# Guide de mise en marche du module LK-Digi (35430)

## Présentation :

Afficheur 4 digits compatible Linker. Il communique avec un microtrôleur type Arduino ou Raspberry avec seulement 2 broches.

Il se raccorde sur deux sorties d'une carte à microcontrôleur avec un cordon non inclus via:
- le shield 35420 pour une carte Arduino ou compatible
- le shield 35421 pour une carte Raspberry B+, 2 et 3

Interface: compatible Linker
Alimentation: 3,3 à 5,5 Vcc
Affichage: 4 digits 7 segments
Hauteur des caractères: 9 mm
Dimensions: 42 x 24 x 14 mm

## Exemple de code Arduino:

Vous devez télécharger et installer la librairie à l'adresse: https://github.com/avishorp/TM1637

Vous pouvez utilisez l'exemple disponible: Fichier -> exemples -> TM1637-master -> TM1637test

# Utilisation du module sous Raspberry:

Commencez par creer le fichier tm1637.py en tapant la commande:
sudo nano tm1637.py
Copier le contenu suivant:

```
import sys
import os
import time
import RPi.GPIO as IO

IO.setwarnings(False)
IO.setmode(IO.BCM)

HexDigits = [0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71]

ADDR_AUTO = 0x40
ADDR_FIXED = 0x44
STARTADDR = 0xC0
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHT_HIGHEST = 7
OUTPUT = IO.OUT
INPUT = IO.IN
LOW = IO.LOW
HIGH = IO.HIGH

class TM1637:
    __doublePoint = False
    __Clkpin = 0
    __Datapin = 0
    __brightnes = BRIGHT_TYPICAL;
    __currentData = [0,0,0,0];

    def __init__( self, pinClock, pinData, brightnes ):
        self.__Clkpin = pinClock
        self.__Datapin = pinData
        self.__brightnes = brightnes;
        IO.setup(self.__Clkpin,OUTPUT)
        IO.setup(self.__Datapin,OUTPUT)
    # end  __init__

    def Clear(self):
        b = self.__brightnes;
        point = self.__doublePoint;
        self.__brightnes = 0;
        self.__doublePoint = False;
        data = [0x7F,0x7F,0x7F,0x7F];
        self.Show(data);
        self.__brightnes = b;              # restore saved brightnes
        self.__doublePoint = point;
    # end  Clear

    def ShowInt(self, i):
        s = str(i)
        self.Clear()
        for i in range(0,len(s)):
            self.Show1(i, int(s[i]))

    def Show( self, data ):
        for i in range(0,4):
            self.__currentData[i] = data[i];

        self.start();
        self.writeByte(ADDR_AUTO);
        self.stop();
        self.start();
        self.writeByte(STARTADDR);
        for i in range(0,4):
            self.writeByte(self.coding(data[i]));
        self.stop();
        self.start();
        self.writeByte(0x88 + self.__brightnes);
        self.stop();
    # end  Show
```

```
  def SetBrightnes(self, brightnes):      # brightnes 0...7
    if( brightnes > 7 ):
      brightnes = 7;
    elif( brightnes < 0 ):
      brightnes = 0;

    if( self.__brightnes != brightnes):
      self.__brightnes = brightnes;
      self.Show(self.__currentData);
    # end if
  # end  SetBrightnes

  def ShowDoublepoint(self, on):          # shows or hides the doublepoint
    if( self.__doublePoint != on):
      self.__doublePoint = on;
      self.Show(self.__currentData);
    # end if
  # end  ShowDoublepoint

  def writeByte( self, data ):
    for i in range(0,8):
      IO.output( self.__Clkpin, LOW)
      if(data & 0x01):
        IO.output( self.__Datapin, HIGH)
      else:
        IO.output( self.__Datapin, LOW)
      data = data >> 1
      IO.output( self.__Clkpin, HIGH)
    #endfor

    # wait for ACK
    IO.output( self.__Clkpin, LOW)
    IO.output( self.__Datapin, HIGH)
    IO.output( self.__Clkpin, HIGH)
    IO.setup(self.__Datapin, INPUT)

    while(IO.input(self.__Datapin)):
      time.sleep(0.001)
      if( IO.input(self.__Datapin)):
        IO.setup(self.__Datapin, OUTPUT)
        IO.output( self.__Datapin, LOW)
        IO.setup(self.__Datapin, INPUT)
      #endif
    # endwhile
    IO.setup(self.__Datapin, OUTPUT)
  # end writeByte

  def start(self):
    IO.output( self.__Clkpin, HIGH) # send start signal to TM1637
    IO.output( self.__Datapin, HIGH)
    IO.output( self.__Datapin, LOW)
    IO.output( self.__Clkpin, LOW)
  # end start

  def stop(self):
    IO.output( self.__Clkpin, LOW)
    IO.output( self.__Datapin, LOW)
    IO.output( self.__Clkpin, HIGH)
    IO.output( self.__Datapin, HIGH)
  # end stop

  def coding(self, data):
    if( self.__doublePoint ):
      pointData = 0x80
    else:
      pointData = 0;

    if(data == 0x7F):
      data = 0
    else:
      data = HexDigits[data] + pointData;
    return data
  # end coding

# end class TM1637
```

Creer ensuite le fichier clock.py en tapant la commande:
sudo nano clock.py
Copier le contenu suivant:

```
import sys
import time
import datetime
import RPi.GPIO as GPIO
import tm1637

Display = tm1637.TM1637(4,5,tm1637.BRIGHT_TYPICAL)

Display.Clear()
Display.SetBrightnes(1)

while(True):
  now = datetime.datetime.now()
  hour = now.hour
  minute = now.minute
  second = now.second
  currenttime = [ int(hour / 10), hour % 10, int(minute / 10), minute % 10 ]

  Display.Show(currenttime)
  Display.ShowDoublepoint(second % 2)

  time.sleep(1)
```

Le module afficheur peut maintenant être raccordé sur la carte Raspberry, ici sur l'entrée D4 et demarré avec la commande suivante:
sudo python clock.py
Le module affiche l'heure.

Si vous rencontrez des problèmes, merci de nous contacter par courriel à :

sav@gotronic.fr