Welcome!

Thank you for purchasing our *AZ-Delivery 4 Digits 7 Segment Display*. On the following pages, we will introduce you to how to use and set-up this handy device.

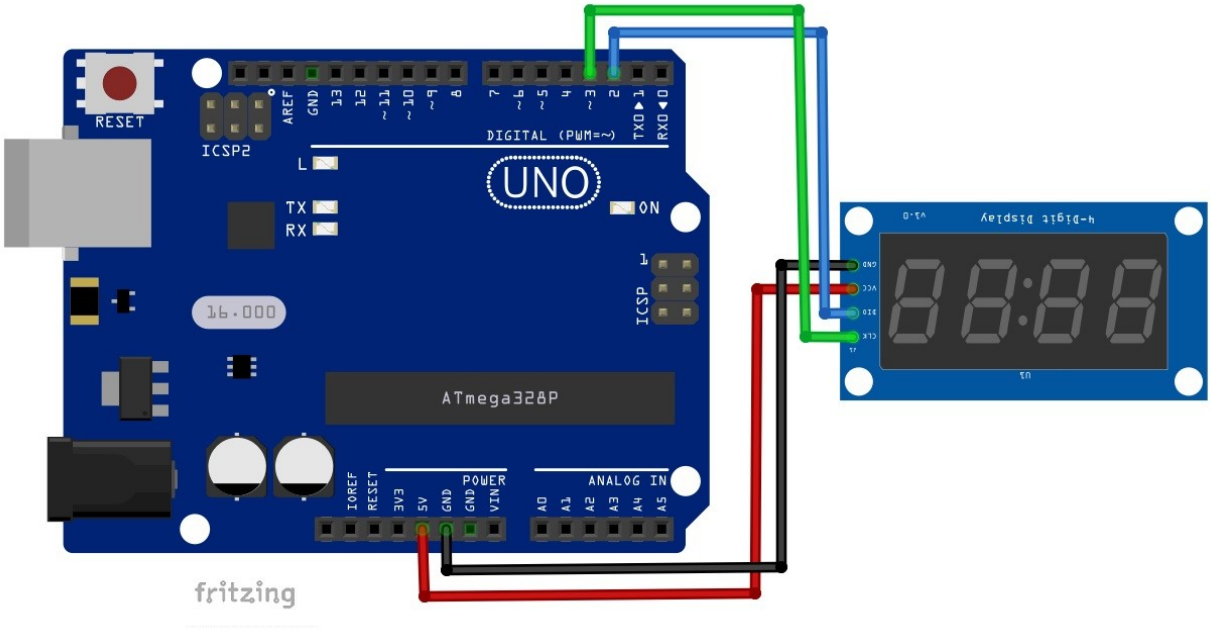
Have fun!

The seven segment display contains seven individually addressable LEDs packed in one package, labeled from *A* to *G*. In the 7 segment 4 digit display there are four seven segment displays, thus we are able to show 4 digits with this module. These kind of modules usually require 12 digital pins in order to control it, but the module we write about in this eBook has just four. This is because of the driver chip, called "*TM1637*". The driver chip is a special circuit for driving and controlling LED displays. To communicate with the driver chip we use some form of two wire interface. It is not I2C interface because the driver chip does not use addressing like in I2C interface. We use 4 wires to connect the module with a microcontroller: any two digital pins for communication, power and ground wires.

Specifications:

» Power supply and logic voltage range:	from 3.3V to 5V DC
» Working DC current:	30mA - 80mA
» LED color:	RED
» Operating temperature:	from -10°C to 80°C
» Changeable brightness:	in software

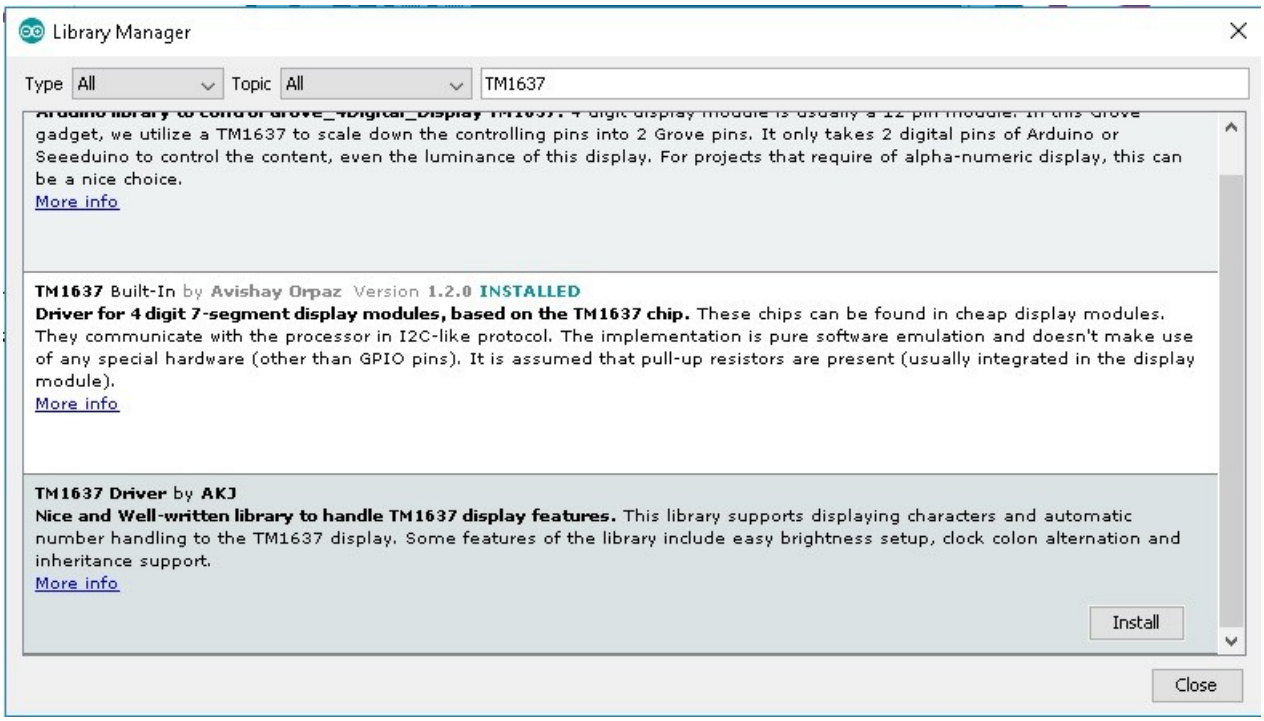
Connecting the module with Uno



Module pin	>	Uno pin	
GND	>	GND	**Black wire**
VCC	>	5V	**Red wire**
DIO	>	D3	**Green wire**
CLK	>	D2	**Blue wire**

Arduino IDE library

In order to use the module with the Uno, it is best if we download a library for it. Open the Arduino IDE and go to *Tools* > *Manage Libraries*, and a new window will open. Type "*TM1637*" in the search box and install a library "*TM1637*" made by *"Avishay Orpaz"* like on image below:



To run an example sketch, open the following sketch:

File > *Examples* > *TM1637* > *TM1637Test*

Sketch code:

```
#include <Arduino.h>
#include <TM1637Display.h>
#define CLK   2
#define DIO   3
#define TEST_DELAY   2000 //  delay in miliseconds
const uint8_t SEG_DONE[] = {
  SEG_B | SEG_C | SEG_D | SEG_E | SEG_G,           // d
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,   // O
  SEG_C | SEG_E | SEG_G,                           // n
  SEG_A | SEG_D | SEG_E | SEG_F | SEG_G };         // E
TM1637Display display(CLK, DIO);
uint8_t data[] = {0xff, 0xff, 0xff, 0xff};
uint8_t blank[] = {0x00, 0x00, 0x00, 0x00};
void setup() {
  display.setBrightness(0x0f);
}
void loop() {
  turnON_allSegments();
  turnON_segment();
  display_numbers();
  brightness_test();
  ON_OFF_test();
  display_DONE();
  delay(5000);
}
void turnON_allSegments() {
  display.setSegments(data);
  delay(TEST_DELAY);
}
```

```
void turnON_segment() {
  // Selectively set different digits
  data[0] = display.encodeDigit(0);
  data[1] = display.encodeDigit(1);
  data[2] = display.encodeDigit(2);
  data[3] = display.encodeDigit(3);
  display.setSegments(data);
  delay(TEST_DELAY);
  display.clear();
  display.setSegments(data+2, 2, 2);
  delay(TEST_DELAY);
  display.clear();
  display.setSegments(data+2, 2, 1);
  delay(TEST_DELAY);
  display.clear();
  display.setSegments(data+1, 3, 1);
  delay(TEST_DELAY);
}
void display_numbers() {
  // How to show decimal numbers in many ways
  display.showNumberDec(0, false); // Expect: ___0
  delay(TEST_DELAY);
  display.showNumberDec(0, true);  // Expect: 0000
  delay(TEST_DELAY);
  display.showNumberDec(1, false); // Expect: ___1
  delay(TEST_DELAY);
  display.showNumberDec(1, true);  // Expect: 0001
  delay(TEST_DELAY);
  display.showNumberDec(301, false); // Expect: _301
  delay(TEST_DELAY);
  display.showNumberDec(301, true); // Expect: 0301
```

```
// one tab
  delay(TEST_DELAY);
  display.clear();
  display.showNumberDec(14, false, 2, 1); // Expect: _14_
  delay(TEST_DELAY);
  display.clear();
  display.showNumberDec(4, true, 2, 2);  // Expect: __04
  delay(TEST_DELAY);
  display.showNumberDec(-1, false);  // Expect: __-1
  delay(TEST_DELAY);
  display.showNumberDec(-12);        // Expect: _-12
  delay(TEST_DELAY);
  display.showNumberDec(-999);       // Expect: -999
  delay(TEST_DELAY);
  display.clear();
  display.showNumberDec(-5, false, 3, 0); // Expect: _-5_
  delay(TEST_DELAY);
  display.showNumberHexEx(0xf1af);        // Expect: f1Af
  delay(TEST_DELAY);
  display.showNumberHexEx(0x2c);          // Expect: __2C
  delay(TEST_DELAY);
  display.showNumberHexEx(0xd1, 0, true); // Expect: 00d1
  delay(TEST_DELAY);
  display.clear();
  display.showNumberHexEx(0xd1, 0, true, 2); // Expect: d1__
delay(TEST_DELAY);
}
```

```
void turn_dots() {
  for(int k = 0; k <= 4; k++) {
    display.showNumberDecEx(0, (0x80 >> k), true);
    delay(TEST_DELAY);
  }
}
void brightness_test() {
  for(int k = 0; k < 4; k++) {
    data[k] = 0xff;
  }
  for(int k = 0; k < 7; k++) {
    display.setBrightness(k);
    display.setSegments(data);
    delay(TEST_DELAY);
  }
  display.setBrightness(4);
}
void ON_OFF_test() {
  for(int k = 0; k < 4; k++) {
    display.setBrightness(7, false);  // Turn off
    display.setSegments(data);
    delay(TEST_DELAY);
    display.setBrightness(7, true); // Turn on
    display.setSegments(data);
    delay(TEST_DELAY);
  }
}
void display_DONE() {
  display.setSegments(SEG_DONE);
}
```

At the beginning of the sketch two ***macros*** are created, called *CLK* and *DIO*. These macros represent digital pin names of the Uno that are used to connect the module with Uno.

After that, another macro is created, called *TEST_DELAY*; it is a time interval between two display functions and it is initialized on *2000* milliseconds, or *2* seconds.

After macros, we create one constant array and two variable arrays. Constant array is called *SEG_DONE* and it is used to store four elements. Every element contains macros for the segment that is turned *ON*, on the seven segment digit of the module. Therefore, to display the letter "*d*" we need to turn *ON* these segments (or to store these macros in one element of the array): *SEG_B*, *SEG_C*, *SEG_D*, *SEG_E* and *SEG_G*.
We use this constant array to display word "*dOnE*".
The other arrays are called `data[]` and `blank[]`. Both arrays contain four elements, one element for each seven segment display onboard the module. `Blank[]` is used to turn *OFF* all the segments on the module. `Data[]` is used to display specific characters on the module. Later in the text we will show examples of this array usage.

In the `setup()` function we use `setBrightness()` function, to adjust the brightness of the display. This function accepts one argument and it is a hexadecimal number; the maximal brightness value is *0x0f*.

In the `loop()` function we call all other functions that we created in the sketch, and wait for 5 seconds between each loop of the `loop()` function.

First function that is used in the `loop()` function is `turnON_allSegments()`, and it does what its name says, it is used to turn *ON* all the segments of the module, and then wait for 2 seconds (*TEST_DELAY* macro value). This is done by using the following line of the code: `display.setSegments(data)`
`setSegments()` is a function that accepts one argument, and it is `data[]` array, that we previously created.

Next function is `turnON_segment()`. It is used to show how to use `setSegments()` function, in different ways. First we encode digits *0*, *1*, *2* and *3*, and store it in the `data[]` array. Then we output that with the following line of the code: `display.setSegments(data)`.
After this, we show three examples of how to shift the `data[]` array on the segments of the module. To clear the display, or to turn *OFF* the modules, you could use another method, and that is to call the following function:
`display.clear()`

The `display_numbers()` function is used to show many methods of how to display a specific number. Each line of this function is commented, explaining what it does.

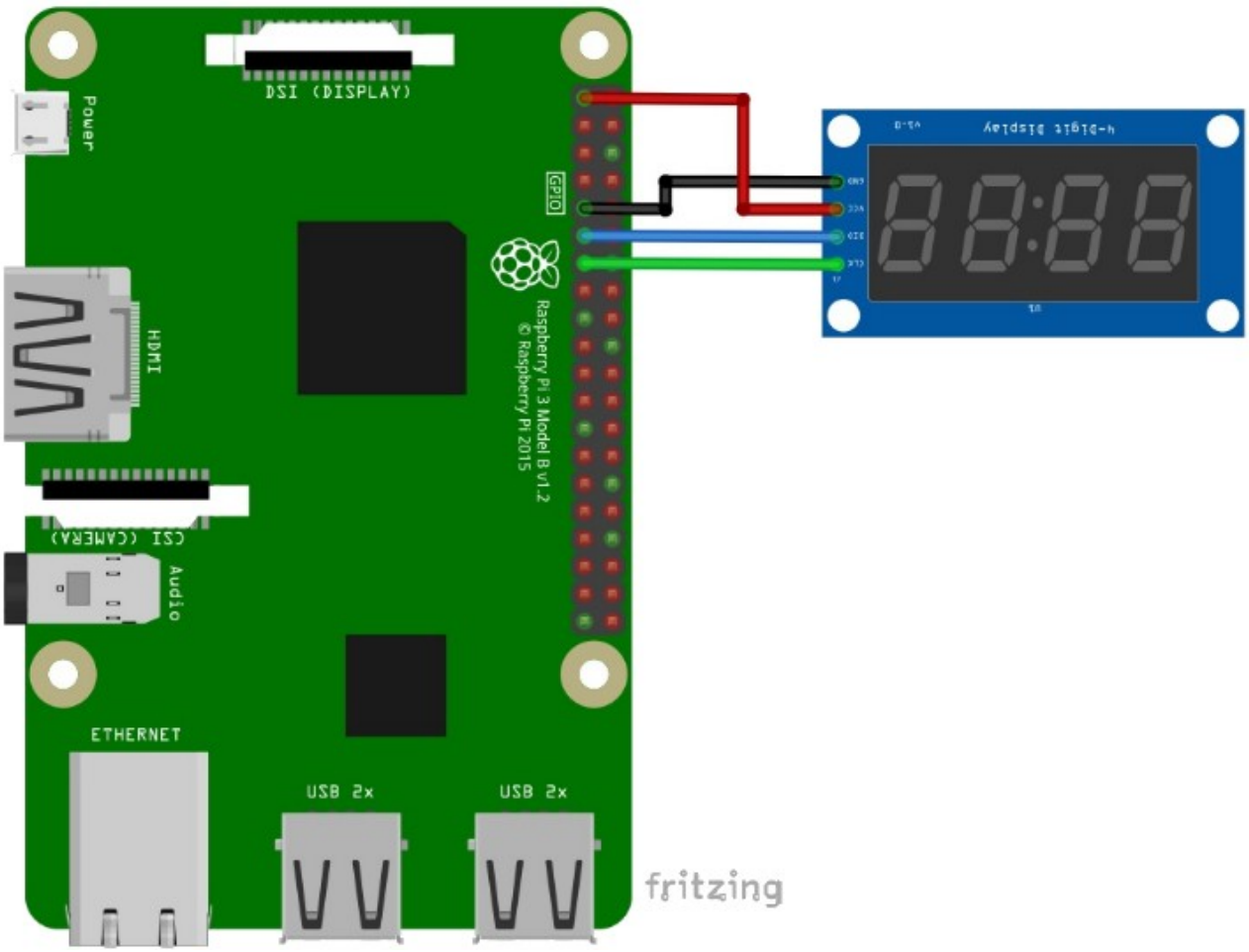
The `turn_dots()` function is used to turn two LEDs between second and third segment of seven segment the module.

The `brightness_test()` function is used to show all brightness levels that can be used for this module.

The `ON_OFF_test()` function is used to show you another method of how to turn *ON* or *OFF* all segments of the module.

And, the `display_DONE()` function is used to display the word "*dOnE*" from the *SEG_DONE* array.

Connecting the module with Raspberry Pi



Module pin	>	Raspberry Pi pin		
VCC	>	3V3	[pin 1]	**Red wire**
GND	>	GND	[pin 9]	**Black wire**
DIO	>	GPIO17	[pin 11]	**Blue wire**
CLK	>	GPIO27	[pin 13]	**Green wire**

Python library

In order to use the module with the Raspberry Pi, it is recommended to download a library. We will use the library called *“raspberrypi-python-tm1637”*. To download and install this library, we need to have installed *wiringPi* and *GIT* app. These tools come preinstalled with the Raspbian operating system, but if you do not have them installed, here is how to do it. Open terminal and run the following commands:

`sudo apt install git -y` - for *git*

and

`sudo pip3 install wiringpi` - for *wiringPi*

To download the library, run the following command:

git clone https://github.com/depklyon/raspberrypi-python-tm1637.git

After that, change directory to downloaded folder with:

`cd raspberry-python-tm1637`

And to install the library run the following command:

`sudo python3 setup.py install`

Python script:

```
import tm1637
from time import sleep
tm = tm1637.TM1637(clk=27, dio=17)
def show():
    tm.write([127, 255, 127, 127]) # all LEDS on "88:88"
    sleep(1)
    tm.write([0, 0, 0, 0]) # all LEDS off
    sleep(1)
    tm.write([63, 6, 91, 79]) # show "0123"
    sleep(1)
    tm.write([0b00111001, 0b00111111, 0b00111111, 0b00111000]) # "COOL"
    sleep(1)
    tm.show('help') # show "HELP"
    sleep(1)
    tm.hex(0xdead) # display "dEAd"
    sleep(1)
    tm.hex(0xbeef) # display "bEEF"
    sleep(1)
    tm.numbers(11, 55) # show "11:55"
    sleep(1)
    tm.number(-955) # show "-955"
    sleep(1)
    tm.temperature(22) # show temperature '22*C'
    sleep(1)
print('[press ctrl + c to stop the script]')
try:
    while True:
        show()
        sleep(1)
except KeyboardInterrup:
    print('Script end!')
finally:
    tm.write([0, 0, 0, 0])
```

Save this code to a script, called "*sevenSegment.py*", and make sure the script is saved in the "*raspberrypi-python-tm1637*" directory. To run the script, open terminal in that directory, and run the following command:

```
python3 sevenSegment.py
```

NOTE: Script code is self explanatory.

You've done it!

You can now use your module for various projects.

Now is the time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us