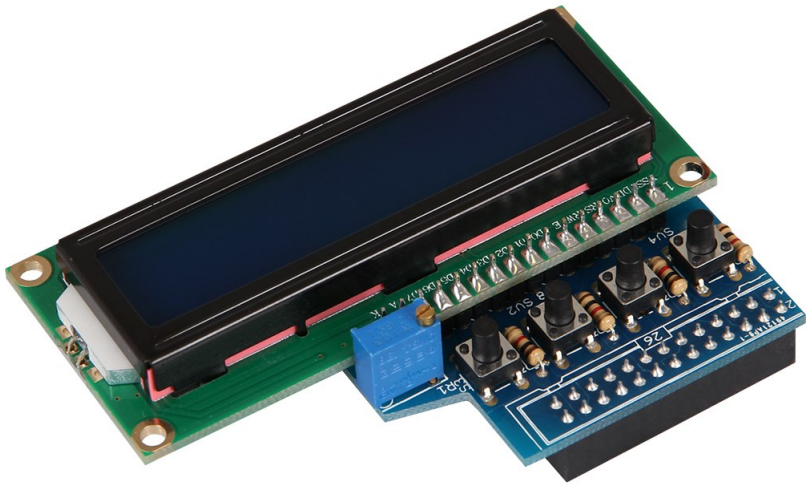
LCD Display 16x2 with Buttons



Index

1. Connecting the Display

2. Installing the System

3. Usage & Example-Code

4. Support

Dear customer

thank you for purchasing our product.
Please find our instructions below.

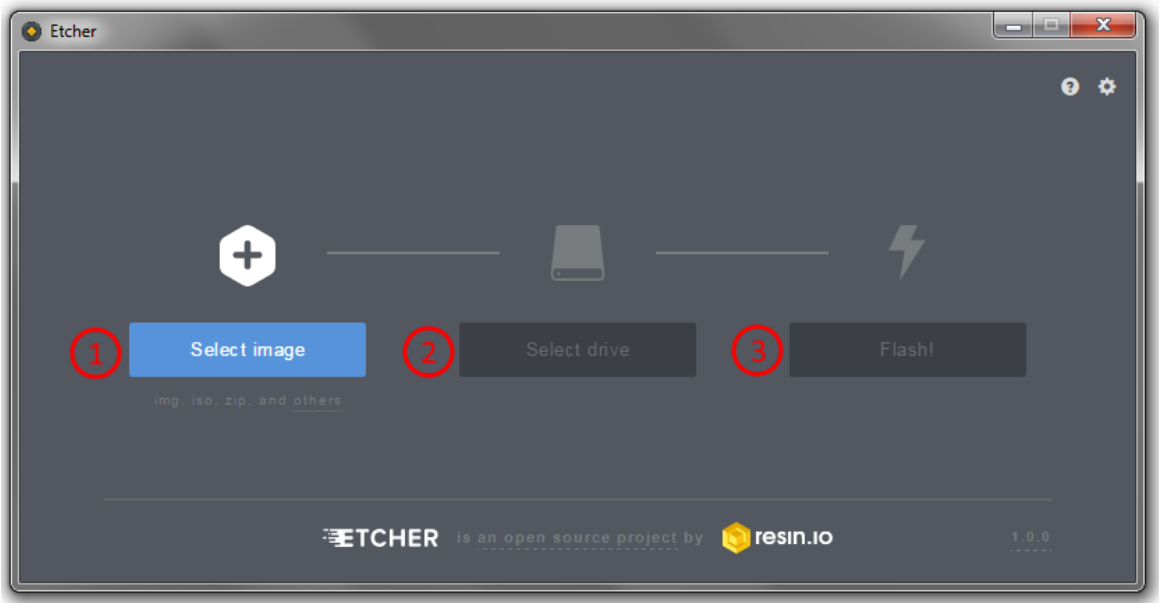
1. Connecting the Display

Plug the display onto the pin header of your Raspberry Pi so that the display is placed over your Raspberry Pi.
As soon as you power your Raspberry Pi, the displays backlight should start to light.

2. Installing the System

You can skip this step, if you are already running the latest Raspbian software on your Raspberry Pi, and continue with the next step.
If not, please follow the instructions.

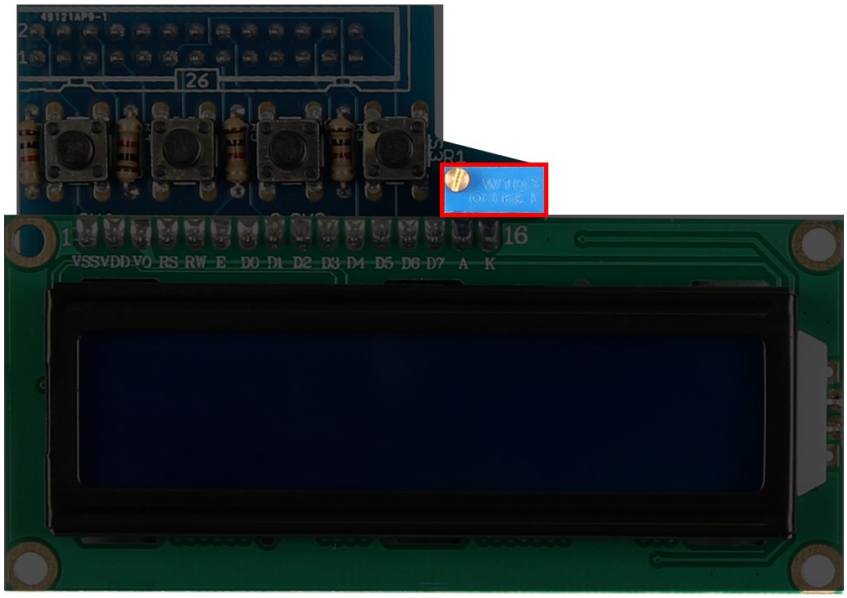
Install the latest Raspbian-System-Image to your SD-Card.
You can download the image here.
Transfer the image with a suitable program (e.g. Etcher).
You can insert the SD-Card to your Raspberry Pi and start the system when the transfer progress is complete.



3. Usage & Example-Code

Your display is ready to use.
No further installations are required.
It may appear that you manually need to adjust the contrast before you are able to read anything on the display.

Adjust the contrast by rotating the screw with a small screw-driver until the contrast is fine.



To use the display, you can either download the example-code or create a new file and insert the code.

To create a new file, enter the following command:

```
sudo nano lcd16x2.py
```

```
import time
import RPi.GPIO as GPIO

# PIN-Configuration
LCD_RS = 7 #GPIO7 = Pi pin 26
LCD_E = 8 #GPIO8 = Pi pin 24
LCD_D4 = 17 #GPIO17 = Pi pin 11
LCD_D5 = 18 #GPIO18 = Pi pin 12
LCD_D6 = 27 #GPIO21 = Pi pin 13
LCD_D7 = 22 #GPIO22 = Pi pin 15
OUTPUTS = [LCD_RS,LCD_E,LCD_D4,LCD_D5,LCD_D6,LCD_D7]

#Button-PINs
SW1 = 4 #GPIO4 = Pi pin 7
SW2 = 23 #GPIO16 = Pi pin 16
SW3 = 10 #GPIO10 = Pi pin 19
SW4 = 9 #GPIO9 = Pi pin 21
INPUTS = [SW1,SW2,SW3,SW4]
#HD44780 Controller Commands
CLEARDISPLAY = 0x01
SETCURSOR = 0x80

#Line Addresses. (Pick one. Comment out whichever doesn't apply)
#LINE = [0x00,0x40,0x14,0x54] #for 20x4 display
LINE = [0x00,0x40] #for 16x2 display
#################################################################################

def InitIO():
 #Sets GPIO pins to input & output, as required by LCD board
 GPIO.setmode(GPIO.BCM)
 GPIO.setwarnings(False)
 for lcdLine in OUTPUTS:
     GPIO.setup(lcdLine, GPIO.OUT)
 for switch in INPUTS:
     GPIO.setup(switch, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def CheckSwitches():
 #Check status of all four switches on the LCD board
 val1 = not GPIO.input(SW1)
 val2 = not GPIO.input(SW2)
 val3 = not GPIO.input(SW3)
 val4 = not GPIO.input(SW4)
 return (val4,val1,val2,val3)
```

```
def PulseEnableLine():
 #Pulse the LCD Enable line; used for clocking in data
 mSec = 0.0005 #use half-millisecond delay
 time.sleep(mSec) #give time for inputs to settle
 GPIO.output(LCD_E, GPIO.HIGH) #pulse E high
 time.sleep(mSec)
 GPIO.output(LCD_E, GPIO.LOW) #return E low
 time.sleep(mSec) #wait before doing anything else

def SendNibble(data):
 #sends upper 4 bits of data byte to LCD data pins D4-D7
 GPIO.output(LCD_D4, bool(data & 0x10))
 GPIO.output(LCD_D5, bool(data & 0x20))
 GPIO.output(LCD_D6, bool(data & 0x40))
 GPIO.output(LCD_D7, bool(data & 0x80))

def SendByte(data,charMode=False):
 #send one byte to LCD controller
 GPIO.output(LCD_RS,charMode) #set mode: command vs. char
 SendNibble(data) #send upper bits first
 PulseEnableLine() #pulse the enable line
 data = (data & 0x0F)<< 4 #shift 4 bits to left
 SendNibble(data) #send lower bits now
 PulseEnableLine() #pulse the enable line

def InitLCD():
 #initialize the LCD controller & clear display
 SendByte(0x33) #initialize
 SendByte(0x32) #set to 4-bit mode
 SendByte(0x28) #2 line, 5x7 matrix
 SendByte(0x0C) #turn cursor off (0x0E to enable)
 SendByte(0x06) #shift cursor right
 SendByte(CLEARDISPLAY) #remove any stray characters on display
##############################################################################

def SendChar(ch):
 SendByte(ord(ch),True)

def ShowMessage(string):
 #Send string of characters to display at current cursor position
 for character in string:
     SendChar(character)
```

```
def GotoLine(row):
 #Moves cursor to the given row
 #Expects row values 0-1 for 16x2 display; 0-3 for 20x4 display
 addr = LINE[row]
 SendByte(SETCURSOR+addr)

################################################################################
# Main Program
print "LCD program starting. Press CTRL+C to stop."
InitIO()
InitLCD()
ShowMessage('Press a button!')
while (True):
 GotoLine(1)
 switchValues = CheckSwitches()
 decimalResult = " %d %d %d %d" % switchValues
 ShowMessage(decimalResult)
# time.sleep(0.2)
```

You can save the file by pressing **CTRL+O** and leave the editor by pressing **CTRL+X**.
You can start the program with the following command:

```
sudo python lcd16x2.py
```

4. Support

We also support you after your purchase.
If there are any questions left or if you encounter any problems, please feel free to contact us by mail, phone or by our ticket-supportsystem on our website.

E-Mail: service@joy-it.net

Ticket-System: http://support.joy-it.net

Phone: +49 (0)2845 98469 – 66 (11- 18 Uhr)

Please visit our website for more informations:

www.joy-it.net