***Working of PID controller with Arduino***

**Introduction**:-

A proportional–integral–derivative controller (PID controller) is a control loop feedback mechanism commonly used in industrial control systems. A PID controller continuously calculates an error value e(t) as the difference between a desired setpoint and a measured process variable and applies a correction based on proportional, integral, and derivative terms (sometimes denoted P, I, and D respectively) which give their name to the controller type.  
  
In today project we will use this mechanism to control two brushless motor in order to calibrate our model. We will put the motors on a balance and calculate the angle using the MPU6050. So in our case the value that we will control is the inclination angle of our model. The e(t) error will be the difference between the right angle of the model and the desired one. The desired one will be 0, which means that the model is perfectly horizontal.

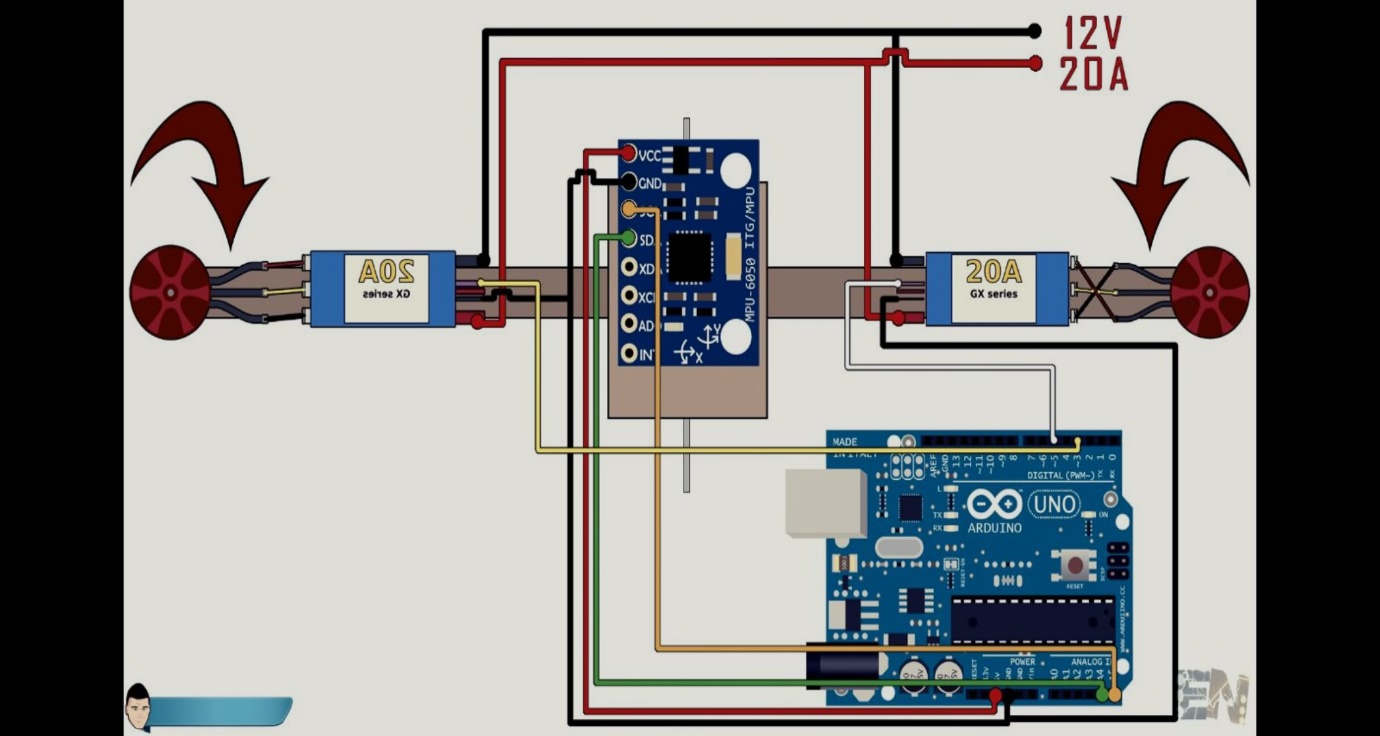
**Part1. Building the balance of arm:-**

So let's say that you have already build the model from the [build your Arduino model](http://electronoobs.com/eng_robotica_tut5.php) project. But in that project, we have use the multiwii platform for our flight controller. Now we want to make our own code to control the model. But the model won't fly steady unless we control each motor in a very precise way. That's why we have to learn how the PID control works. Each brushless motor will have a different power for the same PWM signal because motors are not perfect and they will never be the same. For that we have to constantly measure the angle of our model, compare that value with the desired value and rectify the error if there is one.  
  
So to tune our motors we will first build a balance for just 2 motors. These two motors represent just one axis, in this case will be the y-axis. As you know a model can move in any of the 3D axis, x, y and z. All we want to do with this balance is to find our P, I and D constants. Each of this 3 constants will affect in one way or other the entire PID control, and we have to find the perfect ones.

This is how we will proceed. For the balance we will need:

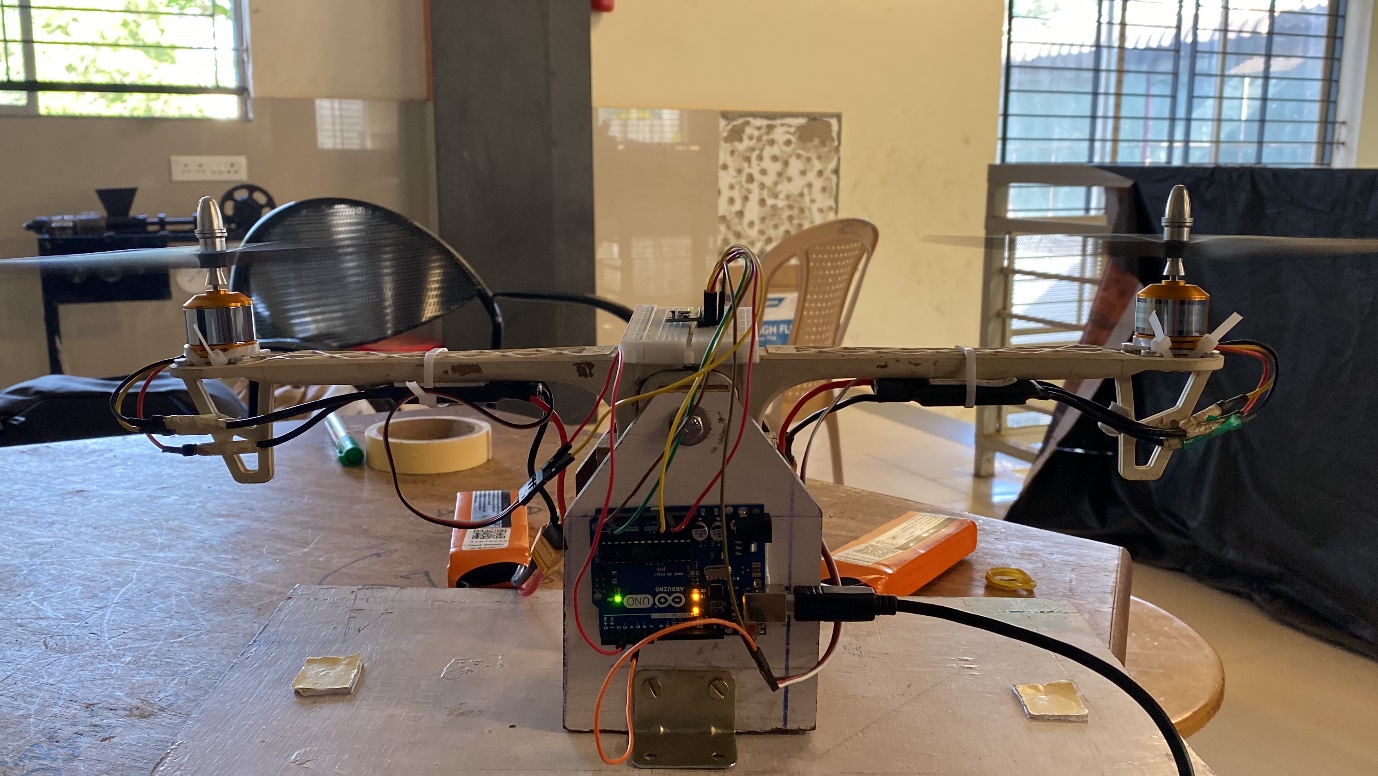
1. Nut Bolt Clamps
2. Plywood piece
3. Radial Ball bearing
4. Model arm
5. M-seal

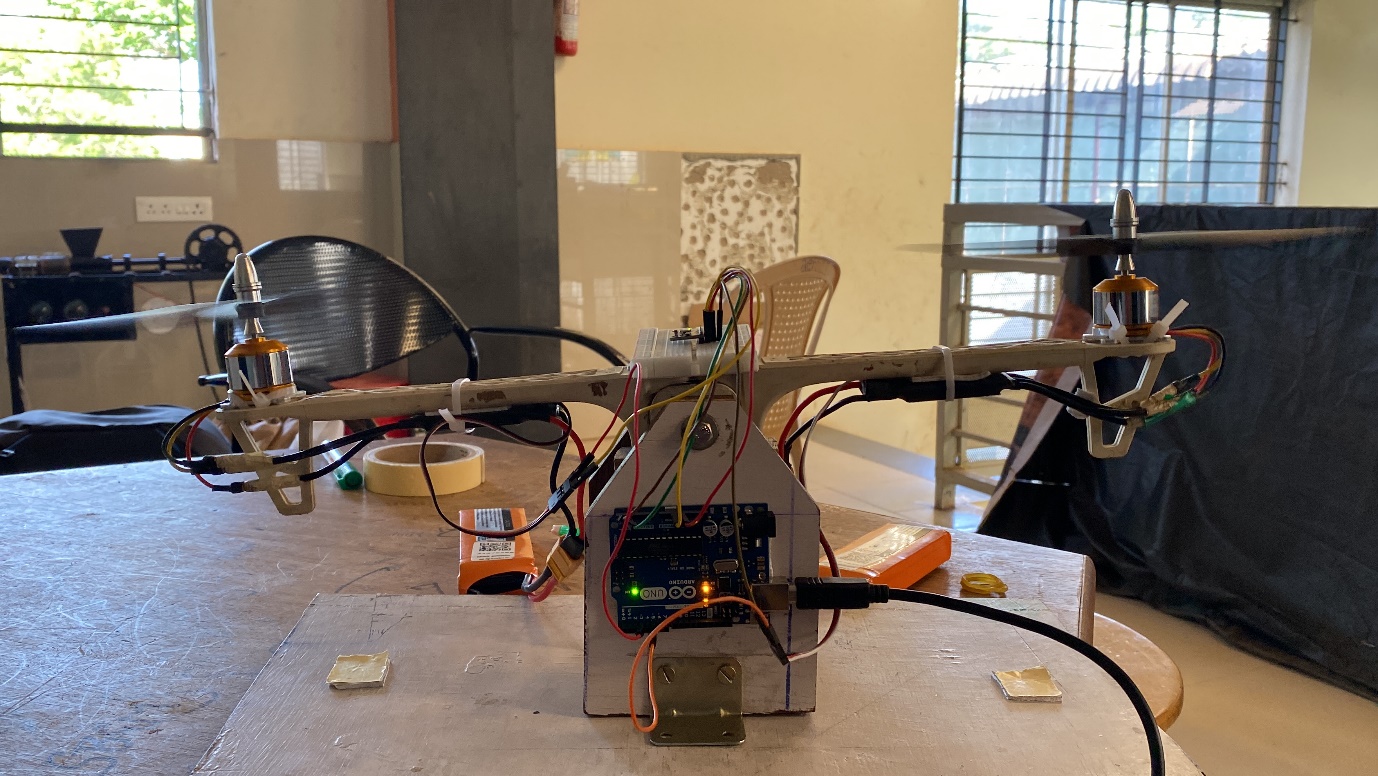
**Part2. Circuit schematic: -**



First of all place the brushless motors in the end of each side of the balance. Make sure that each of the motor is spinning in the direction shown in the schematic below. Also that the screw of the propeller is getting tighten in the opposite direction as thew rotation, otherwise the propeller will fly away during the tests. If the motor is not spinning in the desired direction, just reverse to top and bottom wires from the esc, leave the middle as it is. Make sure that the ESCs are calibrated and that both have the same scale. In this case this this ESCs both have a scale from 1000us to 2000us. use that code to calibrate the ESCs. This is important. If the motors don't have the same scale, this system won't work.  
  
Now we should have the schematic above. Supply 12V to the ESCs. From the ESCs BEC supply 5V to the arduino or use the USB to supply it if you want. Connect the IMU using the i2c connection pins SCL and SDA. In the schematic below you can see with better details the mpu6050 connection to the arduino. Then connect pin 3 to the right ESC and pin 5 to the left one. Also connect ground between the arduino and the ESCs.

Place the MPU6050 module as centered as you can on the balance. We have used a breadboard to make all the connections. Also we connected the i2c cables around a GND cable. This will reduce noise of the i2c communication. For that get two wires of GND from the arduino to the MPU6050 and wire around each of this the SCL and SDA cables of the i2c. Now that everything is ready to go we can start with the code.





LIST OF COMPONENTS

|  |  |  |
| --- | --- | --- |
| SL No. | Name of the components | Quantity |
| 1 | BLDC MOTOR 1000 KV | 2 |
| 2 | 40A ESC | 2 |
| 3 | LIPO BATTERY 3300mAH | 2 |
| 4 | MPU6050 SENSOR | 1 |
| 5 | ARDUINO UNO | 1 |
| 6 | BREAD BOARD | 1 |
| 7 | JUMPER WIRE | 20 |
| 8 | LEAD SCREW 8mm | 1 |
| 9 | PROPELLER 8X45R | 2 |
| 10 | NUT BOLT CLAMPS |  |
| 11 | PLYWOOD 40X20 |  |
| 12 | RADIAL BALL BEARING | 1 |
| 13 | MODEL ARM | 2 |

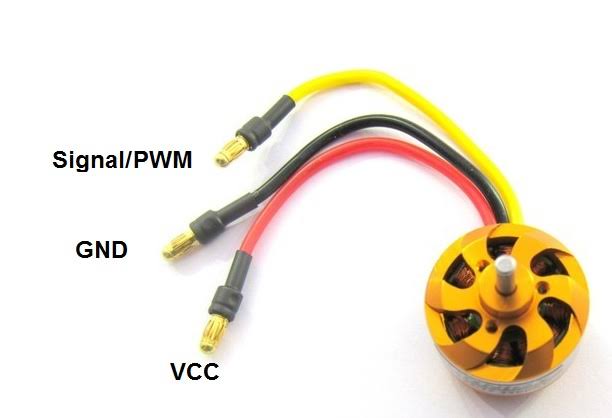
DESCRIPTION OF COMPONENTS

1. BLDC MOTOR 1000 KV :

BLDC motor works on the principle similar to that of a[Brushed DC motor](https://robu.in/product-category/motors-drivers-actuators/dc-motor/). The Lorentz force law states that whenever a current carrying conductor is placed in a magnetic field it experiences a force. As a consequence of reaction force, the magnet will experience an equal and opposite force. In the BLDC motor, the current carrying conductor is stationary and the permanent magnet is moving.

When the stator coils get a supply from source, it becomes electromagnet and starts producing the uniform field in the air gap. Though the source of supply is DC, switching produces an AC voltage waveform with trapezoidal shape. Due to the force of interaction between electromagnet stator and permanent magnet rotor, the rotor continues to rotate.

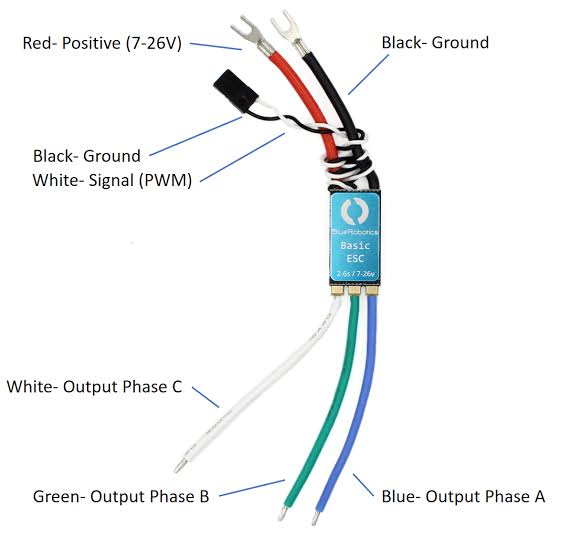
With the switching of windings as High and Low signals, corresponding winding energised as North and South poles. The permanent magnet rotor with North and South poles allign with stator poles which causes the motor to rotate.



1. 40A ESC

An electronic speed control follows a speed reference signal (derived from a throttle lever, joystick, or other manual input) and varies the switching rate of a network of [field effect transistors](https://en.wikipedia.org/wiki/Field_effect_transistor) (FETs). By [adjusting the duty cycle](https://en.wikipedia.org/wiki/Pulse-width_modulation) switching frequency of the transistors, the speed of the motor is changed. The rapid switching of the current flowing through the motor is what causes the motor itself to emit its characteristic high-pitched whine, especially noticeable at lower speeds.

Different types of speed controls are required for [brushed DC motors](https://en.wikipedia.org/wiki/Brushed_DC_motor) and [brushless DC motors](https://en.wikipedia.org/wiki/Brushless_DC_motor). A brushed motor can have its speed controlled by varying the voltage on its armature. A brushless motor requires a different operating principle. The speed of the motor is varied by adjusting the timing of pulses of current delivered to the several windings of the motor.



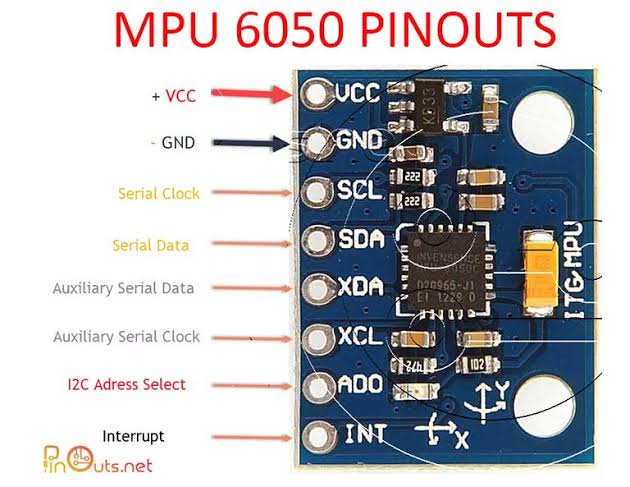
3) LIPO BATTERY 3300mAH

* The 3300mAh Lithium Polymer battery has a voltage of 11.1V.
* The LiPo battery is integrated with a JST-XH balance plug and T discharge plug.
* It is designed to have a Max Continuous Discharge of 35C and Max Burst Discharge of 70C.
* The 11.1V 3300mAh LiPo battery packs are equipped with heavy duty discharge leads that is highly essential to minimize the resistance and sustain high current loads.
* They help in powering Quadcopter, DIY Models, RC Cars, Health and Fitness devices and many more.

4) MPU6050 SENSOR

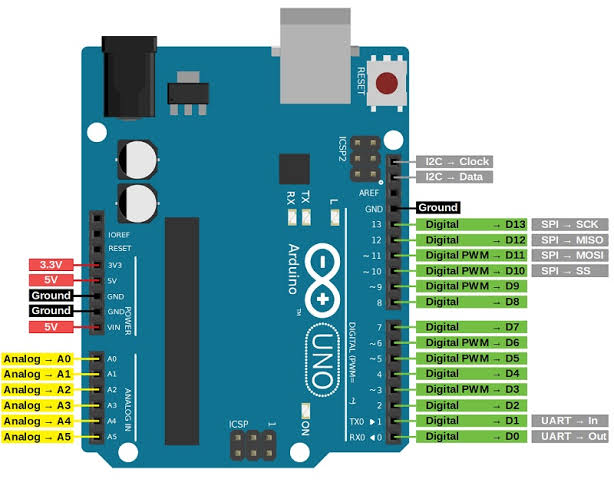
MPU6050 is a MEMS-based 6-axis motion tracking device. It has an on-chip gyroscope and [accelerometer](https://www.elprocus.com/accelerometer-sensor-working-and-applications/) sensors along with [temperature sensor](https://www.elprocus.com/temperature-sensors-applications/). MPU6050 is a digital device. This module is of very small in size, has low power consumption requirements, highly accurate, has high repeatability, high shock tolerance, it has application-specific performance programmability and low consumer price points. MPU6050 can be easily interfaced with other sensors such as [magnetometers](https://www.elprocus.com/magnetometers-types-applications/) and microcontrollers.

* MPU6050 has a 3-axis gyroscope, 3- axis Accelerometer and a Digital motion processor integrated on a single chip.
* It works on the power supply of 3V-5V.
* MPU6050 uses the I2C protocol for communication and transfer of data.
* This module has a built-in 16-bit ADC which provides great accuracy.



5) ARDUINO UNO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. [Arduino boards](https://www.arduino.cc/en/Main/Products) are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the [Arduino programming language](https://www.arduino.cc/en/Reference/HomePage) (based on [Wiring](http://wiring.org.co/)), and [the Arduino Software (IDE)](https://www.arduino.cc/en/Main/Software), based on [Processing](https://processing.org/). Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of [accessible knowledge](http://forum.arduino.cc/) that can be of great help to novices and experts alike. Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.



6) BREAD BOARD

Breadboards are designed to work with through-hole electronic components. These components have long metal leads that are designed to be inserted through holes in a printed circuit board (PCB) that are plated with a thin copper coating, which allows the components' leads to be soldered to the board.

7) JUMPER WIRE

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with [breadboards](https://blog.sparkfuneducation.com/what-is-a-breadboard) and other prototyping tools in order to make it easy to change a circuit as needed.

8) LEAD SCREW 8mm

A leadscrew (or lead screw), also known as a power screw or translation screw, is a screw that is used as a linkage in a machine to translate rotary motion into linear motion. Because of the large sliding contact area between their male and female elements, screw threads have greater frictional energy losses compared to other linkages.

9) PROPELLER 8X45R

Propellers are devices that transform rotary motion into linear thrust. Model propellers provide lift for the aircraft by spinning and creating an airflow, which results in a pressure difference between the top and bottom surfaces of the propeller. This accelerates a mass of air in one direction, providing lift which counteracts the force of gravity.

Propellers for multirotor models such as hexacopter, octocopter and quadcopter propellers, are arranged in pairs, spinning either clockwise or anti-clockwise to create a balance. Varying the speed of these propellers allows the model to hover, ascend, descend, or affect its yaw, pitch and roll.

10)NUT BOLT CLAMPS

Help in fixing of various components .

11)PLYWOOD 40X20

Used to provide support to arm of the model and also acts as base.

12) Radial Ball Bearing

Rolling resistance (friction) is considerably less than sliding resistance. Radial ball bearings consist of a specific number of balls contained within a cavity called a raceway that is formed by two radiused ball races. Although the vast majority of ball bearings have balls made of carbon steel, other materials, such as stainless steel, ceramic or glass are also available. Grease or oil reduces the friction and has a dampening effect on the contact between the balls and the smooth walls of the raceways, and can be contained within the bearing by the use of rubber seals. The bearings may use a metal-faced shield instead of a seal, though the shield does not provide positive sealing of the bearing. The seals or shields also function to keep dirt and dust from affecting bearing rotation. When under load, the bearings settle into the deepest points on the raceways and the load is transferred at the contact points between the raceways and balls. The load can now spin without exerting torque on the shaft.

13) MODEL ARM

Provides support to motors by holding them at the ends and acts as supporting structure.

***OVER ALL WORKING OF THE PROJECT***

**SUMMERY:-**

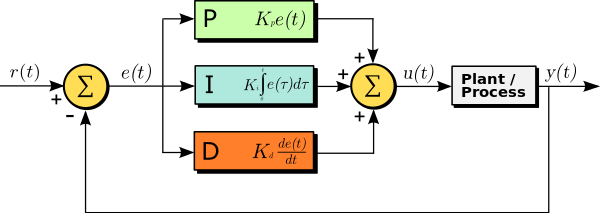
Main motive to consider PID tuning project is that it has industrial control applications to regulate temperature, flow, pressure, speed, stabilization in autonomous system and many more.

A **proportional–integral–derivative controller** (**PID controller** or **three-term controller**) is a [control loop](https://en.wikipedia.org/wiki/Control_loop) mechanism employing [feedback](https://en.wikipedia.org/wiki/Feedback) that is widely used in [industrial control systems](https://en.wikipedia.org/wiki/Industrial_control_system) and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an *error value* {\displaystyle e(t)} as the difference between a desired [setpoint](https://en.wikipedia.org/wiki/Setpoint_(control_system)) (SP) and a measured [process variable](https://en.wikipedia.org/wiki/Process_variable) (PV) and applies a correction based on [proportional](https://en.wikipedia.org/wiki/Proportional_control), [integral](https://en.wikipedia.org/wiki/Integral), and [derivative](https://en.wikipedia.org/wiki/Derivative) terms (denoted *P*, *I*, and *D* respectively), hence the name.

In practical terms, PID automatically applies an accurate and responsive correction to a control function. An everyday example is the [cruise control](https://en.wikipedia.org/wiki/Cruise_control) on a car, where ascending a hill would lower speed if constant engine power were applied. The controller's PID algorithm restores the measured speed to the desired speed with minimal delay and overshoot by increasing the power output of the engine in a controlled manner.

The first theoretical analysis and practical application of PID was in the field of automatic steering systems for ships, developed from the early 1920s onwards. It was then used for automatic process control in the manufacturing industry, where it was widely implemented in at first pneumatic and then electronic [controllers](https://en.wikipedia.org/wiki/Controller_(control_theory)). Today the PID concept is used universally in applications requiring accurate and optimized automatic control.

The distinguishing feature of the PID controller is the ability to use the three *control terms* of proportional, integral and derivative influence on the controller output to apply accurate and optimal control. The block diagram on the right shows the principles of how these terms are generated and applied. It shows a PID controller, which continuously calculates an *error value* {\displaystyle e(t)} as the difference between a desired [setpoint](https://en.wikipedia.org/wiki/Setpoint_(control_system)) {\displaystyle {\text{SP}}=r(t)} and a measured [process variable](https://en.wikipedia.org/wiki/Process_variable) {\displaystyle {\text{PV}}=y(t)}{\displaystyle e(t)=r(t)-y(t)}, and applies a correction based on [proportional](https://en.wikipedia.org/wiki/Proportional_control), [integral](https://en.wikipedia.org/wiki/Integral), and [derivative](https://en.wikipedia.org/wiki/Derivative) terms. The controller attempts to minimize the error over time by adjustment of a *control variable* {\displaystyle u(t)}, such as the opening of a [control valve](https://en.wikipedia.org/wiki/Control_valve), to a new value determined by a [weighted sum](https://en.wikipedia.org/wiki/Weighted_sum) of the control terms.



In this model:

* Term **P** is proportional to the current value of the SP − PV error {\displaystyle e(t)}. For example, if the error is large and positive, the control output will be proportionately large and positive, taking into account the gain factor "K". Using proportional control alone will generally result in an error between the setpoint and the actual process value because it requires an error to generate the proportional response. The controller cannot adjust the system unless there is an error present.
* Term **I** accounts for past values of the SP − PV error and integrates them over time to produce the **I** term. For example, if there is a residual SP − PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.
* Term **D** is a best estimate of the future trend of the SP − PV error, based on its current rate of change. It is sometimes called "anticipatory control", as it is effectively seeking to reduce the effect of the SP − PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or damping effect.

**Tuning** – The balance of these effects is achieved by [loop tuning](https://en.wikipedia.org/wiki/PID_controller#Loop_tuning) to produce the optimal control function. The tuning constants are shown below as "K" and must be derived for each control application, as they depend on the response characteristics of the complete loop external to the controller. These are dependent on the behaviour of the measuring sensor, the final control element (such as a control valve), any control signal delays and the process itself. Approximate values of constants can usually be initially entered knowing the type of application, but they are normally refined, or tuned, by "bumping" the process in practice by introducing a setpoint change and observing the system response.

**Control action** – The mathematical model and practical loop above both use a *direct* control action for all the terms, which means an increasing positive error results in an increasing positive control output correction. The system is called *reverse* acting if it is necessary to apply negative corrective action. For instance, if the valve in the flow loop was 100–0% valve opening for 0–100% control output – meaning that the controller action has to be reversed. Some process control schemes and final control elements require this reverse action. An example would be a valve for cooling water, where the [fail-safe](https://en.wikipedia.org/wiki/Fail-safe) mode, in the case of loss of signal, would be 100% opening of the valve; therefore 0% controller output needs to cause 100% valve opening.

{\displaystyle u(t)=K\_{\text{p}}\left(e(t)+{\frac {1}{T\_{\text{i}}}}\int \_{0}^{t}e(\tau )\,\mathrm {d} \tau +T\_{\text{d}}{\frac {\mathrm {d} e(t)}{\mathrm {d} t}}\right),}

### Selective use of control terms:-

Although a PID controller has three control terms, some applications need only one or two terms to provide appropriate control. This is achieved by setting the unused parameters to zero and is called a PI, PD, P or I controller in the absence of the other control actions. PI controllers are fairly common in applications where derivative action would be sensitive to measurement noise, but the integral term is often needed for the system to reach its target value.

**Applicability:-**

The use of the PID algorithm does not guarantee [optimal control](https://en.wikipedia.org/wiki/Optimal_control) of the system or its [control stability](https://en.wikipedia.org/wiki/Nyquist_stability_criterion) *(see*[*§ Limitations*](https://en.wikipedia.org/wiki/PID_controller#Limitations)*, below)*. Situations may occur where there are excessive delays: the measurement of the process value is delayed, or the control action does not apply quickly enough. In these cases [lead–lag compensation](https://en.wikipedia.org/wiki/Lead%E2%80%93lag_compensator) is required to be effective. The response of the controller can be described in terms of its responsiveness to an error, the degree to which the system [overshoots](https://en.wikipedia.org/wiki/Overshoot_(signal)) a setpoint, and the degree of any system [oscillation](https://en.wikipedia.org/wiki/Oscillation). But the PID controller is broadly applicable since it relies only on the response of the measured process variable, not on knowledge or a model of the underlying process.

### Industrial control:-

The wide use of feedback controllers did not become feasible until the development of wideband high-gain amplifiers to use the concept of negative feedback. This had been developed in telephone engineering electronics by [Harold Black](https://en.wikipedia.org/wiki/Harold_Black) in the late 1920s, but not published until 1934. Independently, Clesson E Mason of the Foxboro Company in 1930 invented a wide-band pneumatic controller by combining the [nozzle and flapper](https://en.wikipedia.org/wiki/Nozzle_and_flapper) high-gain pneumatic amplifier, which had been invented in 1914, with negative feedback from the controller output. This dramatically increased the linear range of operation of the nozzle and flapper amplifier, and integral control could also be added by the use of a precision bleed valve and a bellows generating the integral term. The result was the "Stabilised" controller which gave both proportional and integral functions using feedback bellows. The integral term was called *Reset*. Later the derivative term was added by a further bellows and adjustable orifice.

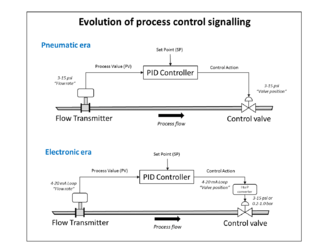
From about 1932 onwards, the use of wideband pneumatic controllers increased rapidly in a variety of control applications. Air pressure was used for generating the controller output, and also for powering process modulating devices such as diaphragm-operated [control valves](https://en.wikipedia.org/wiki/Control_valve). They were simple low maintenance devices that operated well in harsh industrial environments and did not present explosion risks in [hazardous locations](https://en.wikipedia.org/wiki/Electrical_equipment_in_hazardous_areas). They were the industry standard for many decades until the advent of discrete electronic controllers and [distributed control systems](https://en.wikipedia.org/wiki/Distributed_control_system).

With these controllers, a pneumatic industry signaling standard of 3–15 psi (0.2–1.0 bar) was established, which had an elevated zero to ensure devices were working within their linear characteristic and represented the control range of 0-100%.

In the 1950s, when high gain electronic amplifiers became cheap and reliable, electronic PID controllers became popular, and the pneumatic standard was emulated by 10-50 mA and 4–20 mA [current loop](https://en.wikipedia.org/wiki/Current_loop) signals (the latter became the industry standard). Pneumatic field actuators are still widely used because of the advantages of pneumatic energy for [control valves](https://en.wikipedia.org/wiki/Control_valves) in process plant environments.

### Electronic analog controllers:-

Electronic analog PID control loops were often found within more complex electronic systems, for example, the head positioning of a [disk drive](https://en.wikipedia.org/wiki/Disk_drive), the power conditioning of a [power supply](https://en.wikipedia.org/wiki/Power_supply), or even the movement-detection circuit of a modern [seismometer](https://en.wikipedia.org/wiki/Seismometer). Discrete electronic analog controllers have been largely replaced by digital controllers using [microcontrollers](https://en.wikipedia.org/wiki/Microcontrollers) or [FPGAs](https://en.wikipedia.org/wiki/FPGA) to implement PID algorithms. However, discrete analog PID controllers are still used in niche applications requiring high-bandwidth and low-noise performance, such as laser-diode controllers.



### Response to disturbances:-

If a controller starts from a stable state with zero error (PV = SP), then further changes by the controller will be in response to changes in other measured or unmeasured inputs to the process that affect the process, and hence the PV. Variables that affect the process other than the MV are known as disturbances. Generally, controllers are used to reject disturbances and to implement setpoint changes. A change in load on the arm constitutes a disturbance to the robot arm control process.

### Applications:-

In theory, a controller can be used to control any process that has a measurable output (PV), a known ideal value for that output (SP), and an input to the process (MV) that will affect the relevant PV. Controllers are used in industry to regulate [temperature](https://en.wikipedia.org/wiki/Temperature), [pressure](https://en.wikipedia.org/wiki/Pressure), [force](https://en.wikipedia.org/wiki/Force), [feed rate](https://en.wikipedia.org/wiki/Feed_rate), [flow rate](https://en.wikipedia.org/wiki/Volumetric_flow_rate), chemical composition (component [concentrations](https://en.wikipedia.org/wiki/Concentration)), [weight](https://en.wikipedia.org/wiki/Weight), [position](https://en.wikipedia.org/wiki/Position_(vector)), [speed](https://en.wikipedia.org/wiki/Speed), and practically every other variable for which a measurement exists.

## **Loop tuning:-**

*Tuning* a control loop is the adjustment of its control parameters (proportional band/gain, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. Stability (no unbounded oscillation) is a basic requirement, but beyond that, different systems have different behaviour, different applications have different requirements, and requirements may conflict with one another.

Even though there are only three parameters and it is simple to describe in principle, PID tuning is a difficult problem because it must satisfy complex criteria within the [limitations of PID control](https://en.wikipedia.org/wiki/PID_controller#Limitations_of_PID_control). Accordingly, there are various methods for loop tuning, and more sophisticated techniques are the subject of patents; this section describes some traditional, manual methods for loop tuning.

Designing and tuning a PID controller appears to be conceptually intuitive, but can be hard in practice, if multiple (and often conflicting) objectives, such as short transient and high stability, are to be achieved. PID controllers often provide acceptable control using default tunings, but performance can generally be improved by careful tuning, and performance may be unacceptable with poor tuning. Usually, initial designs need to be adjusted repeatedly through computer simulations until the closed-loop system performs or compromises as desired.

Some processes have a degree of [nonlinearity](https://en.wikipedia.org/wiki/Nonlinear_system), so parameters that work well at full-load conditions do not work when the process is starting up from no load. This can be corrected by [gain scheduling](https://en.wikipedia.org/wiki/Gain_scheduling) (using different parameters in different operating regions).

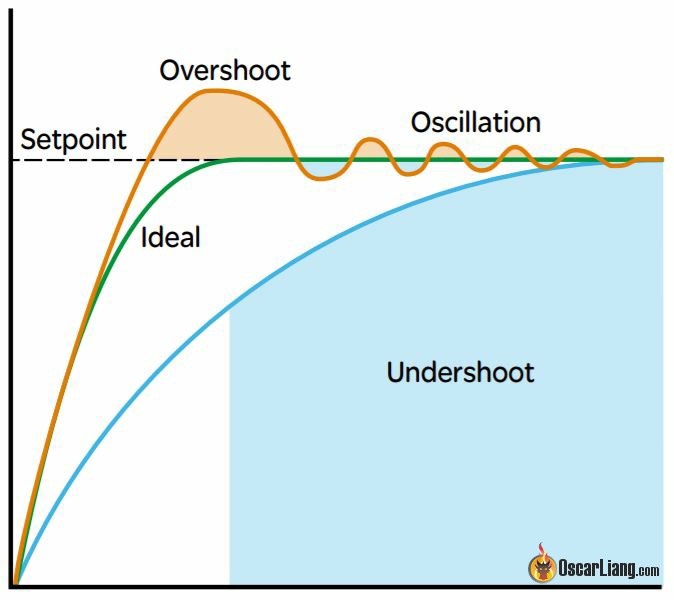
### Stability:-

If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable; i.e., its output [diverges](https://en.wikipedia.org/wiki/Divergence_(computer_science)), with or without [oscillation](https://en.wikipedia.org/wiki/Oscillation), and is limited only by saturation or mechanical breakage. Instability is caused by *excess* gain, particularly in the presence of significant lag.

Generally, stabilization of response is required and the process must not oscillate for any combination of process conditions and setpoints, though sometimes [marginal stability](https://en.wikipedia.org/wiki/Marginal_stability) (bounded oscillation) is acceptable or desired.

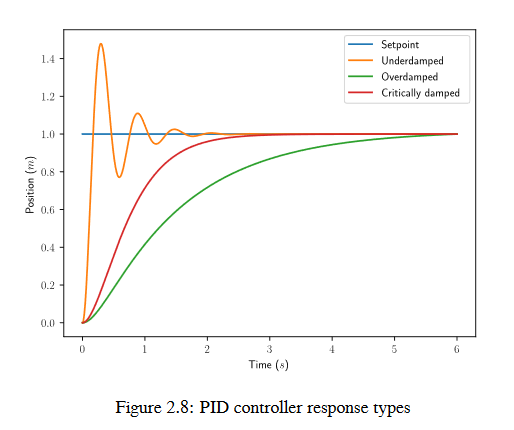
### Manual tuning:-

If the system must remain online, one tuning method is to first set {\displaystyle K\_{i}}Ki and {\displaystyle K\_{d}}Kd values to zero. Increase the {\displaystyle K\_{p}}Kp until the output of the loop oscillates; then set {\displaystyle K\_{p}}Kp to approximately half that value for a "quarter amplitude decay"-type response. Then increase {\displaystyle K\_{i}}Ki until any offset is corrected in sufficient time for the process, but not until too great a value causes instability. Finally, increase {\displaystyle K\_{d}}Kd, if required, until the loop is acceptably quick to reach its reference after a load disturbance. Too much {\displaystyle K\_{d}}Kd causes excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an [overdamped](https://en.wikipedia.org/wiki/Overdamped) closed-loop system is required, which in turn requires a {\displaystyle K\_{p}}Kp setting significantly less than half that of the {\displaystyle K\_{p}}Kp setting that was causing oscillation.



**Effects of increasing a parameter independently**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Rise time** | **Overshoot** | **Settling time** | **Steady state error** | **Stability** |
| **Kp** | **Decrease** | **Increase** | **Small change** | **Decrease** | **Degrade** |
| **Ki** | **Decrease** | **Increase** | **Increase** | **Eliminate** | **Degrade** |
| **Kd** | **Minor change** | **Decrease** | **Decrease** | **No change in theory** | **Improve if Kd is small** |

****

## **Limitations:-**

While PID controllers are applicable to many control problems, and often perform satisfactorily without any improvements or only coarse tuning, they can perform poorly in some applications and do not in general provide [*optimal* control](https://en.wikipedia.org/wiki/Optimal_control). The fundamental difficulty with PID control is that it is a feedback control system, with *constant* parameters, and no direct knowledge of the process, and thus overall performance is reactive and a compromise. While PID control is the best controller in an [observer](https://en.wikipedia.org/wiki/State_observer) without a model of the process, better performance can be obtained by overtly modelling the actor of the process without resorting to an observer.

PID controllers, when used alone, can give poor performance when the PID loop gains must be reduced so that the control system does not overshoot, oscillate or [hunt](https://en.wikipedia.org/wiki/Hunting_oscillation) about the control setpoint value. They also have difficulties in the presence of non-linearities, may trade-off regulation versus response time, do not react to changing process behaviour (say, the process changes after it has warmed up), and have lag in responding to large disturbances.

The most significant improvement is to incorporate [feed-forward control](https://en.wikipedia.org/wiki/Feed-forward_control) with knowledge about the system, and using the PID only to control error. Alternatively, PIDs can be modified in more minor ways, such as by changing the parameters (either [gain scheduling](https://en.wikipedia.org/wiki/Gain_scheduling) in different use cases or adaptively modifying them based on performance), improving measurement (higher sampling rate, precision, and accuracy, and low-pass filtering if necessary), or cascading multiple PID controllers.