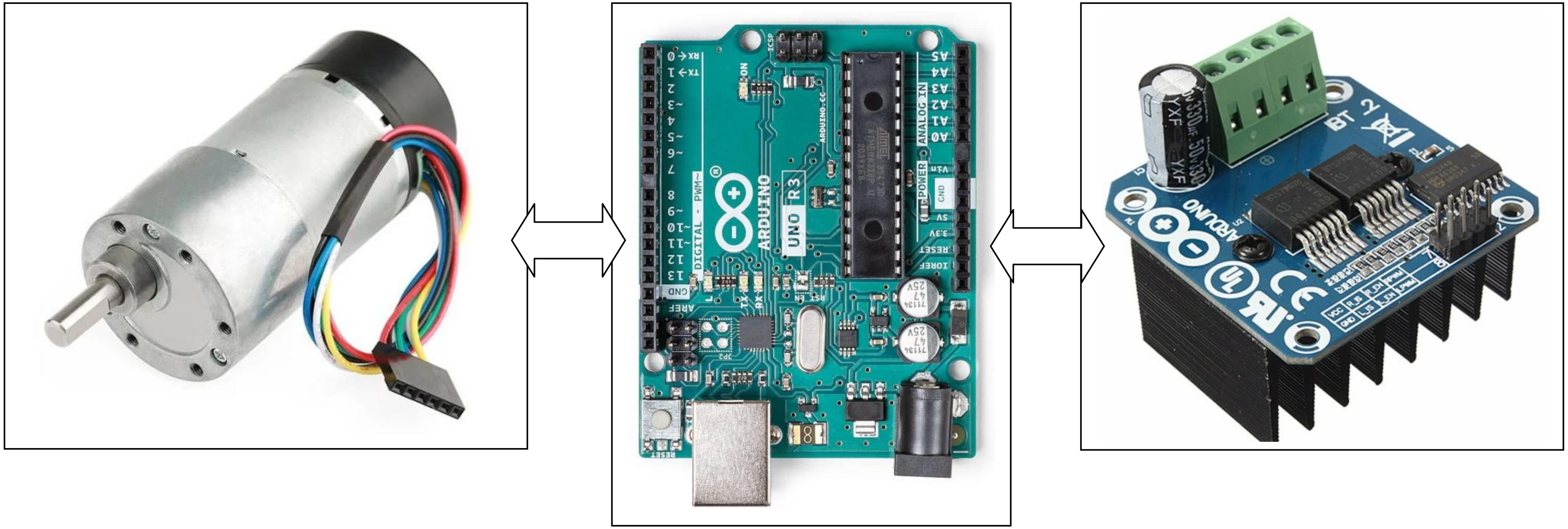
Target

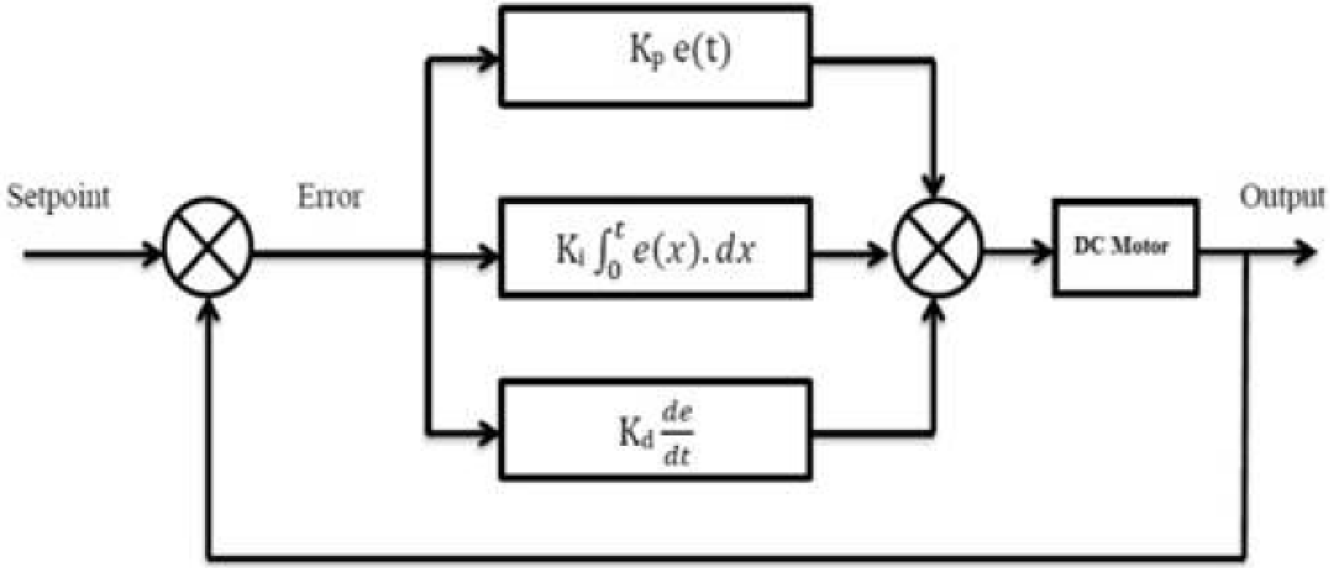
Keep the motor speed constant.

The speed can vary for several reasons, for example:

- reduction of the motor supply voltage (battery)
- increase of the resistant torque to the shaft



PID Controller Design



```
//speed error
e_speed = set_speed - v_speed;   // error speeed
// calculate voltage power for DC motor with P.I.D.
//         proportional      integral          derivative
pwm_pulse = kp * e_speed    +  ki * e_speed_sum  + kd * (e_speed - e_speed_pre)/ deltaT;
// integral error
e_speed_sum += (e_speed * deltaT); //sum of error --> integral
//save last (previous) error for derivate
e_speed_pre = e_speed;
```

```
#include <util/atomic.h>

// Pins for BTD7960 Motor Driver
#define ENCA 2  // decoder A
#define ENCB 4  // decoder B
#define IN1 5   // PWM 1
#define IN2 6   // PWM 2

// Counters for milliseconds during interval
long previousMillis = 0;
long currentMillis = 0;

// globals time var
int pos = 0;
long prevT = 0;
int posPrev = 0;

// Use the "volatile" directive for variables used in an interrupt
volatile int pos_i = 0;
volatile float velocity_i = 0;
volatile long prevT_i = 0;

// Filtered velocity
float v1Filt = 0;
float v1Prev = 0;

//SERIAL INPUT SETUPS
String inputString = "";         // a string to hold incoming data
String Pin;
int iPin;
String State;
boolean stringComplete = false;  // whether the string is complete
long startTime ;                 // start time for stop watch
long elapsedTime ;

//PID variables
double set_speed = 50;    // setpoint to 30 rpm
double v_speed = 0;       // actual speed
double e_speed = 0;       //error of speed = set_speed - v_speed
double e_speed_pre = 0;   //last error of speed
double e_speed_sum = 0;   //sum error of speed
double pwm_pulse = 0;     //this value is 0~255

double kp = 5;
double ki = 20;
double kd = 0.1;

void setup() {
  Serial.begin(9600);

  // Setup  BTD7960 Motor Driver
  pinMode(ENCA,INPUT);
  pinMode(ENCB,INPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING);
}

void loop() {
    // read the position in an atomic block to avoid potential misreads
    ATOMIC_BLOCK(ATOMIC_RESTORESTATE){ pos = pos_i; }

    // Compute velocity DC motor
    long currT = micros();
    float deltaT = ((float) (currT-prevT))/1.0e6;
    float velocity1 = abs((pos - posPrev)/deltaT);
    posPrev = pos;
    prevT = currT;

    // Convert count/s to RPM
    float v1 = velocity1/600.0*60.0;
    // Low-pass filter (25 Hz cutoff)
    v1Filt = 0.854*v1Filt + 0.0728*v1 + 0.0728*v1Prev;
    v1Prev = v1;
    v_speed = v1Filt;  // actual speed
    //PID code
    e_speed = set_speed - v_speed;   // error speeed
    // calculate voltage power for DC motor with P.I.D.
    //         proportional       integral           derivative
    pwm_pulse = kp * e_speed    +  ki * e_speed_sum  + kd * (e_speed - e_speed_pre)/ deltaT;
    e_speed_sum += (e_speed * deltaT); //sum of error --> integral
    e_speed_pre = e_speed;  //save last (previous) error

    // set limit to sum of error (integral)
    if (e_speed_sum >100) {e_speed_sum = 100; }
    else if (e_speed_sum <-100) {e_speed_sum = -100; }

    // set PWM limits
    if(pwm_pulse > 255)    {  pwm_pulse = 255; }
    else if(pwm_pulse < 0) {  pwm_pulse = 0;   }

    // set DC motor speed
    setMotor(pwm_pulse,IN1,IN2);

  // print data
  Serial.print(set_speed); Serial.print(" "); Serial.print(v1Filt); Serial.print(" "); Serial.print(pwm_pulse); Serial.println();

  // check for new setup rpm non serial -> 1=rpm
  CheckSerial();

  delay(10);
}

// SerialEvent occurs whenever a new data comes in the  hardware serial RX.
void serialEvent() {
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}


void CheckSerial(){
  // if Newline arrived on SERIAL
  if (stringComplete) {
    //Serial.println(inputString);

    int id = inputString.indexOf("=");
    if (id>0) {
      Pin = inputString.substring(0, id) ;
      State= inputString.substring(id+1, inputString.length() - id+1);
      iPin= State.toInt();

      // rotation
      if (iPin>=0 && iPin < 255) {
          if (Pin== "1") {
            //Serial.println("DC" + Pin + "=" + State);
            //analogWrite(IN1, iPin);
            //analogWrite(IN2, 0);
            set_speed = iPin;
          }
          else if (Pin== "2") {
            //Serial.println("DC" + Pin + "=" + State);
            //analogWrite(IN1, iPin);
            //analogWrite(IN2, 0);
          }
      }
      else {
            //Serial.println("error " + inputString);
            // STOP DC motor
            analogWrite(IN1, 0);
            analogWrite(IN2, 0);
      }
    }
    // clear the input string:
  inputString = "";
  stringComplete = false;
 }
}

void setMotor(int pwmVal, int in1, int in2){
  analogWrite(in1,pwmVal);
  analogWrite(in2,LOW);
}

void readEncoder(){
 // Read encoder B when ENCA rises
 int b = digitalRead(ENCB);
 int increment = 0;
 if(b>0){
  // If B is high, increment forward
  increment = 1;
 }
 else{
  // Otherwise, increment backward
  increment = -1;
 }
 pos_i = pos_i + increment;

 // Compute velocity with method 2
 long currT = micros();
 float deltaT = ((float) (currT - prevT_i))/1.0e6;
 velocity_i = abs(increment/deltaT);
 prevT_i = currT;
}
```