealt with this problem in this project, there is an overall notion about this model that it does not exactly provide such high frequency.

The open loop speed provides a very smooth line with only a few unexpected values. The PID is also working fine providing a very low settle time of the motor. The motor PID was tested under various voltages keeping the reference speed constant. The voltage change does not change the speed of the motor showing that the PID is working fine. Following are some of the snippets of the final output of the PID.

a) closed loop@110rpm

b) closed loop@120rpm





## Step 8: Special Thanks

This project could not be completed without the help of my group members. I would like to thank them.

Thanking you for viewing the project. Hope it helps you. Stay tuned for more. Till then stay blessed :)

Tahir Ul haq

University of Engineering and Technology UET

Lahore Pakistan