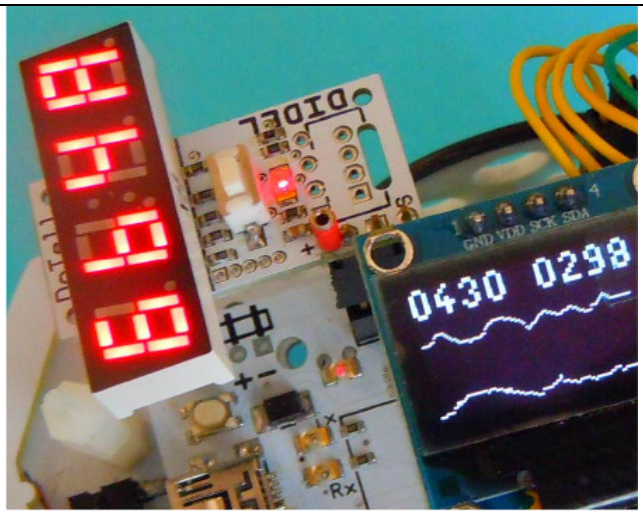
CH-1092 Belmont, Switzerland
Tel +41 21 728-6156
Email info@didel.com, www.didel.com

https://www.didel.com/prof/Tell.pdf

Affichage Tell

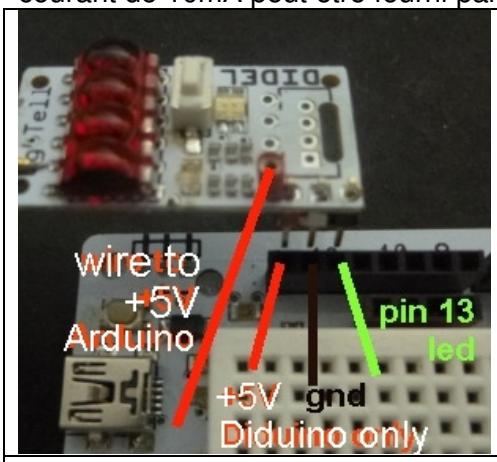
Liens aux docs sous https://www.didel.com/prof/Tell.html

L'affichage Tell remplace le Terminal Arduino pour afficher un nombre de 16 bits ou deux valeurs 8 bits. Il n'utilise qu'une pin et est indispensable pour surveiller des capteur dans une application autonome. Le programme est court (120 bytes) et l'affichage dure 3ms.
Le Oled SSD en comparaison permet d'afficher plus mais l'affichage d'un nombre dure 10ms et le programme prend 2k (librairie OledPix).
Le Terminal demande aussi ~10ms pour afficher un nombre. Son affichage qui défile est gênant pour suivre les valeurs d'un capteur et exige la connexion USB.

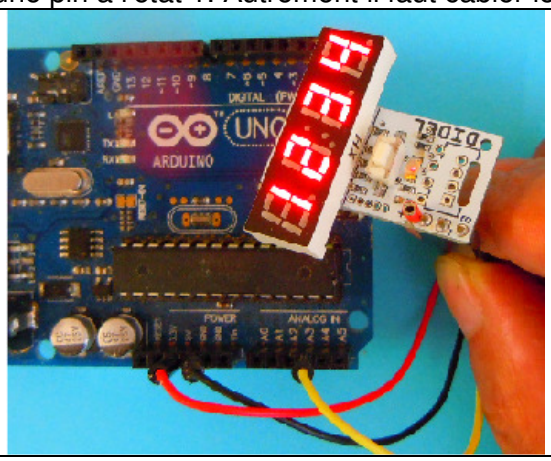


Tell sur la pin 13

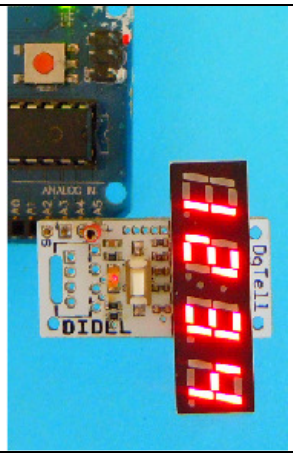
Tell est prévu sur la pin 13 et son utilisation n'empêche pas d'utiliser la pin13 pour clignoter. A côté de la pin 13 il y a un Gnd et à côté une pin exceptionnellement utilisée que l'on peut forcer au +5V sans danger. La carte Diduino a cette pin au +5V, avec une carte Arduino, il faut soit souder un fil dessous, soit utiliser un "jumper wire". Si on a 3 pins consécutives libre, on peut programmer les alimentations, le courant de 10mA peut être fourni par une pin à l'état 1. Autrement il faut câbler le + et le –



Installation sur pin 13 Diduino et Arduino non modifié



Installation sur pins 17



.. sur pins 17,18,19

Installation et appel

Pour pouvoir écrire dans le programme Tell(0x1234); il faut initialiser la pin 13 en sortie, état 1 et avoir la fonction Tell dans le programme, ou chargé par un #include comme dans le programme de test ci-dessous.
La fonction Tell(int); est bloquante et dure 3ms par transfert.

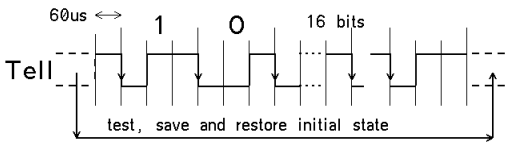
Programme de test (sous https://www.didel.com/prof/Tell.zip)

```
// TestTell.ino Compte 816b
#include "Tell.h"

void setup() {
  TelldirOut;
  TellOff; // high
}
int cnt = 0x1234;
```

```
// LibX Tell.h  affichage 16 bits
#define bTell 5 // pin 13 Arduino PORTB pin 5
#define TellOn   bitClear (PORTB,bTell)
#define TellOff  bitSet  (PORTB,bTell)
#define TelldirOut bitSet  (DDRB,bTell)

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 loop() {
  Tell (cnt++);
  delay (100);
}
```

Le programme envoie 16 impulsions de longueur variable




```
void SetupTell() {
  TelldirOut;
  TellOff; // high
}

void Tell (int dd) {
  byte cc=0;
  //  cli();  //remove jitter interrupt if required (>5us)
  while (cc++ < 16) {
    TellOn; Delt;
    if (!(dd&0x8000)) { TellOff;}
    Delt;
    TellOff; Delt;
    dd <<=1;
  }
  //  sei();  //restore interrupts
  Deltt;
}
```

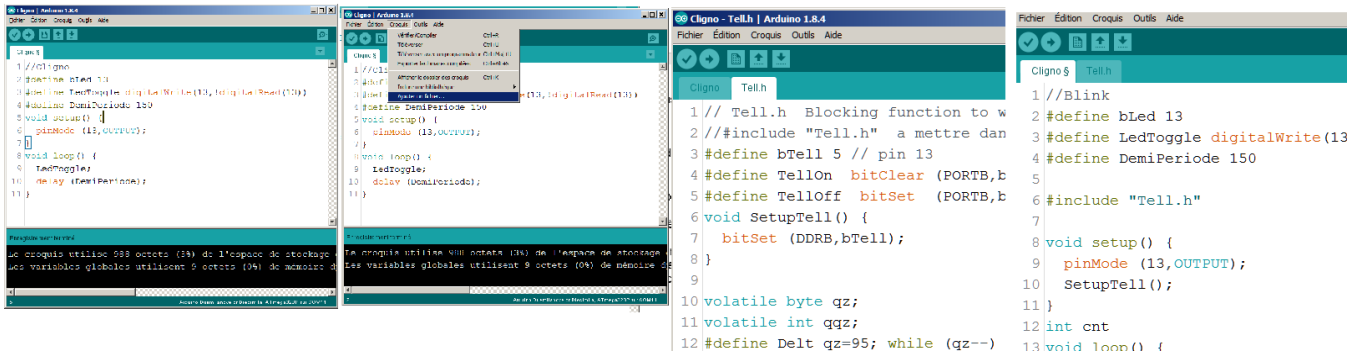
Dans un programme, l'instruction Tell (var) affiche une variable de type int ou byte.
Le module Tell a plusieurs modes d'affichage et démarre dans le dernier mode utilisé. On change de mode en agissant sur le poussoir. Le mode décimal (affiche jusqu'à 9999) est confirmé par les 4 points décimaux allumés. L'affichage hexa est le mode qui suit quand on agit sur le poussoir. Les modes qui affichent 2 bytes différents en décimal sont un peu "tricky" (voir la doc complète en anglais).

C'est souvent utile d'afficher deux variables 8 bits. Le plus simple est d'écrire

```
Tell ((v1≪8)+v2);
```

Utiliser Tell en fichier inclus

Pour inclure Tell.h, il faut aller le chercher sur le disque dans un dossier librairie (LibX) ou dans un croquis qui l'utilise. Cliquer sur "Croquis" puis "Ajouter un fichier" Le fichier inclus peut être lu et édité. On ne peut pas le détruire, mais on peu l'ignorer (et le détruire sous Explorer plus tard) et en charger un autre. Pour utiliser Tell.h, il faut un #include Tell.h et un setupTell();



Tell sur 3 pins

La consommation de courant de Tell (<10mA) permet de l'alimenter pas deux de sortie de l'Arduino. Pour le brancher sur les pins 17,18,19 comme sur la figure plus haut, il faut modifier les définitions et le set-up dans le fichier Tell.h qui est insérer.

```
//TellPin171819.ino
#include "Tell.h"
void setup(){
  SetupTell();
}
int cnt = 0x1234;
void loop() {
  Tell(cnt);
  delay(100);
}
```

```
// Tell.h  Modifié 17 Tell 18 Gnd 19 +5V
//#include "Tell.h"  a mettre dans le pp
#define TellOn    digitalWrite (17,LOW)
#define TellOff   digitalWrite (17,HIGH)
void SetupTell() {
  pinMode (17,OUTPUT);
  pinMode (18,OUTPUT);
  pinMode (19,OUTPUT);
  digitalWrite (17,HIGH);
  digitalWrite (18,LOW);
  digitalWrite (19,HIGH);
}
```

```
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;
  while (cc++ < 16) {
    TellOn;
    Delt; if (!(dd&0x8000)) {
TellOff;}
    Delt; TellOff;
    Delt;
    dd <<=1;
  }
}
```

Programme principal	Fichier inséré. -- Ne jamais toucher aux instructions de la 2e colonne --

Tell en librairie Arduino

On peut mettre Tell.h avec les librairies Arduino, et dans ce cas le #include "Tell" va chercher Tell.h dans la librairie (sous fichier – exemples) et le prend, s'il n'y a pas de fichier inclus, qui a la priorité. C'est génial, le Tell.h standard est en libraire, mais si on en a un inséré, il a la priorité. Pour mettre en librairie, il fait créer un .zip avec Tell.h et éventuellement des programmes .ino.
Dans Arduino, cliquez *Croquis – Inclure une bibliothèque - Ajouter une bibliothèque*
Il suffit alors de déposer le zip comme demandé en 2e ligne du menu. Sélectionnez Tell.zip dans votre dossier. Le transfert est instantané, pas de quittance.
Il faut quitter Arduino et recharger pour que les nouvelles librairies soient accessibles.
Retournez sur *Croquis – Inclure une bibliothèque* vous devez retrouver Tell vers la fin
Du tout le chenit Arduino. C'est prêt.

Décimal ou hexa?

Habituer des élèves à savoir interpréter des nombres hexa est utile dès que l'on doit pénétrer dans les couches profondes de n'importe quel logiciel.
En 8 bits, il faut se souvenir que 16'80 = 128 est la frontière entre les nombres négatifs et positifs (type char ou int8_t). En 16 bits, la frontière est à 32332 et Arduino n'utilise que le type int (signé) en partant de l'idée que les applications utilisent des nombres non signés inférieurs à 32'000 et qu'on peut les mettre dans le même paquet (c'est cacher un concept important).
Avec Tell on a avantage à travailler en hexa, ce qui se justifie par ailleurs puisque c'est un affichage de mise au point des programmes, et on travaille souvent au niveau des bits.

Persistance rétinienne

Afficher plus souvent que toutes les 20ms n'apporte pas plus d'information. A 20ms, on utilise 3/20=15% du temps du processeur.

Interruptions externes

Arduino met en route les interruptions timer qui coupent la fonction Tell toutes les ms environ, et ne perturbent pas. Si d'autre interruptions sont mises en route, il faut activer les instructions cli(); et sei(); que l'on voit en commentaire.

DoTell par interruption

La fonctions DoTell() est appelée par interruption toutes les 60 microsecondes et coupe le programme principal pour 5 µs, (par bouffée de 16 accès toutes les 20ms) Elle utiliser 3% du temps processeur – il n'y a plus de boucle d'attente, mais on perd du temps à quitter le programme principal et à y retourner.

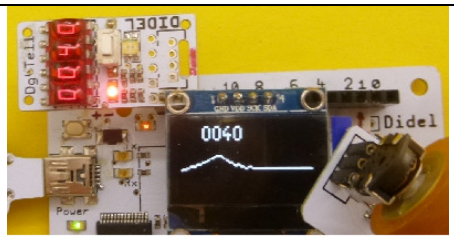
TellI2C compatible I2C

Cet affichage ne peut être connecté que sur les lignes I2C d'un processeur. Il a été développé pour une utilisation pédagogique en TigerJython sur Raspberry.

Références – cliquables dans *www.didel.com/prof/Tell.html*

Vous trouvez plus de détails dans ces documents

DiTellFacile.pdf DiTell – affichage de nombres 16 bits 7p 1606
Note: le terme "facile" veut dire que l'affichage est expliqué d'abord avec les notations Arduino.
DemoTell.pdf Exemples avec la librairie LibX, 8p 1510
En anglais
DiTell.pdf DiTell – a 4-digit display using 1-wire communication, 4p 1510
EasyTellI2C.pdf Easy TellI2C, 1p 1709
DgTelli2c.pdf DgTelli2c – a 4-characters display I2C/SMbus - 3 to 5V, hexa and decimal, plus alpha and segment mode, 4p 1604
PyTell.pdf PyTell – a 4-characters display for I2C/SMbus - 3 to 5V, 2p 1604

Tell est le nom actuel de l'affichage 3 pins. DiTell était utilisé au début pour distinguer avec TellI2C. La librairie Tell.h s'est aussi appellée Debug.h en 2015. La fonction par interruption (machine d'état) s'appelle DoTell dans la librairie DoTell.h. On voit sur des photos un affichage miniature qui n'est malheureusement plus disponible.	

Jdn 180830