Welcome

Thank you for choosing our *LCD screen from AZ-Delivery*. These are available individually or with I2C converter in a bundle in green or blue, and in two sizes: 16x02 and 20x04. In the following pages we will explain how you can set up and use the device.

Have fun!

Areas of application
Education and teaching: Use in schools, universities and training institutions to teach the basics of electronics, programming and embedded systems. Research and development: Use in research and development projects to create prototypes and experiments in the fields of electronics and computer science. Prototype development: Use in the development and testing of new electronic circuits and devices. Hobby and Maker Projects: Used by electronics enthusiasts and hobbyists to develop and implement DIY projects.

Required knowledge and skills
Basic understanding of electronics and electrical engineering. Knowledge of programming, especially in the C/C++ programming language. Ability to read schematics and design simple circuits. Experience working with electronic components and soldering.

Operating conditions
The product may only be operated with the voltages specified in the data sheet to avoid damage. A stabilized DC power source is required for operation. When connecting to other electronic components and circuits, the maximum current and voltage limits must be observed to avoid overloads and damage.

Environmental conditions
The product should be used in a clean, dry environment to avoid damage caused by moisture or dust. Protect the product from direct sunlight (UV), as this can negatively affect the lifespan of the display.

Intended Use
The product is designed for use in educational, research and development environments. It is used to develop, program and prototype electronic projects and applications. The product is not intended as a finished consumer product, but rather as a tool for technically savvy users, including engineers, developers, researchers and students.

Improper foreseeable use
The product is not suitable for industrial use or safety-relevant applications. Use of the product in medical devices or for aviation and space travel purposes is not permitted

disposal
Do not discard with household waste! Your product is according to the European one Directive on waste electrical and electronic equipment to be disposed of in an environmentally friendly manner. The valuable raw materials contained therein can be recycled become. The application of this directive contributes to environmental and health protection. Use the collection point set up by your municipality to return and Recycling of old electrical and electronic devices. WEEE Reg. No.: DE 62624346

electrostatic discharge
The display is sensitive to electrostatic discharge (ESD), which can damage or destroy the electronic components. Please note the following safety instructions to avoid ESD hazards: Attention: Electrostatic charges on your body can damage the display. Note: Ground yourself by wearing an anti-static wrist strap connected to a grounded surface or by touching a grounded metal surface before handling the display. Attention: Use anti-static mats and bags to protect the display. Note: Place the display on an anti-static work mat and store in anti-static bags when not in use. Note: A clean and grounded workplace minimizes the risk of ESD. Action: Keep your workplace clean and free of materials that can generate electrostatic charges. Make sure all surfaces used are grounded.

safety instructions
Although the display complies with the requirements of the RoHS Directive (2011/65/EU) and does not contain any hazardous substances in quantities above the permitted limits, residual chemical hazards may still exist. Please note the following safety instructions: Attention: The back of the display and the circuit board can release chemical residues from manufacturing or during operation. Note: Wear protective gloves when handling or installing the display for a long time to avoid skin irritation. Caution: Electronic components can emit small amounts of volatile organic compounds (VOCs), especially if the display is new. Note: Make sure you work in a well-ventilated area to minimize the concentration of fumes in the air. Caution: Do not use harsh chemicals or solvents to clean the display as they may damage the protective coating or electronics. Note: Use an anti-static cleaning cloth or special electronics cleaner to carefully clean the display. Although the display complies with

the requirements of the RoHS Directive (2011/65/EU) and does not contain any hazardous substances in quantities above the permitted limits, residual chemical hazards may still exist. Please note the following safety instructions: Attention: The back of the display and the circuit board can release chemical residues from manufacturing or during operation. Note: Wear protective gloves when handling or installing the display for a long time to avoid skin irritation. Caution: Electronic components can emit small amounts of volatile organic compounds (VOCs), especially if the display is new. Note: Make sure you work in a well-ventilated area to minimize the concentration of fumes in the air. Caution: Do not use harsh chemicals or solvents to clean the display as they may damage the protective coating or electronics. Note: Use an anti-static cleaning cloth or special electronics cleaner to carefully clean the display. The display contains sensitive electronic components and a top layer. Improper handling or excessive pressure can cause damage to the display or injury. Observe the following safety instructions to avoid mechanical hazards: Attention: The cover of the display is fragile and can break if handled improperly. Note: Avoid applying strong pressure or bending the display. Handle the display carefully and only by the circuit board to avoid breakages. Caution: Drops or impacts can crack the surface of the display and damage the electronic components on the back. Note: Avoid dropping the display and protect it from impacts. Use a soft surface when working to avoid scratches. Attention: If the display breaks, sharp pieces of glass can cause injuries. Note: If the display breaks, handle the fragments carefully and wear protective gloves to avoid cuts. Dispose of the glass pieces safely. Note: Improper attachment can lead to mechanical stress and breakage of the display. Action: Attach the display securely and without excessive pressure. Use appropriate brackets or housings to mount the display stably. Caution: Improper cleaning methods may scratch or damage the surface. Note: Only use soft, anti-static cloths to clean the display. Avoid aggressive cleaning agents and strong friction. The display operates with electrical voltages and currents that, if used improperly, can cause electric shocks, short circuits or fires. Please note the following safety instructions: Attention: Use the product only with the specified voltages. Note: The performance limits of the product can be found in the associated data sheet Note: Improper voltage sources can damage the display or cause dangerous situations. Action: Only use tested and suitable power supplies or batteries to power your circuits. Make sure the voltage source meets the requirements of the display. Caution: Avoid short circuits between the connectors and components of the product Note: Make sure that no conductive objects touch or bridge the circuit board. Use insulated tools and pay attention to the arrangement of connections. Caution: Do not perform any work on the product when it is connected to a power source. Note: Disconnect the product from power before making any circuit changes or connecting or removing components. Note: Look for signs of electrical damage such as smoke, unusual odors, or discoloration. Action: If such signs occur, turn off the power immediately and inspect the circuit thoroughly for errors. The display can generate heat during operation, which could lead to overheating, burns or fire if handled improperly. Please note the following safety instructions: Attention: Some components of the display can heat up during operation or in the event of an error. Measure: After switching off, allow the display to cool down sufficiently before touching the individual components on the back directly. Avoid direct contact with hot components. Caution: Overloading can cause excessive heating of the electronic components. Note: Make sure the power and voltage supply meets the specifications of the display and does not cause overload.

Table of contents

Introduction

A liquid crystal display, also known as an LCD, is a device that uses liquid crystals to display visual characters.

Liquid crystals do not emit any light themselves. LCD screens use a backlight and liquid crystals to block light. When no current flows through the liquid crystals, they are in a chaotic state. Light from the backlight passes effortlessly through liquid crystals. When current flows through liquid crystals, they arrange themselves in a uniform state. This creates a shadow on the screen, which creates objects.

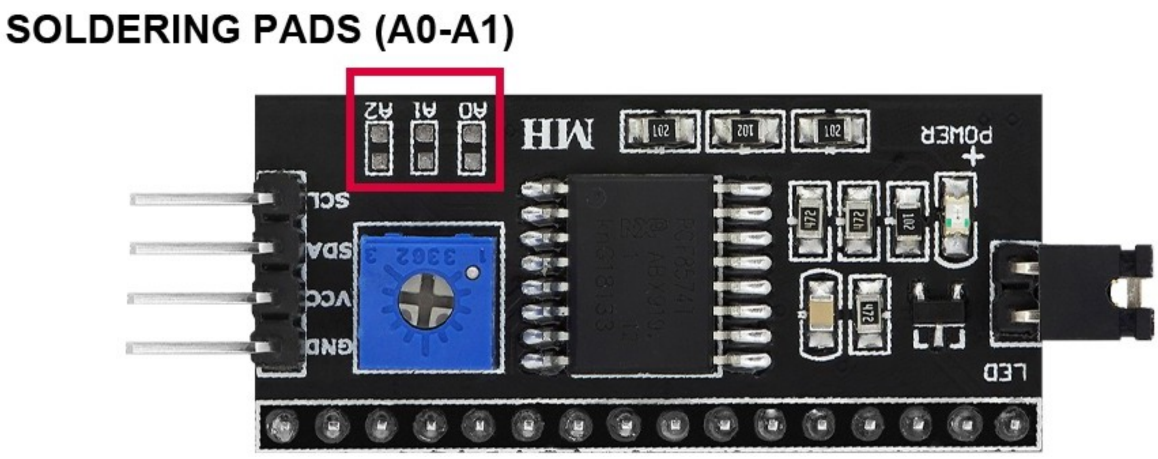
Simply put, LCDs use liquid crystals to transmit or block the light coming from the backlight. This is how pixels are created. They are combined to display any visible 2D objects on the screen. Each pixel area can be switched ON and OFF by supplying electricity via an on-board controller chip.

Each segment of the display with liquid crystals represents a pixel and must be controlled separately. For this reason, a special integrated circuit, a so-called driver chip, is required. The 16x02(20x04) LCD has a driver chip called "HD44780". To control the screen, the microcontroller must communicate with the driver chip. The driver chip uses a type of SPI interface to communicate with a microcontroller.

The I2C adapter

The I2C adapter is a device that simplifies the connection between LCD screens and microcontrollers. It uses the I2C interface to communicate with the microcontroller. Several adapters can be connected to the same I2C interface. The adapter is compatible with LCD screens with integrated HD44780 chips. The I2C adapter is compatible with both 16x02 LCD screens and 20x04 LCD screens offered by AZ-Delivery.

The I2C adapter is supplied with a predefined I2C address, which is *0x27.* However, it can be changed by soldering the pads labelled A0, A1 and A2 on the adapter.



The following table shows how to set a specific I2C address of the adapter:

PADS			I2C ADDRESS
A2	**A1**	**A0**	
C	C	C	0x20
C	C	O	0x21
C	O	C	0x22
C	O	O	0x23
O	C	C	0x24
O	C	C	0x25
O	O	C	0x26
O	O	O	0x27
O - Open			**C - Closed**

Technical Data

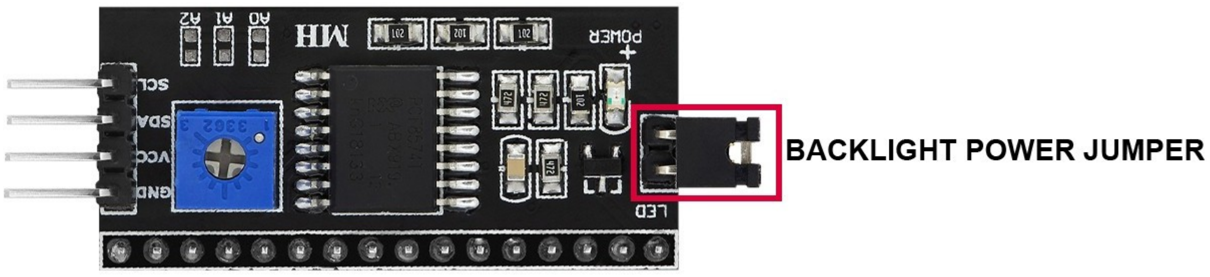
" Operating voltage range:	3.3V to 5V
" Display area:	12 x 56mm
" LCD type: green/blue	STN, positive, transflective,
" Backlight:	ED, white
" Viewing angle:	180°
" Modes:	parallel (8-bit and 4-bit)
" Operating temperature:	-10 °C to 60 °C
" Dimensions:	36x 80 x 12.5mm [1.4 x 3.1 x 0.5in]
" Interface:	I2C/parallel
" I2C address:	0x20 - 0x27
" Adjusting the contrast :	Potentiometer
" Setting the background settings:	Jumper

The voltage for the backlighting is *5V* DC. The current consumption during operation is *1.5mA* for logic functions and *30mA* for the backlight.

The resolution of the screen is *16x02*, which means that characters can be displayed in *2* lines with *16* characters per line. Each character consists of *5x7* pixels.

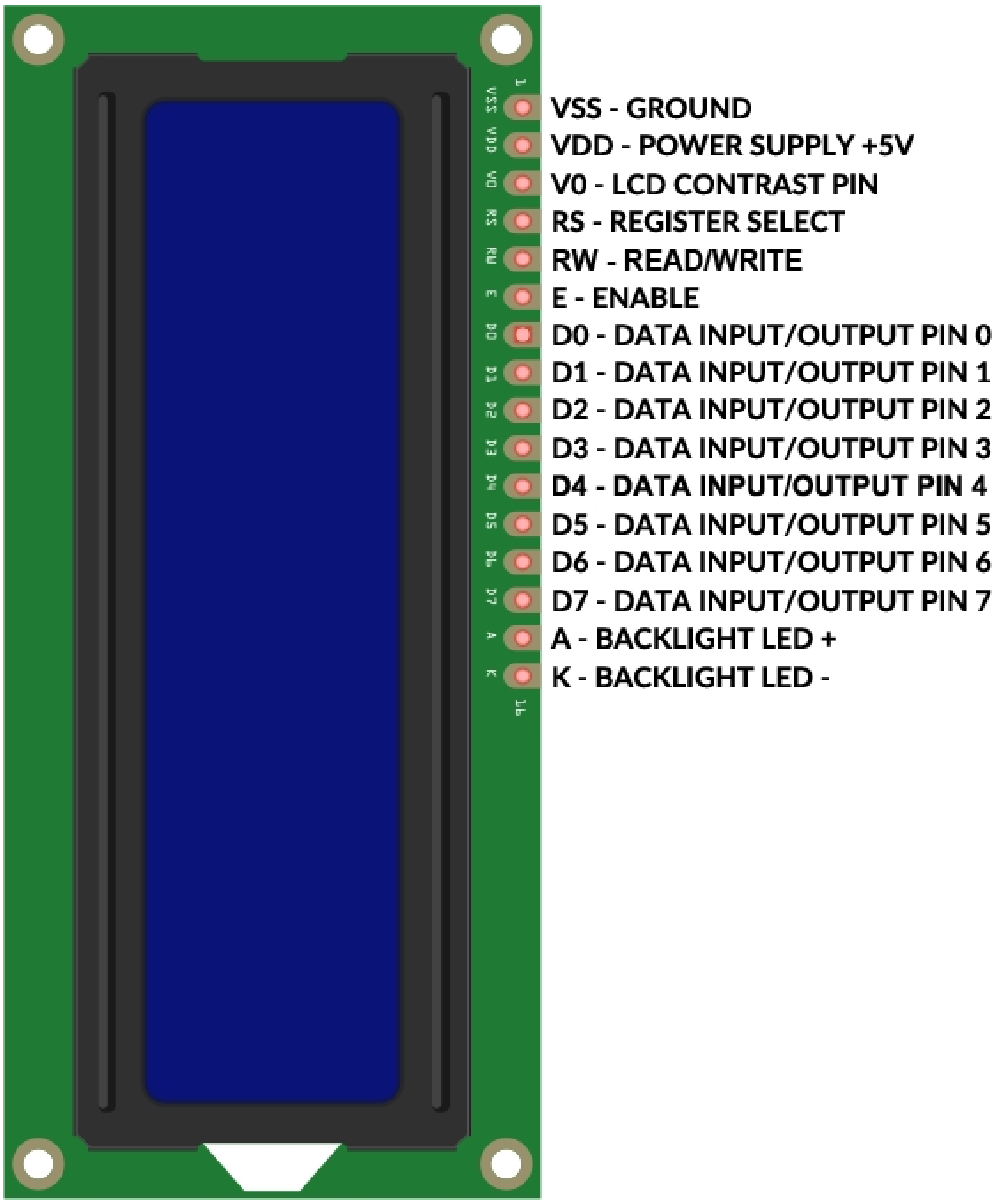
The I2C adapter has a built-in potentiometer to adjust the contrast of the display. A small screwdriver is therefore required for adjustment.

To control the backlight of the LCD screen, the I2C adapter has a power jumper for the backlight. The jumper is used to switch the backlight on/off. When the jumper is connected, the power supply for the backlight is connected. If the jumper is disconnected, the power supply to the backlight is interrupted.



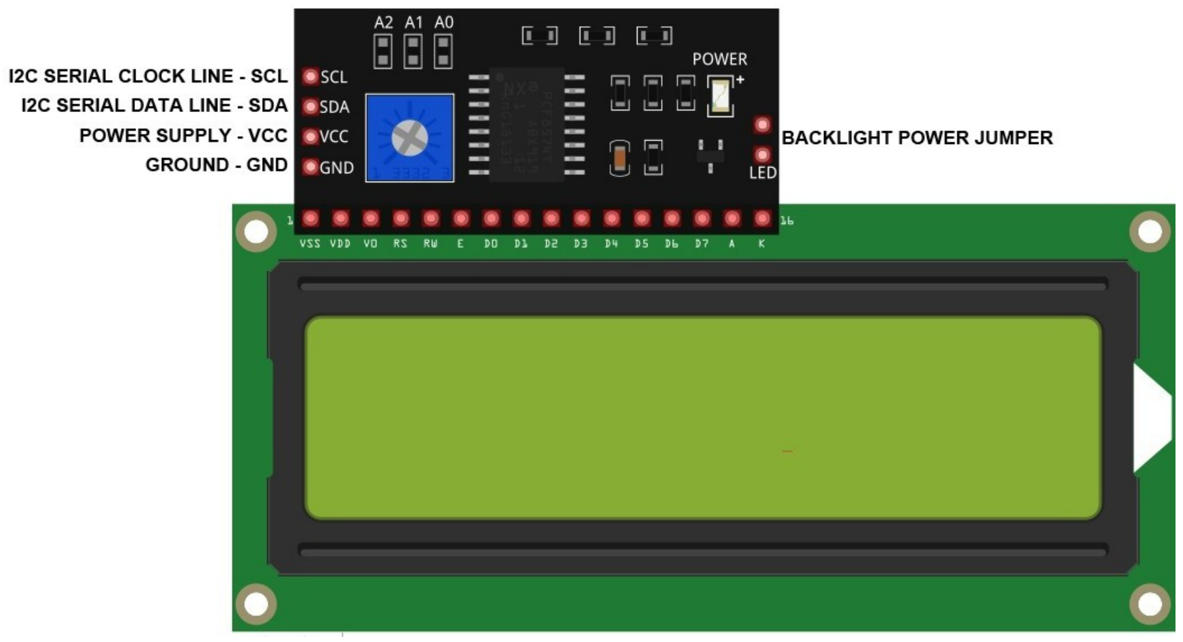
Pin assignment without I2C adapter

The 16x02 LCD has *16* pins. The pin assignment is as follows (also applies to 20x04):


1
VSS
VDD
V0
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
A
K
16
VSS - GROUND
VDD - POWER SUPPLY +5V
V0 - LCD CONTRAST PIN
RS - REGISTER SELECT
RW - READ/WRITE
E - ENABLE
D0 - DATA INPUT/OUTPUT PIN 0
D1 - DATA INPUT/OUTPUT PIN 1
D2 - DATA INPUT/OUTPUT PIN 2
D3 - DATA INPUT/OUTPUT PIN 3
D4 - DATA INPUT/OUTPUT PIN 4
D5 - DATA INPUT/OUTPUT PIN 5
D6 - DATA INPUT/OUTPUT PIN 6
D7 - DATA INPUT/OUTPUT PIN 7
A - BACKLIGHT LED +
K - BACKLIGHT LED -

Pin assignment with I2C adapter

The 16x02 LCD has *16* pins and the I2C adapter has 20. The adapter is connected to the LCD screen as follows (this also applies to the 20x04):

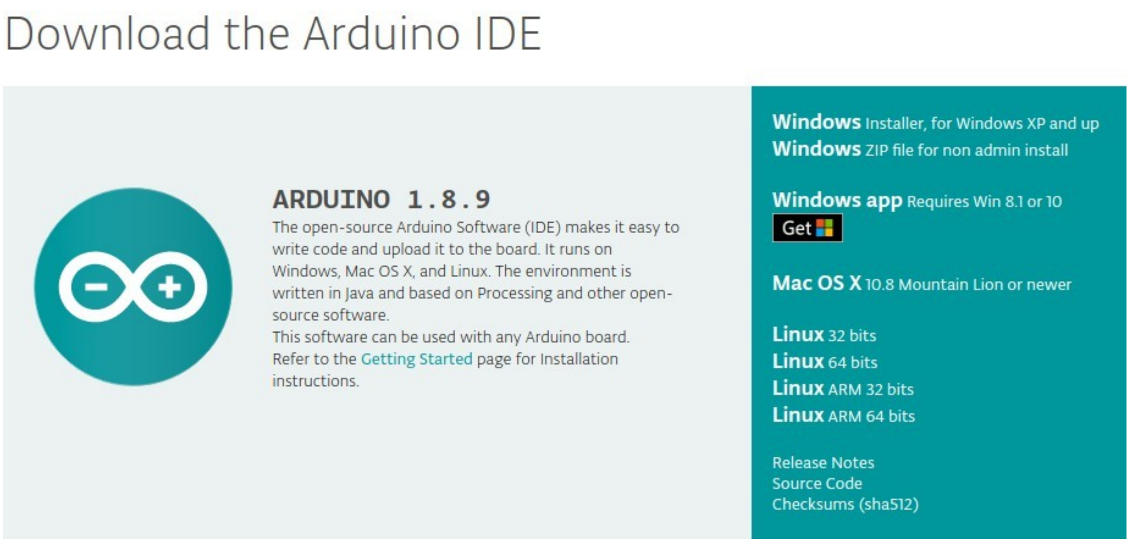

A2 A1 A0
POWER
I2C SERIAL CLOCK LINE - SCL
I2C SERIAL DATA LINE - SDA
POWER SUPPLY - VCC
GROUND - GND
SCL
SDA
VCC
GND
BACKLIGHT POWER JUMPER
LED
1
16
VSS VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K

Note: It is necessary to connect the I2C adapter and the LCD display exactly as shown above. If connected differently, the devices may be damaged.

Note for the Raspberry Pi: The voltage of the TTL logic level of the I/O pins is *5V*. To use the LCD screen and the I2C adapter with the Raspberry Pi, a logic level converter must be used. Otherwise, feeding the signal via the I/O pins of the module to the GPIO pins of the Raspberry Pi may cause damage. You should therefore use the TXS0108E 8ch Logic Level Converter offered by AZ-Delivery.

How to set up the Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of your choice.



Download the Arduino IDE

ARDUINO 1.8.9

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
Get

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Release Notes
Source Code
Checksums (sha512)

For `Windows` users: Double-click click on the downloaded `.`exel *`file`* and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz* that needs to be extracted. Once it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts need to be executed, the first one called *arduino-linux-setup.sh* and the second one called *install.sh.*

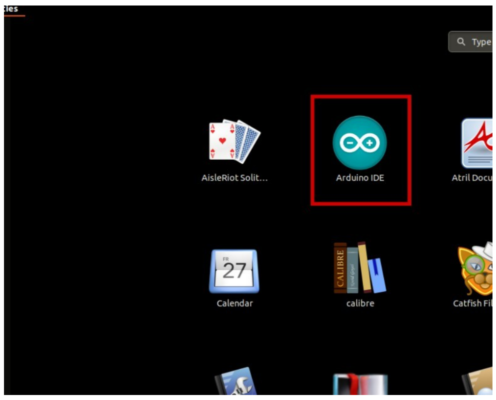
To execute the first script in the terminal, open the terminal in the extracted folder and execute the following command:

```
sh arduino-linux-setup.sh user_name
```

user_name - is the name of a superuser in the Linux operating system. A password for the superuser must be entered when starting the command. Wait a few minutes for the script to complete.

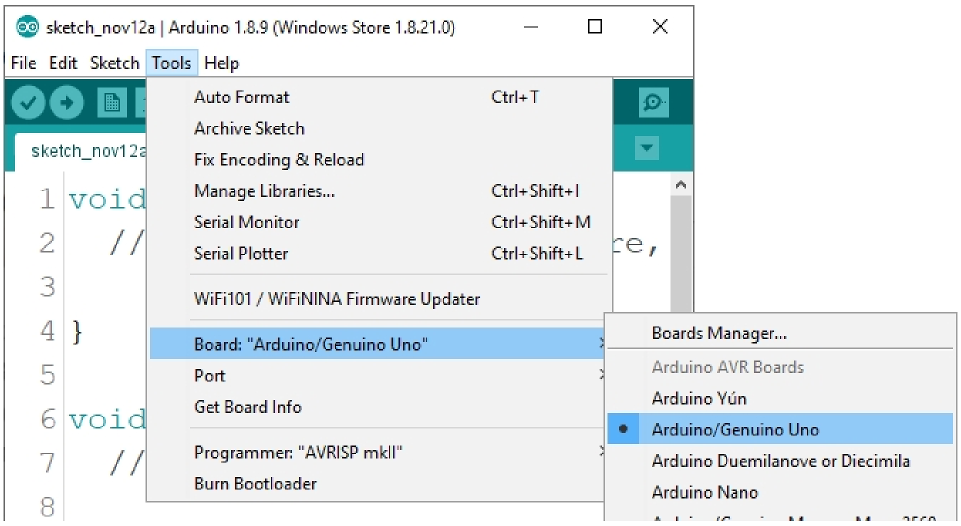
The second script with the name *install.sh script* must be used after the installation of the first script. Execute the following command in the terminal (extracted directory): **sh install.sh**

After installing these scripts, go to *All Apps,* where the *Arduino IDE* is installed.



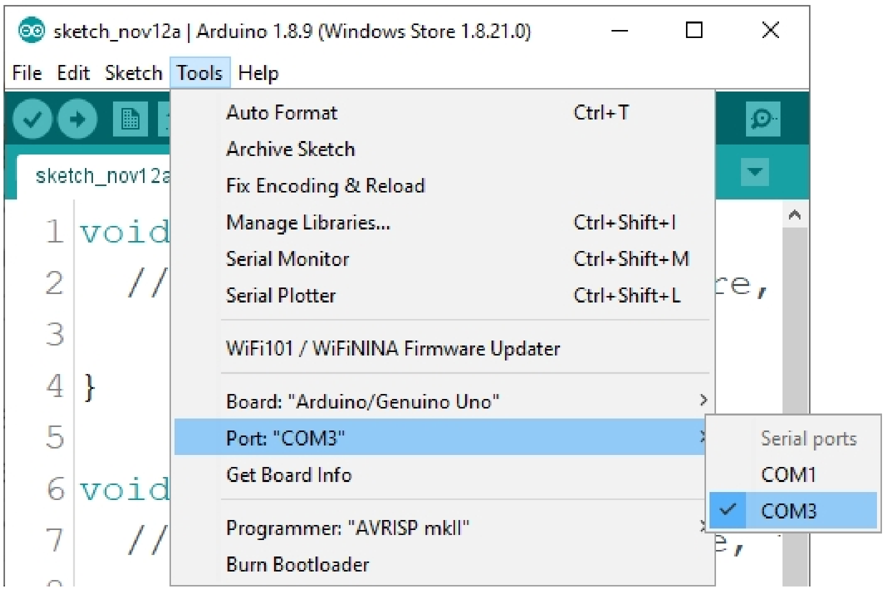
Almost all operating systems come with a pre-installed text editor (e.g. `Windows` with `Notepad`, `Linux` Ubuntu with `Gedit`, `Linux Raspbian` with `Leafpad` etc.). All of these text editors are perfectly fine for the purpose of the eBook.

First check whether your PC can recognise an Arduino board. Open the freshly installed Arduino IDE and go to :
`Tools > Board > {your board name here}`
`{your board name here}` should be the `Arduino/Genuino Uno, as can be seen in` the following picture:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
sketch_nov12a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Yún
Arduino/Genuino Uno
Arduino Duemilanove or Diecimila
Arduino Nano

The port to which the microcontroller board is connected must be selected. Go to: `Tools > Port > {port name goes here}` and if the microcontroller board is connected to the USB port, the port name can be seen in the drop-down menu in the previous picture.

When the Arduino IDE under Windows is used is used, are the port names are as follows:


sketch_nov12a | Arduino 1.8.9 (Windows Store 1.8.21.0)
File
Edit
Sketch
Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM3

For *Linux* users, for example, the port name is */dev/ttyUSBx*, where x is an integer between *0* and *9.*

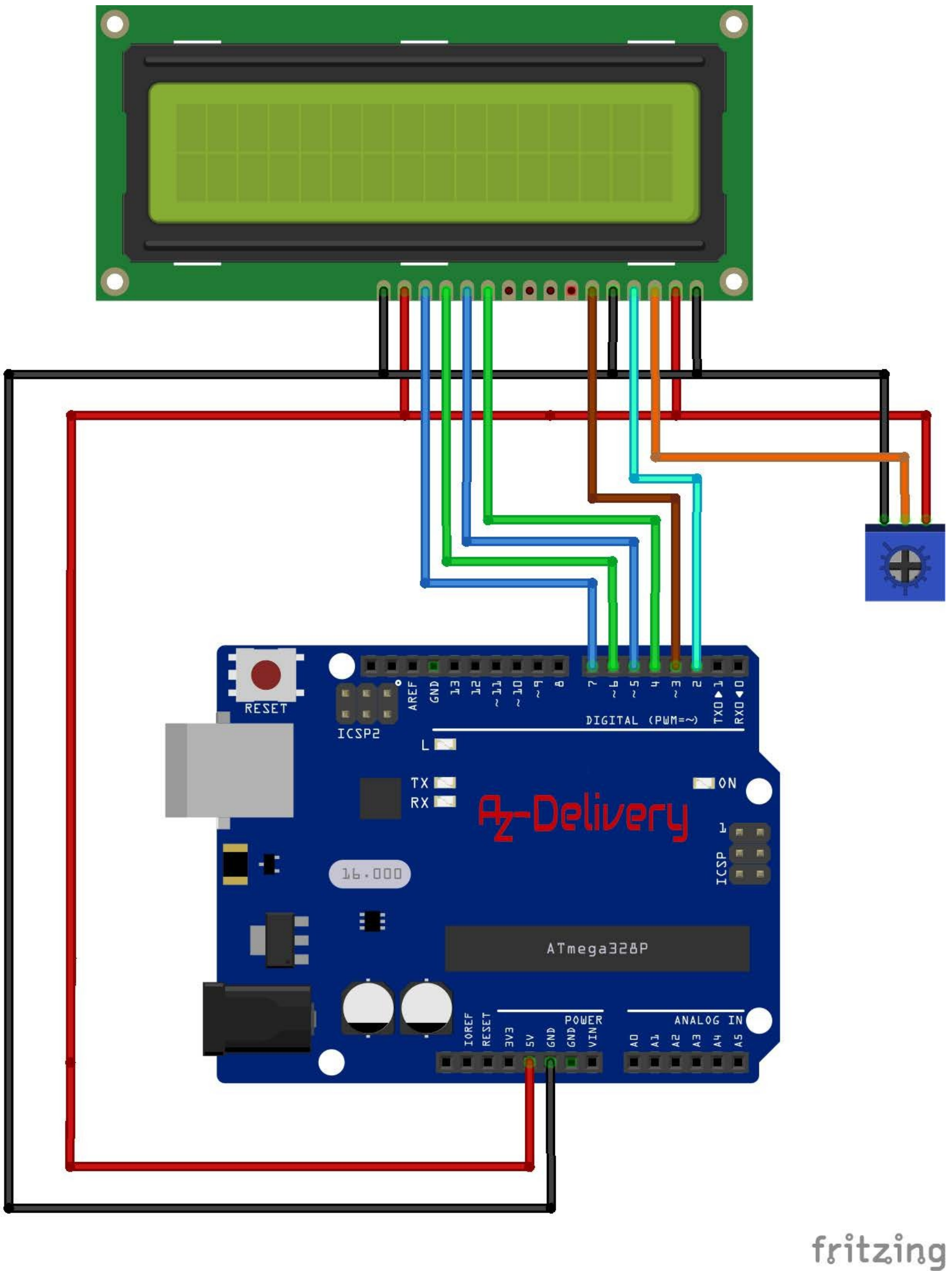
How to set up the Raspberry Pi and Python

The operating system must first be installed for the Raspberry Pi, then everything must be set up so that it can be used in headless mode. Headless mode allows a remote connection to the Raspberry Pi without the need for a PC screen, mouse or keyboard. The only things used in this mode are the Raspberry Pi itself, the power supply and the internet connection. This is all explained in detail in the free eBook:
Raspberry Pi Quick Startup Guide

The operating system Raspbian is with pre-installed *Python* delivered.

Connecting the module to the microcontroller

Connect connect the screen with the microcontroller as shown below (also applies to 16x02):



LCD pin	MC Pin	Wire colour
VSS	GND	**Black wire**
VDD	5V	**Red wire**
V0	Potentio meter	**Orange wire**
RS	D2	**Ochre wire**
RW	GND	**Black wire**
E	D3	**Ochre wire**
D4	D4	**Green wire**
D5	D5	**Blue wire**
D6	D6	**Green wire**
D7	D7	**Blue wire**
K	GND	**Black wire**
A	5V	**Red wire**

Sketch example

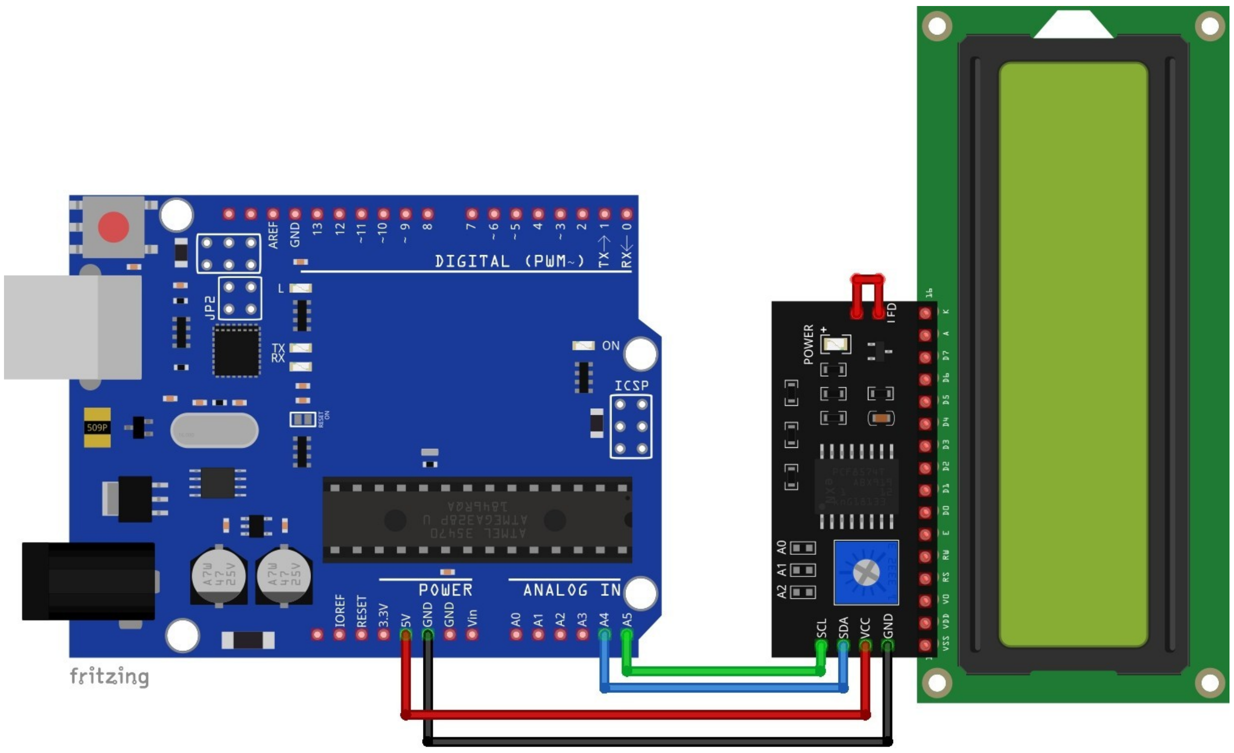
The following sketch example is a modified sketch from the Arduino IDE:

File > Examples > LiquidCrystal > HelloWorld

```
#include <LiquidCrystal.h>
const uint8_t rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {
  lcd.begin(16, 2); // change to "20, 4" if 20x04 is used
  lcd.clear();

}
void loop() {
  lcd.setCursor(0, 0);
  lcd.print("AZ-Delivery");
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
}
```

Connection of the module to the microcontroller with I2C adapter

Connect connect the 16x02 screen and the I2C adapter with the microcontroller as shown below (also applies to 20x04):



I2C adapter pin	MC Pin	Wire colour
SCL	A5	**Green wire**
SDA	A4	**Blue wire**
VCC	5V	**Red wire**
GND	GND	**Black wire**

Library for the Arduino IDE

To use the module with the Arduino IDE, it is recommended to download an external library for it.the library used in this eBook is called *LiquidCrystal_I2C.* To download it, click on this link and download the .zip file.

To integrate the library, go to the Arduino IDE and go to :

Sketch > Include Library > Add .ZIP Library

and when a new window opens, search for and select the downloaded .zip file.

Sketch example

The following sketch example is a modified sketch from the Arduino IDE: File > Examples > LiquidCrystal > HelloWorld

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);
              //Change to "20, 4" if 20x04 is used void
setup() {
  lcd.init();
  lcd.backlight();
  delay(250);
  lcd.noBacklight();
  delay(1000);
  lcd.backlight();
  delay(1000);
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("AZ-Delivery");
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
  delay(100);
}
```

The sketch begins with the inclusion of libraries with the names *`Wire`* and *`LiquidCrystal_I2C`*.

An object with the name lcd is then created. The object represents the display itself, and the following line of code is used to create this object:

```
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 16, 2);
```

Where 0x27 is the I2C address of the I2C adapter. 16 is the number of characters per line and 2 is the number of lines.

In the *`setup()`* function, the *`lcd`* object is initialised with the following line of code: `lcd.init();`

At the end of the *`setup()`* function, the backlight is tested by switching it OFF and ON with a delay of 1000ms (1s).

In the *`loop()`* function, two predefined functions from the *`LiquidCrystal_I2C`* Library is used.

The first function is called `setCursor()`. The function has two arguments and does not return a value. The values of the arguments are integers. The first number represents the *Y position of the cursor*, with values in the range *0* to *1*, where *0* represents the first line and *1* the second line of the screen. The second argument represents the *X position of the cursor*, with values ranging from *0* to *15*, where *0* represents the first column and *15* the last column of the screen.

The function must be used before the `print() function.` To show the `print()` function where the text should be displayed. If you do not use the `setCusrsor()` function, the `print() function displays` the text at position *(0, 0)*.

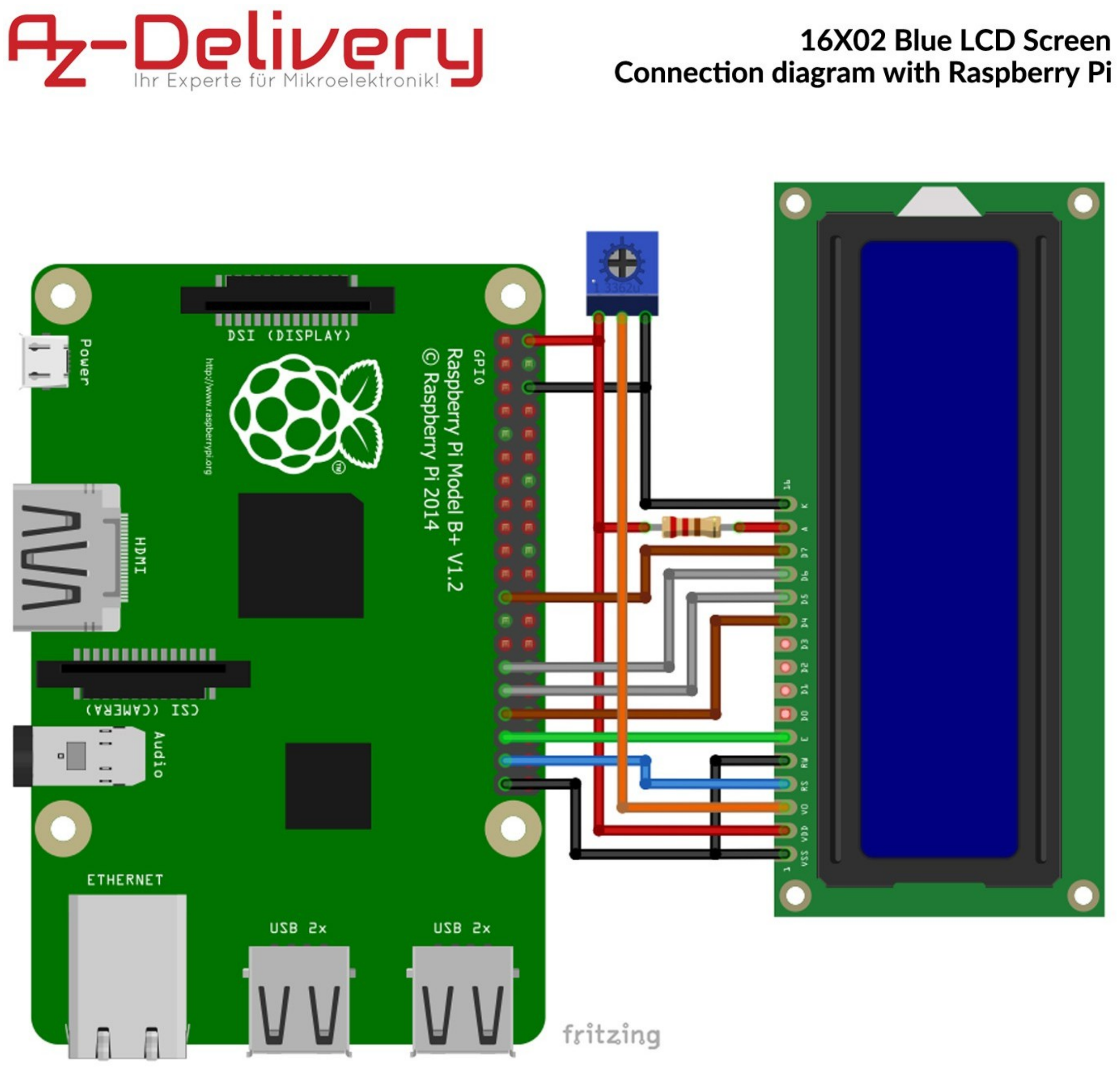
The `print()` function has one argument and does not return a value. The argument represents the text, a string value, that is displayed on the screen.

In the `loop()` function, the cursor is first placed on the first line and then the `print()` function `displays` the message `AZ-Delivery`. Then the cursor is placed on the second line and the number of seconds since the last switch-on or the last reset of the microcontroller is displayed.

NOTE: If your display does not show anything, you must turn the potentiometer on the I2C adapter.

Connecting the screen to the Raspberry Pi

Connect the module to the Raspberry Pi as shown below (also applies to 20x04):



Screen Pin	Raspberry Pi Pin	Physical pin	Wire colour
VSS	GND	6	**Black wire**
VDD	5V	2	**Red wire**
RS	GPIO26	37	**Blue wire**
RW	GND	39	**Black wire**
E	GPIO19	35	**Green wire**
D4	GPIO13	33	**Brown wire**
D5	GPIO6	31	**Grey wire**
D6	GPIO5	29	**Grey wire**
D7	GPIO11	23	**Brown wire**
K	GND	6	**Red wire**
A	5V, via 220Ω resistor	2	**Orange wire**
V0	potentiometer centre pin		

	Potentiometer		
GND	Right pin	6	**Black wire**
5V	Left pin	2	**Red wire**

Python Script

Two scripts are created, one for all functions and the other to use these functions. Below you will find the code for the first script:

```
import RPi.GPIO as GPIO
import time

LCD_RS = 26
LCD_E  = 19
LCD_D4 = 13
LCD_D5 = 6
LCD_D6 = 5
LCD_D7 = 11

# Define some device constants
LCD_WIDTH = 16#    Change maximum characters per line to "20" at
20x04
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
```

```
  # one tab
def lcd_init(RS, E, D4, D5, D6, D7):
  GPIO.setwarnings(False)
  GPIO.setmode(GPIO.BCM)
  global LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7
  LCD_RS = RS
  LCD_E = E
  LCD_D4 = D4
  LCD_D5 = D5
  LCD_D6 = D6
  LCD_D7 = D7
  GPIO.setup(LCD_E, GPIO.OUT) # E
  GPIO.setup(LCD_RS, GPIO.OUT) # RS
  GPIO.setup(LCD_D4, GPIO.OUT) # DB4
  GPIO.setup(LCD_D5, GPIO.OUT) # DB5
  GPIO.setup(LCD_D6, GPIO.OUT) # DB6
  GPIO.setup(LCD_D7, GPIO.OUT) # DB7
  # Initialise display
  lcd_byte(0x33, LCD_CMD) # 110011 Initialise
  # 110010 Initialise
  lcd_byte(0x32, LCD_CMD)
  # 000110 Cursor move direction
  lcd_byte(0x06, LCD_CMD)
  # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x0C, LCD_CMD)
  # 101000 Data length, number of lines, font size
  lcd_byte(0x28, LCD_CMD)
  # 000001 Clear display
  lcd_byte(0x01, LCD_CMD)
  time.sleep(E_DELAY)
  # one tab
def lcd_byte(bits, mode):
  GPIO.output(LCD_RS, mode) # RS
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits & 0x10 == 0x10:
    GPIO.output(LCD_D4, True)
  if bits & 0x20 == 0x20:
    GPIO.output(LCD_D5, True)
  if bits & 0x40 == 0x40:
    GPIO.output(LCD_D6, True)
  if bits & 0x80 == 0x80:
    GPIO.output(LCD_D7, True)
  lcd_toggle_enable() # Toggle 'Enable' pin
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits & 0x01 == 0x01:
    GPIO.output(LCD_D4, True)
  if bits & 0x02 == 0x02:
    GPIO.output(LCD_D5, True)
  if bits & 0x04 == 0x04:
    GPIO.output(LCD_D6, True)
  if bits & 0x08 == 0x08:
    GPIO.output(LCD_D7, True)
  # Toggle 'Enable' pin
  lcd_toggle_enable()
  # one tab
def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)

def lcd_string(message, line):
  # Send string to display
  LCD_LINE_1 = 0x80
  LCD_LINE_2 = 0xC0
  message = message.ljust(LCD_WIDTH, " ")
  if line == 0:
    lcd_byte(LCD_LINE_1, LCD_CMD)
  elif line == 1:
    lcd_byte(LCD_LINE_2, LCD_CMD)
  else:
    print('This lcd has two lines, line 0 and line 1!')
  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]), LCD_CHR)

def lcd_clear():
  lcd_byte(0x01, LCD_CMD)
```

Save the script under the name `lcd16x02.py`.

The code in the script is a modified code from the script under *link*:

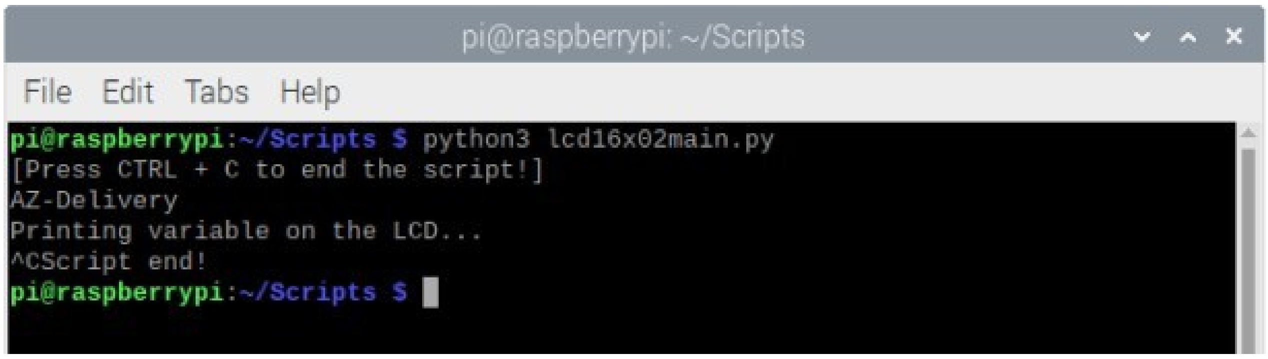
Below you will find the code for the main script:

```
import lcd16x02
from time import sleep
LCD_RS  = 26
LCD_E   = 19
LCD_D4  = 13
LCD_D5  = 6
LCD_D6  = 5
LCD_D7  = 11
# Initialise display
lcd16x02.lcd_init(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7)
i = 0
print('[Press CTRL + C to end the script!]')
try:
  lcd16x02.lcd_string('AZ-Delivery', 0)
  print('AZ-Delivery')
  print('Printing variable on the LCD...')
  while True:
    lcd16x02.lcd_string('{}'.format(i), 1)
    i+=1
    sleep(0.001) # 1 millisecond delay

except KeyboardInterrupt:
  print('Script end!')

finally:
  lcd16x02.lcd_clear()
```

Save the script under the name `lcd16x02main.py` in the same directory as the previous script. To execute the script, open the terminal in the directory where the script is saved and execute the following command:

`python3 lcd16x02main.py`

The output should look like this:



```
pi@raspberrypi:~/Scripts $ python3 lcd16x02main.py
[Press CTRL + C to end the script!]
AZ-Delivery
Printing variable on the LCD...
^CScript end!
pi@raspberrypi:~/Scripts $ 
```

To stop the script, press 'Ctrl + C' on the keyboard.

The first script is used to create all the functions to control the LCD screen, which is not covered in this eBook. Only the main function of the script is explained.

The main script starts with the import of the first script and the import of the *sleep function* from the *time* library.

Six variables are then defined, which represent the pins of the screen that are connected to the pins of the Raspberry Pi.

Next, the screen is initialised with the following line of code:

```
lcd16x02.lcd_init(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7)
```

The variable "*i*" is then created and initialised with the value zero. This is used to show changing data on the display.

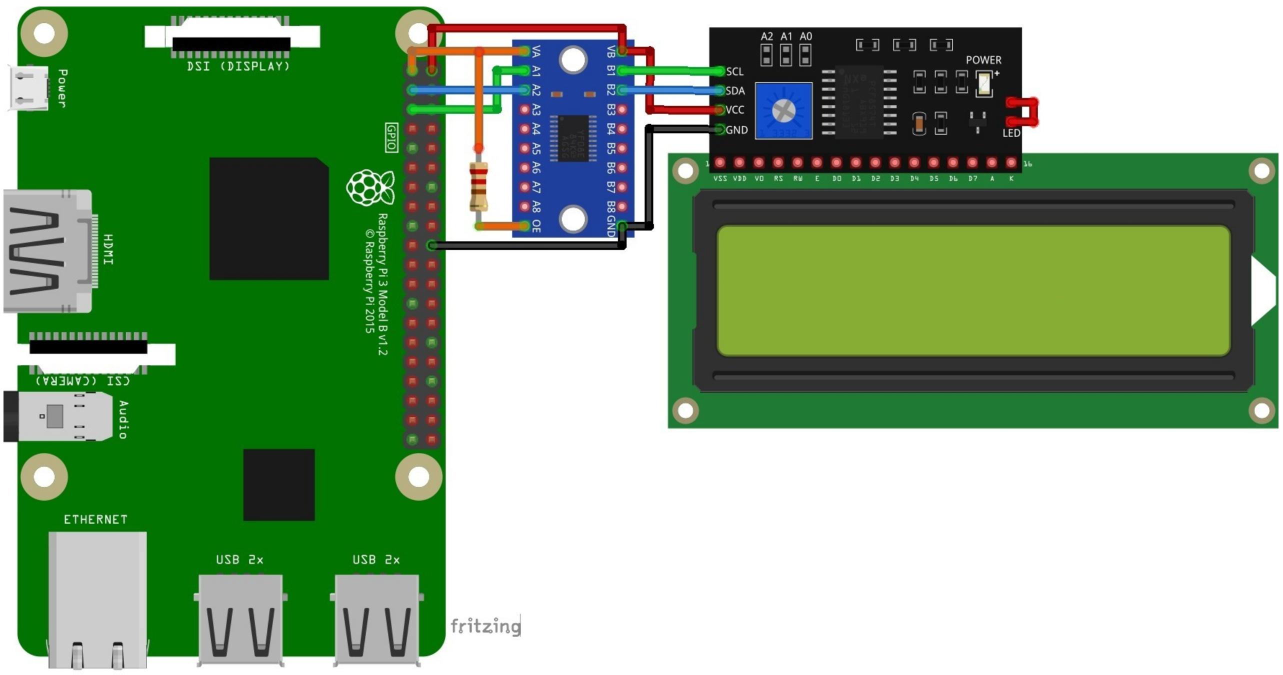
A *try-except-finally* code block is then created. In the *try* block, the message *AZ-Delivery* is first displayed in the first line of the screen, and then an indefinite loop block (*while True:*) is created. In it, the variable *i is* timed on the second line of the display and the value of the variable *i is* increased by *1. There is* a pause of one millisecond between each repetition of the indefinite loop block (*sleep(0.001)*).

The `except` code block is executed with `Ctrl+C.` This is called `KeyboardInterrupt` When this code block has been executed, the message `Script end!` is displayed in the terminal.

The `finally` code block is executed after the script. When the `finally code` block has been executed, the `lcd_clear()` function is called, which clears the data buffer of the screen.

Connecting the screen to the Raspberry Pi with I2C adapter

Connect the 16x02 screen and the I2C adapter to the Raspberry Pi as shown below (also applies to 20x04):



The logic level converter must be used here, as the I2C adapter only works in the 5V range. The logic level converter used in this eBook is called *TXS0108E Logic Level* Converter.

Connect the I2C adapter to the LCD screen as shown in the connection diagram. Make sure that the integrated jumper for the backlight is connected (**red wire**, the right-hand side of the I2C adapter on the connection diagram).

I2C adapter pin	LLC Pin	Wire colour
SCL	B1	**Green wire**
SDA	B2	**Blue wire**
VCC	VB	**Red wire**
GND	GND	**Black wire**

LLC Pin	Raspberry Pi Pin	Physical pin	Wire colour
VA	3.3V	1	**Orange wire**
A1	GPIO3	5	**Green wire**
A2	GPIO2	3	**Blue wire**
OE	3.3V (via resistor)	1	**Orange wire**
GND	GND	20	**Black wire**
VB	5V	2	**Red wire**

Libraries and tools for Python

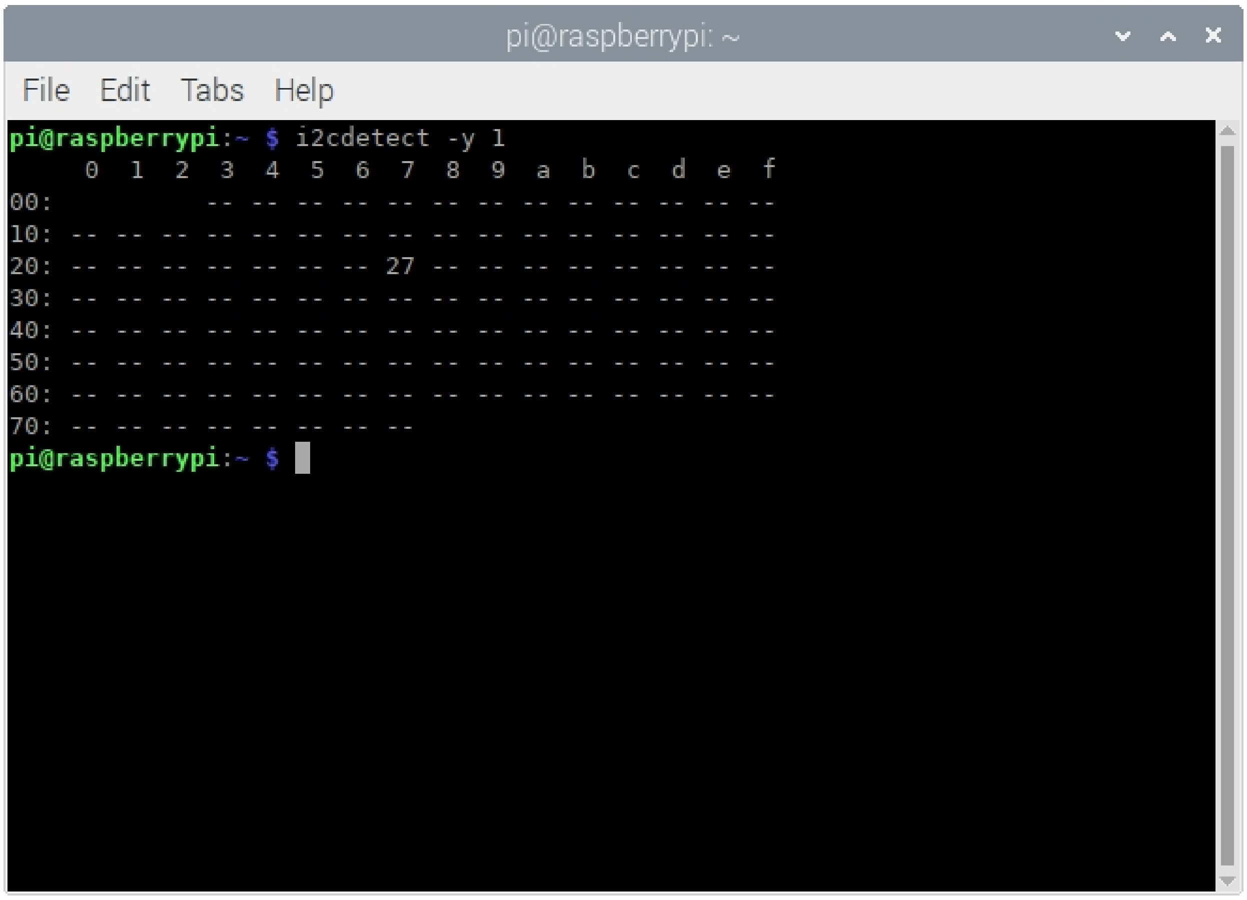
To use the screen with the Raspberry Pi, it is recommended to download an external library.

```
git clone https://github.com/the-raspberry-pi-guy/lcd.git
cd lcd
sudo ./install.sh
```

Then restart the Pi.

To the I2C address of the I2C adapter adapter to determine it, run execute the following command:

`i2cdetec -y 1`



```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

where 0x27 is the I2C address of the I2C adapter.

Python Script

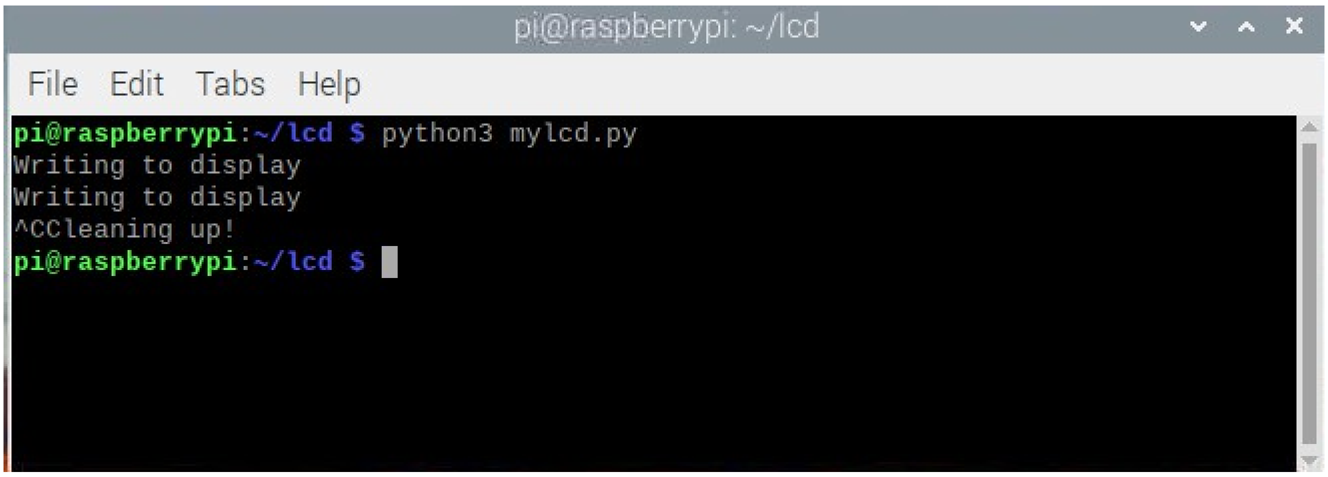
Below you will find the code for the main script:

```
import drivers
from time import sleep

display = drivers.Lcd(0x27)

try:
    while True:
        print("look at display")
        display.lcd_display_string("Demo line 1", 1)
        display.lcd_display_string("Demo line 2", 2)
        display.lcd_display_string("Demo line 3", 3)
        display.lcd_display_string("Demo line 4", 4)
        sleep(2)      # Give time for the message to be read
        display.lcd_clear() # Clear the display of any data
        sleep(2)      # Give time for the message to be read
except KeyboardInterrupt:
    # exit on KeyboardInterrupt (when you press ctrl+c)
    print("exit")
    display.lcd_clear()
```

Save the script under the name *mylcd.py*. Execute the new script with the following command:

`python3 mylcd.py`

The output should look like this:



```
pi@raspberrypi:~/lcd $ python3 mylcd.py
Writing to display
Writing to display
^CCleaning up!
pi@raspberrypi:~/lcd $
```

To stop the script, press 'Ctrl + C' on the keyboard.

Now it's your turn! Develop your own projects and smart home installations. We will show you how to do this in an uncomplicated and understandable way on our blog. There we offer you sample scripts and tutorials with interesting small projects to get you started quickly in the world of microelectronics. The Internet also offers you countless opportunities to learn more about microelectronics.

If you are looking for other high-quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right place for you. We offer you numerous application examples, detailed installation instructions, e-books, libraries and, of course, the support of our technical experts.

https://az-delivery.de

Have fun!

Imprint

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