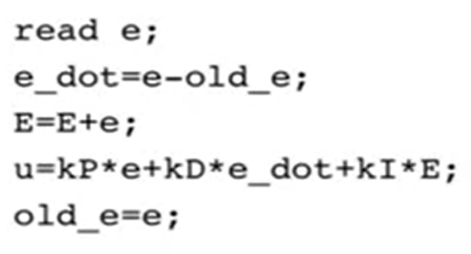
Implementing PID in Arduino and Tuning the Parameters.

Implementing PID in Arduino.

In cycle bot, using PID controller we will control the PWM signal for the speed of reaction wheel to balance the bot. In PID controller it has one INPUT and one OUTPUT. So, the output of the controller will be PWM signal and in input to the controller is error. Now, the where is this error come from.

Error is the difference between the angle measured by the sensors and the desired angle. We used this algorithm to implement PID in arduino given below :-



```
read e;
e_dot=e-old_e;
E=E+e;
u=kP*e+kD*e_dot+kI*E;
old_e=e;
```

Figure 1: Reference of image:- control of mobile robots, GRITS, coursera

where,

$$e = error,$$

$$old_e = previouserror,$$

$$E = integratederror,$$

$$e_{dot} = derivativeerror,$$

$$K_p = proportionalgain,$$

$$K_d = derivativegain,$$

$$K_i = integralgain$$

$$u = PWMsignal.$$

One thing in that algo is that we are not included sampling time($delta_t$) neither on integral part nor on derivative part because it a constant value and can be compensated by choosing the correct gain value for both integral and derivative regulator.

Tuning the Parameters.

There are various methods available you can choose any of them, the condition is your controller should be optimal, stable and robust. We are using two methods to tune the parameters of PID gains i.e **Heuristic A Method and Zeigler-Nichols Method** . You can choose any of the them which give accurate results that you want.

Heuristic A Method

The method consists of the following steps:

1. Set all gains to zero.
2. Increase the K_p until the response to a disturbance is steady oscillation.
3. Increase the T_d until the oscillations stop (i.e. it's critically damped).
4. Repeat steps 2 and 3 until increasing the T_d does not stop the oscillations.
5. Set K_p and T_d to the last stable values.
6. Increase T_i gain until the convergence to the set point occurs with or without overshoot at an acceptable rate.

Zeigler-Nichols Method

The basic tuning steps are:

1. Set all gains to zero.
2. Increase the K_p until the response to a disturbance is steady oscillation. This is called the 'ultimate' gain K_u.
3. Measure the 'ultimate' oscillation period T_u at this steady state.

K_u and T_u can then be used to calculate values for K_p, T_i and T_d, depending on the type of control algorithm implemented, according to the table below.

TABLE TO CALCULATE THE PID GAIN

Control Type	K_p	T_i	T_d
P	$0.5K_u$	–	–
PI	$0.45K_u$	$T_u/1.2$	–
PD	$0.8K_u$	–	$T_u/8$
Classic PID	$0.6K_u$	$T_u/2$	$T_u/8$
Pessen Integral Rule	$0.7K_u$	$T_u/2.5$	$3T_u/20$
Some overshoot	$0.33K_u$	$T_u/2$	$T_u/3$
No overshoot	$0.2K_u$	$T_u/2$	$T_u/3$

Table 1: Reference

K_u and T_u were determined in Step 2 in the previous method . Calculating it all out in a table above provides all the coefficients for the different test settings.