تطبيقات ميكاترونيك -1-

Lecture No. 8

- Interrupts with Arduino
- Control of DC motor with Encoder
- PID controller

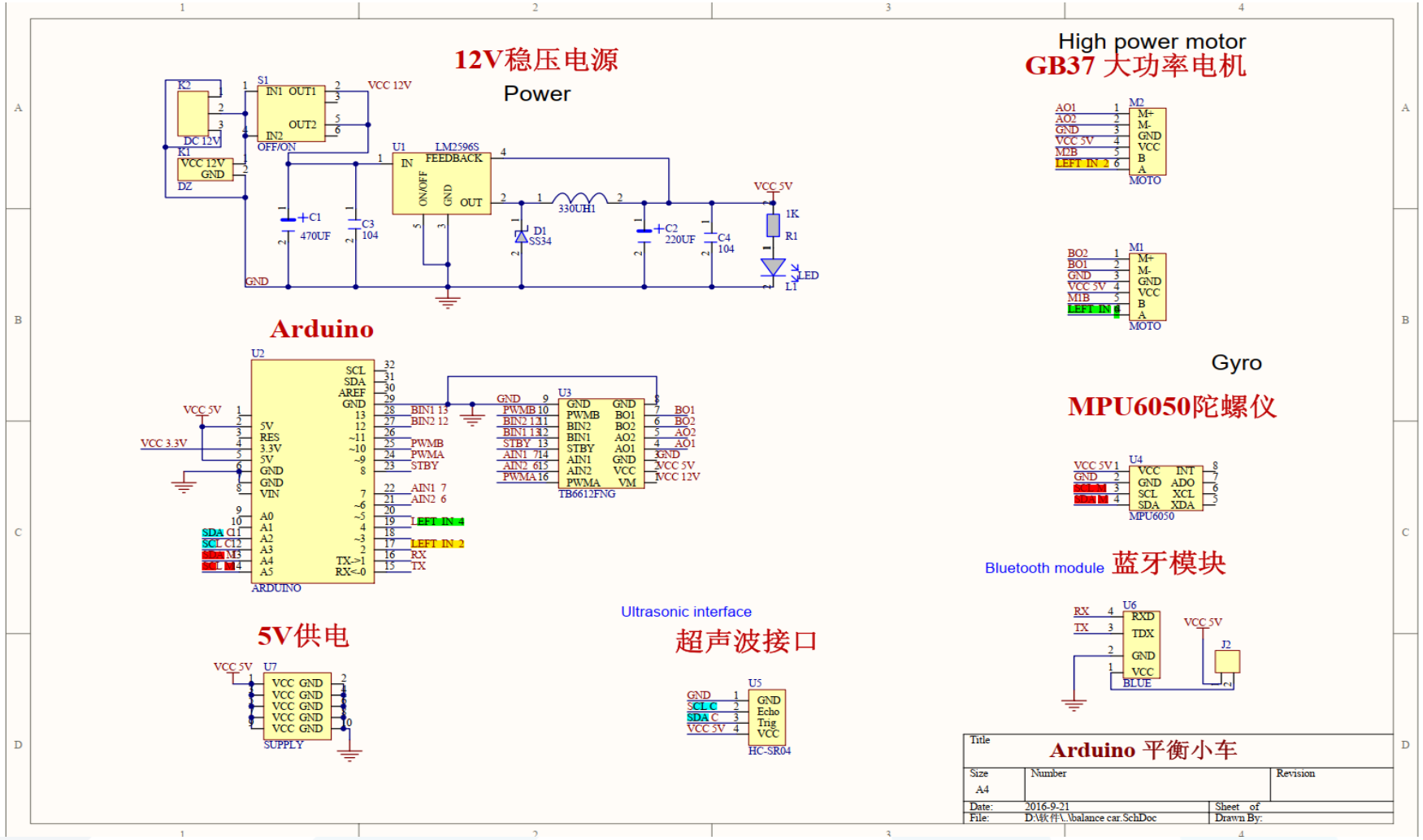
روبوت و أنظمة ذكية - سنة ثالثة

Dr. Eng. Essa Alghannam
Ph.D. Degree in Mechatronics Engineering

2023-2024

الانضمام الى CLASSROOM عبر الرابط التالي

https://classroom.google.com/c/NjM5MDM5NDkzNzM2?cjc=6xp3ajs



GB37 DC gear motor

جهاز استشعار حساس هال بـ13خط ترميز مغناطيسي وإخراج ثنائي الطور.

Rated voltage: 12V

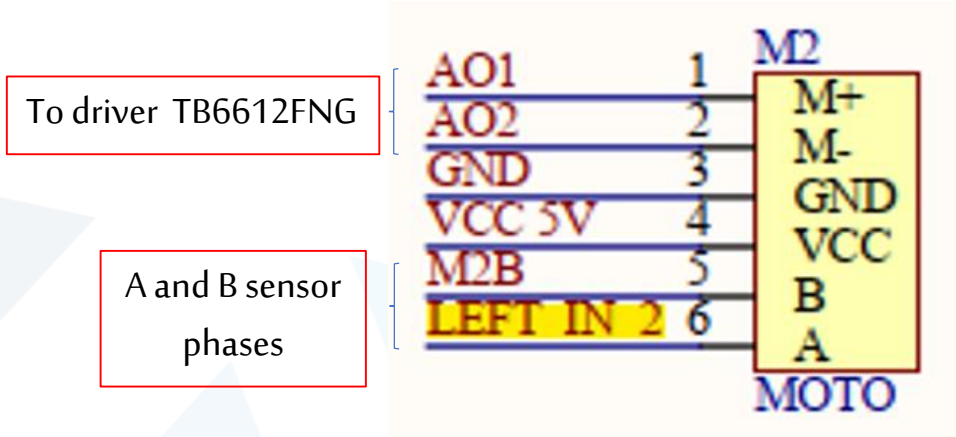
gearbox Shaft Speed : 110RPM

Idle current: 250mA

Power: 4.8w

Maximum locked rotor current: 6.5A

Reduction box length: 22mm



1--- --Motor power cord AO1

2--- --Motor power cord AO2

3--- --Sensor signal line Negative GND

4--- --Sensor Positive 5V VCC

5--- --Sensor signal line B phase

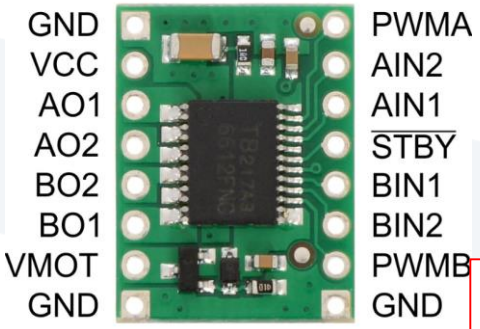
6--- --Sensor signal line A phase

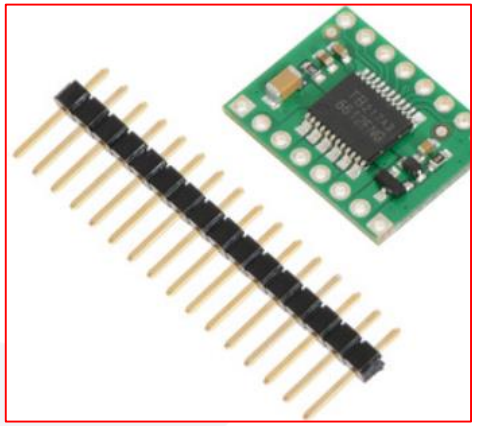
motor with 30:1 reduction ratio. Motor speed is 30*110=3300 RPM

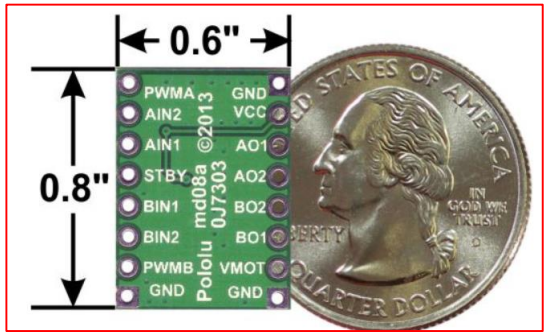
the number of pulses can reach 30*13*2=780 . Single phase can also reach 390 pulses for each turn of the gearbox shaft.

TB6612FNG Dual Motor Driver carrier

https://www.pololu.com/product/713

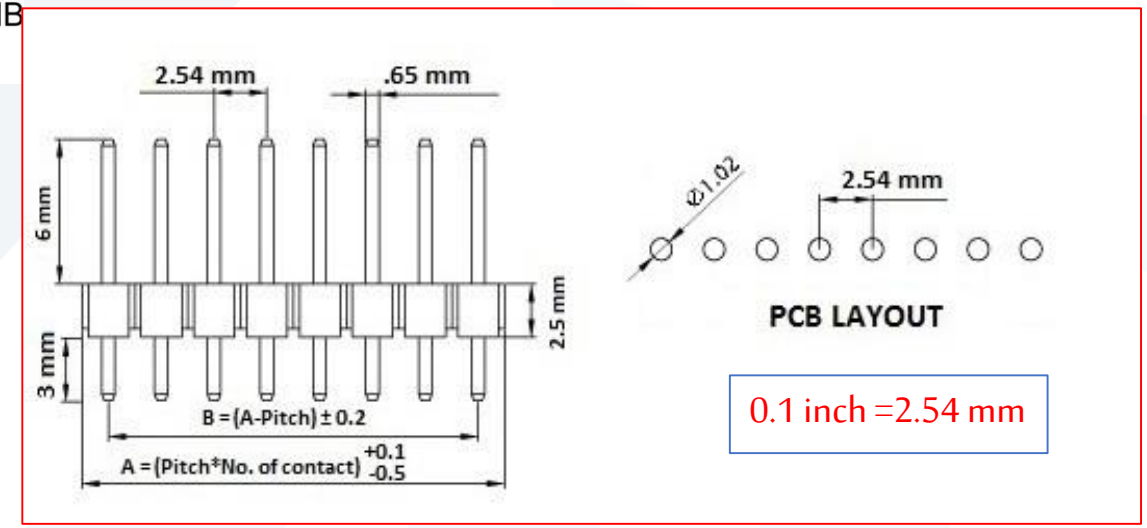






- **Dual-H-bridge motor driver: can drive two DC motors or one bipolar stepper motor**
- **Maximum PWM frequency: 100 kHz**

Characteristics	Symbol	Rating	Unit	Remarks
Supply voltage	VM	15	V	
	Vcc	6		
Input voltage	VIN	-0.2~6	V	IN1,IN2,STBY,PWM pins
Output voltage	Vout	15	V	O1,O2 pins
Output current	Iout	1.2	A	Per 1ch
	Iout (peak)	2		tw=20ms Continuous pulse, Duty≦20%
		3.2		tw=10ms Single pulse



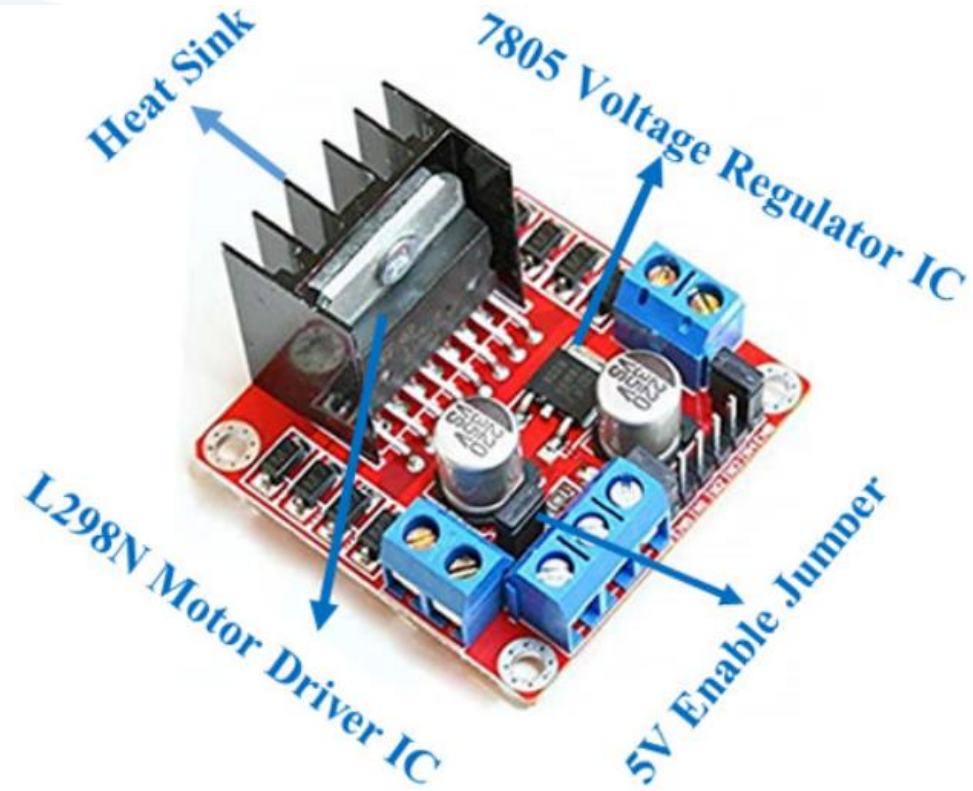
Included hardware

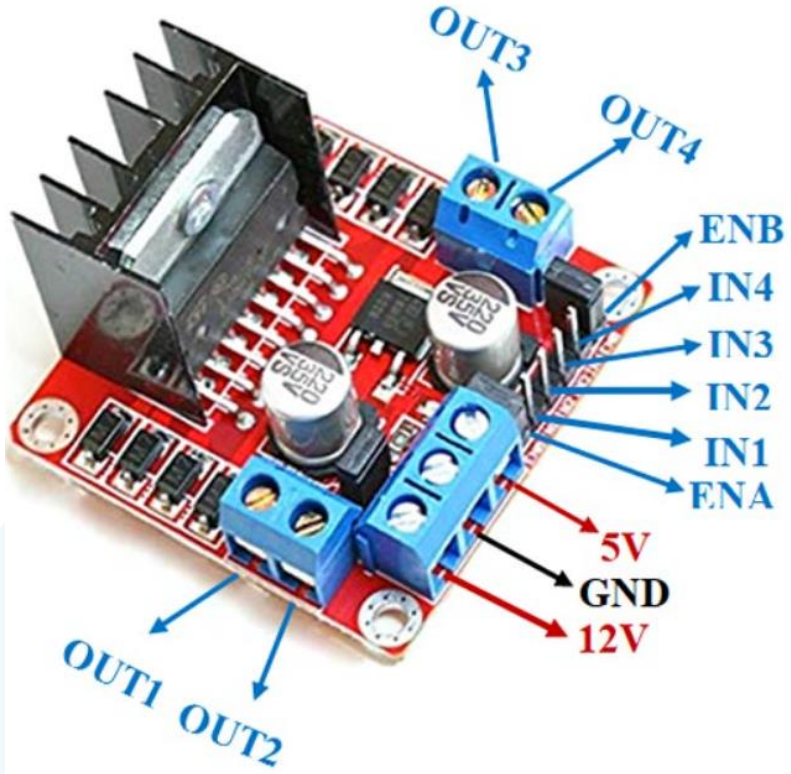
A 1×16-pin breakaway 0.1″ male header strip is included with the TB6612FNG motor driver carrier. This strip can optionally be soldered to the carrier board so that it can be used with perfboards, solderless breadboards, or 0.1″ female connectors. (The headers might ship as two 1×8 pieces or as a single 1×16 piece that can be broken in half.)

L298 Dual Motor Driver carrier

Motor driver:	**L298N**
Motor channels:	**2**
Maximum operating voltage:	**50 V**
Peak output current per channel:	**2 A**
logic voltage:	**4.5-7 V**

Frequency TYPICAL 25kHz - MAX40 KHz



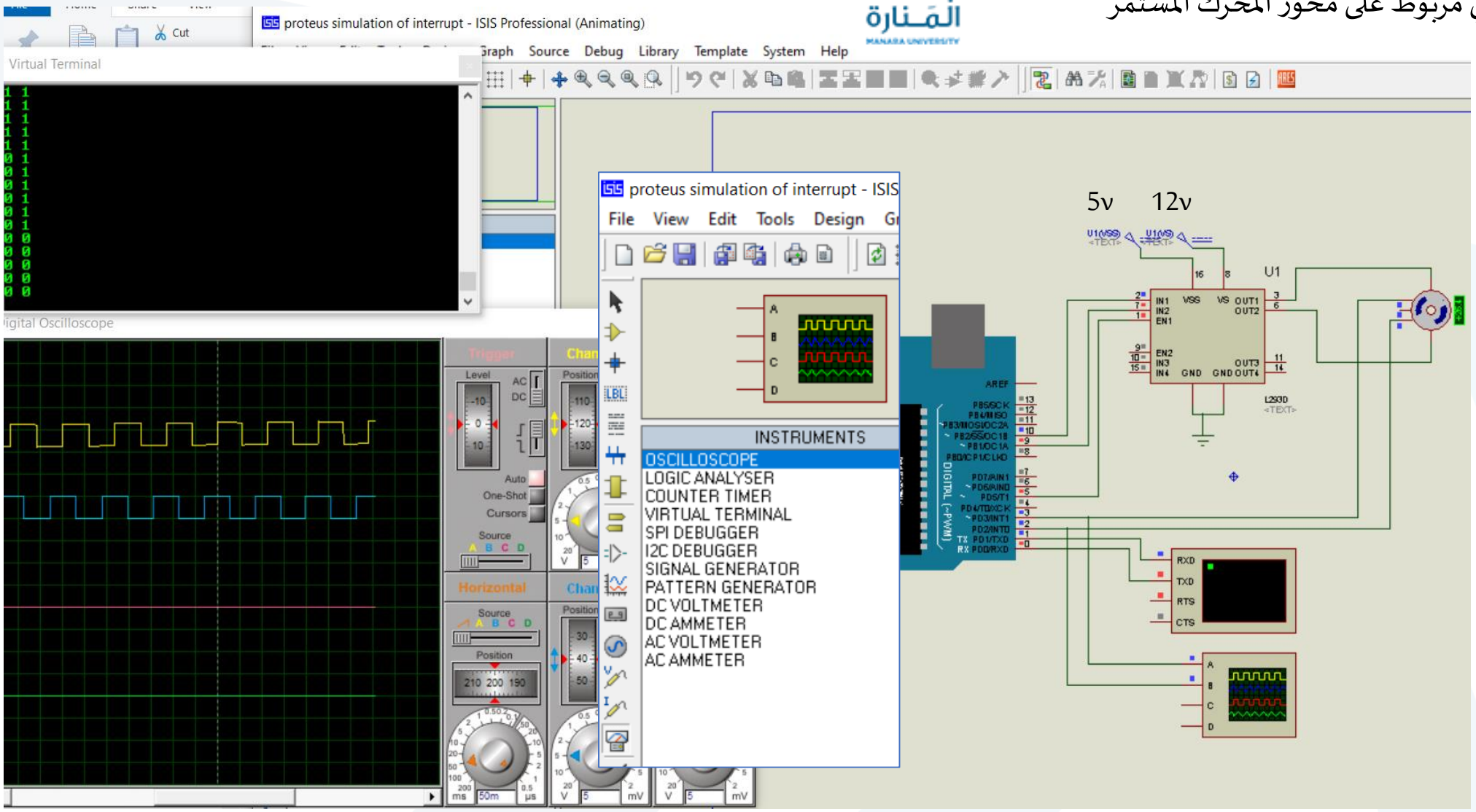


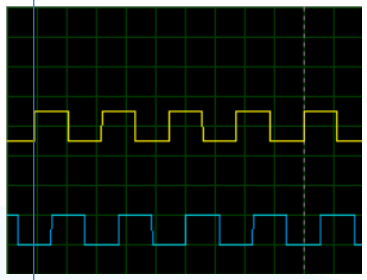
78M05 Voltage regulator will be enabled only when the jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator and the 5V pin can be used as an output pin to power the microcontroller.

The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

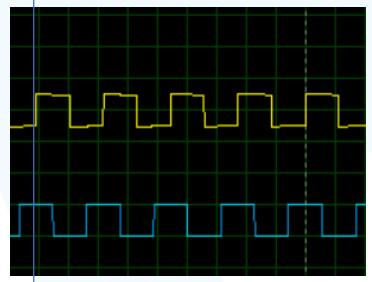
DC MOTOR USING L298 AND ROTARY ENCODER

قراءة إشارة مشفر (انكودر) دوراني مربوط على محور المحرك المستمر





CCW

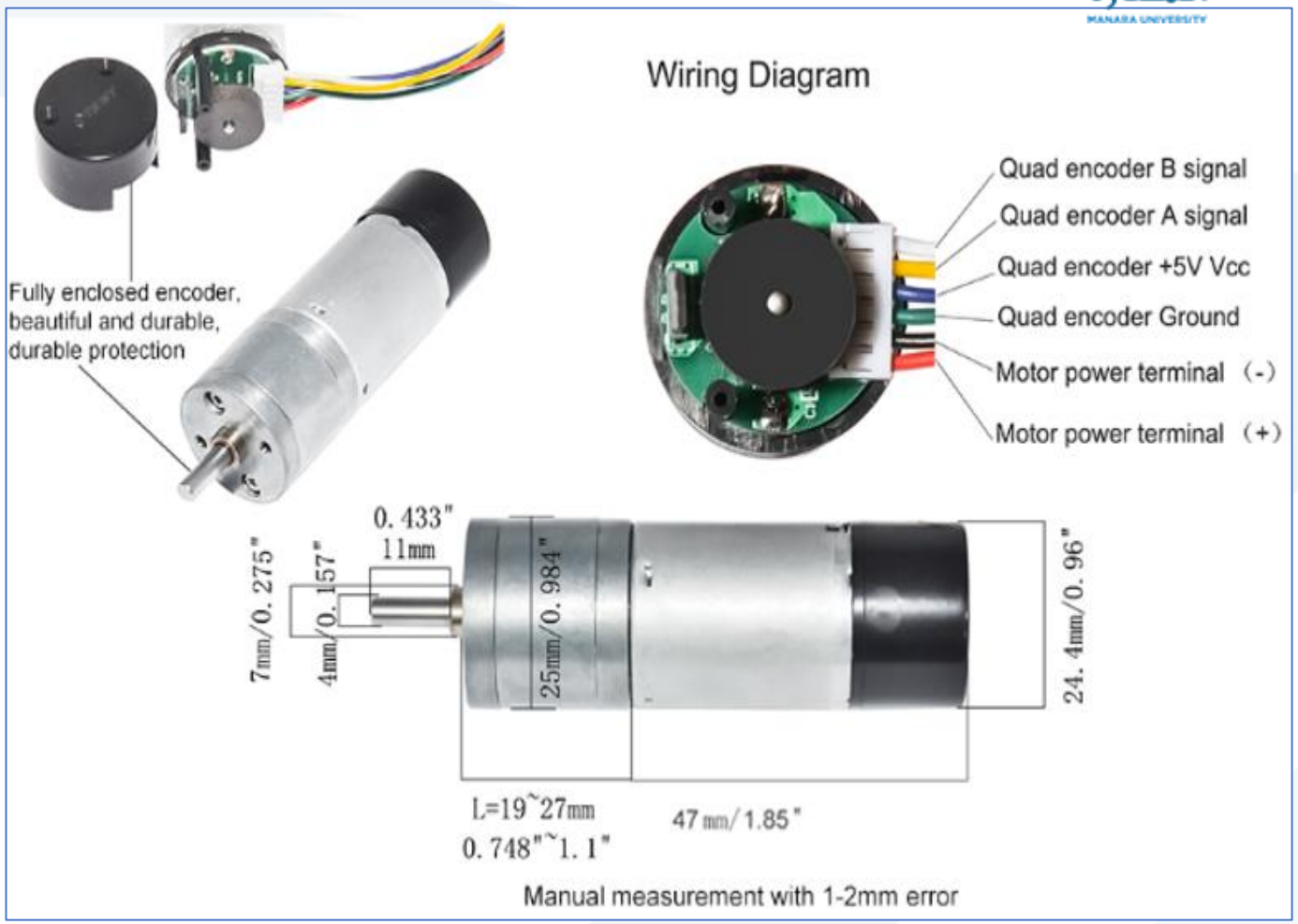


CW

لحظة حدوث جهة صاعدة على القناة الأولى نفحص القناة الثانية فيما اذا كانت HIGH او LOW

```
#define A 2 // pin2 of the Arduino
#define B 3 // Pin3 of the Arduino
int A_DATA;
int B_DATA;
int in1 = 9;
int in2 = 10;
int EN = 5;

void setup() {
  Serial.begin(9600); // Activates Serial communication
  pinMode(A,INPUT); // sets pin2 as the input
  pinMode(B,INPUT); // sets pin3 as the input
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(EN, OUTPUT);
  digitalWrite(in1, HIGH);
  digitalWrite(in2,LOW );
  digitalWrite(EN, HIGH);
}
```

```
void loop() {
  A_DATA = digitalRead(A);
// We simply read Pin2 of the Arduino and store the result in variable A_DATA
  B_DATA = digitalRead(B);
// We simply read Pin3 of the Arduino and store the result in variable b
  Serial.print(A_DATA);
  Serial.print(" ");
  Serial.print(B_DATA);
  Serial.println();
}
```



DC Gear motor with two-channel Hall effect encoder;

Rated Voltage: 12V;

No-Load Speed of the gearbox shaft: 130RPM;

No-Load Current: ≤ 0.15A;

Rated Torque: 1.2kg.cm;

Gear Reduction Ratio: 45:1,

جهاز استشعار حساس هال بت 12خط ترميز مغناطيسي وإخراج ثنائي الطور. Each Loop of the sensor Output Pulses: 12PPR

45*12=540PPR;

Single Output of one channel **540 Pulses Per Revolution of the gearbox shaft**:

- 1000 count encoder is on the shaft of a MOTOR.
- MOTOR is also coupled to a 20:1 gear reducer, with the output to a ball screw with a 0.1" pitch.
- The motor shaft and the encoder must turn 20 times (producing 20,000 quadrature cycles) to advance the load 0.1", and
- will product 200,000 cycles by the time the load has move 1.00".
- The resolution, in terms of load movement is 200,000 counts per inch.
- If the encoder were coupled directly to the ball screw, the functional resolution would be 10,000 counts per inch.

Angular speed ω is also directly related to linear tangential speed v.

$$\omega = 2\pi N / 60 \ (\text{inradians/s when N is in RPM})$$

$$v = r\omega$$

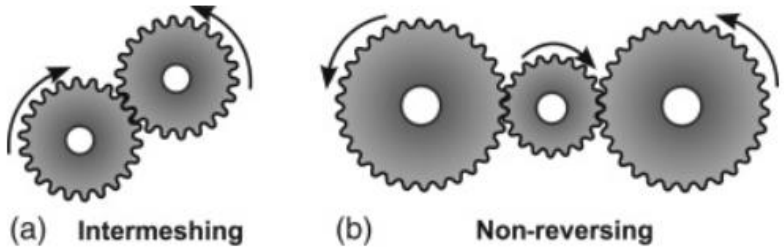
- If the tire outer diameter is 0.80 m (radius of 0.40 m), what is the wheel's rotational speed when the vehicle is moving at 100. km/h? 100. km/h (62.1 mph)

rω = v = (100. km/h)×(1000 m/km)×(1 h/3600 s) = 27.8 m/s.
Hence, ω = v/r = 27.8/0.40 = 70. radians/s or 70. × 60/2π = 670 RPM.

- The rotational speed of the engine must be transformed into the rotational needs of the wheels.
- How can these two different speeds of rotation be reconciled? It is done by a mechanism called a transmission.

A manual transmission is made of several intermeshing toothed gears.

$$\text{Gear Ratio (GR)} = \frac{\text{Input rotation}}{\text{Output rotation}} = \frac{N_1}{N_2} = \frac{d_2}{d_1} = \frac{t_2}{t_1}$$



$$\frac{t2}{t1}=\frac{d2}{d1}=\frac{\omega 1}{\omega 2}=\frac{N1}{N2}=\frac{T2}{T1}=i$$

t1 = number of teeth on driving (pinion) gear

t2 = number of teeth on driven (wheel)gear

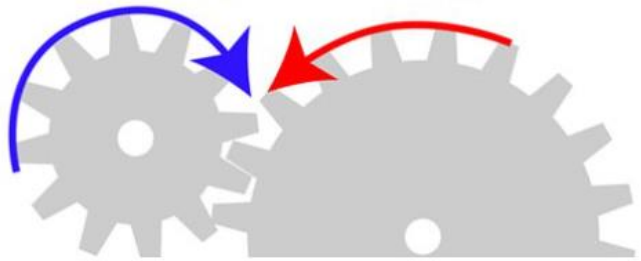
$\omega 1$ = speed of driving gear

$\omega 2$ = speed of driven gear

$N1$=RPM of driving gear

T1 = torque of driving gear

$$RPM_A \times Teeth_A = RPM_B \times Teeth_B$$



To achieve torque multiplication and speed reduction, the driving gear will be smaller than the driven gear

the smaller, driving gear will turn twice for every one revolution of the larger, driven gear.

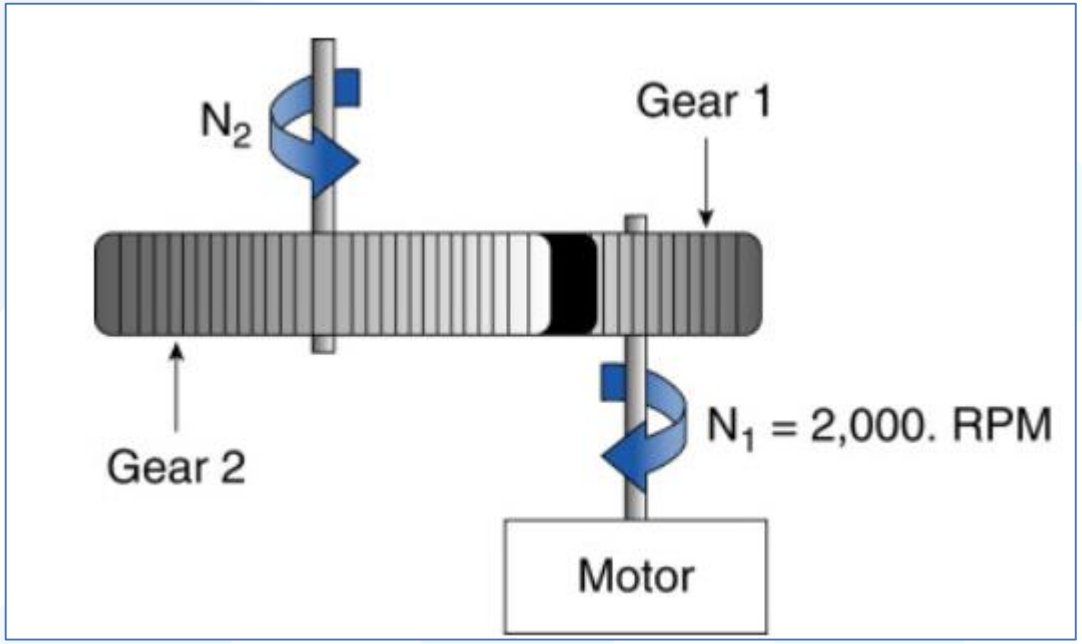
$$\frac{40}{20}=\frac{\omega_1}{\omega_2} \qquad i=\frac{\omega_1}{\omega_2}$$

2:1 means that the speed from the motor is reduced by a factor of 2 and the torque from the motor is multiplied by a factor of 2 (not accounting for any losses due to inefficiencies in the gear train).

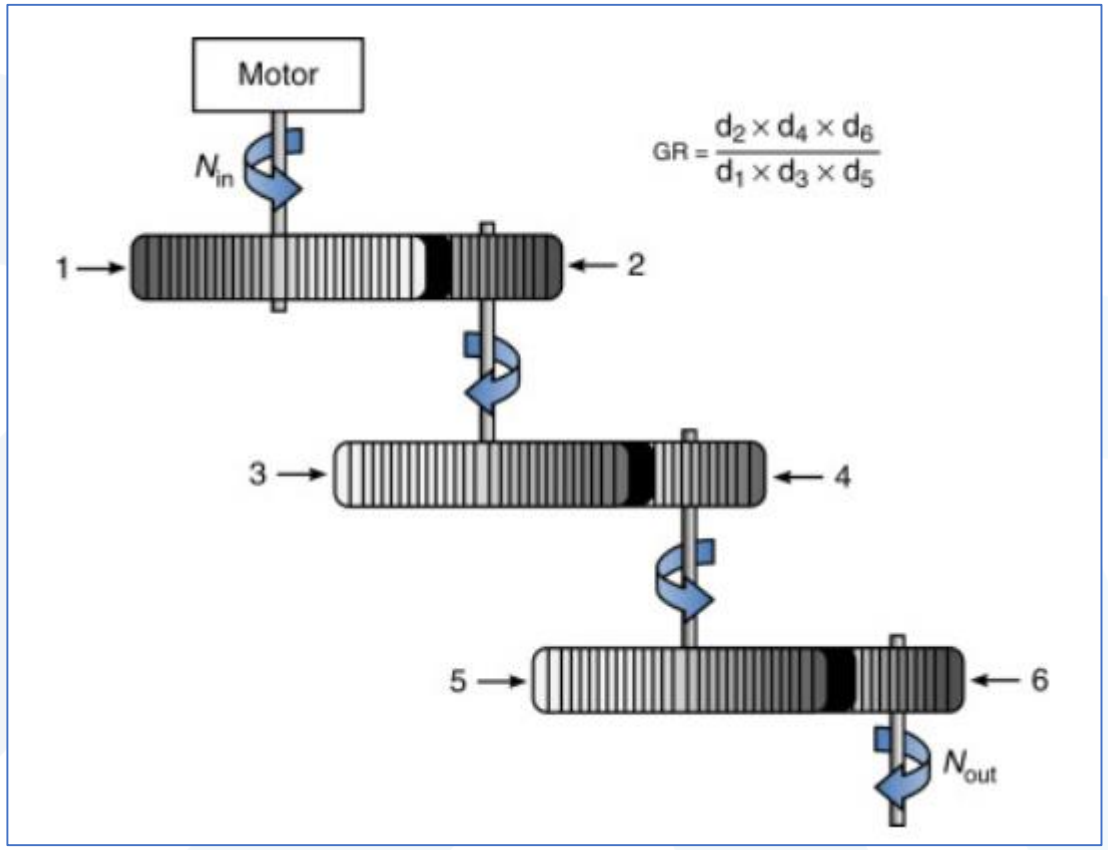
$$\frac{2}{1}=\frac{\omega_1}{\omega_2} \qquad i=2:1$$

A 6.00 cm diameter gear is attached to a shaft turning at 2000. RPM. That gear in turn drives a 40.0 cm diameter gear. What is the RPM of the driven gear?

$N_2 = 2000.\ [\text{RPM}] \times 6.00/40.0\ [\text{cm/cm}] = \mathbf{3.00 \times 10^2\ RPM.}$



$$GR = \frac{\text{Product of diameter or number of teeth of DRIVEN gears}}{\text{Product of diameter or number of teeth of DRIVING gears}}$$

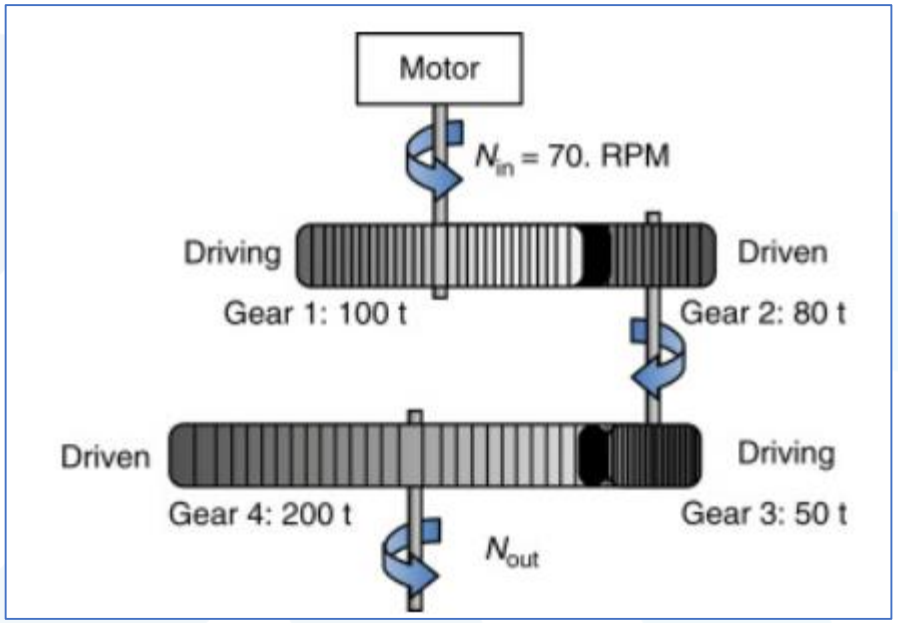


Overall GR = $N_{in}/N_{out} = \dfrac{\text{Product of number of teeth on "driven" gears}}{\text{Product of number of teeth on "driving" gears}}$

Hence, $N_{in}/N_{out} = (t_2 \times t_4)/(t_1 \times t_3) = (80 \times 200)/(100 \times 50) = 3.19$,

and therefore

$N_{out} = 70.0/3.19 = 21.9 \text{RPM}$



Gears not only change the rotation speed, but they also change the torque on the axle.

$$\frac{t2}{t1} = \frac{d2}{d1} = \frac{\omega 1}{\omega 2} = \frac{N1}{N2} = \frac{T2}{T1} = i$$

- For example, in a car with manual transmission, you can first use one set of gears to provide the wheels with high torque and low RPM for initial acceleration (first gear).
- Then you can shift to another set of gears providing the wheels with lower torque and higher RPM as the car speeds up (second gear).
- Then you can shift to a third gear combination that offers low torque and high RPM for cruising along a level highway at 65 miles per hour.
- In a modern automobile there may be four, five, or six forward gears.

Hardware interrupts used in Arduino mega or Uno

External interrupts

BOARD	INT0	INT1	INT2	INT3	INT4	INT5	...	INT15
UNO	Pin 2	Pin 3						
MEGA	Pin 2	Pin 3	Pin 21	Pin 20	Pin 19	Pin 18		
ESP8266	GPIO 0	GPIO 1	GPIO 2	GPIO 3	GPIO 4	GPIO 5	...	GPIO 15

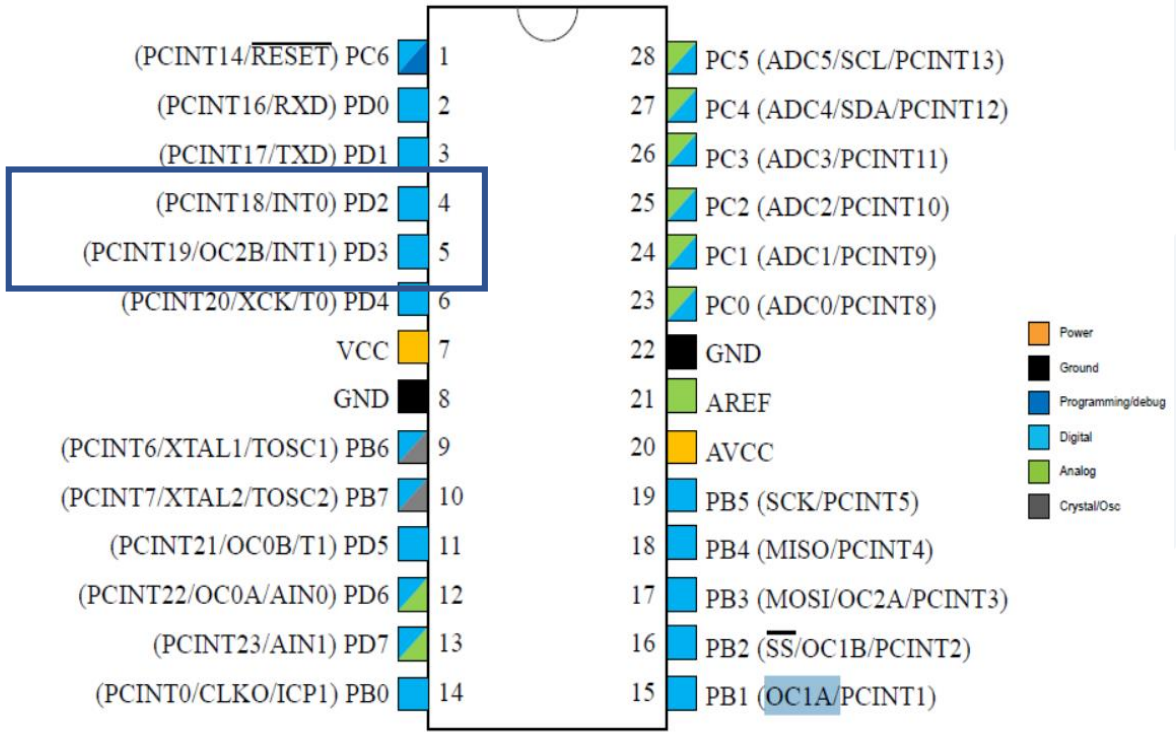
Pin change interrupts

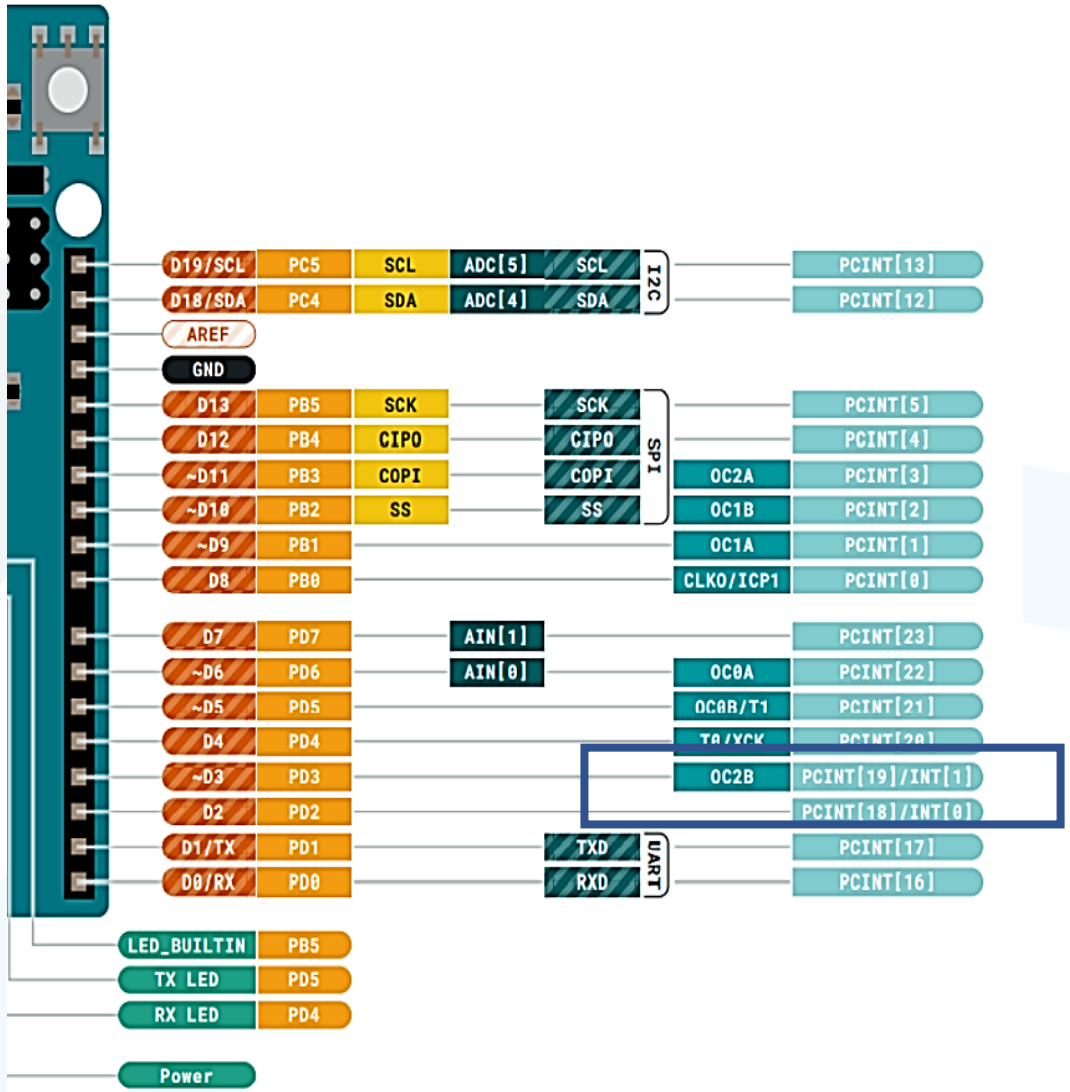
Pin Change Interruptions ISR | PCINT | Arduino101 - YouTube

Arduino Pin Change Interrupts – The Wandering Engineer

Arduino pin change interruptions ISR Tutorial (electronoobs.com)

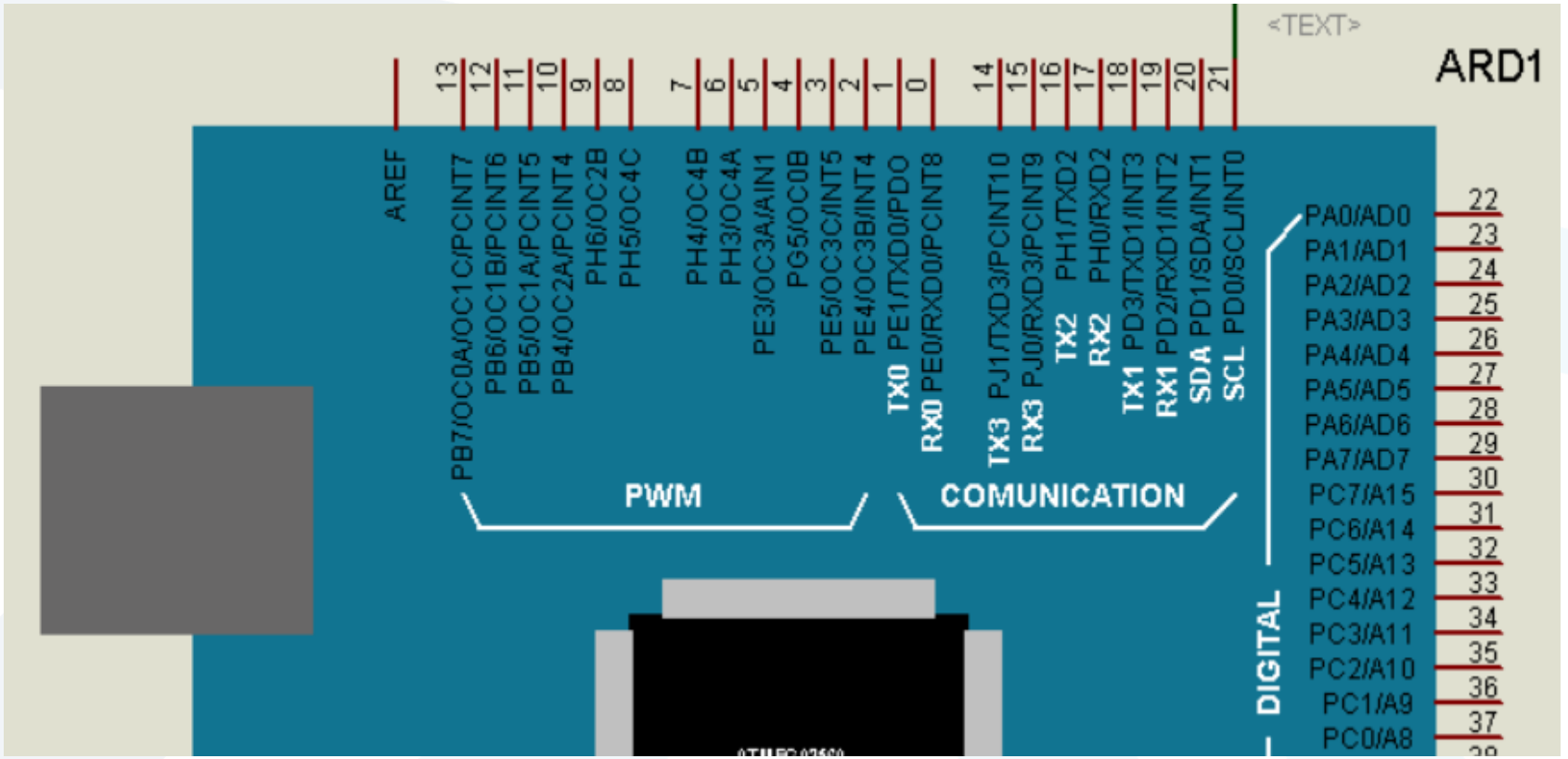
Arduino uno, ,mini, nano: ATmega328P





انتبه في مكتبة الاردينو ميغا في بروتوس ارقام المقاطعات غير صحيحة والصحيح مدون يسارا

```
// for mega
// int.0 ⇔ arduino pin 2
// int.1 ⇔ arduino pin 3
// int.2 ⇔ arduino pin 21
// int.3 ⇔ arduino pin 20
// int.4 ⇔ arduino pin 19
// int.5 ⇔ arduino pin 18

// for uno
// int.0 ⇔ arduino pin 2
// int.1 ⇔ arduino pin 3
```



12.4 Interrupt Vectors in ATmega328 and ATmega328P

Table 12-6. Reset and Interrupt Vectors in ATmega328 and ATmega328P

VectorNo.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 269.

Interrupt:

The number of Interrupt (int). I will be using Pin number 2, so I will simply write 2. As in Arduino interrupt 0 is available on pin number 2 and interrupt 1 is available on pin number 3. In programming, I will use interrupt 0 which is on pin number 2.

ISR:

This is already explained above, this is basically a user-defined function which consists of the instructions that you want to execute when an external hardware interrupt occurs.

The ISR has the following syntax in Arduino:

attachInterrupt(interrupt,ISR,mode);

we use the attachInterrupt() function for creating the external interrupt. This function has three parameters.

Mode:

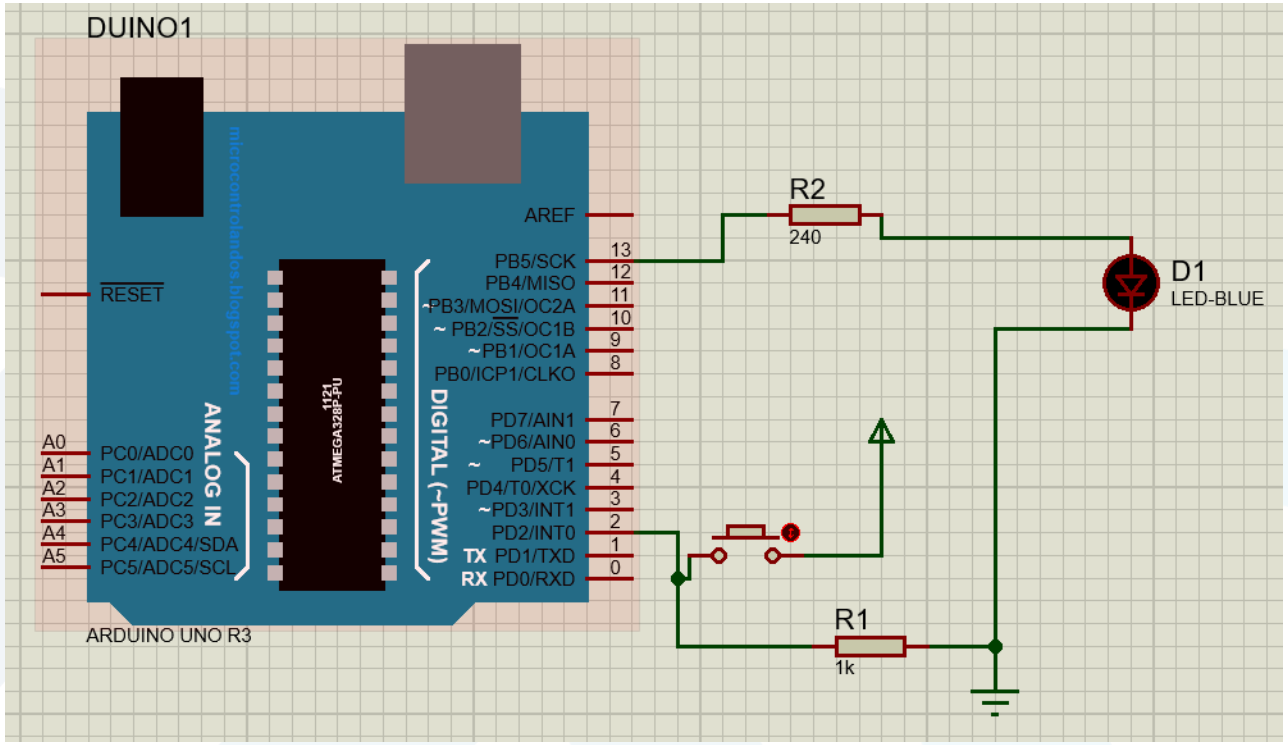
We have 4 modes which are

1. LOW: to trigger the interrupt whenever the pin is low
2. CHANGE: to trigger the interrupt whenever the pin changes the value
3. RISING: to trigger when the pin goes from low to high
4. FALLING: when the pin goes from high to low.

Note: There are things that you need to take care of which are, no delays should be used inside the ISR and keep the code as short as possible.

تطبيق :

```
const byte ledPin = 13;
const byte interruptPin = 0;
volatile byte state = HIGH;

void setup() {
  pinMode(ledPin, OUTPUT);
  //pinMode(interruptPin, INPUT_PULLUP);
  //attachInterrupt(0, ISR1, FALLING);

  //attachInterrupt(digitalPinToInterrupt(interruptPin), blink, RISING);
  attachInterrupt(0, blink, FALLING);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

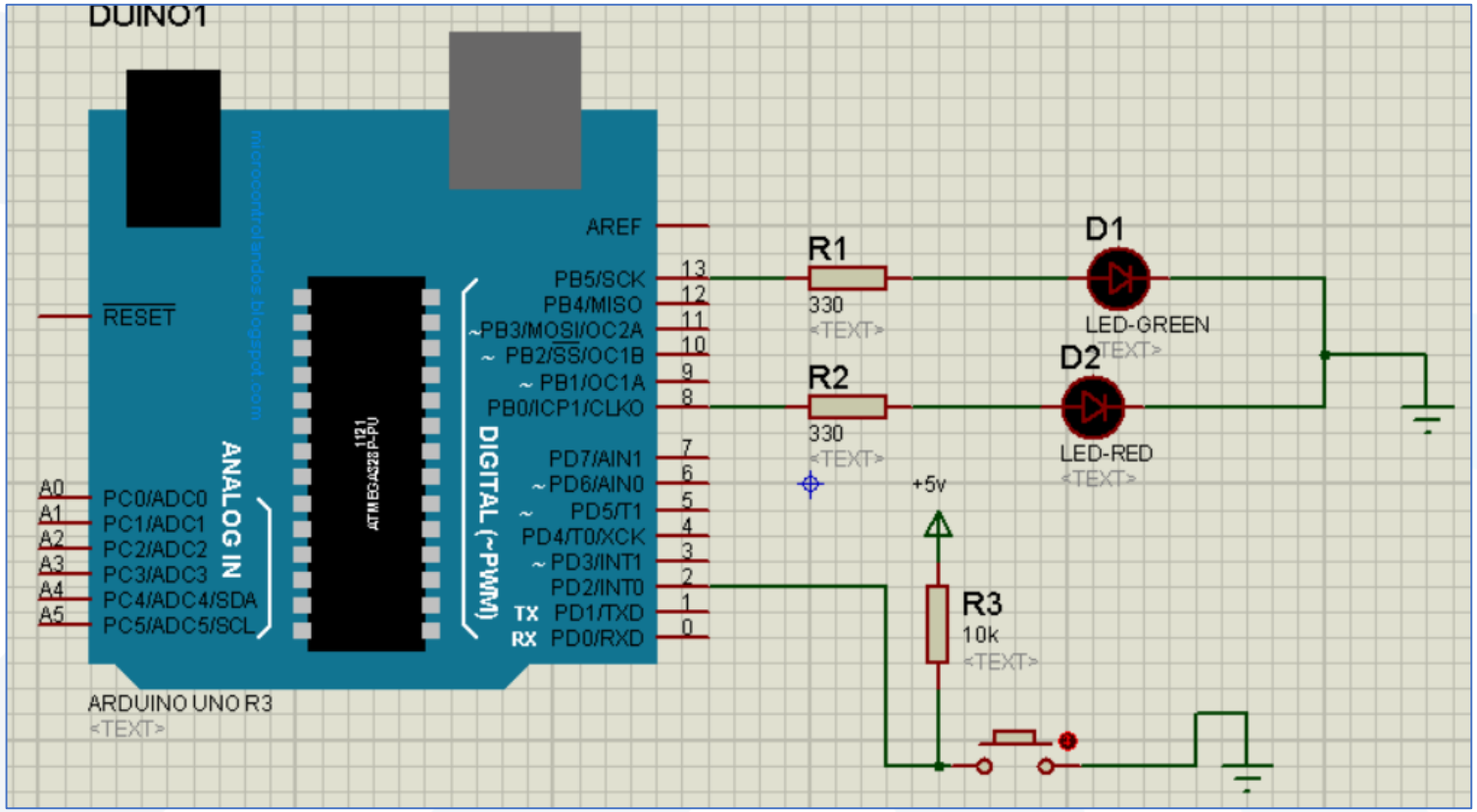


The volatile modifier ensures that changes to the state variable are immediately visible in loop().

```
int led1 = 8;
int led2 = 13;
int in = 2;
 void setup()
{
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
//  pinMode(in, INPUT);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  attachInterrupt(0, ISR1, FALLING);

}
 void loop()
{
  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led2, LOW);
  delay(500);
}

void ISR1()
{
  digitalWrite(led2,LOW);
  digitalWrite(led1,HIGH);
  while(!digitalRead(2));
  digitalWrite(led1,LOW);
}
```

جَامعة المَنارة
MANARA UNIVERSITY

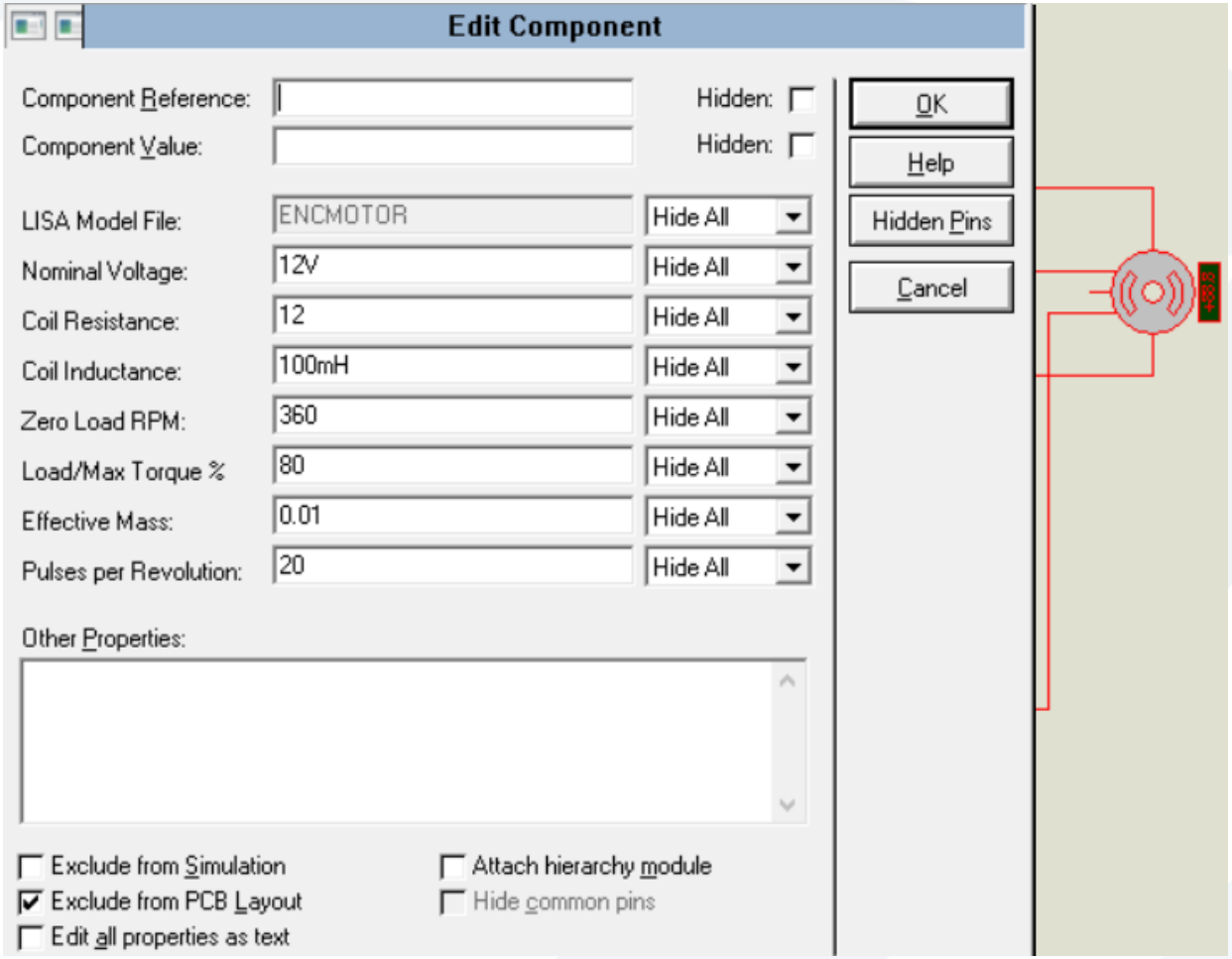
تطبيق :

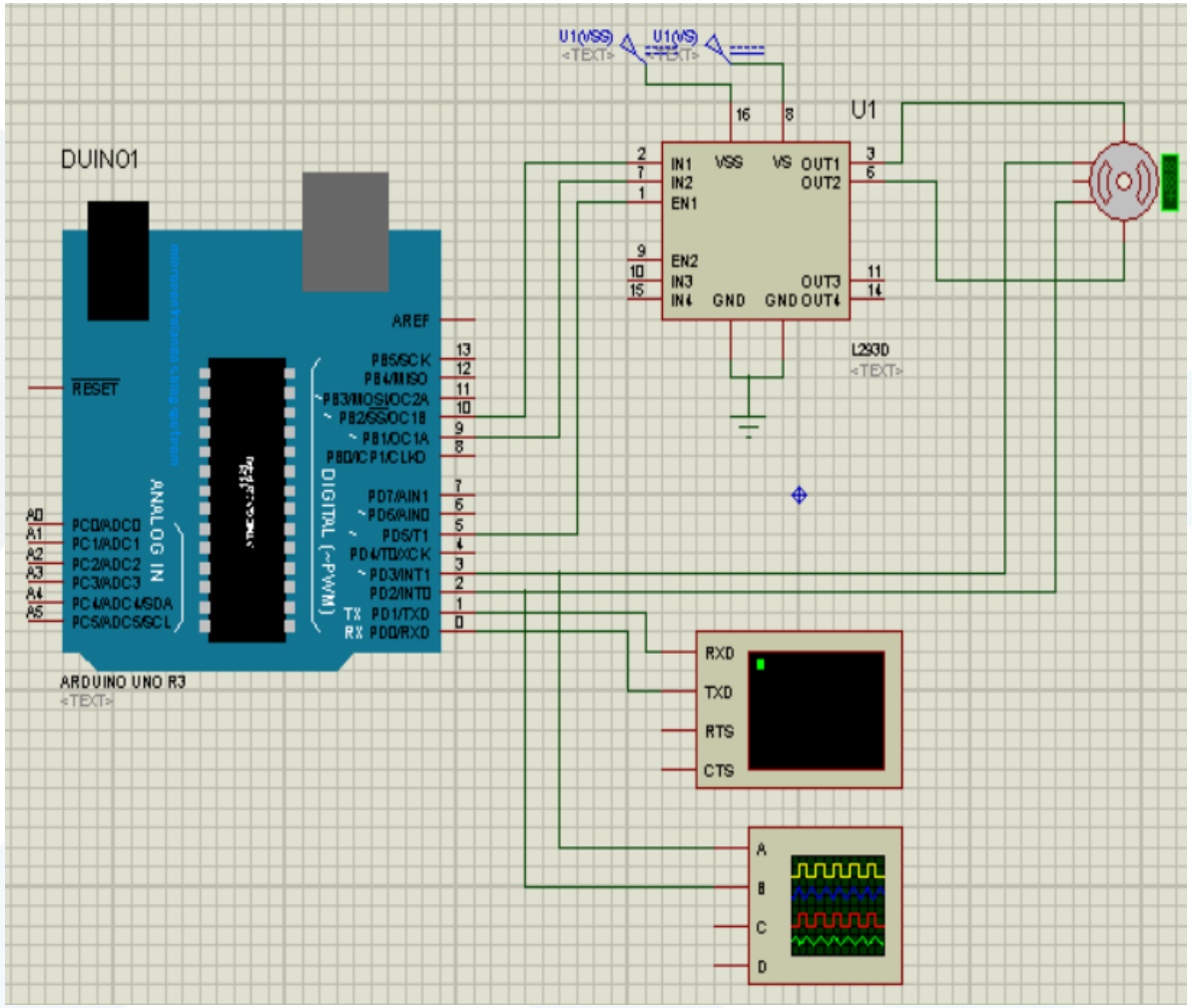


قياس زاوية دوران محرك
تيار مستمر

تطبيق :

MOTOR ENCODER

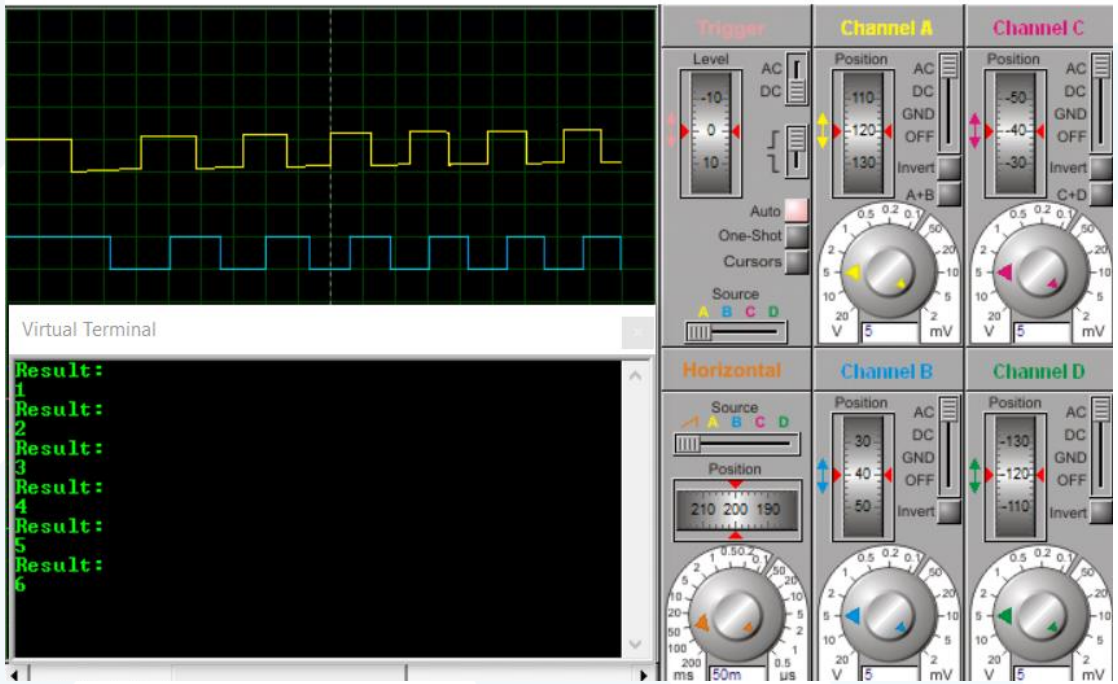




```
#define channel_A 2 // pin2 of the Arduino
#define channel_B 3 // pin 3 of the Arduino

int in1 = 9;
int in2 = 10;
int EN = 5;

int Count_pulses = 0;
void setup() {
Serial.begin(9600); // activates the serial communication
pinMode(channel_A,INPUT); // sets the channel_A pin as the input
pinMode(channel_B,INPUT); // sets the channel_B pin as the input
attachInterrupt(digitalPinToInterrupt(channel_A),Motor_Encoder_ISR,RISING);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
  pinMode(EN, OUTPUT);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(EN, HIGH);
}

void loop() {

}
```

```
void Motor_Encoder_ISR(){
  int b = digitalRead(channel_B);
  if(b > 0){
    Count_pulses++;
  }
  else{
    Count_pulses--;
  }
   Serial.println("Result: ");
  Serial.println(Count_pulses);
}
```

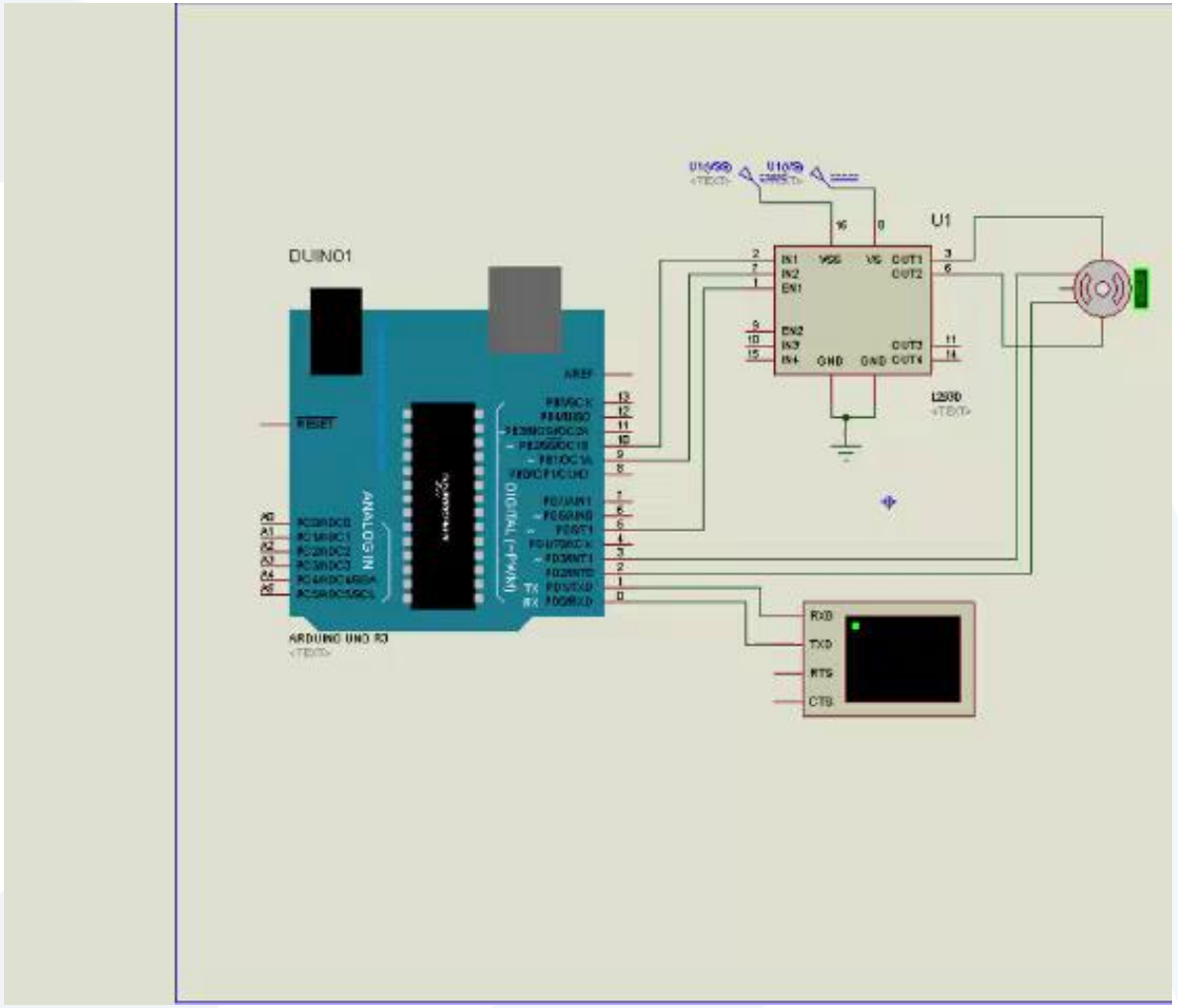
Digital Oscilloscope



تحكم باتجاه دوران المحرك وقياس زاوية الدوران

```
#define channel_a 2
#define channel_b 3
#define PWM 5
#define IN2 9
#define IN1 10
int pos = 0;
void setup() {
  Serial.begin(9600);
  pinMode(channel_a,INPUT);
  pinMode(channel_b,INPUT);
  attachInterrupt(digitalPinToInterrupt(channel_a),readEncoder,RISING);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(PWM, OUTPUT);}
 void loop() {
  setMotor(1, 255, PWM, IN1, IN2);
  delay(2000);
  //Serial.println(pos);
  setMotor(0, 255, PWM, IN1, IN2);
  delay(2000);
  //Serial.println(pos);
  setMotor(-1, 255, PWM, IN1, IN2);
  delay(2000);
  setMotor(0, 255, PWM, IN1, IN2);
  delay(6000);
  //Serial.println(pos);
}
```

```
void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
  analogWrite(pwm,pwmVal);
  if(dir == 1){
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
  }
  else if(dir == -1){
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
  }
  else{
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
  }
}

void readEncoder(){
  int b = digitalRead(channel_b);
  if(b > 0){
    pos++;
  }
  else{
    pos--;
  }
  Serial.println(pos);
}
```


DUINO1
U1
L293D
ARDUINO UNO R3
RESET
AREF
ANALOG IN
DIGITAL (PWM~)
RXD
TXD
RTS
CTS

PID Control

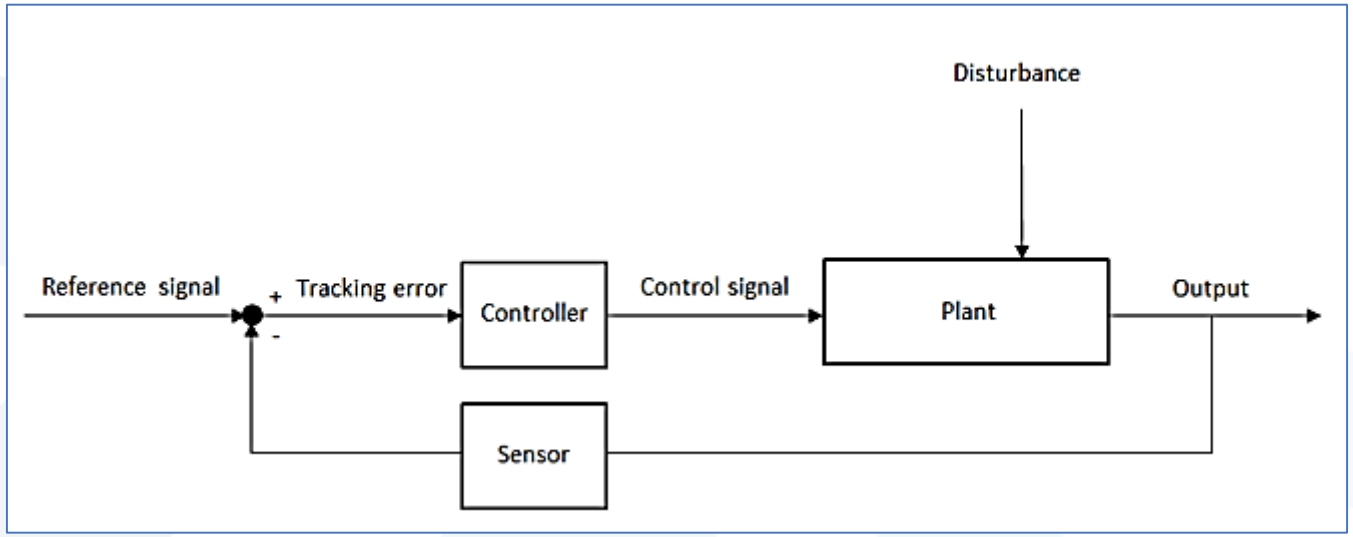
$$u(t) = K_p \cdot e(t) + K_I \int_0^t e(\zeta) \cdot \delta\zeta + K_D \cdot \frac{de(t)}{dt}$$

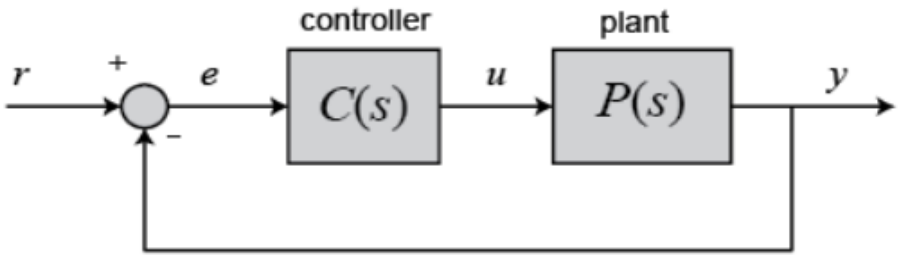
$u(t)$	= PID control variable
K_p	= proportional gain
$e(t)$	= error value
K_i	= integral gain
$de(t)$	= change in error value
dt	= change in time

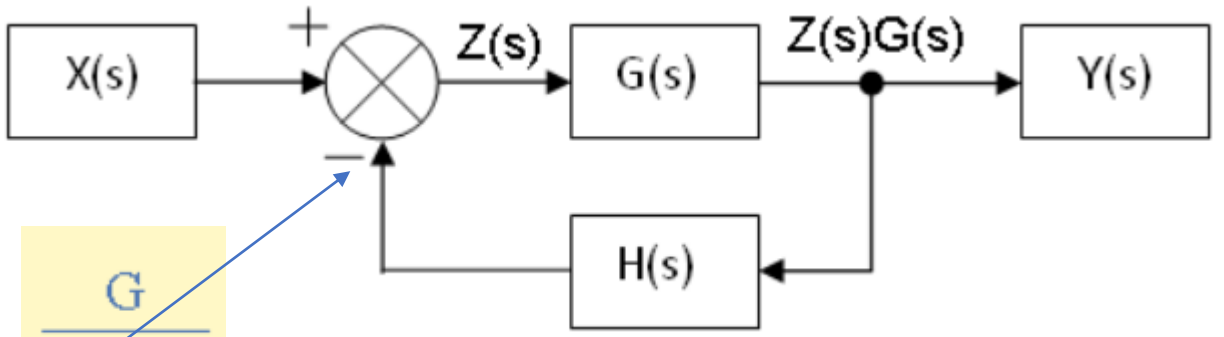
Kp = Proportional gain

KI = Integral gain

Kd = Derivative gain







$$\frac{G}{1+GH}$$

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

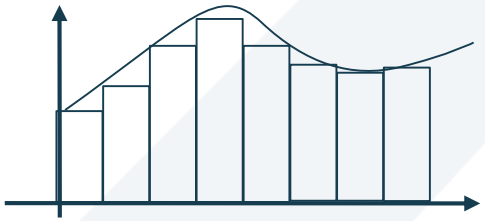
Implementation in discrete system

$$u(t) = K_p \cdot e(t) + K_I \int_0^t e(\zeta) \cdot \delta\zeta + K_D \cdot \frac{de(t)}{dt}$$

a good approximate of e': store my old error, compute new error ,take the difference and divide it by dt then I have

What is the integral ?

The integral is the sum under the curve and to approximate , we can sum up all those little blocks.



$$\int_0^t e(\zeta)\delta\zeta \simeq \sum_{k=0}^{N} e(k\delta t)\delta t = \delta t \cdot E$$

$$\delta t \cdot E_{new} = \delta t \cdot \sum_{k=0}^{N} e(k\delta t) = \delta t e\big((N+1)\delta t\big) + \delta t \cdot E_{old}$$

$$E_{new} = E_{old} + E$$

Implementation in discrete system

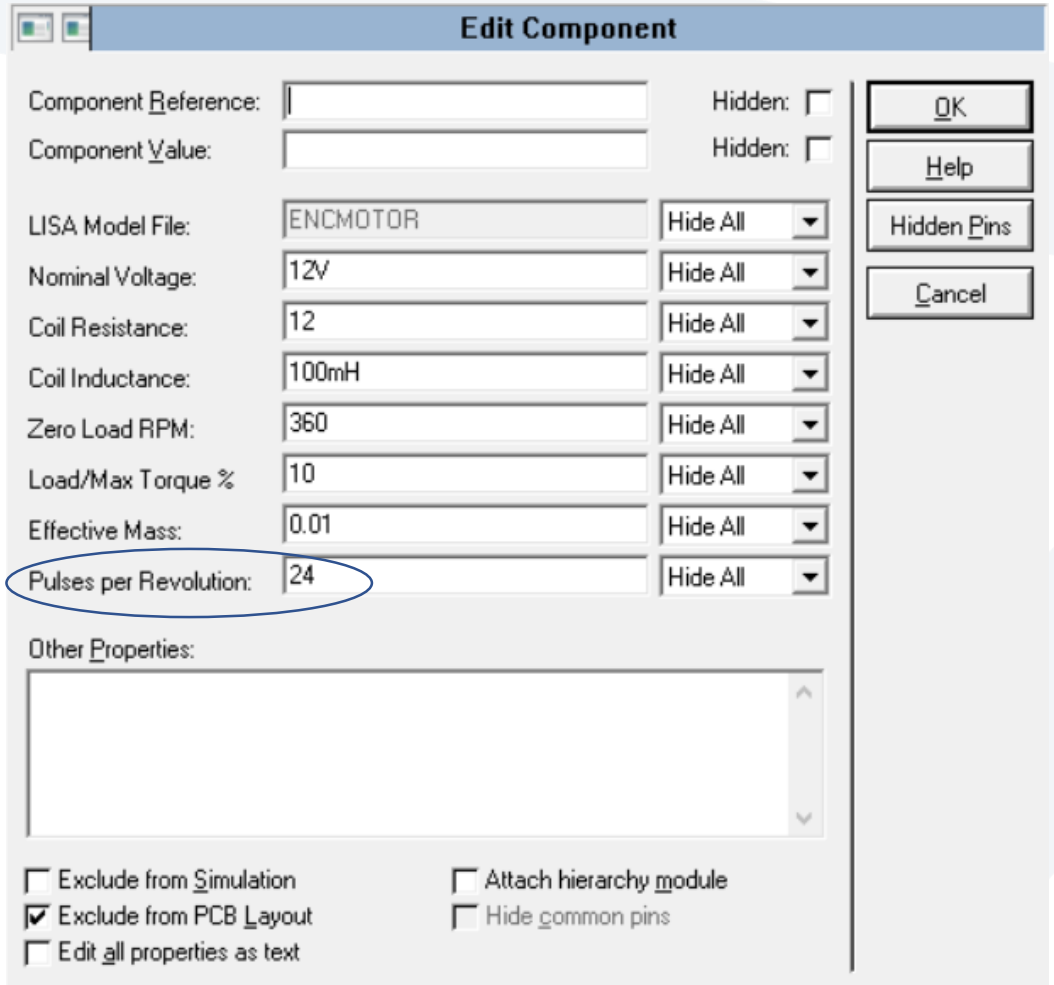
```
read e ;

e_dot = e  −  old_e ;

E = E + e ;

U = Kp * e + Kd * e_dot + Ki * E ;

old_e = e ;
```

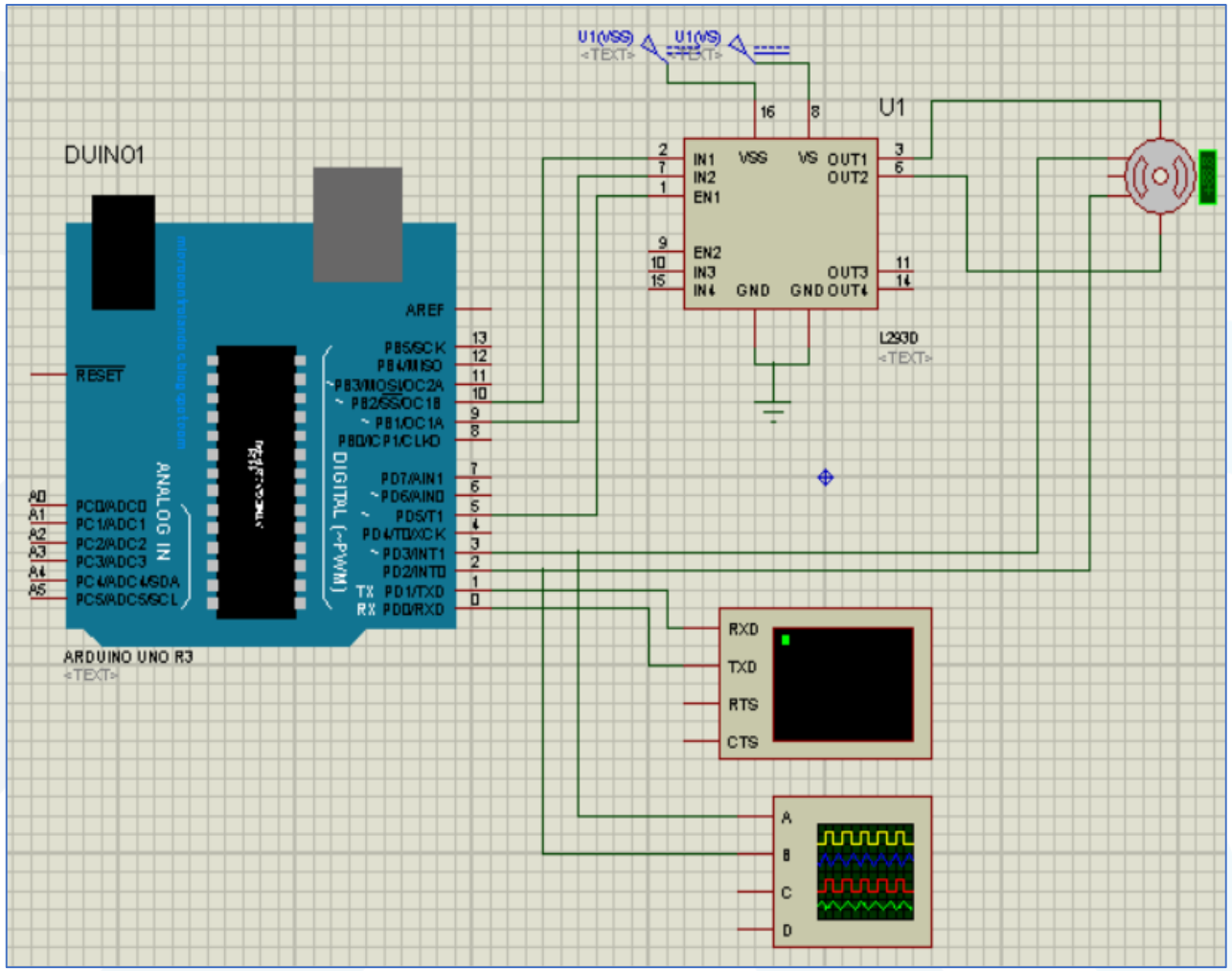
Notes: we should consider (dt)

$$K_D \cdot \dot{e} = K_D \cdot \frac{e_{new} - e_{old}}{\Delta t} \rightarrow K_D = \acute{K_D} \cdot (e_{new} - e_{old})$$

$$K_I \cdot \int_0^t e\,(\zeta)\delta\zeta = K_I \cdot (\Delta t \cdot E) \rightarrow K_I = \acute{K_I} E$$

جَامعة
المَنارة
MANARA UNIVERSITY





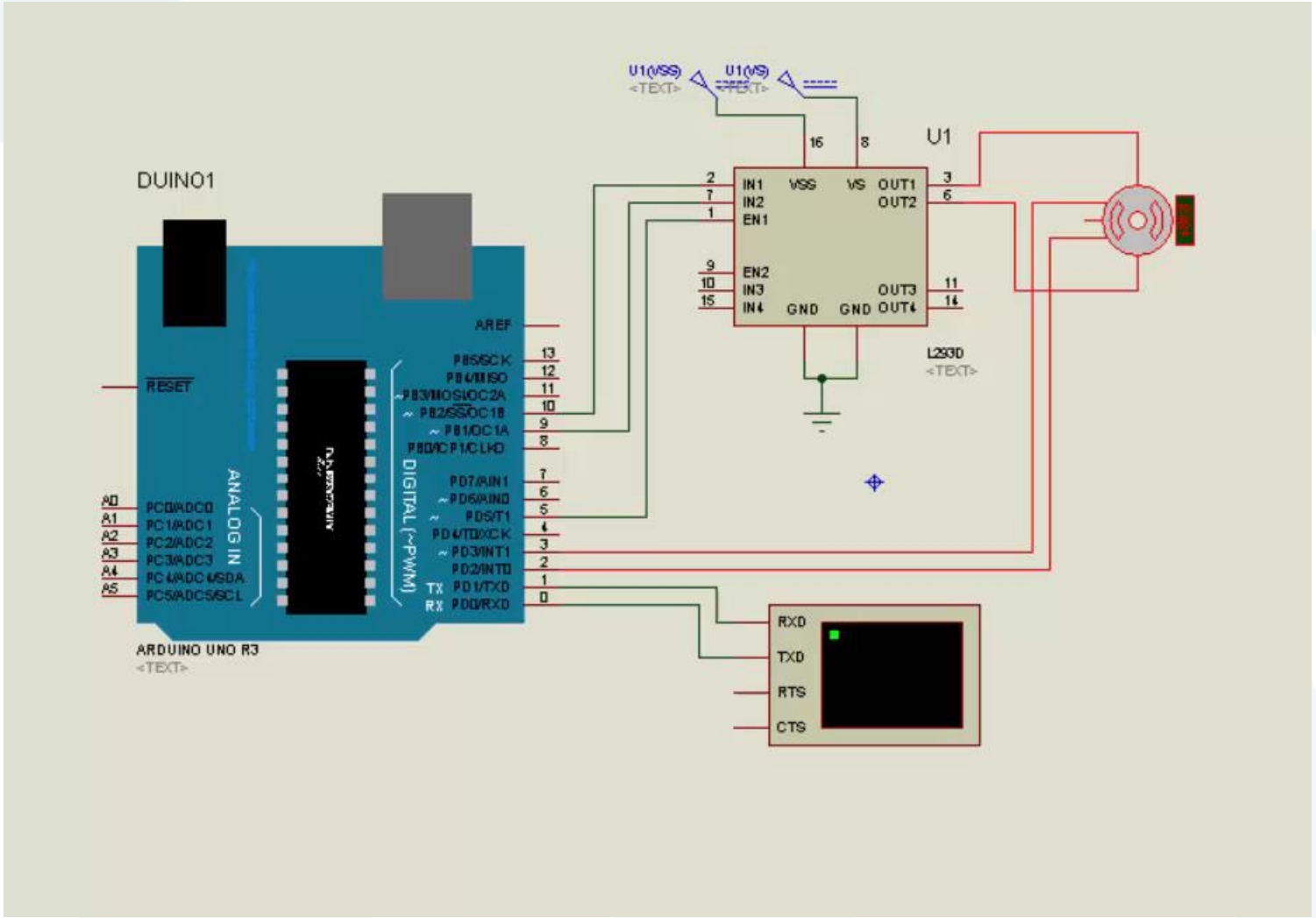
```
#define ENCA 2
#define ENCB 3
#define PWM_PIN 5
#define IN2 9
#define IN1 10

int pos = 0;
long prevT = 0;
float eprev = 0;
float eintegral = 0;

void setup() {
  Serial.begin(9600);
  pinMode(ENCA,INPUT);
  pinMode(ENCB,INPUT);

  pinMode(IN2,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(PWM_PIN,OUTPUT);

  attachInterrupt(digitalPinToInterrupt(ENCA),ISR_encoder,RISING);
  Serial.println("target pos");
}
```

```
void loop() {
  int target = 300;// set target position
  //target = 250*sin(prevT/1e6);

  // PID constants
  float kp = 6;   //10
  float kd = 0.1;//0.122
  float ki = 0.1;//0
  // time difference
  long currT = micros();
  float deltaT = ((float) (currT - prevT))/( 1.0e6 );
  prevT = currT;
  // error
  int e = pos-target;
  // derivative
  float dedt = (e-eprev)/(deltaT);
  // integral
  eintegral = eintegral + e*deltaT;

  // control signal
  float u = kp*e + kd*dedt + ki*eintegral;
```

```
  // motor power
  float pwr = fabs(u);
  if( pwr > 255 ){
    pwr = 255;
  }

  // motor direction
  int dir = 1;
  if(u<0){
    dir = -1;
  }

  // signal the motor
  setMotor(dir,pwr,PWM_PIN,IN1,IN2);

  // store previous error
  eprev = e;
  Serial.print(target);
  Serial.print("    ");
  Serial.print(pos);
  Serial.print("    ");
  Serial.print(pwr);
  Serial.print("    ");
  Serial.print(dir);
  Serial.println();
}
```

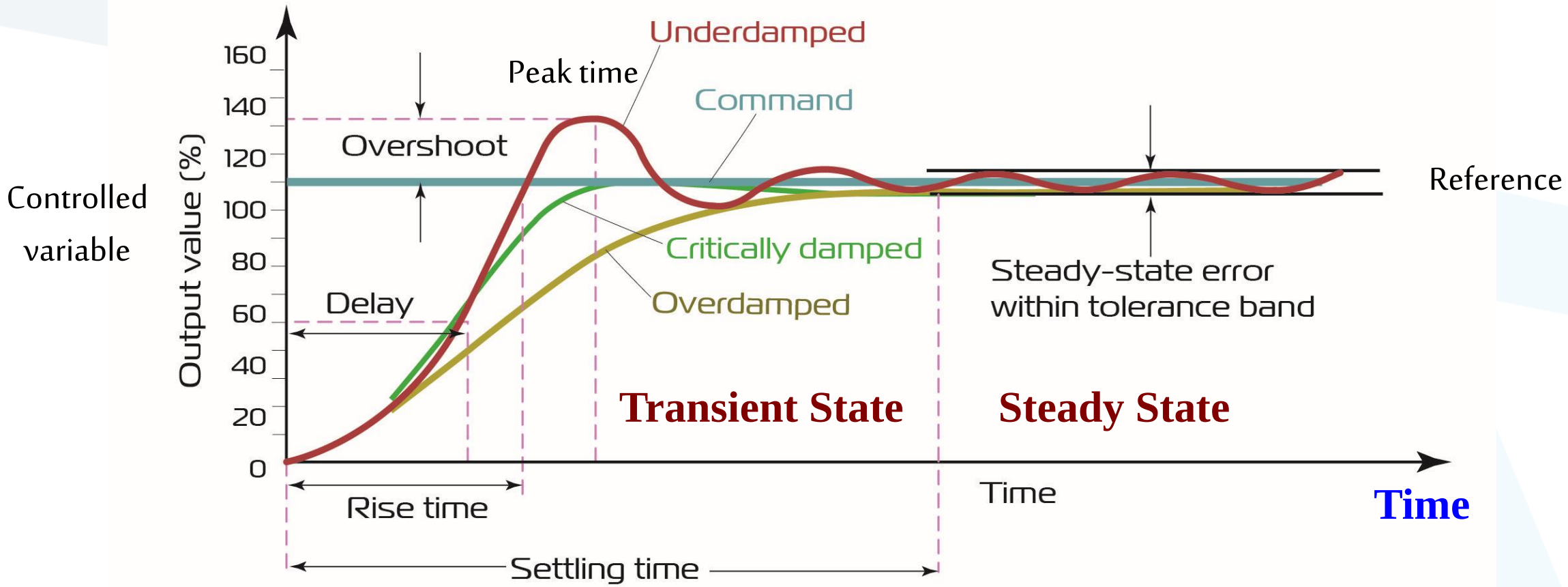
```
void setMotor(int dir, int pwmVal, int pwm_pin, int in1, int in2){

  if(dir == -1){
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    //Serial.println();
    //Serial.print("mmmmmmmmmmmmmmmmm");
    //Serial.println();
  }
  else if(dir == 1){
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    //Serial.println();
    //Serial.print("fffffffffffff");
    //Serial.println();
  }
  else{
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
  }
  analogWrite(pwm_pin,pwmVal);
}
```

```
void ISR_encoder(){
  int b = digitalRead(ENCB);
  if(b > 0){
    pos++;
  }
  else{
    pos--;
  }
}
```



U1(VSS)
U1(VS)
U1
DUINO1
IN1
IN2
EN1
EN2
IN3
IN4
VSS
VS
GND
GND
OUT1
OUT2
OUT3
OUT4
L293D
AREF
RESET
ANALOG IN
DIGITAL (~PWM)
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
PD7/AIN1
PD5/T1
PD3/INT1
PD2/INT0
PD1/TXD
PD0/RXD
TX
RX
ARDUINO UNO R3
RXD
TXD
RTS
CTS

Performance specifications



جَامعة
المَنارة
MANARA UNIVERSITY

انتهت المحاضرة