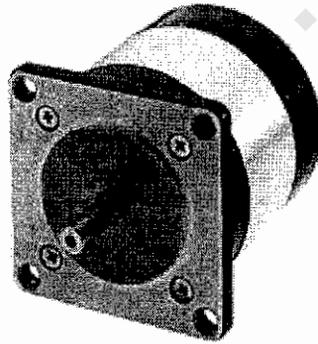


الفصل العاشر

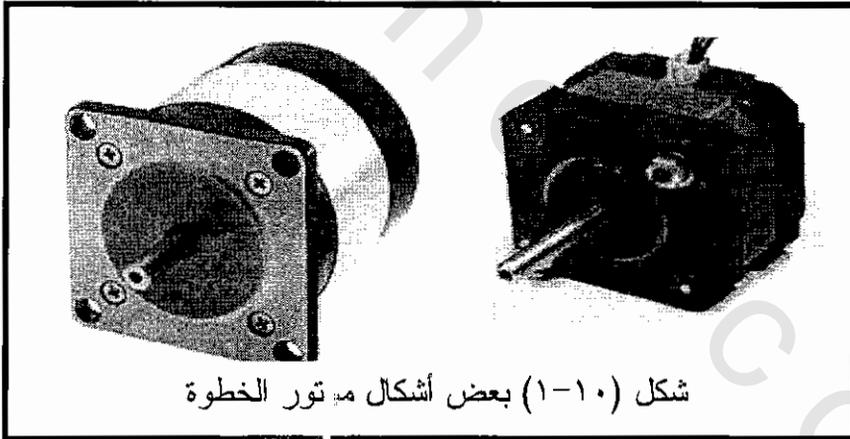
موتور الخطوة

Stepper Motor



١-١٠ مقدمة

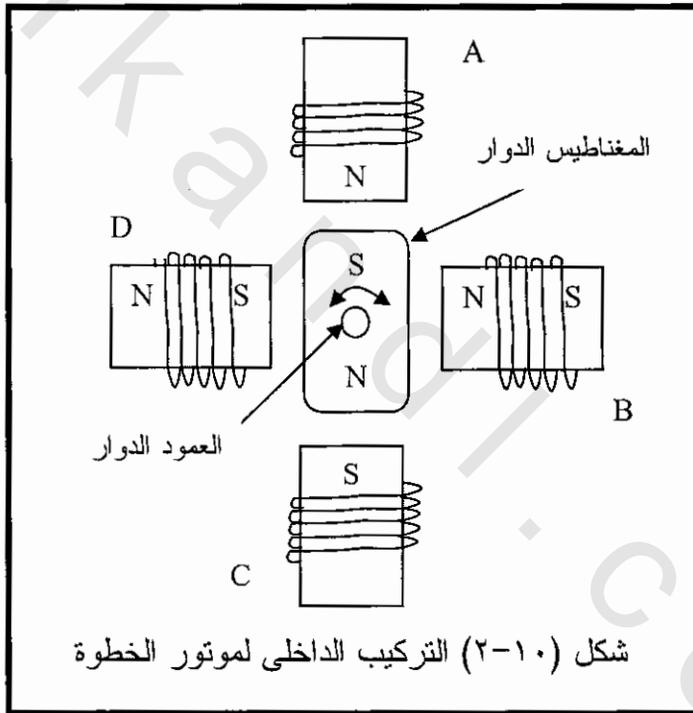
يستخدم موتور الخطوة فى الكثير من التطبيقات التى تتطلب حساب عدد اللغات التى يلفها الموتور بدقة ، أو حتى التى تتطلب وضع الموتور عند كسور صغيرة من اللفة . من مميزات موتور الخطوة على باقى أنواع الموتورات أنه يمكن إدارته باستخدام إشارة رقمية ذات قدرة مناسبة . وعلى ذلك يمكن إدارته باستخدام معالج أو متحكم أو حاسب بسهولة من خلال برنامج ينفذ بأى لغة من لغات البرمجة . شكل (١-١٠) يبين شكلا عاما لبعض مواير الخطوة المتاحة فى السوق . تستخدم مواير الخطوة فى إدارة الطابعات ، والمساحات الضوئية ، والأحجام الصغيرة من الماكينات المحكومة بالحاسب CNC ، وفى الكثير من التطبيقات التى تحتاج للدقة فى وضع الموتور وعدد اللغات وسرعة اللف . هذه المواير لا تحتوى فرش brushes لذلك فإنها لا تحتوى أى احتكاكات ميكانيكية قد تؤثر على أدائها أو الصيانة المستمرة لها . هذا بجانب أن مواير التيار المستمر DC تحتاج لدوائر معقدة للتحكم فى سرعتها ومكان وقوفها ، وهذه الدوائر قد تفوق ثمن الموتور نفسة عند الحاجة للدقة العالية . لذلك فإنه يفضل دائما استخدام موتور الخطوة كلما كانت الأحمال تسمح بذلك .



١-٢ التركيب الأساسى لموتور الخطوة وكيفية عمله

هناك أنواعا عديدة من مواير الخطوة ، ولكن فكرة عملها كلها سهلة وبسيطة وسنقوم بتقديمها فى هذا الجزء . شكل (١-٢) يبين التركيب الأساسى لموتور الخطوة . نلاحظ من هذا الشكل أنه يتكون من ٤ مغناطيسات كهربية مثبتة stator وهى المغناطيسات A و B و C و D . بمرور التيار الكهربى فى الملف

المحيط بأى واحد من هذه المغناطيسات يمكن مغنطة هذا المغناطيس بحيث يكون أحد أطرافه هو القطب الشمالي والآخر هو القطب الجنوبي للمغناطيس . يوجد في مركز الموتور كما نرى في شكل (٢-١٠) مغنطيس دائم حر الدوران حول مركزه الذى يمثل العمود الدوار للموتور rotor . أى أنه بدوران هذا المغنطيس يدور العمود وبالتالي يمكن إدارة الحمل الموصل على الموتور . بمرور التيار الكهربى فى المغناطيسين A و C فى الاتجاه المناسب فإنه يمكن جعل A قطب شمالي و C قطب جنوبي . ونتيجة وجودهما بجانب المغناطيس الدوار فإن المغناطيس الدوار سيعدل من وضعه بحيث يكون قطبه الجنوبي ناحية A وقطبه الشمالي ناحية C كما فى شكل (٢-١٠) لأن الأقطاب المتشابهة تتنافر والمختلفة تتجاذب كما نعرف من قواعد المغناطيسية البسيطة . أى أن الجزء الدوار سيعدل من وضعه بحيث يكون رأسيا كما فى الشكل .

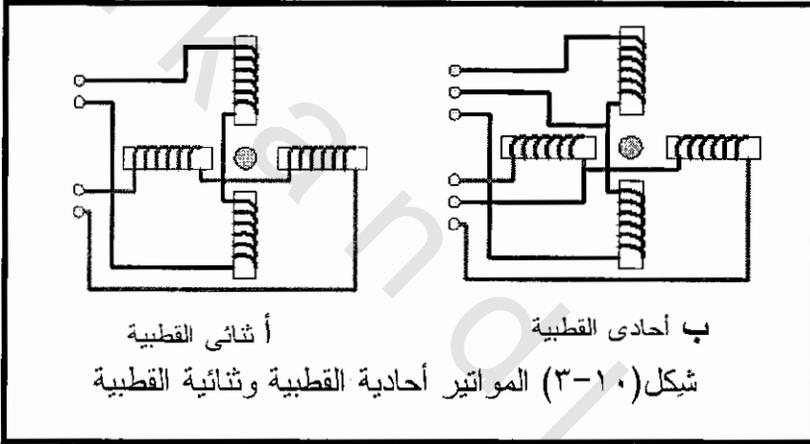


إذا فصلنا التيار عن كل من A و C ، ووصلناه على كل من B و D فإن نفس الشيء سيحدث أيضا بحيث سيدور المغناطيس الدوار ليأخذ الوضع الأفقى فى هذه المرة وسيكون اتجاه الدوران عكس أو مع عقارب الساعة على حسب اتجاه التيار فى كل من B و D . فإذا كان D قطب شمالي و B قطب جنوبي فإن اتجاه الدوران سيكون عكس عقارب الساعة . بتوالى مرور التيار فى الاتجاهات

المناسبة يمكن الاستمرار في دوران المغناطيس الدوار في الاتجاه المطلوب في خطوات كل منها ٩٠ درجة .

١٠-٣ المواتير أحادية وثنائية القطبية

هذا النوع من التغذية لملفات الجزء الثابت من الموتور الذى شرحناه مسبقا يسمى التغذية ثنائية القطبية حيث يتم تغذية ملفين في نفس الوقت كما فى شكل (١٠-١٣). كما نرى من هذا الشكل فإن الموتور ثنائى القطبية يخرج منه ٤ أسلاك ، سلكان من كل ملفين حيث يتم تغذية الملفين مع بعضهما فى كل مرة . وتكون التغذية بحيث يكون أحد الملفين قطب شمالي والآخر جنوبي لتثبيت الجزء الدوار بينهما فى وضع معين .



هناك التغذية الأحادية ، أو الموتور أحادى القطبية الذى يحتوى أيضا ٤ ملفات ولكن يتم تغذية كل ملف على حدة ولذلك يخرج منه ٦ أسلاك حيث يكون الطرف الآخر للتغذية هو الأرضى . أحيانا يتم توصيل طرفى الأرضى من داخل الموتور بحيث يخرج من الموتور ٥ أسلاك بدلا من ٦ . شكل (١٠-٣ب) يبين شكل توضيحي لملفات الموتور أحادى القطبية .

شكل (١٠-٤) يبين شكل توضيحي لطريقة دوران الموتور تبعا لتتابع تغذية الملفات حيث كما نرى فى هذا الشكل فإن مقدار الخطوة سيكون ربع لفة . شكل (١٠-٥) يبين طريقة أخرى لتغذية ملفات الموتور حيث يتم تغذية ملفين متجاورين فى نفس الوقت فينتج عن ذلك وقوف الدوار بين الملفين وسيكون مقدار الخطوة ربع لفة أيضا فى هذه الحالة . يمكن إدارة الموتور عن طريق تغذية ملف واحد حيث يقف الجزء الدوار فى مواجهته ، ثم تغذية نفس الملف

والمجاور له فيقف الجزء الدوار بينهما ، ثم تغذية الملف الأخير وحده فيقف الدوار في مواجهته ، وهكذا حيث نرى أن خطوة الدوران في هذه الحالة ستكون ثمن لفة . شكل (١٠-٦) يبين رسم توضيحي لذلك .

Step	Coil 4	Coil 3	Coil 2	Coil 1	
a.1	on	off	off	off	
a.2	off	on	off	off	
a.3	off	off	on	off	
a.4	off	off	off	on	

شكل (١٠-٤) تغذية موتور أحادى القطبية بحيث يكون مقدار الخطوة ربع لفة

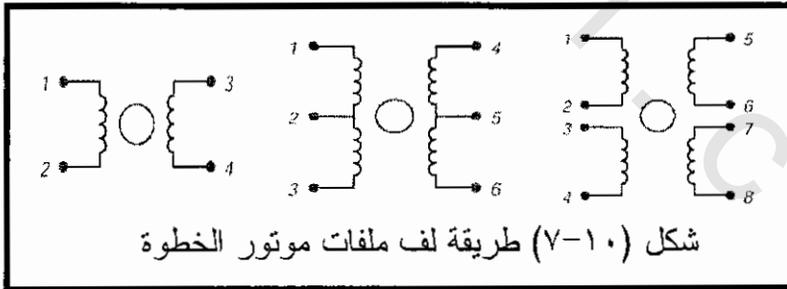
Step	Coil 4	Coil 3	Coil 2	Coil 1	
b.1	on	on	off	off	
b.2	off	on	on	off	
b.3	off	off	on	on	
b.4	on	off	off	on	

شكل (١٠-٥) تغذية ملفين متجاورين في نفس الوقت ، مقدار الخطوة ربع لفة أيضا في هذه الحالة

Step	Coil 4	Coil 3	Coil 2	Coil 1
a.1	on	off	off	off
b.1	on	on	off	off
a.2	off	on	off	off

شكل (٦-١٠) تغذية ملف ، ثم ملفين ، ثم ملف ، وهكذا بحيث ينتج عن ذلك خطوة مقدارها ثمن لفة .

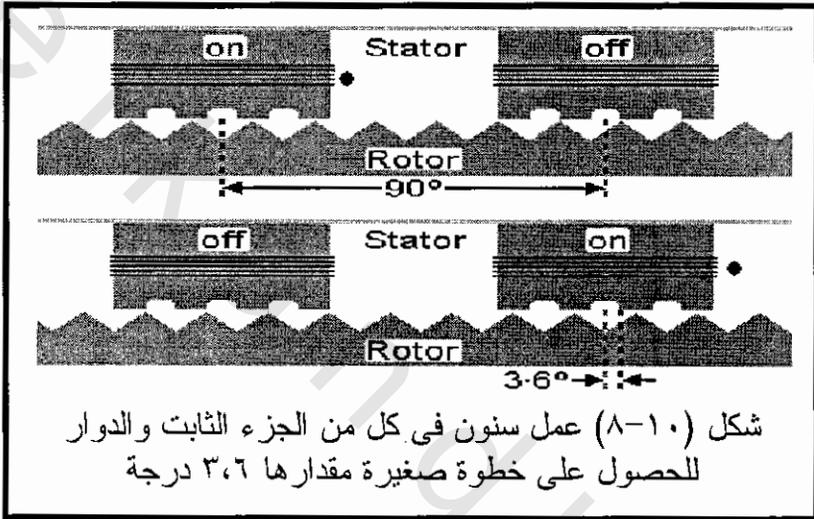
موتور الخطوة يمكن أن يوجد في السوق وله ٤ أطراف (ثنائي القطبية) ، أو خمسة (أو ستة) أطراف (أحادي القطبية) ، أو ثمانية أطراف (عام ، Universal) حيث يمكن توصيله خارجيا في أي من الحالتين السابقتين كما رأينا . شكل (٧-١٠) يبين هذه الملفات .



١٠-٤ مقدار خطوة الموتور

يمكن زيادة عدد الخطوات في اللفة عن طريق عمل سنون في كل من المغناطيس الدوار والثابت كما في شكل (٨-١٠) . إذا كان عدد السنون في الدوار ٢٥ سنة وكل مغناطيس من الأربعة المثبته به ٤ سنون ، فإنه عند تغذية

أحد الملفات الثابتة تكون أسنان الدوار في منتصف أسنان الثابت ومقابلة لها تماما كما في حالة المغناطيس العلوى الأيسر (on) والسفلى الأيمن (on) كما في شكل (٨-١٠) . في نفس الوقت تكون أسنان الدوار قريبة من أسنان الثابت الذى عليه الدور في التغذية بمقدار ربع سنة من أسنان الجزء الثابت . عند تغذية الملف الذى عليه الدور تتحرك أسنان الدوار بحيث تتطابق مع أسنان هذا الثابت وستكون حركتها بمقدار ربع سنة من أسنان الجزء الثابت . وعلى ذلك فإن مقدار الزاوية التى يتحركها الدوار تكون $360 \div (4 \times 25) = 3.6$ درجة . أى أنه سيكون هناك ١٠٠ خطوة فى كل لفة .



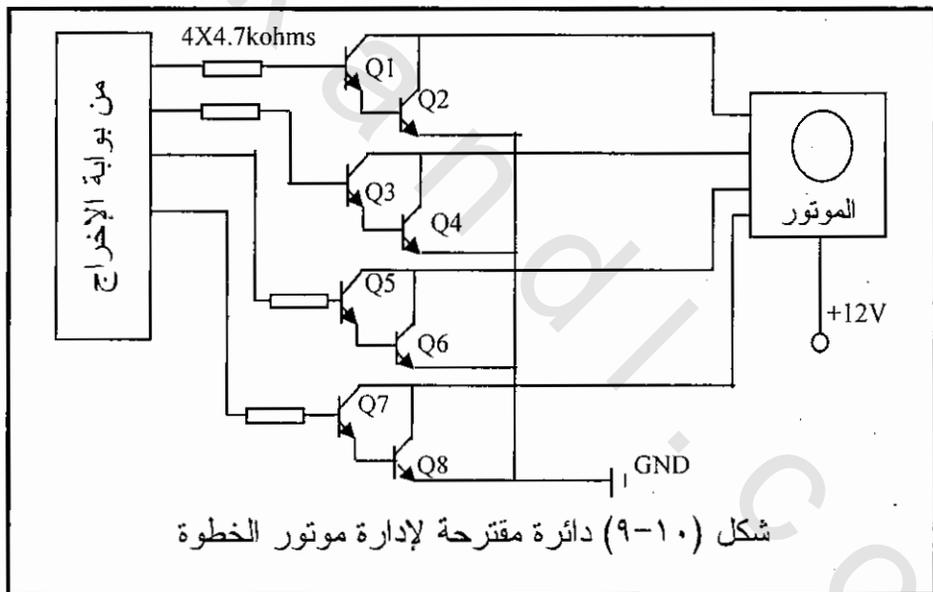
تتحدد سرعة الموتور بسرعة تغير تغذية هذه الملفات . كلما زادت هذه السرعة كلما كانت حركة الموتور أكثر استمرارا ومع تقليل هذه السرعة تكون حركة الموتور فى صورة خطوات . يجب ألا تزيد سرعة تغير تغذية الملفات الثابتة عن حد معين بحيث يستطيع الجزء الدوار ملاحقة هذه السرعة . إذا زادت السرعة عن هذا الحد ربما تسمع زنة للموتور وقد يدور بسرعة أبطأ بكثير جدا من المتوقعة نتيجة الخطوات المفقودة لأن الموتور غير قادر على ملاحقة تتابع تغذية الملفات نتيجة القصور الذاتى للأجزاء الدوارة .

١٠-٥ إدارة موتور الخطوة

شكل (٩-١٠) يبين دائرة مقترحة لإدارة موتور الخطوة . إن الخرج الرقمى القادم من بوابة الإخراج من أى متحكم أو معالج مباشرة لا يكون مناسباً لإدارة

الموتور لأن ملفات الموتور تحتاج لتيار أكثر مما يستطيع هذا الخرج توفيره . لذلك فإننا نقدم هذه الدائرة كمثال فقط لدائرة الموائمة لموتور الخطوة وهناك العديد من الدوائر الأخرى والشرائح التي يمكن استخدامها لهذا الغرض . في الدائرة الموجودة في شكل (٩-١٠) الترانزستورات Q1 و Q2 تمثل زوج دارلنجتون Darlington pair التي تتميز بتوفير التيارات العالية ، وبنفس الطريقة أزواج الترانزستورات الأخرى .

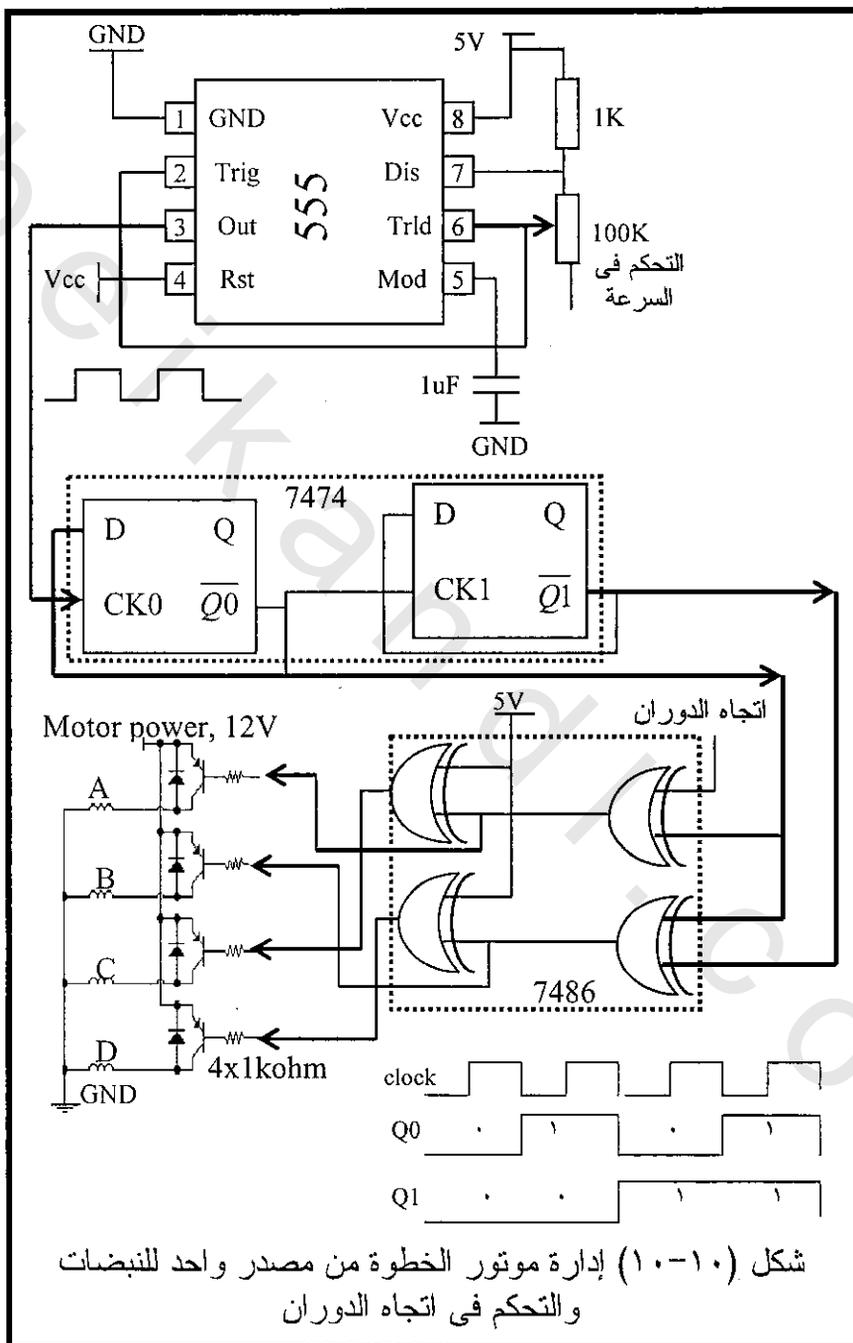
الترانزستورات Q1 و Q3 و Q5 و Q7 من النوع 2N5551 و الترانزستورات Q2 و Q4 و Q6 و Q8 من النوع TIP31 . يمكن استخدام شرائح دارلنجتون الجاهزة في شريحة واحدة بدلا من تركيبها من ٢ ترانزستور كما في الشكل السابق . في الدائرة الموجودة في شكل (٩-١٠) تم تغذية التتابع في صورة ثنائية من بوابة إخراج . شكل (١٠-١٠) يبين دائرة تتحكم في إدارة موتور من موجة واحدة مربعة يتم تحويلها إلى تتابعات من ٤ بت اللازمة لإدارة الموتور باستخدام عدادين ثنائيين .



شكل (١٠-١٠) يبين أيضا التتابعات الناتجة حيث تم استخدام الشريحة 555 في وضع عديم الاستقرار للحصول على موجة مربعة نتحكم في ترددها وبالتالي في سرعة الموتور عن طريق تغيير المقاومة ١٠٠ كيلو أوم المتغيرة . هذه النبضات تم إدخالها على عدادين ثنائيين كل منهما يقسم الدخل على ٢ . حيث كما نرى من شكل (١٠-١٠) سيكون التتابع الناتج على الخرج Q وبالتالي يمكن استنتاجه على \bar{Q} لكل من القلايين هو :

$Q1Q0=00, 01, 10, 11, 00, \dots$

بعد ذلك تم إدخال هذا التتابع على دوائر XOR لتحويله من شفرة ثنائية إلى شفرة رباعية بالتتابع التالي :



1100, 0110, 0011, 1001, 1100, ...

هذه الشفرة الناتجة هي اللازمة لإدارة الموتور . لاحظ وجود خط التحكم في اتجاه الدوران الذى حينما يكون صفر يجعل الموتور يدور فى اتجاه معين ، وإذا كان واحد يجعل الموتور يدور فى الاتجاه الآخر .

مثال ١٠-١ :

برنامج إدارة موتور الخطوة عكس عقارب الساعة على أن يظل الموتور يدور إلى أن يتم ضرب أى زر من لوحة المفاتيح ، بلغة C من الممكن أن يكون كما يلي :

جدول ٥-٢ المتتابع الثنائى المطلوب
للدوران عكس عقارب الساعة

D	C	B	A
1	0	1	0
0	1	1	0
0	1	0	1
1	0	0	1
بداية تكرار المتتابع			

جدول ٥-٣ المتتابع الثنائى المطلوب
للدوران مع عقارب الساعة

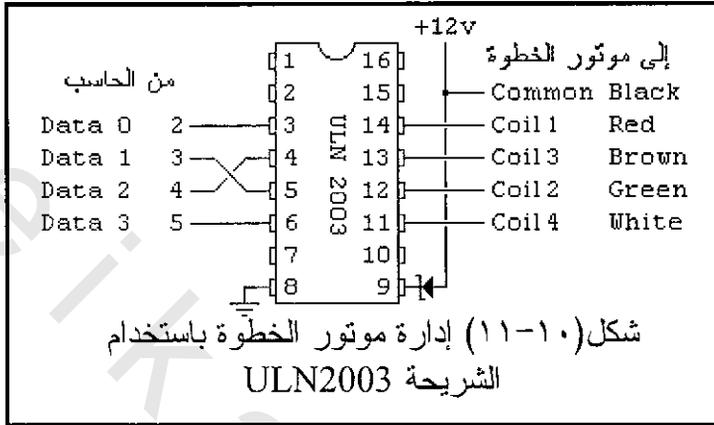
D	C	B	A
1	0	0	1
0	1	0	1
0	1	1	0
1	0	1	0
بداية تكرار المتتابع			

```
#define delay 0.005
int main(void)
{ do{ outportb(0x300H,0AH);
  sleep(delay);
  outportb(0x300H,06H);
  sleep(delay);
  outportb(0x300H, 05H);
  sleep(delay);
  outportb(0x300H, 09);
} while(!kbhit());
}
```

حاول تنفيذ هذا البرنامج مع استخدام أزمنة تأخير مختلفة قصيرة بشكل ملحوظ ، وطويلة بشكل ملحوظ أيضا لترى استجابة الموتور لسرعة تتابعات التغذية . يمكنك تجربة هذا البرنامج من المخرج المتوازي للحاسب .

شكل (١٠-١١) يبين استخدام الشريحة ULN2003 لإدارة الموتور تبعا للمتتابع السابق القادم من المخرج المتوازي للحاسب . نلاحظ من هذا الشكل أن الشريحة

لها أربع مداخل توصل على الأربع مخارج القادمة من المخرج التوازي ، وأربع أسلاك أخرى توصل على ملفات الموتور كما في الشكل . يجب توصيل الدايمود الموصل على الطرف ٩ في الشريحة لحماية الشريحة من أى تيارات عكسية نتيجة تغير الجهد على ملفات الموتور .



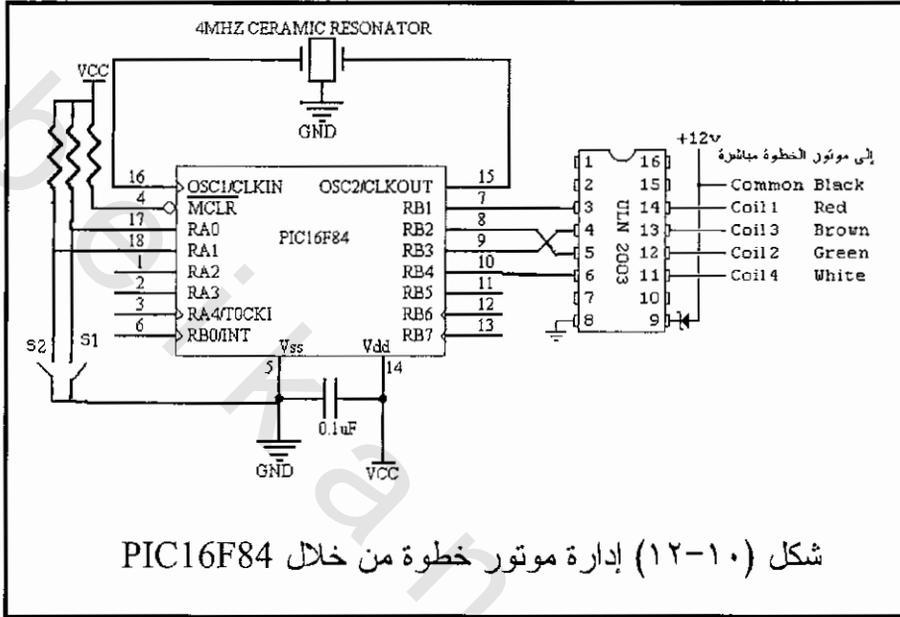
مثال ١٠-٢

برنامج إدارة موتور الخطوة باستخدام المتحكم PIC16F84 . سنستخدم البوابة B في هذا المتحكم كبوابة إخراج نستخدم منها فقط ٤ أطراف نخرج عليهم الشفرات المطلوبة لإدارة الموتور . سنستخدم طرفان من البوابة A كأطراف دخل توصل على الطرف الأول منهما مفتاح S1 عندما يكون مضغوط يتم تشغيل الموتور . المفتاح الثاني S2 موصل على الطرف A2 بحيث يتم اختيار اتجاه الدوران هل سيكون لليمين أم لليسار . شكل (١٠-١٢) يبين الدائرة المقترحة لذلك .

البرنامج فكرته بسيطة حيث سيكون عبارة عن حلقة لا نهائية تقرأ الطرف A0 فإذا كان واحد فإن ذلك يعنى أن المفتاح S1 مفتوح (غير مضغوط) وفى هذه الحالة لا تعمل شئ سوى قراءة نفس الطرف مرة أخرى مع جعل الموتور واقف عن طريق تفسير أطراف الموتور . إذا تم ضغط المفتاح S1 فإن الطرف A0 يصبح صفراً وفى هذه الحالة يتم الاستمرار فى البرنامج حيث يتم اختبار الطرف A1 لنرى هل المفتاح S1 بصفراً أم واحد لكى نقرر هل سيكون الدوران لليمين أم لليسار ، حيث بعدها نخرج شفرات الدوران على حسب الاتجاه المطلوب وبين كل شفرة والثانية زمن تأخير مقداره ٢٥ ميلي ثانية .

البرنامج الذى يحقق ٢٥ ميلي ثانية ينادى على برنامج فرعى آخر يحقق ٢,٥ ميلي ثانية ١٠ مرات . البرنامج الفرعى الذى يحقق ١ ميلي ثانية يحتوى حلقتين يتم تنفيذ الداخلية فيهما ٢٤٩ مرة بحيث أن مجموع دورات الأمر فى هذا

البرنامج الفرعى هو ٢٥٠١ دورة أمر . بفرض أن المتحكم يعمل بنبضات تزامن مقدارها ٤ ميغاهرتز كما فى شكل (٥-٦) فإن زمن مجموع الأوامر فى هذا البرنامج الفرعى سيكون (2501x1usec=2.5msec) . قائمة أوامر البرنامج ستكون كما يلى :



```

;***** Label Definition *****
ra0          equ    00          ;PA0 bit
cnt500u     equ    0c          ;500usec counter Address
cnt25m      equ    0e          ;100msec counter Address
;***** Program Start *****
org 0       ;Reset Vector
goto init
org 4       ;Interrupt Vector
goto init
;***** Initial Process *****
org 5
init
    bsf STATUS,RP0 ;Change to Bank1
    movlw h'ff'    ;Set input mode data
    movwf TRISA    ;Set PORTA to Input mode
    clrf TRISB     ;Set PORTB to Output mode
    bcf STATUS,RP0 ;Change to Bank0

```

```

    movlw h'00'      ;إخراج أصفار لوقف الموتورر
    movwf PORTB      ;Output data.
keyscan
    btfss PORTA,ra0  ;RA0 ON(Low lebel) ?
    goto Drive
    goto keyscan

Drive
    btfss PORTA,ra1
    goto anticlockwise

Clockwise
    movlw h'0A'
    movwf PORTB
    call t25m
    movlw h'06'
    movwf PORTB
    call t25m
    movlw h'05'
    movwf PORTB
    call t25m
    movlw h'09'
    movwf PORTB
    call t25m
    goto keyscan

Anticlockwise
    movlw h'09'
    movwf PORTB
    call t25m
    movlw h'05'
    movwf PORTB
    call t25m
    movlw h'06'
    movwf PORTB
    call t25m
    movlw h'0A'
    movwf PORTB
    call t25m
    goto keyscan

```

```

;***** 25msec Timer Subroutine *****
T25m    movlw d'10'      ;Set loop counter

```

```

movwf cnt25m      ;Save loop counter
tm2lp             call t1m          ;1msec subroutine
                  decfsz cnt25m,f   ;cnt100m - 1 = 0 ?
                  goto tm2lp        ;No. Continue
                  return            ;Yes. Count end
;***** 2.5msec Timer Subroutine *****
t1m               movlw d'2'        ;(1) Set loop cnt1
                  movwf cnt1m      ;(1) Save loop cnt1
tm1lp1            movlw d'249'      ;(1)*2 Set loop cnt2
                  movwf cnt500u    ;(1)*2 Save loop cnt2
tm1lp2            nop              ;(1)*249*2 Time adjust
                  nop              ;(1)*249*2 Time adjust
                  decfsz cnt500u,f  ;(1)*249*2 cnt500u-1=0 ?
                  goto tm1lp2      ;(2)*248*2 No, continue
                  decfsz cnt1m,f    ;(1)*2 cnt1m-1=0 ?
                  goto tm1lp1      ;(2) No. Continue
                  return            ;(2) Yes. Cnt end
;Total 2501*1usec=2.5msec
end

```

مثال ١٠-٣

سنحاول في هذا المثال التحكم في سرعة موتور الخطوة واتجاه دورانه عن طريق تحديد هذه البيانات على شاشة في الباسيك المرئى Visual Basic ثم إرسال هذه البيانات من خلال المخرج المتوالى للحاسب إلى المتحكم 8051 الذى يقوم بدوره بإدارة الموتور بالسرعة المطلوبة وفي الاتجاه المطلوب . هذا المثال يعتبر تدريباً جيداً على التعامل مع المخرج المتوالى للحاسب والمتحكم وموتور الخطوة . الدائرة التى سنستخدمها هنا هي نفس الدائرة التى استخدمناها من قبل فى الفصل التاسع فى شكل (٩-١٣) والتى تم إعادتها هنا مع بعض التعديل لتوصيل الموتور بدلاً من لمبات البيان كما فى شكل (١٠-١٣) .

شكل (١٠-١٤) يبين محتويات شاشة التصميم فى الباسيك المرئى والتى من خلالها سيتم الاتصال بالمتحكم من خلال المخرج المتوالى . فى هذه الشاشة من المفروض أن يبدأ المستخدم بوضع زمن التأخير لكل خطوة من خطوات الموتور والتى بها يحدد سرعة الموتور فى مربع النص الموجود فى الشاشة بحيث أنه كلما زاد هذا الرقم كلما كان الموتور أبطأ . بعد ذلك يختار المستخدم الاتجاه المطلوب وذلك بتعليم مربع العلامة check box اليمين أو اليسار ثم يضرب الزرار RUN . فى الحال من المفروض أن يبدأ الموتور فى الدوران . يمكن

للمستخدم أن يوقف الموتور بالضرب على الزرار STOP . لا ننسى إضافة أداة التحكم في الاتصالات Comm. Port كما في الشكل .

يبدأ المستخدم باختيار اتجاه الدوران عن طريق النقر على أحد مربعي التعليم والشفرة لكل منهما ستكون كالتالي :

Private Sub Option1_Click()

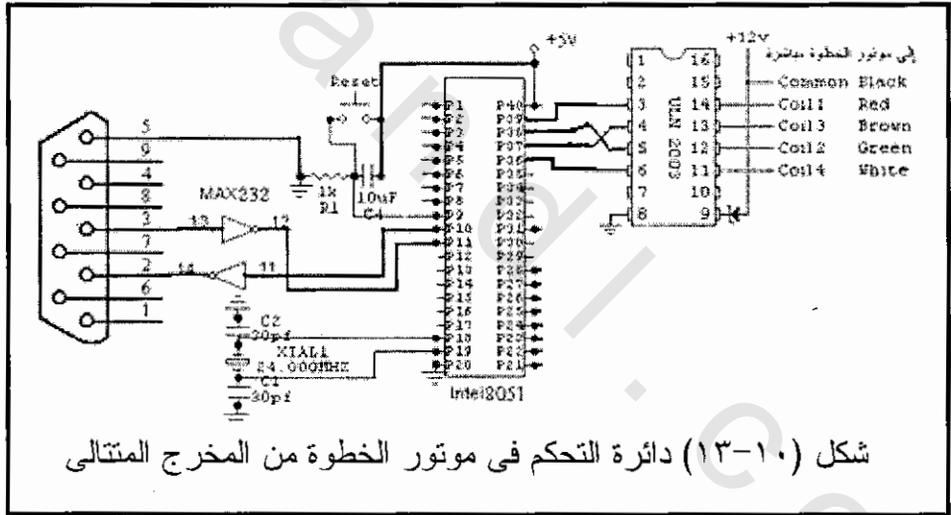
Direction = 1 تعنى اختيار الاتجاه اليمين

End Sub

Private Sub Option2_Click()

Direction = 0 تعنى اختيار الاتجاه لليساار

End Sub



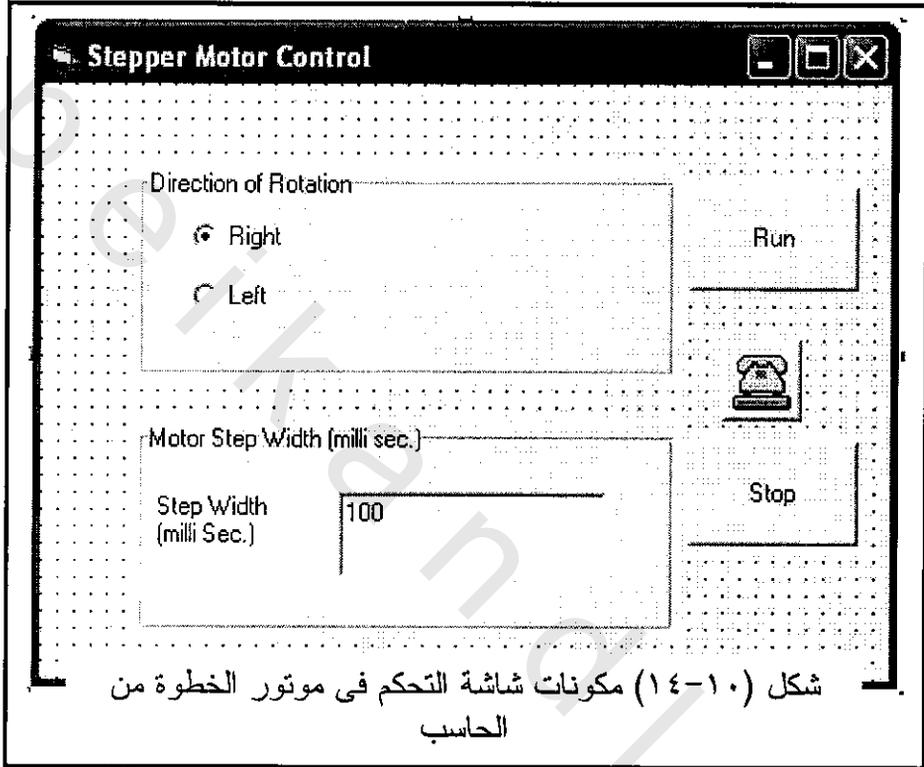
بعد ذلك يحدد المستخدم زمن خطوة الموتور بوحدة الميلي ثانية في مربع النص الموضح . فمثلا عندما يكتب المستخدم الرقم ١٠٠ فذلك يعني أن الموتور سيتحرك ١٠ خطوات في الثانية الواحدة . بذلك تكون كل المعلومات المطلوبة متاحة ، لذا عند ضغط المستخدم للمفتاح Run فإن الحاسب يرسل الشفرة ٧٧ للمتحكم الدقيق والتي تعني تشغيل الموتور ، شفرة ذلك ستكون كالتالي كما في شفرة الزرار RUN :

Private Sub Command1_Click()

```

If MSComm1.PortOpen = False Then
    MSComm1.PortOpen = True
End If
MSComm1.Output = Chr(77)
End Sub

```



بعد ذلك يقوم المتحكم الدقيق بطلب كل من اتجاه الدوران وزمن الخطوة من الحاسب ويتم ذلك من خلال حدث الاستقبال للمخرج المتوالي . الشفرة التالية هي شفرة أداة التحكم في الاتصالات :

```

Private Sub MSComm1_OnComm()
    MSComm1.InputLen = 1
    MSComm1.RThreshold = 1
    Dim command As Integer
    Select Case MSComm1.CommEvent
        Case comEvReceive
            commdata = MSComm1.Input
            command = Asc(commdata)
            Select Case command

```

Case 88 هذا الحرف يعنى سؤال عن الاتجاه

MSComm1.Output = Direction

Case 89 سؤال عن مقدار الخطوة

MSComm1.Output = Char(Val(Text1.Text))

End Select

End Select

End Sub

نلاحظ من هذه الشفرة أنه بمجرد أن يضرب المستخدم الزرار RUN فإن الحاسب يرسل الحرف 77 إلى المتحكم ، فيرد عليه المتحكم بالحرف 88 الذى يعنى طلب اتجاه الدوران فيرسله الحاسب (1 أو 0) ، بعدها يطلب المتحكم مقدار زمن التأخير فيرد الحاسب بإرسال الرقم الموجود في مربع النص . بذلك يصبح المتحكم جاهزا لإدارة الموتور كما في الشفرة التالية التى سيتم حرقها في ذاكرة البرمجة للمتحكم :

؛stepper motor Control

؛CRYSTAL 11.05920 MHZ.

*****؛

ORG 0000H

LJMP MAINProg للقفز عبر عناوين خدمة المقاطعة.

ORG 0080H

*****؛

MAINProg:

الجزء التالي من الشفرة يهيئ البوابة المتوالية ويحدد معدل إخراج البتات (baud rate) . السطر الأول من الشفرة يحدد التعامل مع المؤقت T1 فقط ، السطر التالي يضع هذا المؤقت في وضع إعادة التحميل الأتوماتيكي . السطر الثالث يحمل المؤقت بالرقم FDH الذي يجعل معدل إخراج البيانات ٩٦٠٠ بت/ ثانية ، ولكن ذلك يحتاج أيضا كما علمنا إلي جعل البت smod صفرا وهو ما تم في السطر الرابع :

ANL tmod, #0fh

orl tmod, #20h

MOV TH1, #FDH

anl pcon, #7Fh

أما الشفرة التالية فتحدد شكل الإرسال المتوالي وهو بت للبدائية ثم ٨ بتات بيانات ثم بت واحدة للنهاية كما تجعل المتحكم الدقيق جاهزا لاستقبال البيانات عن طريق تفعيل البت (REN = Receive Enable) . السطر الذي يليه يبدأ تشغيل المؤقت T1 لإخراج التردد المطلوب.

MOV SCON,#50H
SETB TR1

الجزء التالي من الشفرة يجعل المتحكم الدقيق في انتظار لاستقبال شفرة الأمر الأول من الحاسب . وعند وجود شفرة ذلك الأمر في المسجل SBUF فإنه يتم نقله إلي المرسم لتحديد طبيعة الأمر .

here:

JNB RI, here
Command: MOV A,SBUF
CLR RI

السطر التالي يقارن البايت المستقبلية بالرقم ٧٧ فإن ساوته فإن البرنامج يقوم بالقفز إلي جزء آخر منه خاص بتشغيل الموتور . أما إذا كانت البايت المستقبلية غير ذلك فإن البرنامج يوقف المحرك ويتم ذلك عن طريق القفز إلي بداية البرنامج وانتظار أي أمر جديد دون فعل أي شيء . الشفرة التالية تحقق ذلك :

CJNE A,#77,STOP
LJMP run

Stop:

LJMP here

الجزء التالي من البرنامج هو المسئول عن تشغيل الموتور . أولاً يقوم البرنامج بإرسال الرقم 88 للحاسب والذي يعني أن المطلوب هو إرسال اتجاه الدوران . بعد ذلك ينتظر البرنامج هذه المعلومة من الحاسب وعند استقبالها فإن البرنامج يخزنها في المسجل R0 . الشفرة التالية توضح ذلك :

run:

MOV A,#88
ACALL send

here2:

JNB RI,here2
MOV R0,SBUF
CLR RI

ثانياً يرسل البرنامج الرقم 89 إلي الحاسب والذي يعني أن المطلوب هو زمن الخطوة بالميللي ثانية . بعد ذلك ينتظر البرنامج هذه المعلومة أيضاً وعند استقبالها فإن البرنامج يخزنها في المسجل R1 . الجزء التالي من الشفرة يوضح ذلك :

MOV A,#89
ACALL send

here3:

JNB RI,here3

```
MOV R1,SBUF
CLR RI
```

بداية من هذه النقطة فإن البرنامج تكون لديه كل المعلومات المطلوبة . يقوم البرنامج أولا بالاعداد لعملية التشغيل عن طريق وضع الرقم 08H في المرمك والذي يقابل 00001000 في النظام الثنائي . من المعروف أنه لتشغيل موتور الخطوة فإنه يتم إرسال الرقم 1000 له ثم ترحيل هذا الواحد جهة اليمين حتي النهاية ثم تكرار ذلك ، وعند عكس اتجاه الترحيل فإن الموتور يدور في الاتجاه المعاكس . البرنامج هنا يقوم بذلك عن طريق تكوين الرقم وترحيلاته في المرمك تباعا مع ارسالها إلي البوابة P0 . الشفرة التالية تقوم أولا بتحميل المرمك بالرقم 08H ثم تختبر المسجل R0 فإن كان لا يساوي الواحد فإن البرنامج يقوم بالقفز لجزء آخر يقوم بتشغيل الموتور جهة اليسار وإلا فإنه يقوم بتشغيله جهة اليمين :

```
MOV A,#08H
CJNE R0,#01H,TURN_LEFT
```

TURN_RIGHT:

الجزء التالي من البرنامج يوضح عملية تشغيل الموتور جهة اليمين حيث يقوم البرنامج باخراج الرقم 08H ثم استدعاء برنامج فرعي للتأخير بالميللي ثانية حيث يتحدد زمن التأخير فيه بمحتويات المسجل R1 . بعد ذلك يرسل البرنامج محتويات المرمك جهة اليمين لتوليد الرقم التالي الذي سيرسل إلي البوابة P0 . نلاحظ أن البرنامج يقوم باختبار هذه المحتويات لأنه بعد 4 خطوات فإن محتويات المرمك تصبح صفرا وبالتالي يجب إعادة وضع الرقم 08H في المرمك مرة أخرى . كما يجب أيضا تصفير علم الحمل . الجزء التالي من البرنامج يوضح ذلك :

```
MOV P1,A
LCALL DELAY
RRC A
CJNE A,#00H,NEXT
MOV A,#08H
CLR C
```

نظرا لأنه من الممكن طلب إيقاف الموتور من قبل الحاسب في أي لحظة لذا فإن البرنامج يختبر علم الاستقبال فإن كان صفرا، فإن البرنامج يولد الخطوة التالية للموتور أما إن كان واحدا فذلك يعني وجود أمر من الحاسب إما بسرعة أو اتجاه جديدين للدوران أو أمر بإيقاف الموتور لذا فإن البرنامج يقوم بالقفز إلي بداية البرنامج لقراءة الباييت المستقبلية ووضعها في المرمك وتكرار الجزء السابق شرحه :

```
NEXT: JNB RI,TURN_RIGHT
LJMP COMMAND
```

الجزء التالي من البرنامج يقوم بتشغيل الموتور جهة اليسار وهو مشابه لدرجة كبيرة للجزء الخاص بدوران الموتور جهة اليمين :

```
TURN_LEFT:
    CLR C
    MOV P1,A
    LCALL DELAY
    RLC A
    CJNE A,#10H,NEXT2
    MOV A,#01H
```

```
NEXT2:
    JNB RI,TURN_LEFT
    LJMP COMMAND
```

البرنامج الفرعي التالي يقوم بإرسال البايت الموجودة بالمركم عبر المخرج المتوالي للمتحكم الدقيق وعند الانتهاء فإن علم نهاية الإرسال TI يصبح واحدا :

```
send:
    MOV SBUF,A
```

```
h:
    JNB TI,h
    CLR TI
```

```
RET
البرنامج الفرعي التالي يعطي زمن تأخير مساوي لمحتويات R1 بالميللي ثانية :
```

```
delay:
    MOV R2,#250
again: DJNZ R2,again
    DJNZ R1,delay
```

```
RET
END
```