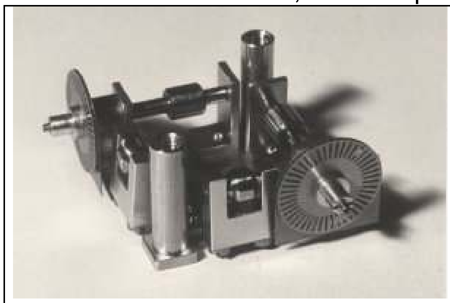
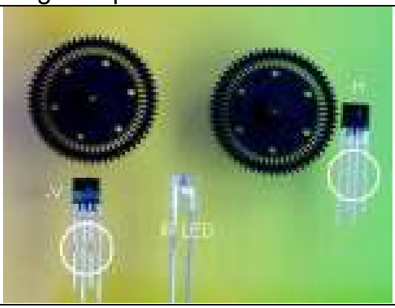
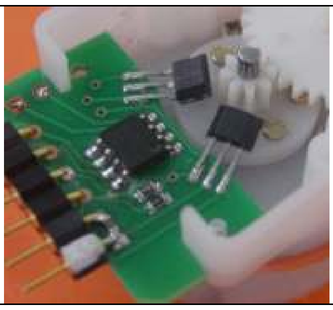
CH-1092 Belmont, Switzerland

Email info@didel.com, www.didel.com

www.didel.com/mot/RomEnco.pdf

Measuring speed and position with a quadrature encoder

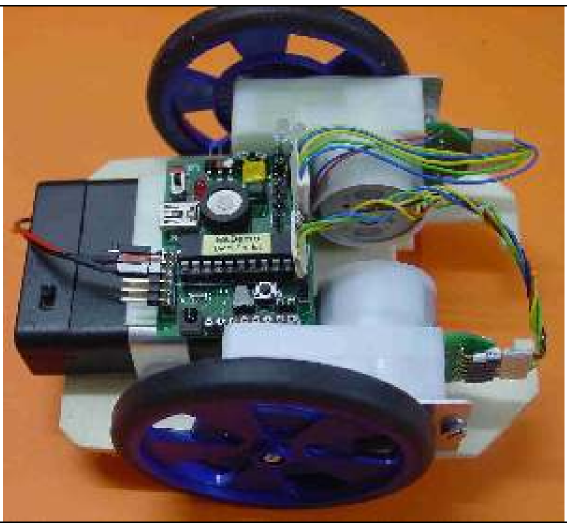
Till recently, all computer mice had two quadrature encoders. Optical encoders developed by Logitech are available from Didel, with compatible gear/optical masks.

		
Archeo-mouse encoder (1978)	7001V and 7001H circuits 120 transistions per turn	RomEnco 12 transistions per turn www.didel.com/Rome.pdf

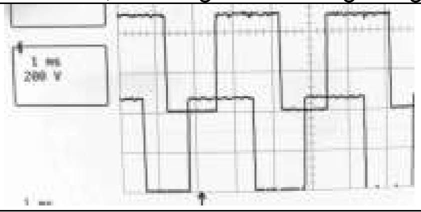
For a robot, there is no need for a very high resolution, which imply dedicated circuit or processor. The Bo10/Gm17 motor is a low cost and effective solution for measuring precisely distances and speed, missing on most robots.
see www.didel.com/mot/Rome.pdf
and www.didel.com/mot/RomEnco.pdf

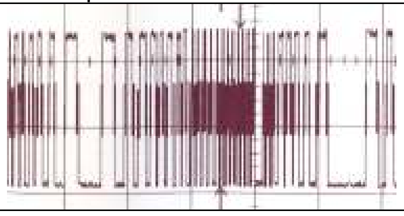
Nouveau www.didel.com/kidules/KiEnco.pdf
Avec des programmes en C/Pinguino/Arduino

Translation soon. Ask if you are impatient
info@didel.com



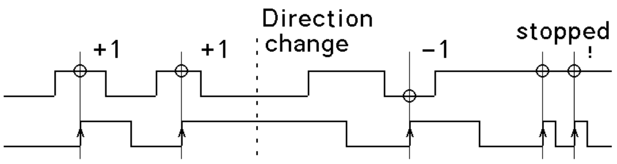
A quadrature encoder uses two sensors to generate two signals shifted by 90 degrees, so that the transition of one signal occurs in the midlle phase of the other signal. If the motor spins in one direction, one signal is enough to get the speed and distance.





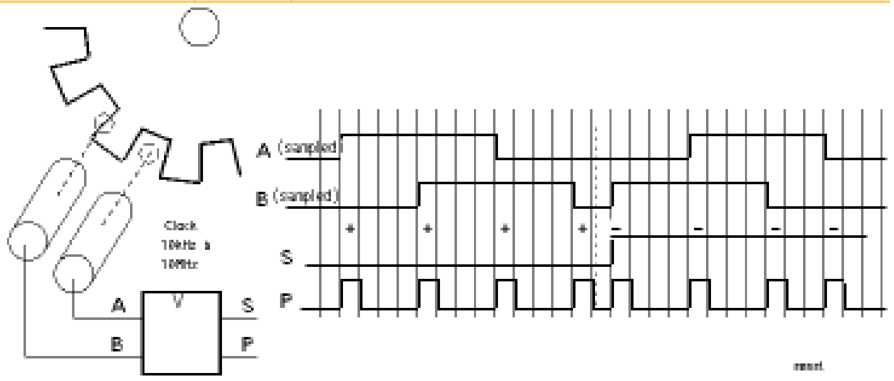
The distance is obtained by counting the pulses. If the motor stops completely before changing direction, the sign of the motor power can decide if the distance is increasing or decreasing, but this is not fully reliable. The speed is either obtained by counting pulses in a given time, or measuring with a timer the period of the pulse.

When the motor changes direction, oscillate around a position, one sensor will just give erratic values. Two sensors and a simple software (but fast enough) will never be lured.
A simple solution is to take e.g. the positive transition on one line, and sample at that moment the other signal. If high, the encoder turns in one direction. If low, it turns the other. But now if you get small oscillations of the sampling edge when the motor is stopped, the count will change. Bad.



Doing things correctly is not so difficult. Most interfacing books present the state machine, which can be asynchronous (interrupts at each edge) or synchronous (signals is samples frequently enough not to miss a transition). Let us take that synchronous approach, and test both signals for changes.

Testing the signal combination at every change allows to decide for the direction. Counting the change pulses will be used for defining the angle/distance and speed..



Any processor can be programmed to sample and decode the quadrature signals and update an up/down counter. The routine needs only 10-20 instructions, but must be executed frequently enough not to miss a transition. If a mechanical rotary encoder is read, sampling must be every 1-5 millisecond for filtering contact bounces.

The following explanations develop what is found on page 15-16 of our document on PIC programming : http://www.didel.com/picg/doc/PicSoft.pdf

The principle is to keep the previous value A- and B- and compute the exclusive OR of combined signals.

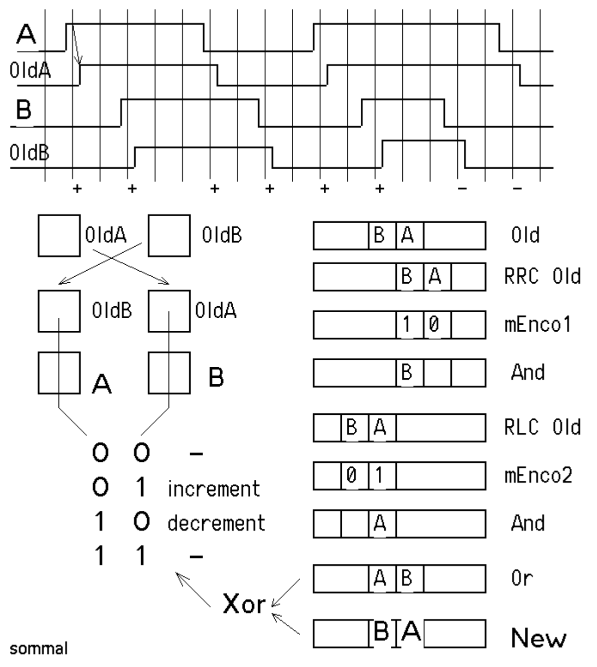
We propose two algorithms, developped in the 80's by Rene Sommer, chief engineer at Logitech and we give examples for the Microchip 10F/12F/16F microcontrollers.

If you prefer solutions documented on the Web, see http://www.mcmanis.com/chuck/robotics/projects/lab-x3/quadratrak.html for another description of our algorithm, slightly less compact.
For a not very efficient solution in C for Arduino, see http://www.arduino.cc/playground/Main/RotaryEncoders

Sommer algorithm

Swapping bits and calculating the exclusive or is easy if the two encoder bits are wired on two next bits of the same port. If two channels, take 4 consecutive bits.
As shown in the figure, this allow to shift, mask and XOR with the new encoder value, which is then saved under the Old name.

The routine in CALM assembler is available from www.didel.com/pic/libX/XEnco.asi



Fastest Sommer algorithm

The second algorithm is more efficient ; routine and tables use 35 bytes to update an 8-bit up/down counter. PICs handle tables in a very efficient manner. The program uses two level of tables, the second table calling the routine to be performed when a step is done, usually incrementing or decrementing. It is a very efficient program given later.

Quadrature encoder program – 16-bit position counter
; (to be called about every 1ms if less than 1kHz step frequency)
; lasts 12/13 us (single channel, and 12/19 us (two channels).

CALM notations

```
.macro    dd         ; prepare a table 000pbyyy
          RetMove #2'%1,W
.endmacro
; PortA input encoder
; PortB out counter low
.Loc      16'20
OldPort:  .Blk.16   1
Temp:     .Blk.16   1
CntLow:   .Blk.16   1
CntHigh:  .Blk.16   1
.Loc      0
Begin:
          Move      #2'11111111,W      ; inputs
          Move      W,TrisA
          Clr       W
          Move      W,TrisB   ; Outputs
          Move      PortA,W
; For one channel
          And       #2'11,W
          Move      W,OldPort
Loop:
          Move      OldPort,W
          Call      TaSwap
          Move      W,Temp
          Move      PortA,W   ; bits 1,0
          And       #2'11,W
          Move      W,OldPort
          Xor       Temp,W
          Call      TaJump
; Test: display value on PortB
          Move      CntLow,W
          Move      W,PortB
          Jump      Loop
TaSwap:
          Add       W,PCL
          d         00
          d         10
          d         01
          d         11
TaJump:
          Add       W,PCL
          Ret                 ; Nop
          Jump      I1        ; Increment
          Jump      D1        ; Decrement
          Ret                 ; Nop
I1:       Inc       CntLow
          Skip,NE
          Inc       CntHigh
          Ret
D1:       Move      #1,W
          Sub       W,Cnt2Low
          Skip,CS
          Dec       Cnt2High
          Ret
.End
```

Microchip notations (not tested)

```
macro dd
      retlw  \1
endm
; PortA input encoder
; PortB out counter low

OldPort  Equ  20h
Temp     Equ  21h
CntLow   Equ  22h
CntHigh  Equ 23h

Begin
         Movlw   0b11111111
         Tris    PortA
         Clrw
         Tris     PortB
         Movf    PortA,0

         Andlw   0b11
         Movwf   OldPort
Loop
         Movf     Oldport,W
         Call      TaSwap
         Movwf   Temp
         Movf      PortA,W ; bits 1,0
         Andlw     0b11
          Movwf   OldPort
          Xorwf   Temp,0
           Call      TaJump
; Test: display value on PortB
          Movf     CntLow,W
          Movwf   PortB
          Goto     Loop
TaSwap
           Addwf PCL,1
          d 00
          d 10
          d 01
          d 11
TaJump
          Addwf PCL,1
           Return        ; Nop
           Goto   I1     ; Increment
           Goto   D1    ; Decrement
           Return        ; Nop
I1          Incf    CntLow,f
            Btfsc   Status,2
            Incf    CntHigh,f
            Return
D1          Movlw  1
            Subwf  CntLow,f
             Btfsc  Status,1
             Decf CntHigh,f
             Return
end
```

Documentation on testing the RomEnco with the Microdule M877 and using it with the Starlet (16F690 robotic board) and the PicStar board (18F4550) on
www.didel.com/starlet/StarletRome.pdf

jdn100425/110725