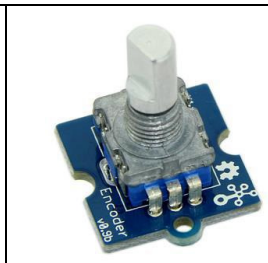
CH-1092 Belmont, Switzerland
Tel +41 21 728-6156, Fax 728-6157
Email info@didel.com, www.didel.com

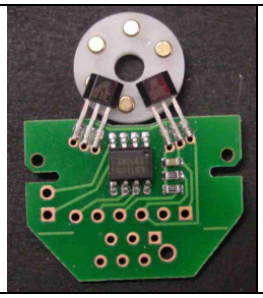
www.didel.com/EncoderSoft.pdf

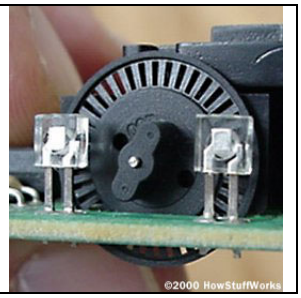
Encoder software

1 Introduction

Encoders may looks like potentiometers, but they have at least 4 wires, 2 for power supply and 2 for quadrature phase signals. They are also found on motor shafts or robot wheels in order to measure angles or displacements. They were the hart of the old mice.



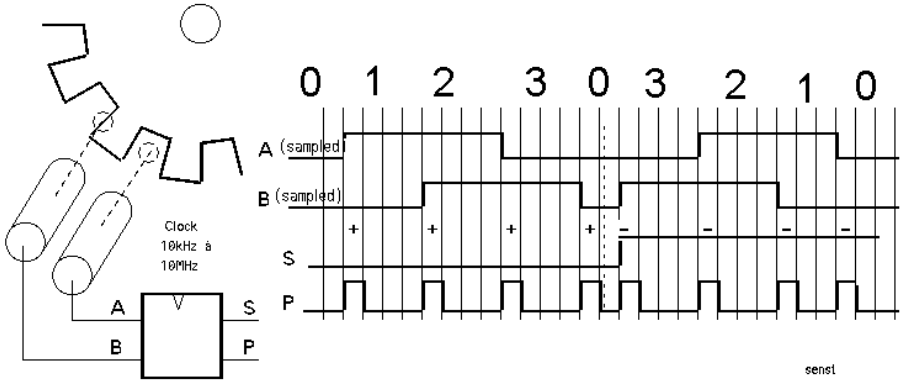


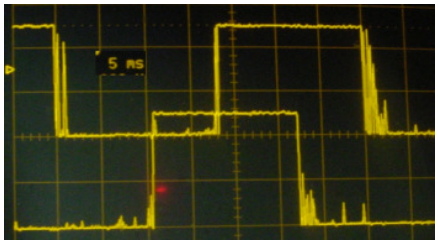


This text shows to beginners in C / Arduino how to do efficient programming.
It should also interest the "experts" in Arduino, who sometimes lack a good understanding of the real time and a concern for optimization of the code. External interrupt is frequently proposed, but is inapropriate if there are bounces. Some programs do not decode all transitions.

2 Principle of signal decoding

The signals are generated by two optical, magnetic or mechanical sensors. Mechanical sensors have contact bounces, eliminated by adequate sampling. Filter with an electronic circuit is absurd if one can do it by software.
It should be understood that there is a sampling period, which allows the program to compare and track the signal change, and a maximum duration of contact rebounds. The sampling period must be longer than the duration of the bounces.



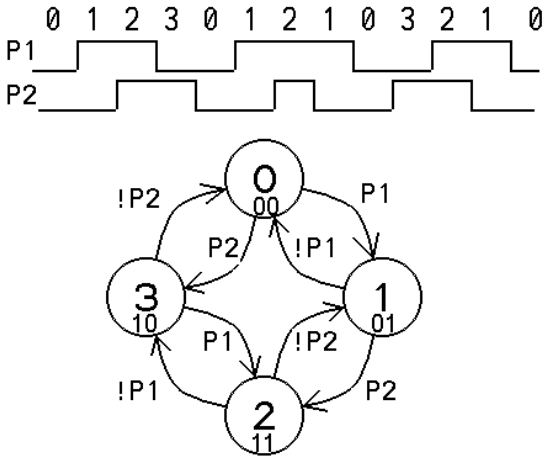


The controller samples A and B signals and decides to increment or decrement a counter according to the phase difference. The simplest algorithm is based on the state graph.

The C structure "switch-case" makes it possible to decide for each state what to do if there is a change of state. In state 0 for example, there are 3 possibilities because one cannot have P1 and P2 which pass together to 1:

P1 = 1 we go to state 1 and count
P2 = 1 we go to state 3 and decount
P1 = 0 and P2 = 0 one remain in state 0

If the signals have contact bounces, as for a mechanical encoder, sampling must be done every 5 to 10ms, which limits the speed of rotation, anyway low enough with an encoder replacing a potentiometer.



In order to test the algorithm, we have to choose how to display the result. Arduino offers the serial terminal, which calls the Serial library. But the problem is that one has to sample every 10 ms with a manual encoder, so we do not miss steps. But if we print each time, it will fill the screen and be difficult to check the count. The solution is to perform 100 encoder cycles, with their sampling at 10ms, before

displaying the result on the serial terminal. But be careful, how long for printing the value? If it's too long, we're going to miss steps!
At 9600bit / s, it takes 1ms to display a letter. It takes about the same time to convert the binary value of the 16-bit variable to decimal, so the display of the 6 digits (at most) will take 7 milliseconds. Every second, the sampling will be done with a period of 17ms, that is what will limit the rotation speed.

3 First test program

Sorry, comments are in French.

```
// TestEncodeurArduino.ino
// (le programme se trouve sous www.didel.com/Encodeur.zip)
// L'encodeur est câblé sur les pins Arduino 14 et 15
#define P1 14  //const Int P1=14; si vous préférez
#define P2 15
int cnt =0 ; byte etat = 0;

void setup()
{
  Serial.begin(9600);
  pinMode (P1, INPUT);
  pinMode (P2, INPUT);
}

void loop () {
  for (int i=0;i<100;i++) {  // display every 100 scan
    delay (10);// sample delay (debounce)
    switch (etat) {
     case 0 : // P1=0 P2=0
      if (digitalRead (P1)) { etat = 1 ; cnt++ ; }
      if (digitalRead (P2)) { etat = 3 ; cnt-- ; }
      break;
     case 1 : // P1=1 P2=0
      if (!digitalRead (P1)) { etat = 0 ; cnt-- ; }
      if (digitalRead  (P2)) { etat = 2 ; cnt++ ; }
      break;
     case 2 : // P1=1 P2=1
      if (!digitalRead (P1)) { etat = 3 ; cnt++ ; }
      if (!digitalRead (P2)) { etat = 1 ; cnt-- ; }
      break;
     case 3 : // P1=0 P2=1
      if (digitalRead (P1))  { etat = 2 ; cnt-- ; }
      if (!digitalRead (P2)) { etat = 0 ; cnt++ ; }
      break;    } // end switch
  } // end for
  Serial.println(cnt);
}
```

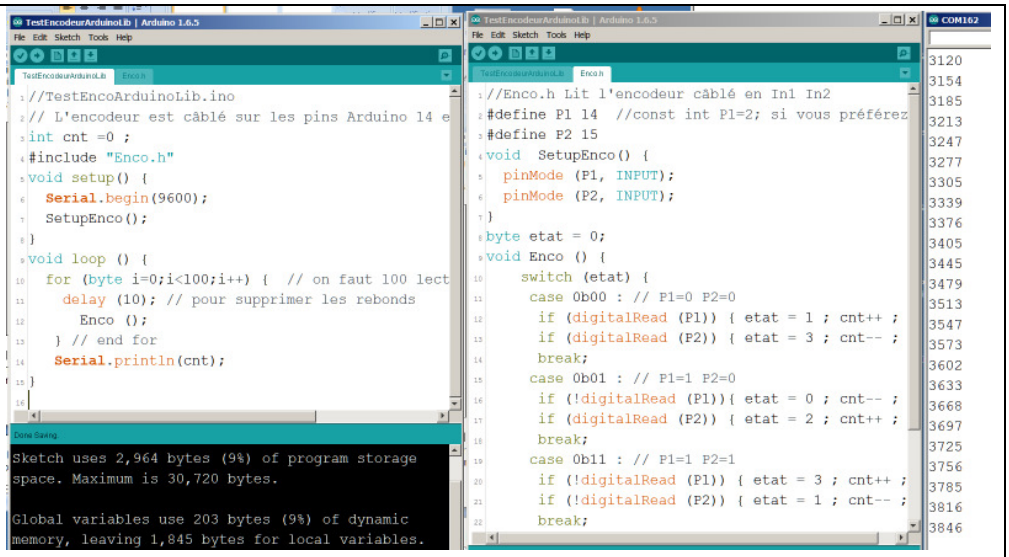
4 Warning

If this program does not work, first look at the signals with a good oscilloscope before saying it is not usable! It does work with 2 pushbutton, You depress the buttons following the sequence and you see the result on the terminal. It should works with an encoder that spin smootly. We noticed it did not work with a Grove encoder having a detent, that gives the 4 edges on each detent, and some edges may be as close as 1ms.
It does not work either with a high resolution motor encoder like the RomEnco (Didel-Solarbotics), because the debouncing kill the reading. We need a sample delay due to the use of the Arduino terminal. We can reduce the sample delay to 100 microseconds, and get up to 10'000 transitions per seconds (RomEnco goes up to 4000)). we can increase the iterations of the for loop to display the count every second, but each time the serial teminal display a number, counts will be lost. You will not notice them, it will be a small percentage, just be aware of what you do.

5 EncoArduino.h function

Now that we have a program that works, we must "encapsulate", that is create an Enco () function; (the first version will be named EncoArduino ()). Fron now on, we just need to remember 2 things: the function `EncoArduino ();` updates the variable `int cnt;` this function uses the signals P1 P2 that must be declared and initialized calling the SetupEnco () file. The EncoArduino.h file is included as a library, but is part of the sketch. It is visible in a separate window.
This practice is not common, but it is very effective (see appendix).

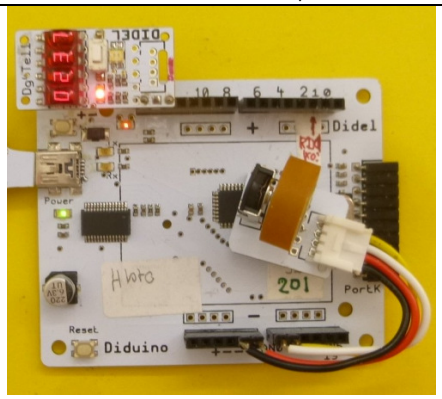
We see that the main program only shows what we want to do:
Read the encoder and display on the terminal every 100 cycles.
The variable cnt is global, it is used by the file Enco.h and by the program, so declared at the beginning.
The state variable is local to Enco.h, so defined in Enco.h and can be forgotten. Arduino hides these concepts, bearing essential to make a serious application.



6 Display on Tell 4-digit display

Let's use the embedded DiTell bmodule instead of the screen (Tindie DiTell b). No more need of a screen if the robot is moving.
Tell displays 4 digits with a function that takes 3ms.
The cabling is shown on next picture, and with our modular approach we must include the Tell.h library, call its setup, and know that function `Tell (xx);` will displays the variable `int xx;`

```
//TestEncoTell.ino
int cnt =0 ;
#include "Enco.h"
#include "Tell.h"
void setup() {
  SetupEnco();
  SetupTell();
}
void loop () {
   delay (7); // suppr  rebonds
   Enco ();
   Tell (cnt);
}
```

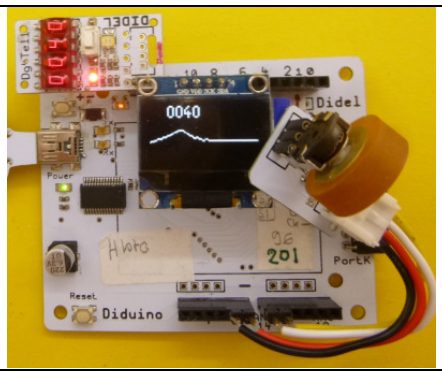


It can be seen that the for loop has been deleted because the display does not scroll; It lasts 3ms. With the suppression of the rebounds, one is always limited to 100 transistions per seconds.
Note that the Enco function has been modified to no longer use the Arduino functions, so be portable and fast. If you are not comfortable with logical operations, trust, as with many other functions that you have not thought to understand.
The EncoArduino () function; Lasts 10us with the functions Arduino, Enco (); Lures1.5 us with direct C instructions.

7 Display on an Oled SSD1306

Let's add an Oled SSD1306 display, on pins 8 to 11 (notice you can power from signal pins). We reserve pins 4 to 7 for two motors. A second encoder can be added on pins 16 and 17. This will require a second Enco2.h function with new names (P3 ,P4, cnt2)
Oled offers a freedom of representation that we do not have with Tell. Our compact lib is under github.com/nicoud/Oled. The write time is 25ms with the program below, so we are limited to 40 transitions per second, which is compatible with all potentiometer-encoder.

```
  //TestEncoTellOled.ino
int cnt =0 ;
#include "Enco.h"
#include "Tell.h"
#include "OledI2Cbb.h"
#include "OledPix.h"
void setup() {
  SetupEnco();
  SetupTell();
  SetupI2Cbb();
  SetupOledPix();
}
```



```
byte x;
void loop () {
   Enco ();
   Tell (cnt); // 3ms
   LiCol(1,20);BigHex16(cnt);
   DDot (x,(cnt&0x63));
  if (x++==128) {x=0; Clear();}
}
```

8 Interrupts

The usual approach to use interrupt is to create an interrupt at each transition of the two inputs. Microcontrollers can usually do this. The problem is bounces also trigger interrupts and it gets complicated. Our approach is to use a synchronous interrupt, which has no limitation in the number of encoders, and which makes it possible to easily add other sources of interruptions.

The Timer2 is initialized to create a regular interrupt. The interrupt routine calls the Enco(); function, often enough not to lose steps, but not too often to filter the bounces.
The document www.didel.com/coursera/LC5.pdf explains the AVR328 timer 2 and gives the essential knowledge we need.
The program samples at 100 Hz. The interruption lasts 2.5 microseconds every 10 ms and updates the variable `cnt`. What happens if in the main program, the Tell function and the Oled functions are called? Can they be interrupted for 2.5 microseconds? The answer is yes, but it has to be read in the specifications. The waiting loop in the main program no longer depends on Enco(), but only on our visual perception of the display.

```
//TestEncoInter.ino
int cnt =0 ;
#include "Enco.h"
void setup() {
  Serial.begin(9600);
  SetupEnco();
    TCCR2A = 0; // setup Timer2
    TCCR2B = 0b00000111; // clk/1024
    TIMSK2 = 0b00000001; // TOIE2
    sei(); // autorise les interruptions
}
volatile int cnt;
ISR (TIMER2_OVF_vect) { // interrupt task
  TCNT2 = -160; // pour 100 hz si clk/1024
  Enco();
}
void loop () {
  Serial.println(cnt);
  delay (500);
}
```

If you have a Tell or an Oled, easy to add them (`TestEncoInterTellOled.ino`).

For a fast motor encoder, modify timer initialization. With an interrupt every 60 µs , the Rome BO10 motor with RomEnco (12 edges per turn) could spin at 1300 turns/second (not RPM) using 2.5 µs of processor time every 60 µs.

9 More inside the interrupt routine

Using a single timer to handle multiple tasks is called synchronous programming. It avoid the problem of multiple colliding interrupts. Add a second, third,.. encoder? Add PFM on motors, PWM on Leds? Programming in C is very powerfull. See LibX (github nicoud Libx) which may be called DidLib when fully documented (there are already several LibX).

jdn 170621

Include files are a little tricky with Arduino. See www.didel.com/coursera/FichiersInclus.pdf
We hope to translate soon: www.didel.com/IncludeFiles.pdf