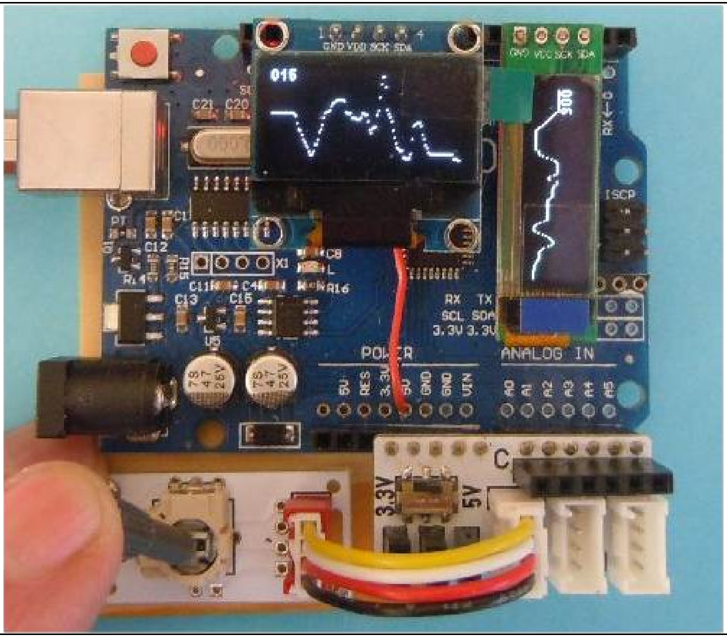
Oled I2C bitbang library
Two or more oleds on the same board

First part of this document under https://www.didel.com/OledLib.pdf

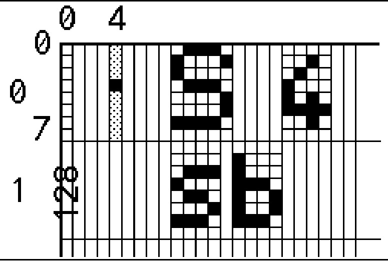
I2C is a rather slow transfer protocol, easy to program in master mode. In addition, The Oled I2C SSD1306 accepts only write commands, Acknowledge can be ignored.
As shown on the picture, the two analog values read on the pot are shown in real time (min 5ms to display a dot) on the two screens simultaneously.
You can also note that the Oleds are powered by Arduino pins set at 0 and 5V; current on the Oled is max 3-4mA and voltage can go doen to 2.5V.

We assume you have understood the OledPix library. OledMap is not considered, sice it needs 1 kbyte of memory per screen, and AVR328 has only 2k total.



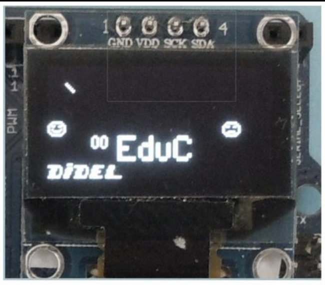
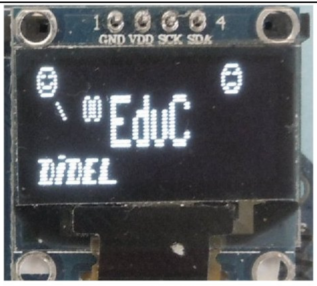
Test lines and scan lines

Remember that the 64x128 Oled must be seen as 8 lines of vertical bytes and is well suited to display 7x5 characters. First bytes of these line numbered 0,1,..7 have a byte address 0, 128, 256, etc.
The hardware of the circuit consider also scan lines of one pixel high, ordinate 0..63.

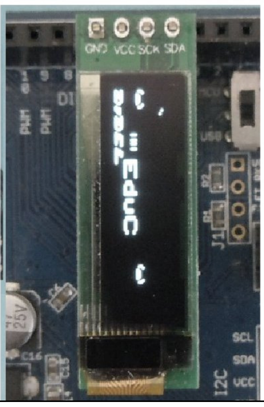
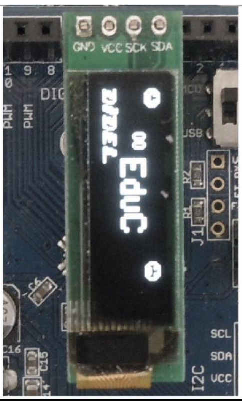


Two Oled sizes

A single command, allows to change the continuous transfer of the local bitmap . In the normal 64x128 mode, the 32x128 oled display one every tho lines – difficult to read. The 32x128 mode uses the second half of the inside bitmap, that is text lines 4,5,6,7, bytes 512 till 1023.
A 64x128 Oled spread these 32 lines over the height of the screen, ignoring one every two lines. We use it frequently and name it DoubleH mode; it shows more readable text and numbers, without any software change.

Normal Mode

Double Height Mode

Oled commands

I2C 8-bit SSD1306 Oled address is 0x78 (7-bit address 0x3C).. The address is followed by a command (0x00) and by control sequences. For normal mode, I2C transfer is:

```
   0x78  0x00  0xda  0x12
```

For 32x64 or double height 64x128 mode, I2C transfer is:

```
   0x78  0x00  0xda  0x02
```

If you use Adafruit library, send that info using Wire lib, eg

```
     Wire.beginTransmission(0x78/2);  // 0x3C
     Wire.write(0x00);
     Wire.write(0xda);
     Wire.write(0x02);  // double height
     Wire.endTransmission();
```

OledI2Cbb.h

BitBang needs Sck and Sda signals be declared on any 2 pins. We have selected hereArduino pin 5 for SCK and pin 4 for SDA. In addition, we define pin 7 at zero volt and pin 6 at + 5V. So we can insert the Oled on pins 7654 and power it witout additional wires.
As explained in OledLib, pinMode is too slow. A general template define the port and the pins, as in the example below, with the Oled wired on pins 7,6,5,4 (program `TestOledbb7654.ino)`

```
#define Ddr  DDRD
#define Port PORTD
#define Pin  PIND
#define bCk 5  // pin 5
#define bDa 4  // pin 4
#define bGnd 7
#define bVcc 6
#define Ck1  bitClear (Ddr,bCk)
#define Ck0  bitSet (Ddr,bCk)
#define Da1  bitClear (Ddr,bDa)
#define Da0  bitSet (Ddr,bDa)

void Setupbb() { //Avec OledPix
  Ddr  |=  (1<<bGnd|1<<bVcc|1<<bCk|1<<bDa) ;
  Port |=  (1<<bVcc) ;
  Port &= ~(1<<bGnd|1<<bCk|1<<bDa) ;
}
```

Notice that only the first 7 lines have to be modified if the 4 pins are on the same ports
Program `TestOledbb7654.ino` takes 1816 bytes and 47 variables.

Add another Oled

The second Oled will be defined with a similar set of definitions and the setup will have to initialize both.
Now, we do not want to duplicate the OledPix library.
The OledPix lib uses 3 I2C functions to communicate with the Oled: Start, Stop, Write.
We need to have these functions including a switch to select the transfer toward Oled #1 or #2.
Let uns name the variable `selOled`, with values `0` and `1` for two Oleds.
The Bb primitives, tested separately for the 2 Oleds, are `Start0;`, `Start1();` etc
The Oled Start(); function or macro just need to include an if statement.

```
  void Start(){
     if(SelOled==0) {Start0();}
     else  { Start1();}
  }
```

Etc.. Write carefully and it will work..
Our example in the zip www.didel.com/OledBbLib.zip uses macros for Start0();, etc., which does not seems to be worth, but we did not compare precisely code size and execution time.

jdn 180421