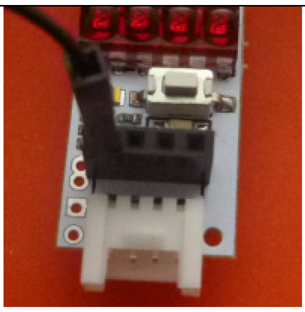
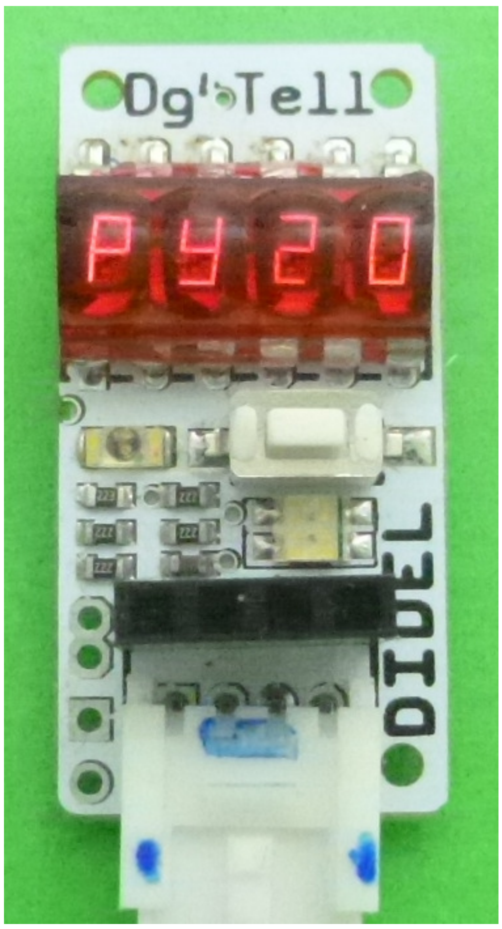
http://www.didel.com/ info@didel.com

PyTell – a 4-characters display for I2C/SMbus - 3 to 5V supported by Aplus Python library

The ***PyTell*** is a low power miniature 4-digit display that works from 3V (20mA) to 5.5V (30mA). Its simple I2C interface makes it easy to program under Arduino or any other microcontroller – they all have I2C on 2 of their pins. Its originality is you can control all segments individually. If you want to display numbers or text, you have to do your own character generator. Of course, the powerful library developped by Prof A.Pluess (http://www.aplu.ch) http://www.aplu.ch/download/pytell.zip provides all the facilities. Using the Wire library on Arduino is easy, but use the DgTellI2C if you do not want to program the character generator yourself.





Grove connector pinout:
Gnd Vcc SDA SCL

Pushbutton and leds are not used by PyTell software.

The PyTell respond to four commands at its 7-bit I2C address 0x20:

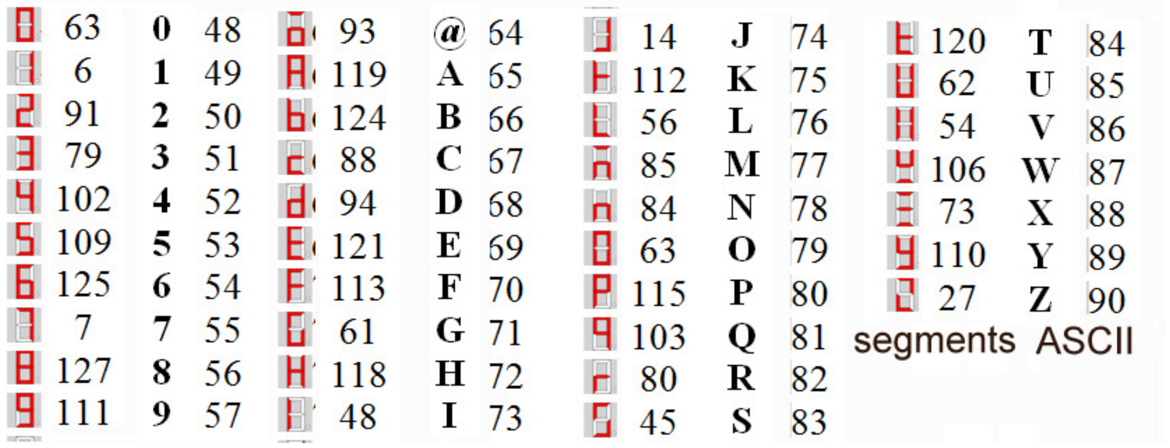
1) Direct read access provide the Id number, 0xA0 (maybe 0xA1 for a future release).
   `SMbus Receive, Python read_byte(0x20)`
   Also `Python read_byte_data(0x20,0)`

2) Block write of 4 bytes with command 1
   `Python write_block_data (0x20,1,txt[4]`

3) Word write with command 2 and 3 in case block write is not supported, or looks easier

```
Python write_word_data (0x20,2,"AB")
Python write_word_data (0x20,3,"CD")
```

Segment codes for the Ascii characters



segments		ASCII
63	0	48
6	1	49
91	2	50
79	3	51
102	4	52
109	5	53
125	6	54
7	7	55
127	8	56
111	9	57
93	@	64
119	A	65
124	B	66
88	C	67
94	D	68
121	E	69
113	F	70
61	G	71
118	H	72
48	I	73
14	J	74
112	K	75
56	L	76
85	M	77
84	N	78
63	O	79
115	P	80
103	Q	81
80	R	82
45	S	83
120	T	84
62	U	85
54	V	86
106	W	87
73	X	88
110	Y	89
27	Z	90

Test program for Arduino/Diduino

```
//TestWritePyTell.ino
#include <Wire.h>  // Arduino library
#define AdTell 0x20
void setup()   {
  Wire.begin();
}
uint8_t  id ;
void loop() {
  Wire.requestFrom(AdTell,1);
  id = Wire.read();

  Wire.beginTransmission(AdTell);
  Wire.write(1);  // command
  Wire.write(4); // block length (SMbus)
  Wire.write(119);  // A
  Wire.write(124);  // b
  Wire.write(88);   // c
  Wire.write(94);   // d
  Wire.endTransmission();

  for (;;); //stop
}
```

Command summary

Command	Read	Write
0	**Id=0xC0**	
1		**Bloc 4 segments**
2		**Write word A B …**
3		**Write word … C D**

Reprogramming the I2C address

Depending on what else must be connected on the I2C, it may be useful to change the address. Within an environment with student, we opted to a simple hardware solution that does not need to reprogram the chip (easy with a Pickit2).
Two programming pins available on the programming connectore are used. At reset, thes pins are open with pull-ups. The software takes the address form the EeProm.
If the status of these 2 pins labeled below 7 and 6 (they corresponf to RB7 and RB6 if you are familiar with the 16F882) is different, one of the 3 predefined address will be set and saved. The change is confirmed by a blinking.

```
Connector   7  6  –  +  c   - pin is used to force a zero on pin 6 and/or 7
            1  0   AdI2C = 0x40  7-bit address
            0  1   AdI2C = 0x30
            0  1   AdI2C = 0x20
            1  1   no change
```

jdn160410