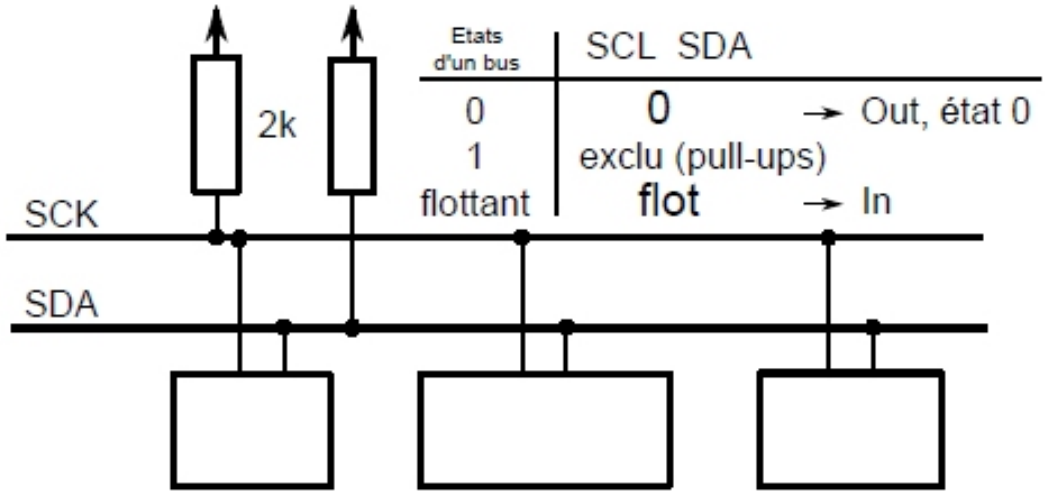
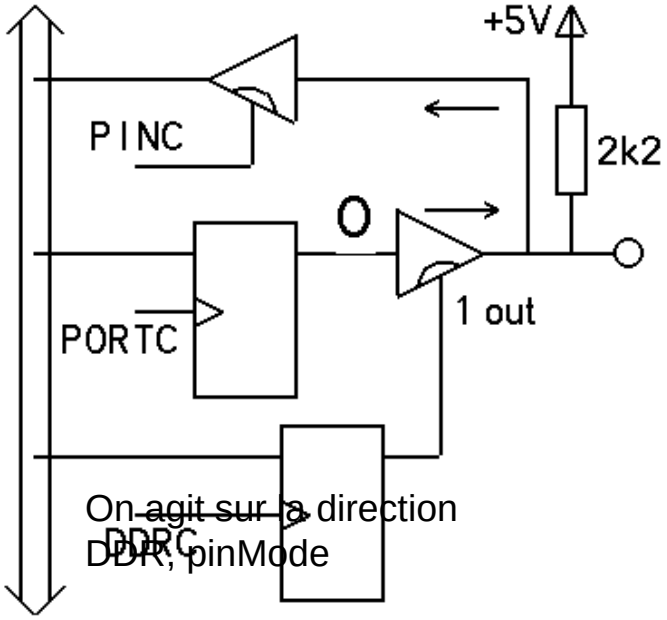
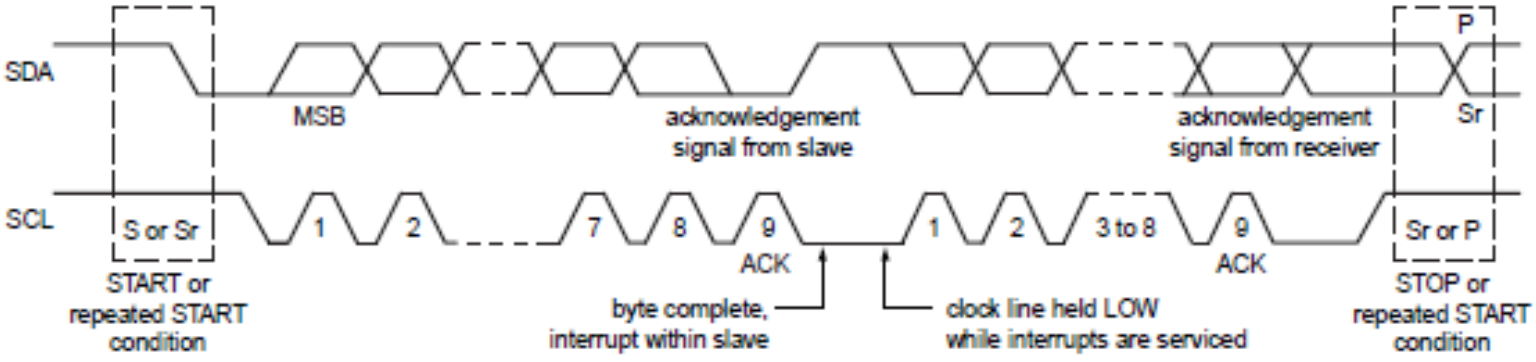
Microclub 9 septembre 2016

Bon à savoir - SMbus et Python

J.D. Nicoud



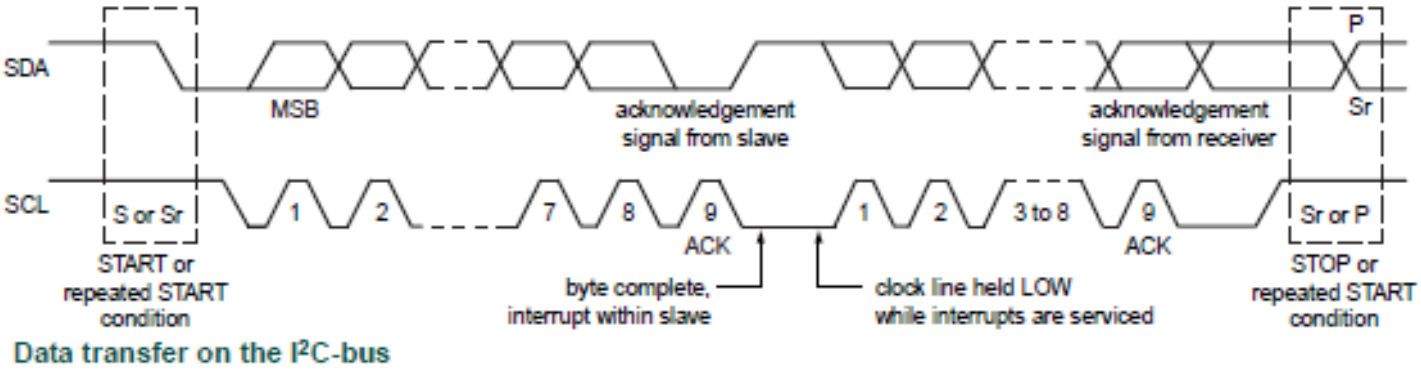




Data transfer on the I²C-bus

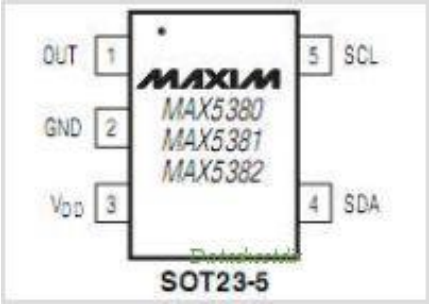
Low level macros/functions

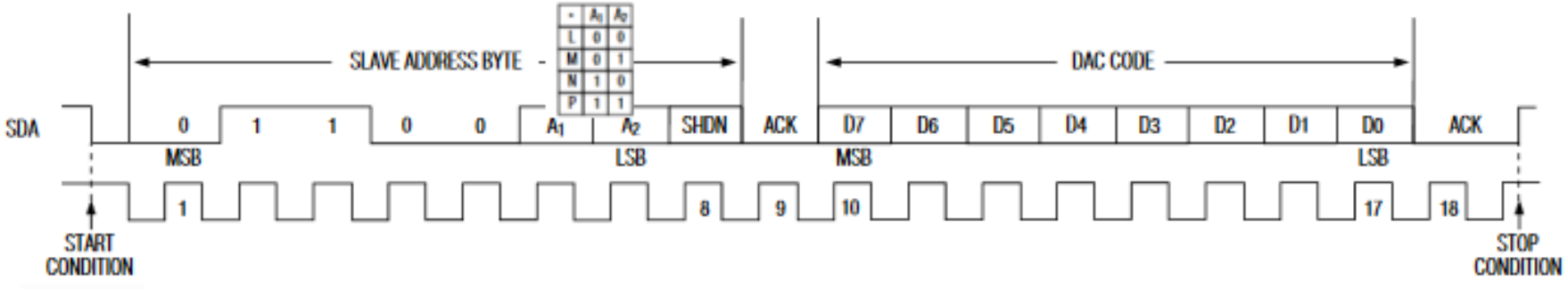
Start
Stop
Restart
Write8
  GetAck
Read8
  GiveAck



Data transfer on the I²C-bus

DA MAX5382 8-bit DAC

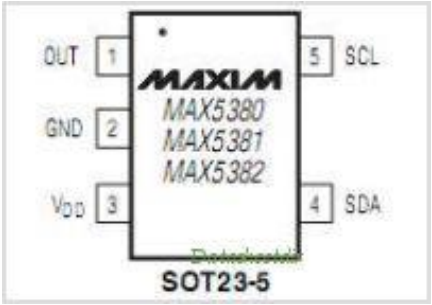


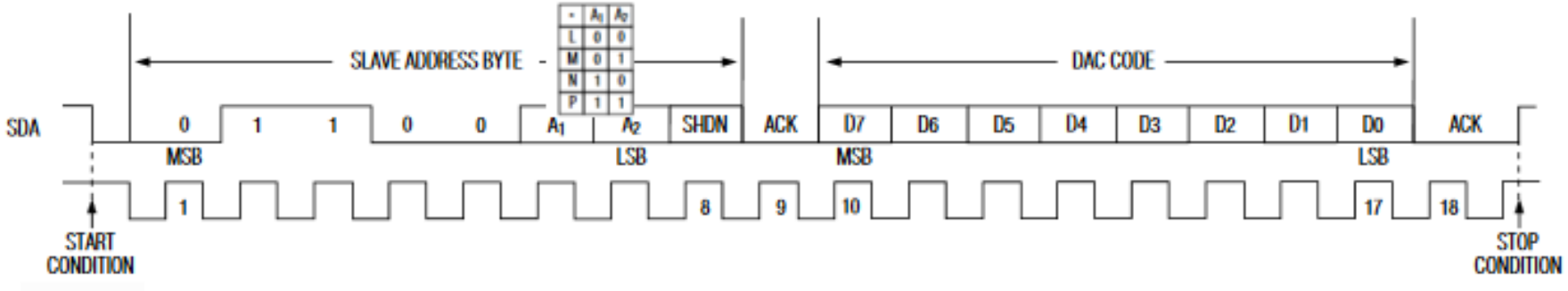


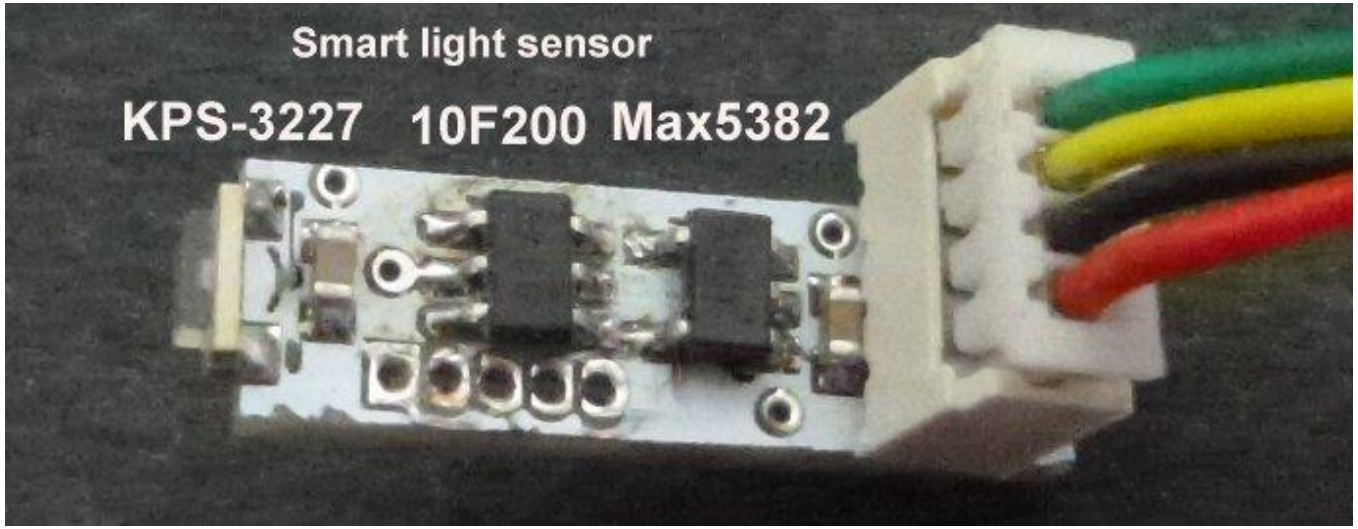
Assembleur PIC 10F200
Ecriture MCP5383 (D/A)
160 bytes (10+13 lignes+macros)

```
.Macro  Start
        Set     d0c1    ; SDA '"\.......
        Nop2
        Set     d0c0    ; SCL '""""\.....
        Nop
.Endmacro
```

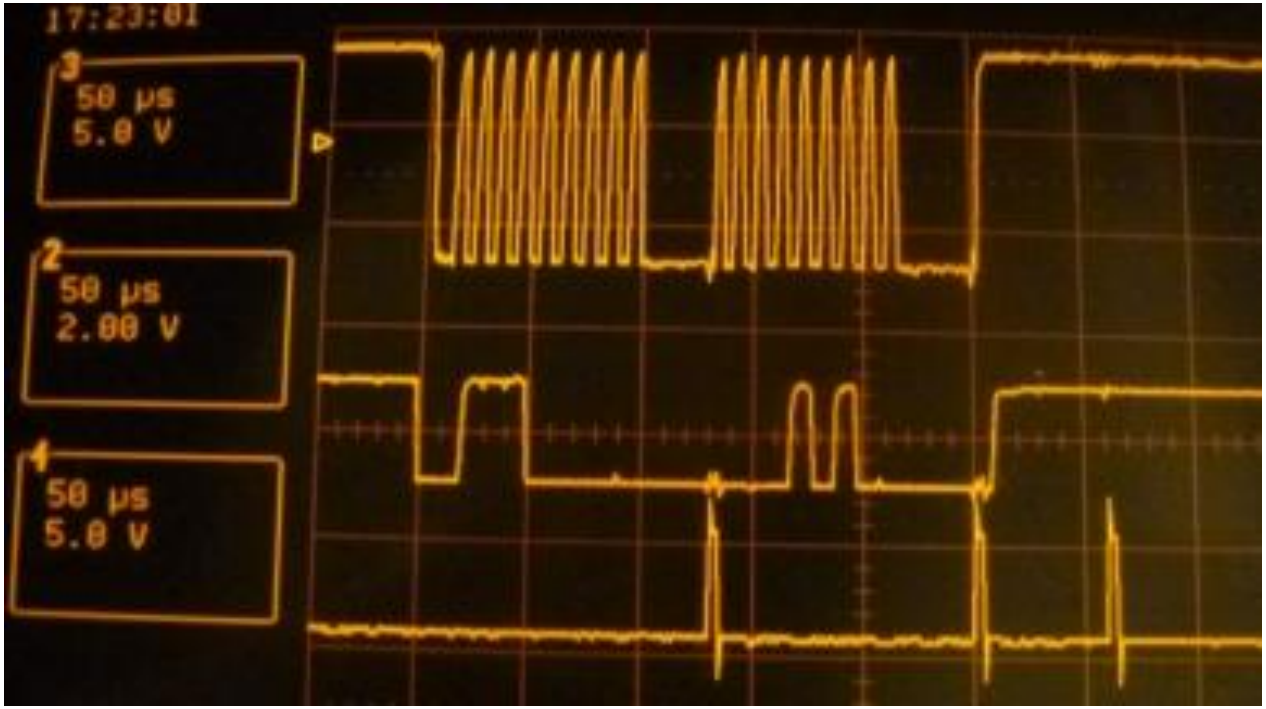
DA MAX5382 8-bit DAC







AVR TWI **module (flag ou interrupt)**



Arduino Librairie I2C

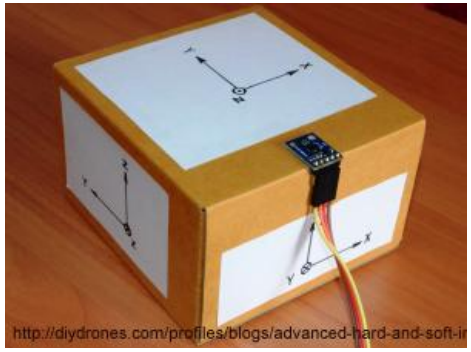
```
#include <Wire.h>
setup
Wire.begin();
écriture
Wire.beginTransmission(addr);    //start et adresse
Wire.write (data);
Wire.endTransmission();          //stop
 lecture
 Wire.requestFrom(addr,2);    //start adresse, long
 Data = Wire.read();
 Data = Wire.read();    // stop ajouté selon long

 On peut faire une boucle for :
 Wire.requestFrom(addr,long);    //start adresse
 while (available()  {      //tant que  tampon non vide
   Data = Wire.read();
 }
```

Arduino

```
#include <Wire.h>
void setup() {
   Wire.begin();
}
#define AdI2C 0x30  // Max5382
byte data=0;

void loop()  {
  Wire.beginTransmission(AdI2C);
  Wire.write(data);
  Wire.endTransmission();

  data++;
  delay(100);
}
```

Python

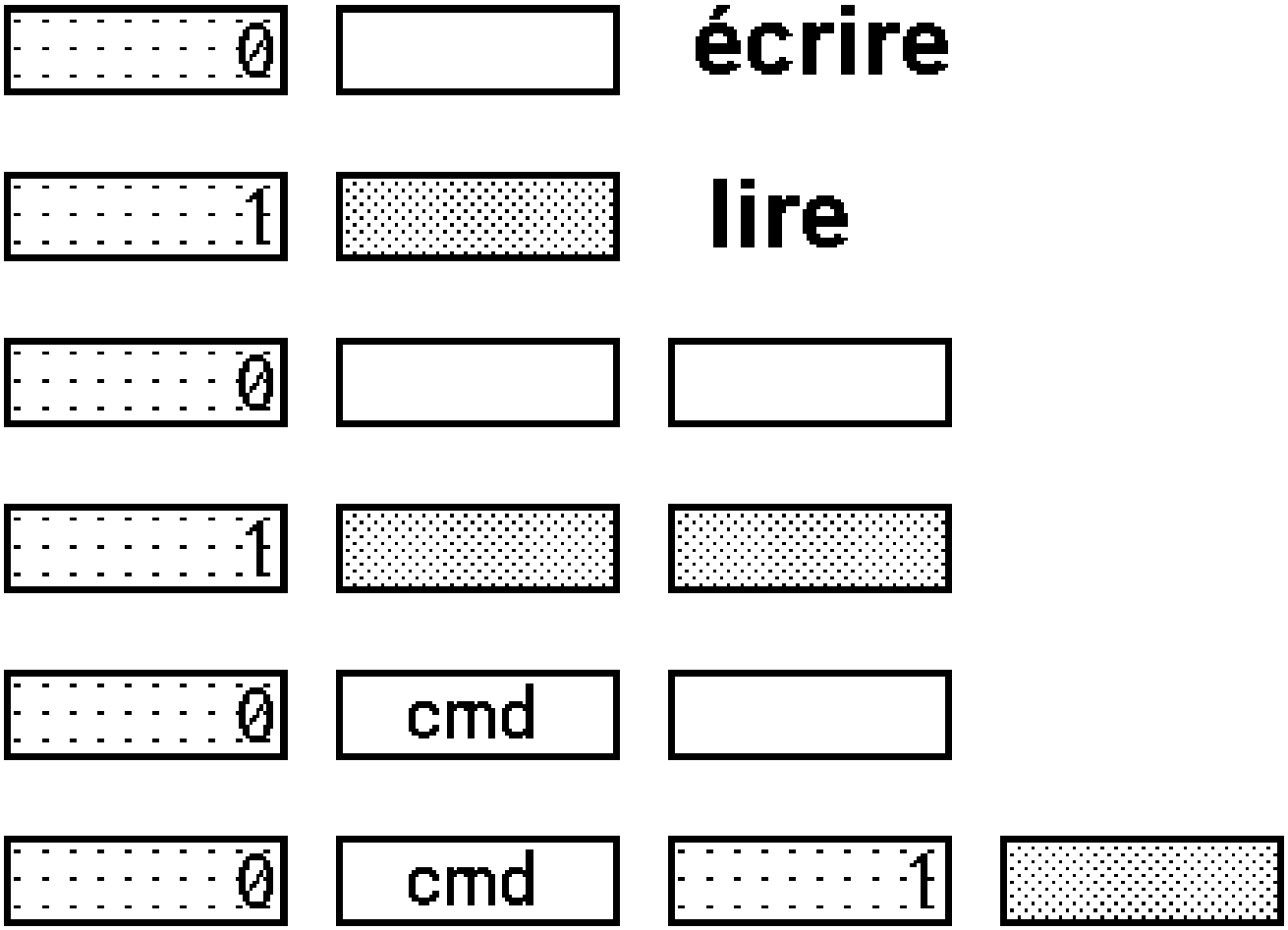
```
import smbus
bus = smbus.SMBus(1)
import time

AdI2C = 0x30    #Max5382
data = 0

while True:

    bus.write_byte(AdI2C,data)


    data = data+1
    time.sleep(0.1)
```


http://diydrones.com/profiles/blogs/advanced-hard-and-soft-ir


1
I²C
DI
Addr
Byte
Write
1 byte
4
65


GND VCC C D
Klaue
Oled
Didel


10 DOF


0
écrire
1
lire
0
1
0
cmd
0
cmd
1

6.1. Register Descriptions

Si 7021 température et humidité

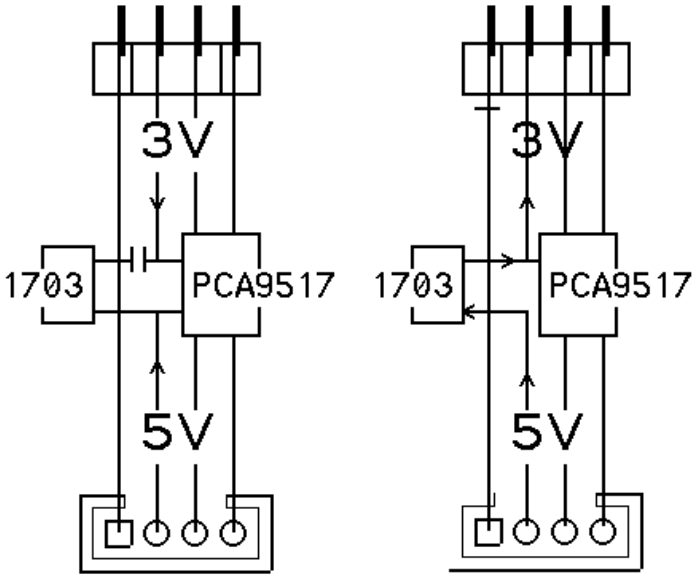
Register 1. User Register 1

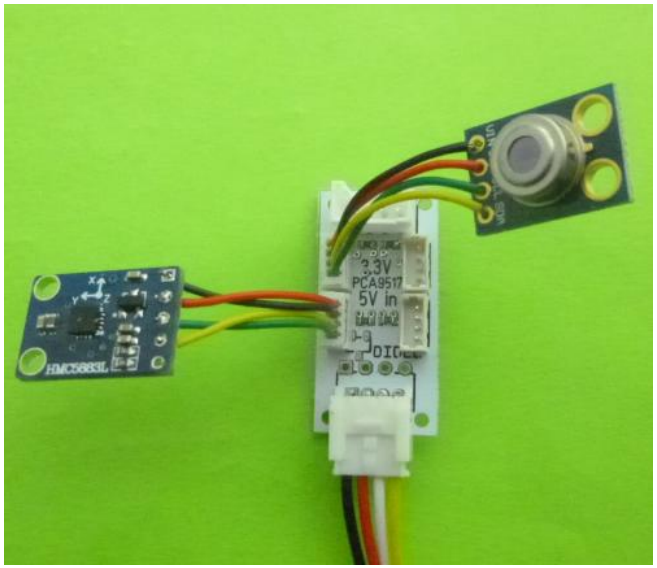
Bit	D7	D6	D5	D4	D3	D2	D1	D0
Name	RES1	VDDS	RSVD	RSVD	RSVD	HTRE	RSVD	RES0
Type	R/W	R	R/W	R/W		R/W	R/W	R/W

Reset Settings = 0011_1010

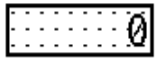
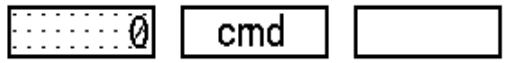
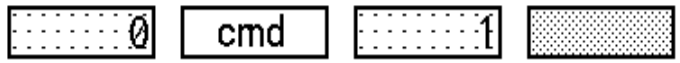
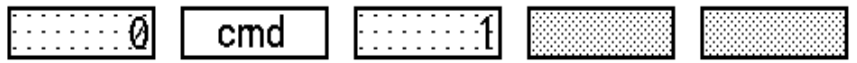
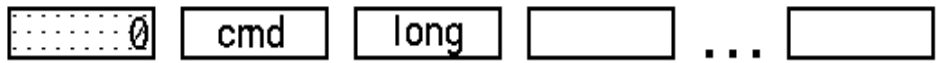
Bit	Name	Function
D7; D0	RES[1:0]	Measurement Resolution: RH / Temp 00: 12 bit / 14 bit 01: 8 bit / 12 bit 10: 10 bit / 13 bit 11: 11 bit / 11 bit
D6	VDDS	VDD Status: 0: V_{DD} OK 1: V_{DD} Low The minimum recommended operating voltage is 1.9 V. A transition of the VDD status bit from 0 to 1 indicates that VDD is between 1.8 V and 1.9 V. If the VDD drops below 1.8 V, the device will no longer operate correctly.
D5, D4, D3	RSVD	Reserved
D2	HTRE	1=On-chip Heater Enable 0=On-chip Heater Disable
D1	RSVD	Reserved

5V/3V PCA9517

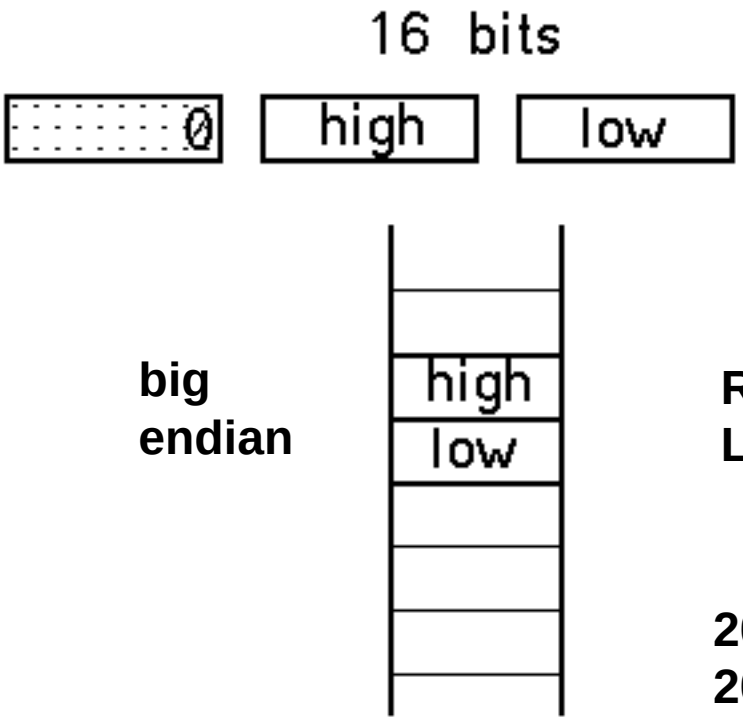




python:smbus

write_quick(addr)	
read_byte(addr)	
write_byte(addr,val)	
read_byte_data(addr,cmd)	
write_byte_data(addr,cmd,val)	
read_word_data(addr,cmd)	
write_word_data(addr,cmd,val)	
read_block_data(addr,cmd)	
write_block_data(addr,cmd,vals)	

read_i2c_block_data(addr,cmd)
write_i2c_block_data(addr,cmd,vals)

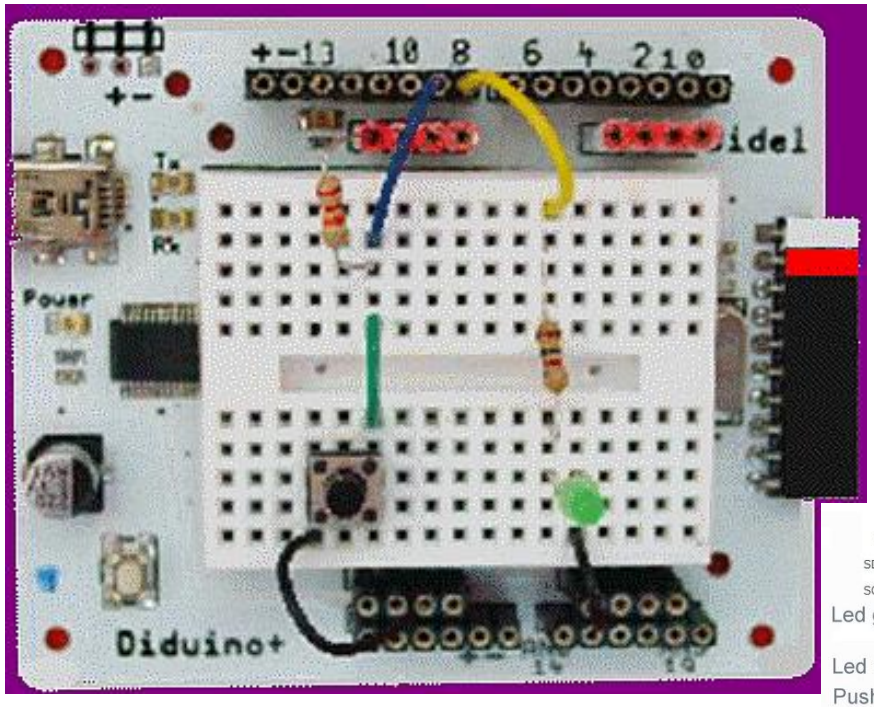


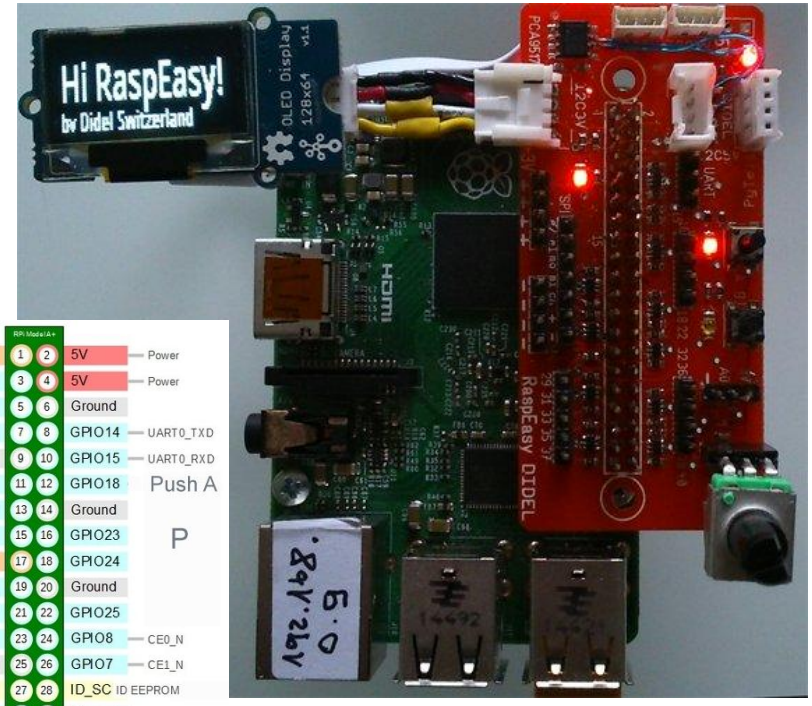
big endian

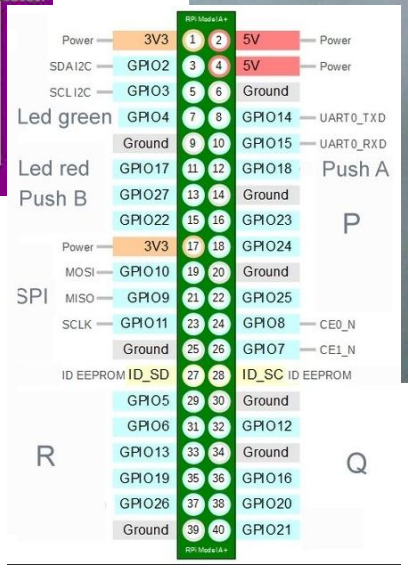
Raspberry/Python est Little endian

20.9.2016 little endian (EU)
2016.9.20 big endian (ISO, chine)
9-20-2016 middle endian

# Exchanges high and low bytes
data = ((rd & **0xFF**) << **8**) | ((rd & **0xFF00**) >> **8**)


Didel
Power
Diduino+


Hi RaspEasy!
by Didel Switzerland
OLED Display
128x64
RaspEasy DIDEL

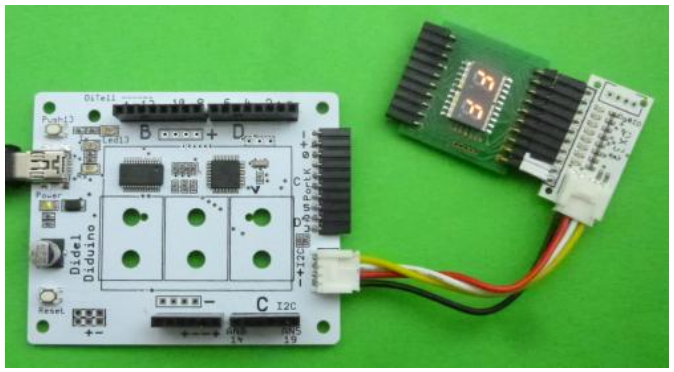

Power 3V3 1 2 5V Power
SDA I2C GPIO2 3 4 5V Power
SCL I2C GPIO3 5 6 Ground
Led green GPIO4 7 8 GPIO14 UART0_TXD
Ground 9 10 GPIO15 UART0_RXD
Led red GPIO17 11 12 GPIO18 Push A
Push B GPIO27 13 14 Ground
GPIO22 15 16 GPIO23 P
Power 3V3 17 18 GPIO24
MOSI GPIO10 19 20 Ground
SPI MISO GPIO9 21 22 GPIO25
SCLK GPIO11 23 24 GPIO8 CE0_N
Ground 25 26 GPIO7 CE1_N
ID EEPROM ID_SD 27 28 ID_SC ID EEPROM
GPIO5 29 30 Ground
GPIO6 31 32 GPIO12
R GPIO13 33 34 Ground Q
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
Ground 39 40 GPIO21

Raspberry Python vs Arduino

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)   # ! pins
GPIO.setup(P_BUTTON, GPIO.IN)
GPIO.setup(P_LED, GPIO.OUT)

GPIO.input(P_BUTTON)
GPIO.output(P_LED, GPIO.HIGH)


def  copy:
    time.sleep(0.2)
    GPIO.output(P_LED, GPIO.HIGH)

if GPIO.input(P_BUTTON) == GPIO.HIGH

while True:

for  num in range ( 0,10)
```

```
pinMode (pin,OUT);


digitalWrite (pin, HIGH);
digitalRead (pin)

void copy (int var) {
   delay (var);
   digitalWrite (Led,High);
}

If (PousOn == 0) {  }

while (1) { }

for (int i=0; i<10;i++) ;
```

```
// TestWrite8574.ino  ecrit 8 bits
#include <Wire.h>
#define AdI2C 0x38  // or 0x20
void setup() {
  Wire.begin;
}
void loop()  {
  Wire.beginTransmission(AdI2C); // start+adresse
  Wire.write(0x55);              // data
  Wire.endTransmission();        // stop
  for(;;);
}
```



```
byte data;
void loop() {
  Wire.requestFrom(Ad8574,1);
  data =  Wire.read();
  delay(100);
}
```

```
import smbus
bus = smbus.SMBus(1)
import time
AdI2C = 0x38    # 8574
data = 0
while True:
    bus.write_byte(AdI2C,data)
    data = data+1
    time.sleep(0.1)
```

```
import smbus
import time
print "starting..."
bus = smbus.SMBus(1)
adc_address = 0x48   # ADC0
while True:     # Reads word (16 bits) as int
rd = bus.read_word_data(adc_address, 0)

# Exchanges high and low bytes
data = ((rd & 0xFF) << 8) | ((rd & 0xFF00) >> 8)

# Ignores two least significiant bits
data = data >> 2
print "data:", data
time.sleep(1)
```

```
// ScanI2C.ino  3812/455
#include <Wire.h>
void setup() {
  Serial.begin (9600);
  Wire.begin();
}
byte count;
void loop() {
  Serial.println ("Scan");
  for (byte i = 1; i < 120; i++)
  {
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0) {
      Serial.print ("Address: ");
      Serial.print ("0x");  Serial.print (i, HEX);
      Serial.print (" ("); Serial.print (i,(DEC));
Serial.println (")");
      count++;
      } // end of good response
  } // end of for loop
  Serial.println ("Done.");
  Serial.print ("Found ");
  Serial.print (count, DEC);
  Serial.println (" device(s).");

  for (;;);  // reset pour nouvelle analyse
}
```

Python