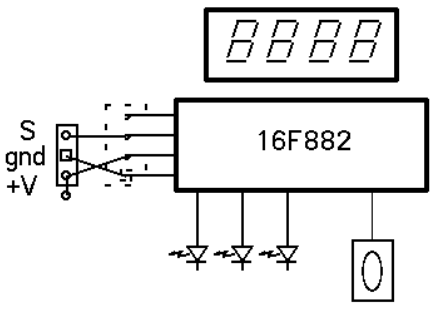
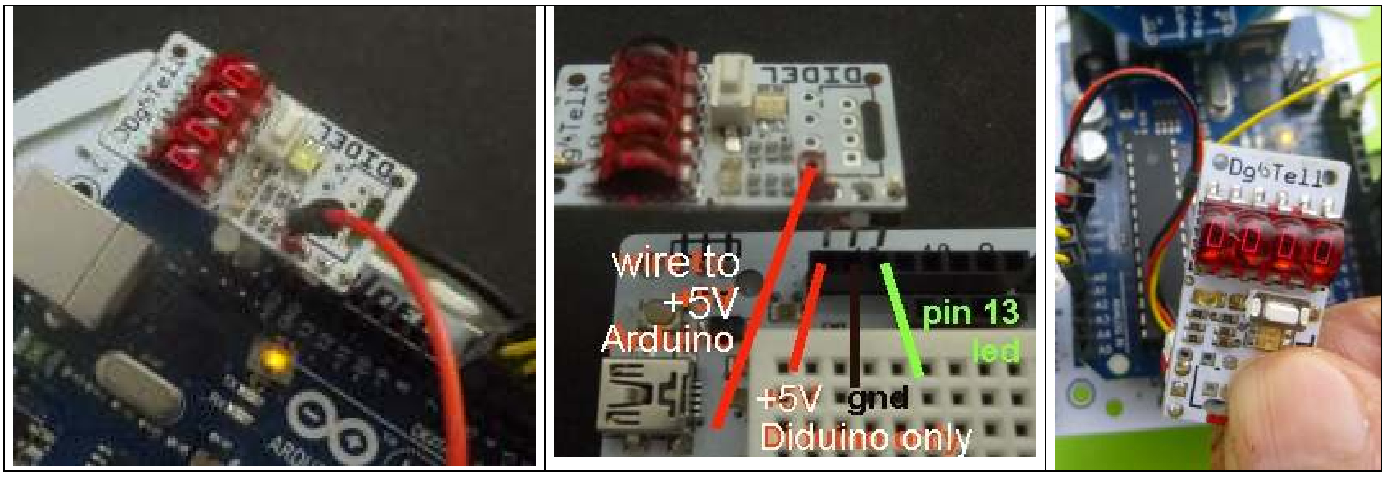
http://www.didel.com/ info@didel.com

www.didel.com/diduino/DiTell.pdf

DiTell – a 4-digit display using 1-wire communication with hexa and decimal presentation

Prefer I2C ? – see www.didel.com/digrove/DgTelli2c.pdf

The ***DiTell*** role is to spy two 8-bit sensor values or a 16-bit variable, showing their values in hexadecimal or decimal. It needs only one wire, plus a 3 to 5V supply, 15 to 40mA. A 3-pin optional connector plugs on the right top pins of the Arduino, using pin 13, usually reserved for blinking the led and helping debugging. With the ***DiTell*** connected, you get a powerful debugging tool and easy control display. Use a 3-wire cable, up to 50cm long, and put the ***DiTell*** any place if the connector on pin 13 does not suit you. What's great is the possible local conversion to decimal. One thinks decimal while programming, but variables are stored, and transferred in hex. Local conversion to decimal is instantly selected, depressing on a button.	

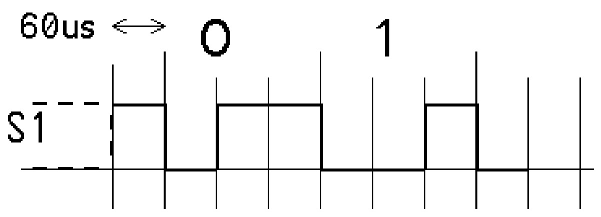


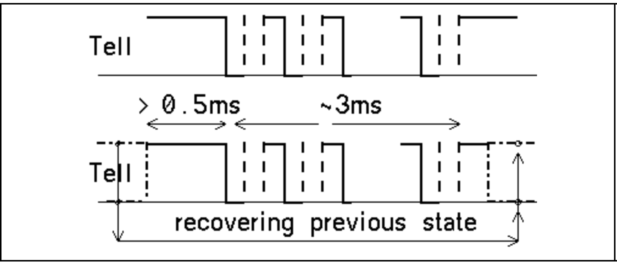
The 1.wire transfer protocol is similar to Dallas 1 wire. The16.bit transfer is done in 3 milliseconds with a few lines C function to be called when you need to update the displayed value. It adds 92 bytes to your code, 132 if you wish to blink.. Compare with any other solution !

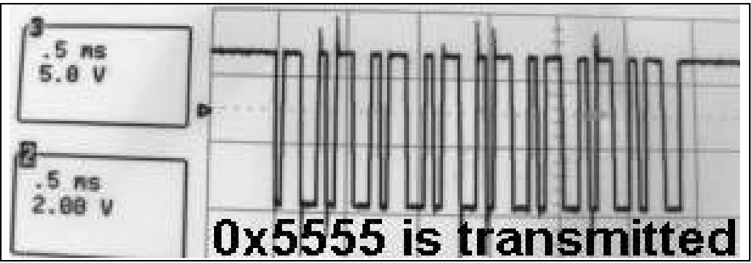
The ***DiTell*** display is supported by the LibX library that makes displaying a value transparent to the user, See www.didel.com/xbot/DemoTell.pdf (en français) . Doc on Github coming.

Didel software is written in C, not using Arduino "facilities", and is easy to adapt to any robot, to any microcontroller programmed in C.

Timing

Every bit lasts 180 us Zero pulses are 60us low, 120 us high One pulses are 120us low, 60 us high Transfer time ~3 ms Spacing between16-bits transfers must be > 1 ms (display refresh period is usually 20ms). The state before the burst can ba saved and restored, allowing the usual blinking on pin13.	





Changing the delays allows to use e.g. a 1MHz Tiny AVR processor.

Code for AVR328 16MHz and AtTiny (qz constants according to speed)
Sketch under www.didel.com/diduino/DiTell.zip

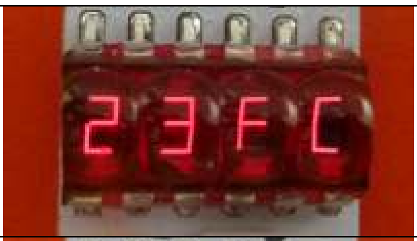
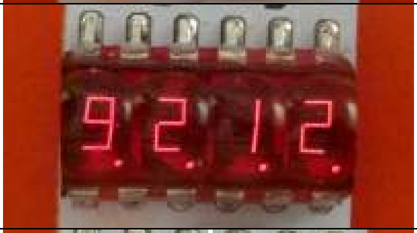
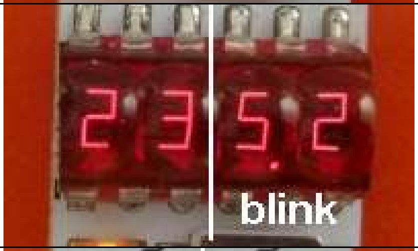
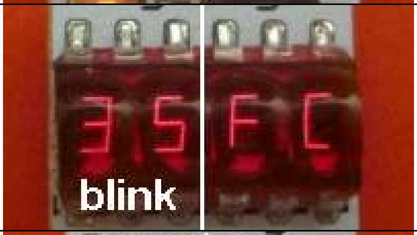
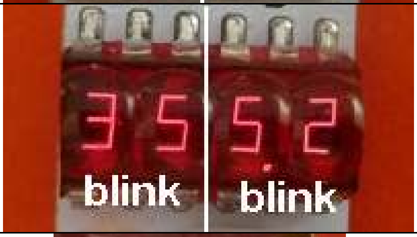
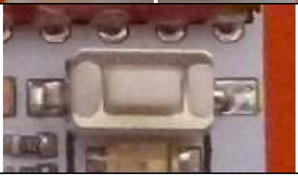
```
// TestTell send 16 bits, 16 MHz processor, Arduino
#define bS1 5 // pin 13
#define S1On  bitClear (PORTB,bS1)
#define S1Off   bitSet   (PORTB,bS1)
#define S1dirOut  bitSet  (DDRB,bS1)

volatile byte qz; volatile int qqz;
#define Delt qz=95; while (qz--) {}  // 60us à 16 MHz
#define Deltt  qqz=600; while (qqz--) {}  // 600us à 16 MHz

void Tell (int dd) {
  byte cc=0;
     sd = PORTB;  S1Off;  // save , set high  // optional recovery
     Deltt;   // min 600 us                   // instructions
  // cli();  //remove jitter interrupt if required (>5us)

  //  cli();   //remove jitter interrupt if required (>5us)
  while (cc++ < 16) {
    S1On;
    Delt; if (!(dd&0x8000)) { S1Off;}
    Delt; S1Off;
    Delt;
    dd <<=1;
  }
      if (!(sd & (1<<bS1))) { S1On; }  // recovery instruction
// sei();   //restore interrupts
}
```

```
// Arduino 814 bytes with recovery
void setup () {
   S1dirOut;
}
void loop () {  // test blink
   S1Off;
   Tell (0x5533);
   delay(1000);
   S1On;
   Tell (0x4321);
   delay(1000);
}
```

```
// AtTiny24 with AVRstudio 286 bytes (qz = 6 at 1MHz clock)
int main () {  // AVR studio
  DaOut;
  for (;;) {
    Tell (0x1234);
    delMs (10);
  }
}
```

This optimised `Tell()` function needs 92 bytes of code + 40 with recovery
See GitHub DiTell for details and options (under preparation Nov 2015)

Hex and Decimal modes
4-digit displays are traditionally showing hexadecimal values, which can be decimal if converted before transmission.
The DgTell do local conversion according to 5 modes, providing the frexibility to work with one 16 bits word or two 8-bit numbers.

Mode 0 is hexadecimal.	
Mode 1 is decimal 4 digits, with the limitation that if if you send 0x270F it will be converted to 9999, and a greater hex number cannot be displayed. Dashes will be shown -. -. -. -. The 4 dots are lighted when the number is displayed in decimal.	
Mode 2 assumes you work with two 8-bit numbers, and it happens the high byte is preferably shown in hex, and the low byte in decimal. How to make the difference? Bytes in decimal are blinking. How are shown values up to 255? From 100 to 199, the left dot in on, from 200 to 255, it is the right dot. On the right 2 digits, 0xFC has been converted to 252, notice the right dot is on.	
Mode 3 assumes you work also with two 8-bit numbers, and it happen the low byte is preferably shown in hex, and the high byte in decimal. 0x23 has been converted to 35 (2*16+3).	
Mode 4 assumes you work also with two 8-bit numbers, and you need to see both of them in decimal.	
The push button goes through the 5 modes any time. The selected mode is saved on Eeprom and restored at power up.	

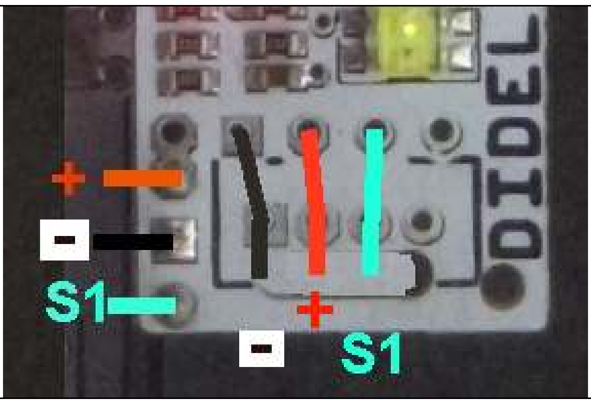
Leds

There are one red led on the DiTell and a bicolor led, unused. The red led blings at pouwer-up, and then show the state of the S1 line, led on if active low. If connected on pin 13, the led connected to pin 13 is active high, so the 2 leds are of opposite phase.

.

Wiring options

I

DiTell can be controlled by three connectors. The side one with 3 pins is compatible with Arduino pin. Check you have power on last pin., or bring power on the additional + pins The 4-pins connectors on the board have the minus on the side, then + and the S1 signal You can put a Grove connector and use only the signal on pin 3. You can put a male of female 2.54mm 3-pin connector, osolder long wires directly.	

Comments

The DgTell is not the solution for all problems. The idea was to use a single pin to display a 16-bit word, or two bytes. The target microcontroller was an Arduino compatible board, with not so many pins. On Arduino, one pin must be kept for debugging, usually pin13 with its led you can blink. The DgTell uses that pin, and it is still possible to blink (see Github DiTell). Serial transfers on that pin disturb the led for 3ms, but it is still easy to recognize slow blinks.

A display needs to be powered. Next to pin13 is a Gnd pin. Next pin is the Aref pin, very seldomly used. On the Diduino, that pin is connected to the +5V. On Arduino boards, there is no arm to connect that pin to the +5V available on the other side of the board.
See the pictures on the top of this document.

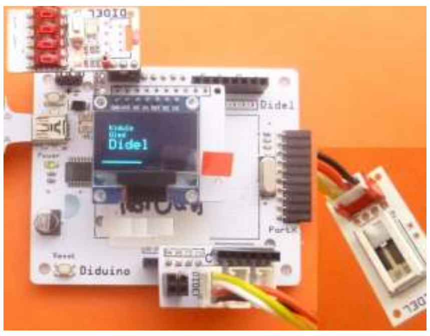
As mentionned earlier, the `Tell(val);` function must be called to update the display. This is a blocking function that will be disturbed by an interrupt of more than 5 microsecond duration. In this case, one should deactivate interrupts for the 3ms of the transfer.

Didel proposes also an Oled module and an adapter to connect it on PORTB. Pin 13 of the ***MsOled*** is shared and a 3-pin connector is provided for the ***DiTell***. The graphic is very slow, but as a didactic tool, it offers the possibility to show the evolution of a sensor. ***DiTell*** gives the instant value, the ***MsOled*** uses the Adafruit library and is heavy.(7000 bytes) See http://www.didel.com/xbot/Oled.pdf	

Use the DiTell on your Tiny24 and Tiny25 projects

Debugging a small processor is always delicate. The one pin ***DiTell***. makes it easy to find its place, and frequently that pin can be shared with another signal, e.g. a push-button or a led. In case of a push button active low, changing the direction is enough, no delay is required before the transfer.
Push.button active high may work if there is a >1k resistor toward the +5 or + 3.3V.

Interrupt transfers under LibX

DiTell has been designed to be controlled by interrupt. The **LibX** library, written in portable C (not Arduino), uses a simple timer, available on all processors, that will interrupt the main program every 60 microsecond and go through a serie of tasks, that are steps of state machines. The ***DiTell*** transfer needs 50 steps, that is last 3ms, but each step is less than 15 instructions and takes 2 microseconds. Other tasks are also cut in small pieces and you can really do a lot of work before the interrupt routine takes 30 us, and use half of the processor time. In the main program loop, asking ***DiTell*** to display a new value needs only one instruction	

jdn 151031