360 度旋轉編碼器模組

工作原理:

增量編碼器是一種將旋轉位移轉換為一連串數字脈衝信號的旋轉式傳感器。
這些脈衝用來控制角位移。
在 Eltra 編碼器中角位移的轉換採用了光電掃描原理。讀數係統以由交替的透光窗口和不透光窗口構成的徑向分度盤（碼盤）的旋轉為依據，
同時被一個紅外光源垂直照射，光把碼盤的圖像投射到接收器表面上。接收器覆蓋著一層衍射光柵，它具有和碼盤相同的窗口寬度。
接收器的工作是感受光盤轉動所產生的變化，然後將光變化轉換成相應的電變化。
再使低電平信號上升到較高電平，並產生沒有任何干擾的方形脈衝，這就必須用電子電路來處理。
讀數係統通常採用差分方式，即將兩個波形一樣但相位差為 180°的不同信號進行比較，以便提高輸出信號的質量和穩定性。
讀數是再兩個信號的差別基礎上形成的，從而消除了乾擾。

工作電壓：5V 一圈脈衝數：20

顺时针运动	逆时针运动
A B	A B
1 1	1 1
0 1	1 0
0 0	0 0
1 0	0 1

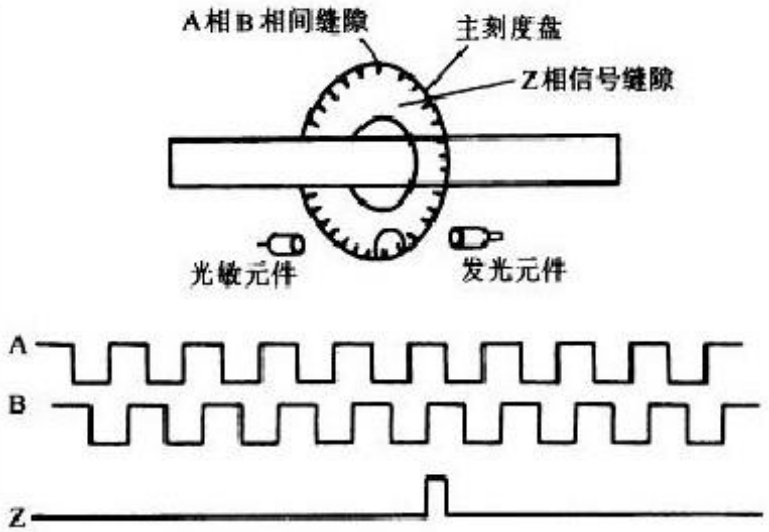
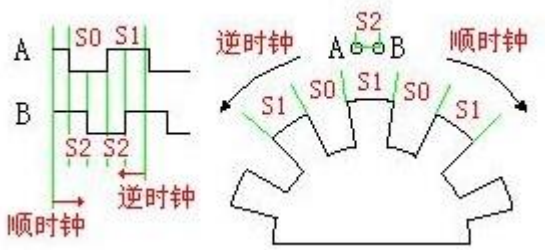
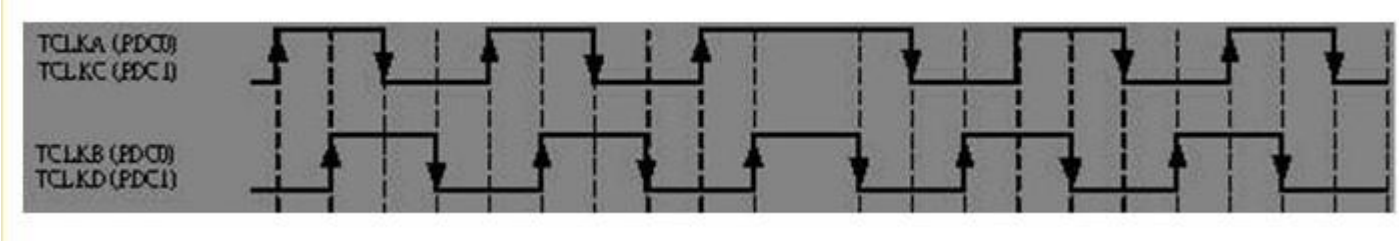


图 3 光电编码器工作原理图及输出波形

增量編碼器給出兩相方波，它們的相位差 90 °，通常稱為 A 通道和 B 通道。其中一個通道給出與轉速相關的信息，與此同時，通過兩個通道信號進行順序對比，得到旋轉方向的信息。還有一個特殊信號稱為 Z 或零通道，該通道給出編碼器的絕對零位，此信號是一個方波與 A 通道方波的中心線重合。



量型編碼器精度取決於機械和電氣兩種因素，這些因素有：光柵分度誤差、光盤偏心、軸承偏心、電子讀數裝置引入的誤差以及光學部分的不精確性。確定編碼器精度的測量單位是電氣上的度數，編碼器精度決定了編碼器產生的脈衝分度。以下用 360 °電氣度數來表示機械軸的轉動，而軸的轉動必須是一個完整的周期。要知道多少機械角度相當於電氣上的 360 度，可以用下列公式來計算：電氣 360 =機械 360 ° /n °脈衝/轉



圖：A、B 換向時信號

編碼器分度誤差是以電氣角度為單位的兩個連續脈衝波的最大偏移來表示。誤差存在於任何編碼器中，這是由前述各因素引起的。Eltra 編碼器的最大誤差為± 25 電氣角度（在已聲明的任何條件下），

相當於額定值偏移± 7%，至於相位差 90 °（電氣上）的兩個通道的最大偏差為± 35 電氣度數相當於額定值偏移± 10%左右。

UVW 信號增量型編碼器

除了 上述傳統的編碼器外，還有一些是與其它的電氣輸出信號集成在一起的增量型編碼器。與 UVW 信號集成的增量型編碼器就是實例，它通常應用於交流伺服電機的反饋。這些磁極信號一般出現在交流伺服電機中，UVW 信號一般是通過模擬磁性原件的功能而設計的。在 Eltra 編碼器中，這些 UVW 信號是用光學方法產生，並以三個方波的形式出現，它們彼此偏移 120 °。為了便於電機啟動，控制電動機用的啟動器需要這些正確的信號。這些 UVW 磁極脈衝可在機械軸旋轉中重複許多次，因為它們直接取決於所連接的電機磁極數，並且用於 4、6 或更多極電機的 UVW 信號。

Arduino 範例程式

```
int redPin = 2;
int yellowPin = 3;
int greenPin = 4;
int aPin = 6;
int bPin = 7;
int buttonPin = 5;

int state = 0;
int longPeriod = 5000; // Time at green or red
int shortPeriod = 700; // Time period when changing
int targetCount = shortPeriod;
int count = 0;

void setup()
{
  pinMode(aPin, INPUT);
  pinMode(bPin, INPUT);
  pinMode(buttonPin, INPUT);
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

void loop()
{
  count++;
  if (digitalRead(buttonPin))
  {
    setLights(HIGH, HIGH, HIGH);
  }
  else
```

```
  {
    int change = getEncoderTurn();
    int newPeriod = longPeriod + (change * 1000);
    if (newPeriod >= 1000 && newPeriod <= 10000)
    {
      longPeriod = newPeriod;
    }
    if (count > targetCount)
    {
      setState();
      count = 0;
    }
  }
  delay(1);
}

int getEncoderTurn()
{
  // return -1, 0, or +1
  static int oldA = LOW;
  static int oldB = LOW;
  int result = 0;
  int newA = digitalRead(aPin);
  int newB = digitalRead(bPin);
  if (newA != oldA || newB != oldB)
  {
    // something has changed
    if (oldA == LOW && newA == HIGH)
    {
      result = -(oldB * 2 – 1);
    }
  }
  oldA = newA;
  oldB = newB;
  return result;
}

int setState()
  {
    if (state == 0)
    {
      setLights(HIGH, LOW, LOW);
```

```
      targetCount = longPeriod;
      state = 1;
    }
    else if (state == 1)
    {
      setLights(HIGH, HIGH, LOW);
      targetCount = shortPeriod;
      state = 2;
    }
    else if (state == 2)
    {
      setLights(LOW, LOW, HIGH);
      targetCount = longPeriod;
      state = 3;
    }
    else if (state == 3)
    {
      setLights(LOW, HIGH, LOW);
      targetCount = shortPeriod;
      state = 0;
    }
  }

void setLights(int red, int yellow, int green)
{
  digitalWrite(redPin, red);
  digitalWrite(yellowPin, yellow);
  digitalWrite(greenPin, green);
}
```