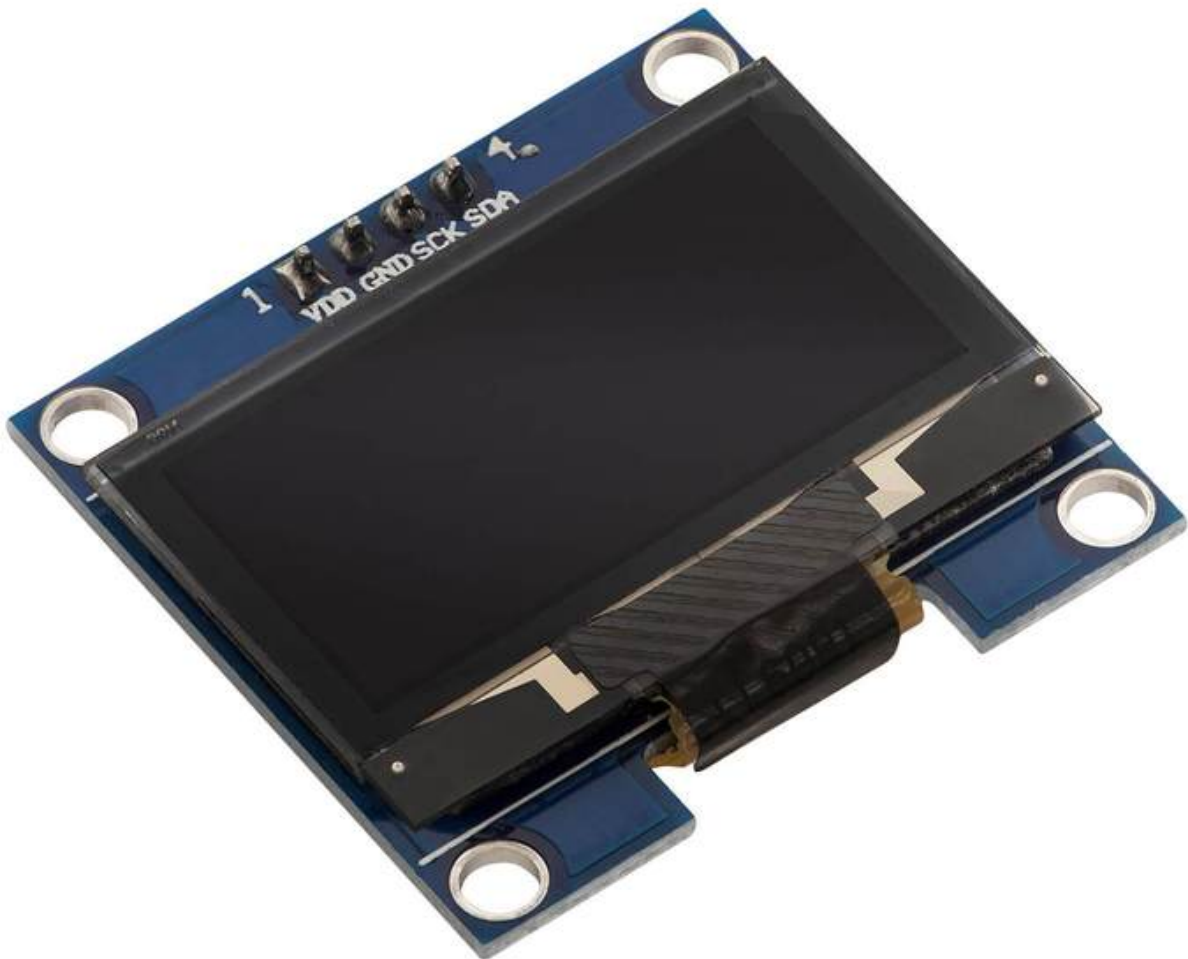


AZ-Delivery

Benvenuto!

Grazie per aver acquistato il nostro *Schermo OLED I2C 1,3 pollici AZ-Delivery*. Nelle pagine seguenti, ti illustreremo come utilizzare e configurare questo pratico dispositivo.

Buon divertimento!





Indice dei Contenuti

Introduzione.....	3
Specifiche:.....	4
Come configurare l'Arduino IDE.....	5
Come configurare il Raspberry Pi e il Python.....	9
La piedinatura.....	10
Collegamento dello schermo con Atmega328P Board.....	11
Libreria per Arduino IDE.....	12
Esempio di sketch.....	13
Collegamento dello schermo con Raspberry Pi.....	25
Abilitazione dell'interfaccia I2C.....	26
Librerie e strumenti per Python.....	27
Script Python.....	29



Introduzione

OLED sta per Organic Light Emitting Diodes (diodi organici ad emissione di luce). Gli schermi OLED sono schiere di LED impilati insieme in una matrice. Lo schermo OLED da 1,3 pollici ha 128x64 pixel (LED). Per controllare questi LED abbiamo bisogno di un circuito pilota o di un chip. Lo schermo ha un chip del driver chiamato SH1106. Il chip del driver ha un'interfaccia I2C per la comunicazione con il microcontrollore principale.

Lo schermo OLED e il chip del driver SH1106 funzionano nella gamma 3.3V. Ma c'è un regolatore di tensione a bordo da 3,3V, il che significa che questi schermi possono funzionare nel campo dei 5V.

Le prestazioni di questi schermi sono molto migliori dei tradizionali LCD. La semplice comunicazione I2C e il basso consumo energetico li rendono più adatti ad una varietà di applicazioni.



Specifiche

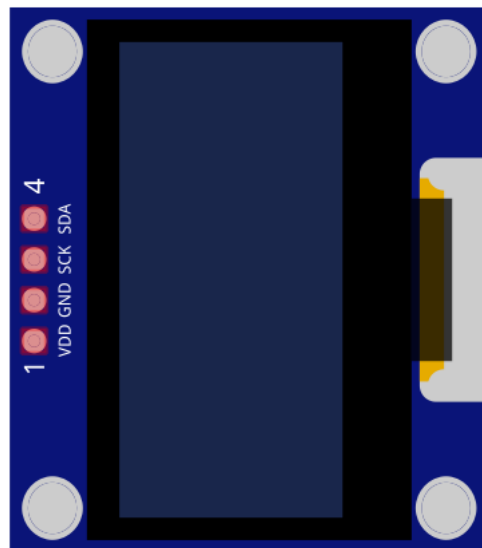
- » Tensione di alimentazione: da 3.3V a 5V
- » Interfaccia di comunicazione: I2C
- » Colore pixel: Bianco
- » Temperatura di esercizio: da -20 a 70 °C
- » Basso consumo energetico: < 11mA
- » Dimensioni: 36 x 34 x 3mm [1.4 x 1.3 x 0.1inch]

Per prolungare la vita dello schermo, è comune l'uso di uno "Screen saver". Si raccomanda di non utilizzare informazioni costanti per un lungo periodo di tempo, perché questo accorcia la durata dello schermo e aumenta il cosiddetto effetto "Screen burn".

La piedinatura

Lo schermo OLED da 1,3 pollici ha quattro pin. La piedinatura è mostrata nell'immagine seguente:

I2C Serial Data Line - SDA
I2C Serial Clock Line - SCK
Ground - GND
Power Supply - VDD



Az-Delivery

Lo schermo ha un regolatore di tensione a bordo da 3,3V. I pin dello schermo OLED da 1,3 pollici possono essere collegati all'alimentazione a 3,3V o a 5V senza pericolo per lo schermo stesso.

NOTA: Quando si utilizza il Raspberry Pi, l'alimentazione deve essere fornita solo da pin a 3,3V.

Come configurare l'Arduino IDE

Se l'Arduino IDE non è installato, seguire il [link](#) e scaricare il file di installazione del sistema operativo scelto.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle with a white infinity symbol containing a minus and a plus sign. To its right, the text reads: **ARDUINO 1.8.9**, followed by a description of the IDE as open-source software written in Java, and a note that it can be used with any Arduino board, referring to the 'Getting Started' page for installation instructions. On the right side, there is a teal sidebar with links for different operating systems: Windows (Installer and ZIP file), Windows app (with a 'Get' button), Mac OS X (10.8 Mountain Lion or newer), Linux (32 bits, 64 bits, ARM 32 bits, ARM 64 bits), Release Notes, Source Code, and Checksums (sha512).

Per gli utenti *Windows*, fare doppio clic sul file `.exe` scaricato e seguire le istruzioni nella finestra di installazione.

Az-Delivery

Per gli utenti *Linux*, scaricare un file con estensione *.tar.xz*, che è necessario estrarre. Quando lo si estrae, andare nella directory estratta, e aprire il terminale in quella directory. È necessario eseguire due script *.sh*, il primo chiamato *arduino-linux-setup.sh* e il secondo chiamato *install.sh*.

Per eseguire il primo script nel terminale, eseguire il seguente comando:

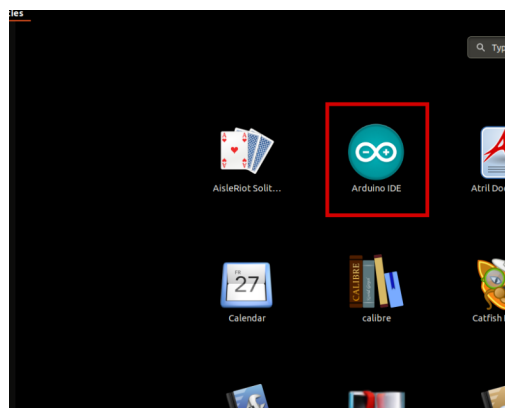
sh arduino-linux-setup.sh user_name

user_name - è il nome di un superutente nel sistema operativo Linux. Vi verrà richiesto di fornire la password per il superutente. Aspettate qualche minuto che lo script completi tutto.

Dopo l'installazione del primo script, si deve eseguire il secondo script chiamato *install.sh*. Nel terminale, eseguire il seguente comando:

sh install.sh

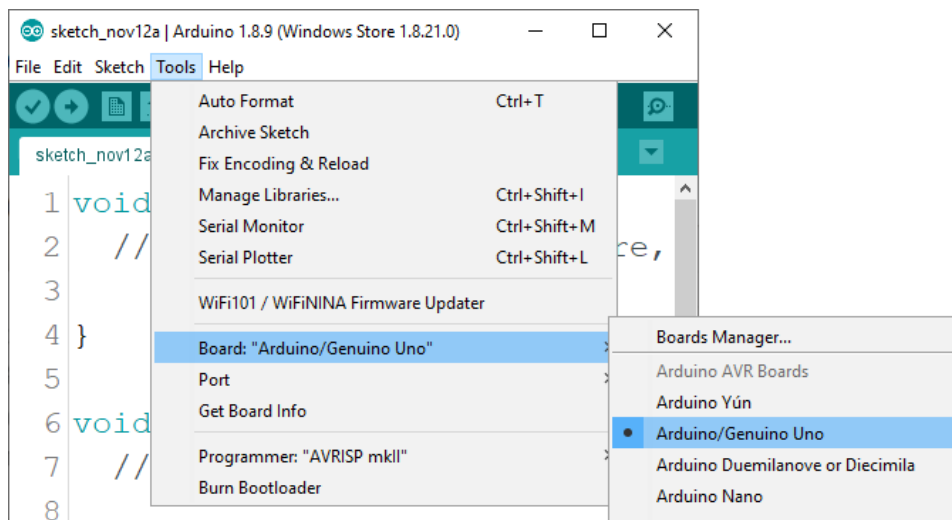
Dopo l'installazione di questi script, andare su *Tutte le App*, dove troverai l'Arduino IDE installato.



Az-Delivery

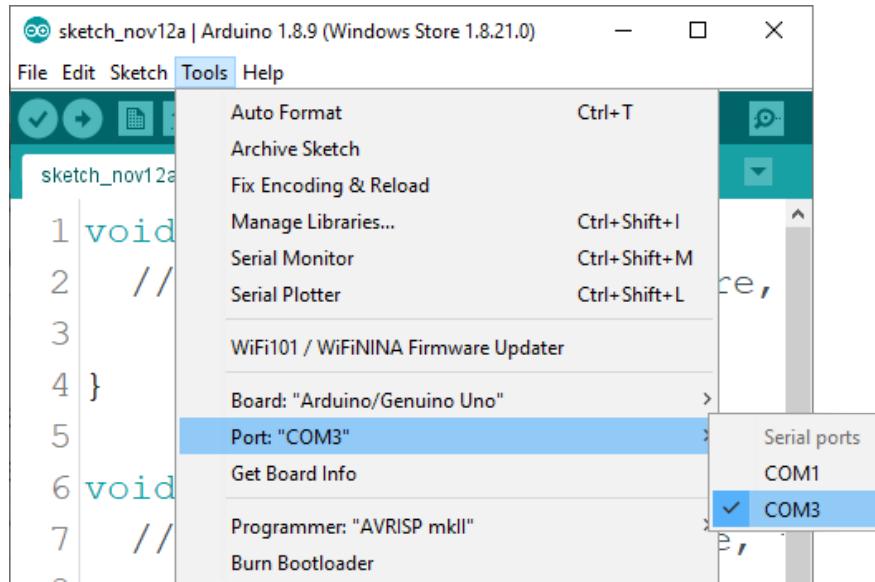
Quasi tutti i sistemi operativi sono dotati di un editor di testo preinstallato (ad esempio *Windows* viene fornito con *Notepad*, *Linux Ubuntu* viene fornito con *Gedit*, *Linux Raspbian* viene fornito con *Leafpad*, ecc..). Tutti questi editor di testo sono perfettamente adatti allo scopo dell'eBook.

La prossima cosa da fare è controllare se il PC è in grado di rilevare la scheda microcontrollore. Aprite l'Arduino IDE appena installato e andate su: *Strumenti > Scheda > {your board name here}*
{your board name here} dovrebbe essere l'*Arduino/Genuino Uno*, come si può vedere nella seguente immagine:



È necessario selezionare la porta alla quale è collegata la scheda microcontrollore. Vai su: *Strumenti > Porta > {port name goes here}*
e se avete collegato la scheda microcontrollore sulla porta usb dovreste vedere il nome della porta.

Se si utilizza l'Arduino IDE su Windows, i nomi delle porte sono i seguenti:



Per gli utenti *Linux*, il nome della porta è */dev/ttyUSBx* per esempio, dove x rappresenta un numero intero compreso tra 0 e 9, per esempio.



Come configurare il Raspberry Pi e il Python

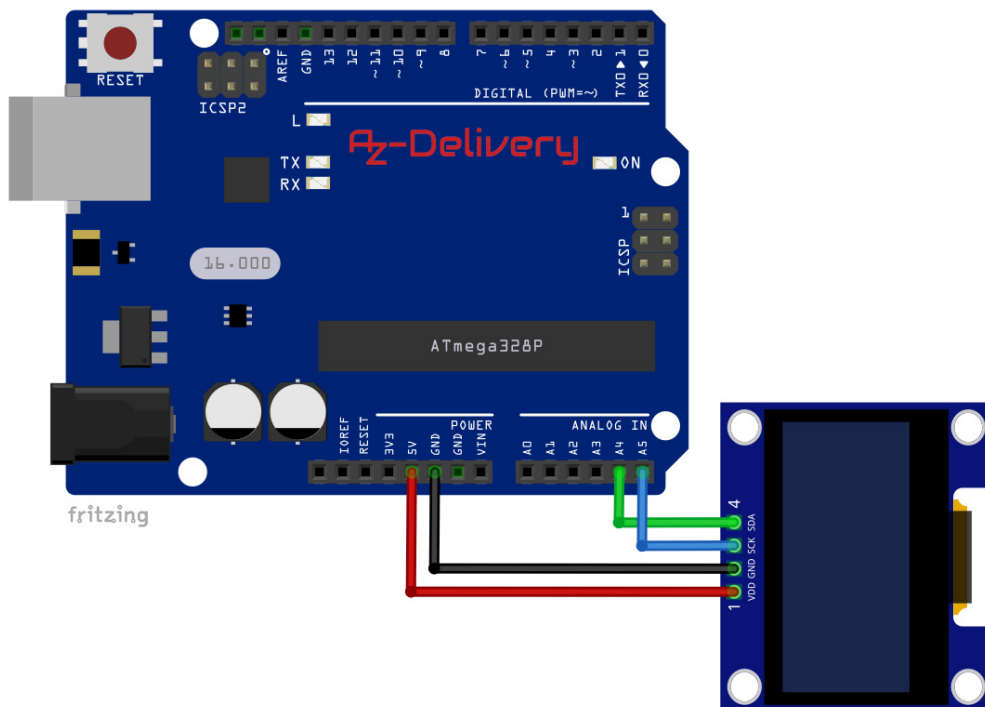
Per il Raspberry Pi, prima deve essere installato il sistema operativo, tutto deve essere impostato in modo da poter essere utilizzato in modalità Headless. La modalità Headless consente la connessione remota al Raspberry Pi, senza la necessità di uno schermo di un PC, mouse o tastiera. Le uniche cose di cui avete bisogno per questa modalità sono il Raspberry Pi, l'alimentazione e la connessione internet. Tutto questo è spiegato in dettaglio nell'eBook gratuito:

[Raspberry Pi Quick Startup Guide](#)

Il sistema operativo Raspbian viene fornito con il Python preinstallato.

Collegamento dello schermo con Uno

Collegare lo schermo da 1.3 pollici OLED con Uno come indicato nel seguente schema di collegamento:



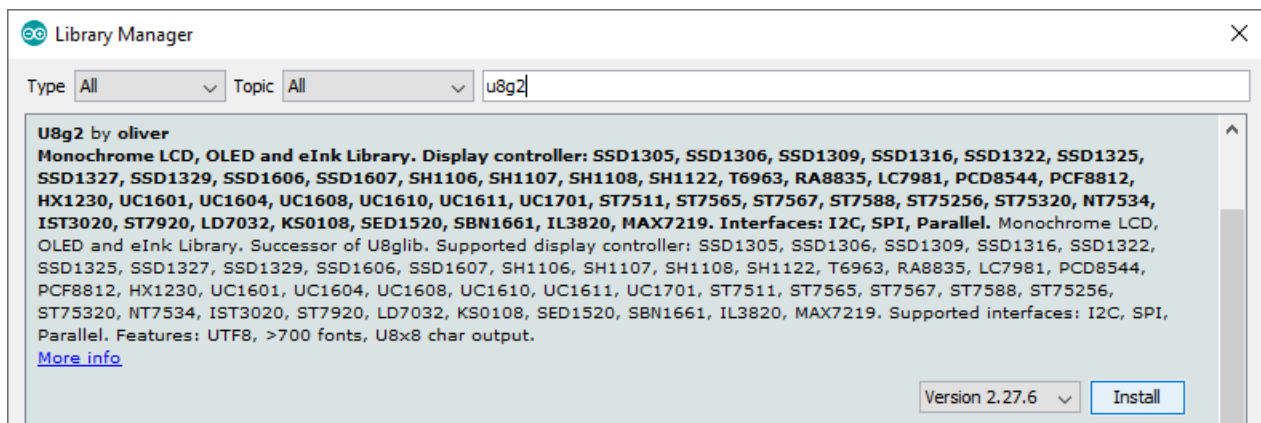
Pin schermo	Pin Uno	Colore filo
SDA	A4	Filo verde
SCK	A5	Filo blu
GND	GND	Filo nero
VCC	5V	Filo rosso

Libreria per Arduino IDE

Per utilizzare lo schermo con Uno, si consiglia di scaricare una libreria esterna per esso. La libreria utilizzata in questo eBook si chiama *U8g2*. Per scaricarla e installarla, aprire l'Arduino IDE e andare su:

Strumenti > Gestione Librerie.

Quando si apre una nuova finestra, digitare *u8g2* nella casella di ricerca e installare la libreria *U8g2* realizzata da oliver, come mostrato nella seguente immagine:



Con la libreria vengono forniti diversi esempi di sketch, per aprirne uno, andate su:

File > Esempi > U8g2 > full_buffer > Test grafico

Lo schermo può essere testato con questo esempio di sketch. In questo eBook, il codice dello sketch viene modificato per creare una versione più facile da usare per i principianti.

Az-Delivery

Esempio di sketch

```
#include <U8g2lib.h>
#include <Wire.h>
#define time_delay 2000
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);

const char COPYRIGHT_SYMBOL[] = {0xa9, '\\0'};
void u8g2_prepare() {
    u8g2.setFont(u8g2_font_6x10_tf);
    u8g2.setFontRefHeightExtendedText();
    u8g2.setDrawColor(1);
    u8g2.setFontPosTop();
    u8g2.setFontDirection(0);
}
void u8g2_box_frame() {
    u8g2.drawStr(0, 0, "drawBox");
    u8g2.drawBox(5, 10, 20, 10);
    u8g2.drawStr(60, 0, "drawFrame");
    u8g2.drawFrame(65, 10, 20, 10);
}
void u8g2_r_frame_box() {
    u8g2.drawStr(0, 0, "drawRFrame");
    u8g2.drawRFrame(5, 10, 40, 15, 3);
    u8g2.drawStr(70, 0, "drawRBox");
    u8g2.drawRBox(70, 10, 25, 15, 3);
}
void u8g2_disc_circle() {
    u8g2.drawStr(0, 0, "drawDisc");
    u8g2.drawDisc(10, 18, 9);
    u8g2.drawStr(60, 0, "drawCircle");
    u8g2.drawCircle(70, 18, 9);
}
```

Az-Delivery

```
void u8g2_string_orientation() {
    u8g2.setFontDirection(0);
    u8g2.drawStr(5, 15, "0");
    u8g2.setFontDirection(3);
    u8g2.drawStr(40, 25, "90");
    u8g2.setFontDirection(2);
    u8g2.drawStr(75, 15, "180");
    u8g2.setFontDirection(1);
    u8g2.drawStr(100, 10, "270");
}

void u8g2_line() {
    u8g2.drawStr(0, 0, "drawLine");
    u8g2.drawLine(7, 20, 77, 32);
}

void u8g2_triangle() {
    u8g2.drawStr(0, 0, "drawTriangle");
    u8g2.drawTriangle(14, 20, 45, 30, 10, 32);
}

void u8g2_unicode() {
    u8g2.drawStr(0, 0, "Unicode");
    u8g2.setFont(u8g2_font_unifont_t_symbols);
    u8g2.setFontPosTop();
    u8g2.setFontDirection(0);
    u8g2.drawUTF8(10, 20, "☀");
    u8g2.drawUTF8(30, 20, "☁");
    u8g2.drawUTF8(50, 20, "☂");
    u8g2.drawUTF8(70, 20, "☂");
    u8g2.drawUTF8(95, 20, COPYRIGHT_SYMBOL); //COPYRIGHT SYMBOL
    u8g2.drawUTF8(115, 15, "\xb0"); // DEGREE SYMBOL
}
```

Az-Delivery

```
#define image_width 128
#define image_height 21
static const unsigned char image_bits[] U8X8_PROGMEM = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x06, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xfc, 0x1f, 0x00, 0x00,
    0xfc, 0x1f, 0x00, 0x00, 0x06, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0xfe, 0x1f, 0x00, 0x00, 0xfc, 0x7f, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x07, 0x18, 0x00, 0x00, 0x0c, 0x60, 0x00, 0x00,
    0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x18, 0x00, 0x00,
    0x0c, 0xc0, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x03, 0x18, 0x00, 0x00, 0x0c, 0xc0, 0xf0, 0x1f, 0x06, 0x63, 0x80, 0xf1,
    0x1f, 0xfc, 0x33, 0xc0, 0x03, 0x18, 0x00, 0x00, 0x0c, 0xc0, 0xf8, 0x3f,
    0x06, 0x63, 0xc0, 0xf9, 0x3f, 0xfe, 0x33, 0xc0, 0x03, 0x18, 0x00, 0x00,
    0x0c, 0xc0, 0x18, 0x30, 0x06, 0x63, 0xc0, 0x18, 0x30, 0x06, 0x30, 0xc0,
    0xff, 0xff, 0xdf, 0xff, 0x0c, 0xc0, 0x18, 0x30, 0x06, 0x63, 0xe0, 0x18,
    0x30, 0x06, 0x30, 0xc0, 0xff, 0xff, 0xdf, 0xff, 0x0c, 0xc0, 0x98, 0x3f,
    0x06, 0x63, 0x60, 0x98, 0x3f, 0x06, 0x30, 0xc0, 0x03, 0x18, 0x0c, 0x00,
    0x0c, 0xc0, 0x98, 0x1f, 0x06, 0x63, 0x70, 0x98, 0x1f, 0x06, 0x30, 0xc0,
    0x03, 0x18, 0x06, 0x00, 0x0c, 0xc0, 0x18, 0x00, 0x06, 0x63, 0x38, 0x18,
    0x00, 0x06, 0x30, 0xc0, 0x03, 0x18, 0x03, 0x00, 0x0c, 0xe0, 0x18, 0x00,
    0x06, 0x63, 0x1c, 0x18, 0x00, 0x06, 0x30, 0xc0, 0x00, 0x80, 0x01, 0x00,
    0xfc, 0x7f, 0xf8, 0x07, 0x1e, 0xe3, 0x0f, 0xf8, 0x07, 0x06, 0xf0, 0xcf,
    0x00, 0xc0, 0x00, 0x00, 0xfc, 0x3f, 0xf0, 0x07, 0x1c, 0xe3, 0x07, 0xf0,
    0x07, 0x06, 0xe0, 0xcf, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x30, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0,
    0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0xe0, 0x00, 0xfc, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0x00, 0xfc, 0x1f, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f };
```


Az-Delivery

```
void u8g2_bitmap() {  
    u8g2.drawXBMP(0, 5, image_width, image_height, image_bits);  
}  
void setup(void) {  
    u8g2.begin();  
    u8g2_prepare();  
}  
float i = 0.0;  
void loop(void) {  
    u8g2.clearBuffer();  
    u8g2_prepare();  
    u8g2_box_frame();  
    u8g2.sendBuffer();  
    delay(time_delay);  
  
    u8g2.clearBuffer();  
    u8g2_disc_circle();  
    u8g2.sendBuffer();  
    delay(time_delay);  
  
    u8g2.clearBuffer();  
    u8g2_r_frame_box();  
    u8g2.sendBuffer();  
    delay(time_delay);  
  
    u8g2.clearBuffer();  
    u8g2_prepare();  
    u8g2_string_orientation();  
    u8g2.sendBuffer();  
    delay(time_delay);  
  
    u8g2.clearBuffer();  
    u8g2_line();  
    u8g2.sendBuffer();  
    delay(time_delay);  
}
```

Az-Delivery

```
// one tab
u8g2.clearBuffer();
u8g2_triangle();
u8g2.sendBuffer();
delay(time_delay);

u8g2.clearBuffer();
u8g2_prepare();
u8g2_unicode();
u8g2.sendBuffer();
delay(time_delay);

u8g2.clearBuffer();
u8g2_bitmap();
u8g2.sendBuffer();
delay(time_delay);

u8g2.clearBuffer();
u8g2.setCursor(0, 0);
u8g2.print(i);
i = i + 1.5;
u8g2.sendBuffer();
delay(time_delay);
}
```

Az-Delivery

All'inizio dello sketch vengono importate due librerie, *U8g2lib* e *wire*.

Successivamente, viene creato l'oggetto chiamato *u8g2*, con la seguente linea di codice:

```
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, U8X8_PIN_NONE);
```

L'oggetto creato rappresenta il sinottico stesso e viene utilizzato per controllare il sinottico. La libreria *U8g2* può essere utilizzata per molti altri schermi OLED, quindi ci sono molti costruttori negli esempi di sketch della libreria.

Dopo di che, viene creata la funzione chiamata *u8g2_prepare()*, che non ha argomenti e non restituisce alcun valore. All'interno di questa funzione vengono utilizzate cinque funzioni della libreria *u8g2*.

La prima funzione è chiamata *setFont()* che ha un solo parametro e non restituisce alcun valore. L'argomento rappresenta il font *u8g2*. L'elenco dei font disponibili si trova al seguente [link](#).

La seconda funzione è chiamata *setFontRefHeightExtendedText()* che non ha argomenti e non restituisce alcun valore. Viene utilizzata per disegnare i caratteri sullo schermo. Una spiegazione più dettagliata di questa funzione si trova al seguente [link](#).

Az-Delivery

La terza funzione è chiamata *setDrawColor()* che ha un solo argomento e non restituisce alcun valore. Il valore dell'argomento è un numero intero che rappresenta un indice di colore per tutte le funzioni di disegno. Le procedure di disegno dei font utilizzano questo argomento per impostare il colore di primo piano. Il valore predefinito è 1. Se è impostato a 0, allora lo spazio intorno al carattere è illuminato, e il carattere non lo è. Si può usare anche il valore di argomento 2, ma non c'è differenza rispetto a 0.

La quarta funzione è chiamata *setFontPosTop()* che non ha argomenti e non restituisce alcun valore. Questa funzione controlla la posizione dei caratteri in una riga del testo. La funzione ha un paio di versioni. Il primo è *setFontPosBaseLine()* il secondo è *setFontPosCenter()*. Il terzo è *setFontPosBottom()* e il loro scopo è quello di cambiare la posizione dei caratteri in una riga.

La quinta funzione è chiamata *setFontDirection()*, che ha un solo argomento e non restituisce alcun valore. L'argomento è un numero intero che rappresenta la direzione del testo. Il valore è un numero intero nell'intervallo da 0 a 3, dove 0 = 0°, 1 = 90°, 2 = 180° e 3 = 270°.

Az-Delivery

La funzione chiamata *drawStr()* ha tre argomenti e non restituisce alcun valore. Viene utilizzata per visualizzare una stringa costante sullo schermo. I primi due argomenti rappresentano la posizione *X* e *Y* del cursore, dove viene visualizzato il testo. Il terzo argomento rappresenta il testo stesso, un valore di stringa costante. Si dovrebbero usare le funzioni che impostano il layout del testo prima di usare la funzione *drawStr()*, altrimenti la funzione *drawStr()* usa impostazioni predefinite per il carattere, la dimensione e il layout generale del testo.

Per visualizzare le forme, vengono utilizzate funzioni specifiche per ogni forma:

La funzione chiamata *drawFrame()*, ha quattro argomenti e non restituisce alcun valore. Si usa per visualizzare una cornice, un rettangolo vuoto. I primi due argomenti rappresentano la posizione *X* e *Y* dell'angolo in alto a sinistra della cornice. Il terzo argomento rappresenta la larghezza del frame e il quarto argomento rappresenta l'altezza della cornice.

La funzione chiamata *drawRFrame()* ha cinque argomenti e non restituisce alcun valore. Viene utilizzata per visualizzare una cornice con angoli arrotondati. I primi due argomenti rappresentano la posizione *X* e *Y* dell'angolo in alto a sinistra della cornice. I secondi due argomenti rappresentano la larghezza e l'altezza della cornice e il quinto argomento rappresenta il raggio dell'angolo.

Az-Delivery

La funzione chiamata *drawBox()* ha quattro argomenti e non restituisce alcun valore. Viene utilizzata per visualizzare un rettangolo pieno. I primi due argomenti rappresentano la posizione *X* e *Y* dell'angolo in alto a sinistra del rettangolo. I secondi due argomenti rappresentano rispettivamente la larghezza e l'altezza del rettangolo.

La funzione chiamata *drawRBox()* ha cinque argomenti e non restituisce alcun valore. Viene utilizzato per visualizzare un rettangolo riempito con bordi arrotondati. I primi due argomenti rappresentano la posizione *X* e *Y* dell'angolo in alto a sinistra del rettangolo. I secondi due argomenti rappresentano rispettivamente la larghezza e l'altezza del rettangolo. Il quinto argomento rappresenta il raggio d'angolo.

La funzione chiamata *drawCircle()* ha tre argomenti e non restituisce alcun valore. Viene utilizzata per visualizzare un cerchio. I primi due argomenti rappresentano le posizioni *X* e *Y* del punto centrale del cerchio. Il terzo argomento rappresenta il raggio del cerchio.

La funzione chiamata *drawDisc()* ha tre argomenti e non restituisce alcun valore. Si usa per visualizzare un disco. I primi due argomenti rappresentano le posizioni *X* e *Y* del punto centrale del disco. Il terzo argomento rappresenta il raggio del disco.

Az-Delivery

La funzione chiamata *drawTriangle()* ha sei argomenti e non restituisce alcun valore. Viene utilizzata per visualizzare un triangolo pieno. I primi due argomenti rappresentano la posizione *X* e *Y* del punto del primo angolo del triangolo. I secondi due argomenti rappresentano le posizioni *X* e *Y* del secondo punto d'angolo del triangolo. Gli ultimi due argomenti rappresentano le posizioni *X* e *Y* dell'ultimo punto d'angolo del triangolo.

La funzione chiamata *drawLine()* ha quattro argomenti e non restituisce alcun valore. Viene utilizzata per visualizzare una linea. I primi due argomenti rappresentano la posizione *X* e *Y* del punto di partenza della retta. I secondi due argomenti rappresentano la posizione *X* e *Y* del punto finale della retta.

La funzione chiamata *drawUTF8()* ha tre argomenti e restituisce un valore. Viene utilizzata per visualizzare un testo, il valore della stringa che può contenere un carattere codificato come carattere Unicode. I primi due argomenti rappresentano la posizione *X* e *Y* del cursore e il terzo rappresenta il testo stesso. I caratteri Unicode possono essere visualizzati in un paio di modi. Il primo è quello di copiare e incollare il carattere esistente nello sketch, come nella seguente riga del codice:

```
u8g2.drawUTF8(50, 20, "☂")
```

Il secondo è quello di creare un array di caratteri, che ha due valori: il primo valore è un numero esadecimale del carattere *Unicode*, e il secondo valore è un carattere nullo (" "). Questo può essere fatto utilizzando l'array di caratteri chiamato `COPYRIGHT_SYMBOL`, come nella seguente riga di codice:

```
const char COPYRIGHT_SYMBOL[] = {0xa9, '\0'}  
u8g2.drawUTF8(95, 20, COPYRIGHT_SYMBOL); //COPYRIGHT SYMBOL
```

Az-Delivery

Il terzo modo di utilizzare la funzione è quello di utilizzare un numero esadecimale per il carattere stesso, come nella seguente riga di codice:

```
u8g2.drawUTF8(115, 15, "\xb0"); // DEGREE SYMBOL
```

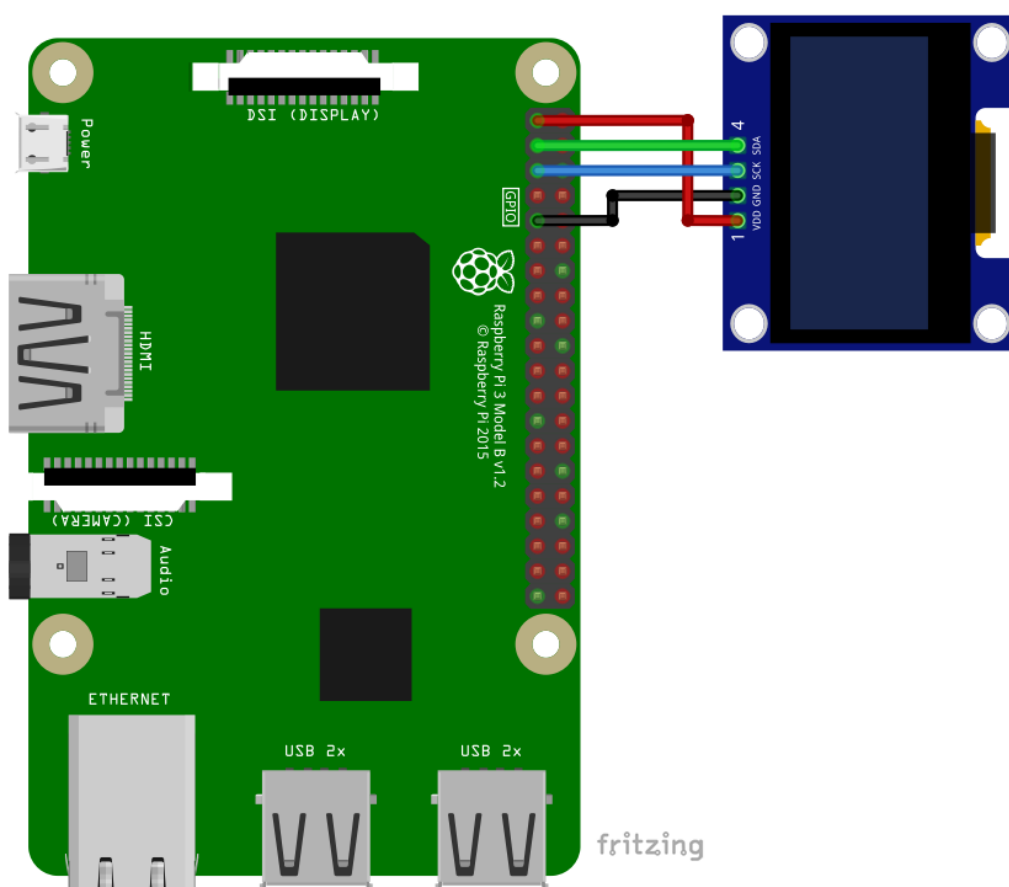
La funzione restituisce un valore, un numero intero che rappresenta la larghezza del testo (stringa).

Per visualizzare qualcosa sullo schermo, il buffer di dati dello schermo deve essere prima cancellato, poi viene impostato un nuovo valore (un'immagine) per il buffer di dati, poi un nuovo valore del buffer di dati viene inviato allo schermo. In questo modo, una nuova immagine viene visualizzata sullo schermo. Per vedere questo cambiamento, la funzione *delay()* deve essere usata per spostare il prossimo cambiamento del buffer di dati, come nelle righe di codice seguenti:

```
u8g2.clearBuffer();  
u8g2_bitmap(); // setting the data buffer  
u8g2.sendBuffer();  
delay(time_delay);
```


Collegamento dello schermo con Raspberry Pi

Collegare lo schermo con il Raspberry Pi come indicato nel seguente schema di collegamento:

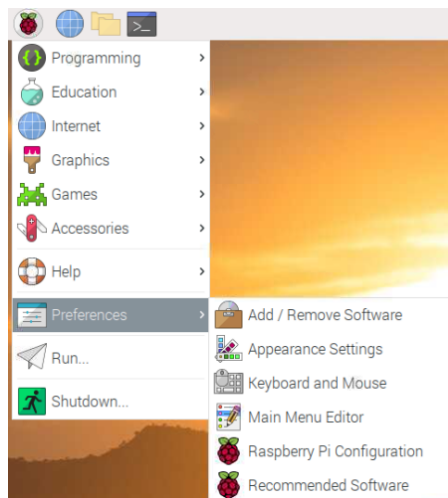


Pin schermo	Pin Raspberry Pi	Pin fisico	Colore filo
SDA	GPIO2	3	Filo verde
SCL	GPIO3	5	Filo blu
GND	GND	9	Filo nero
VCC	3V3	1	Filo rosso

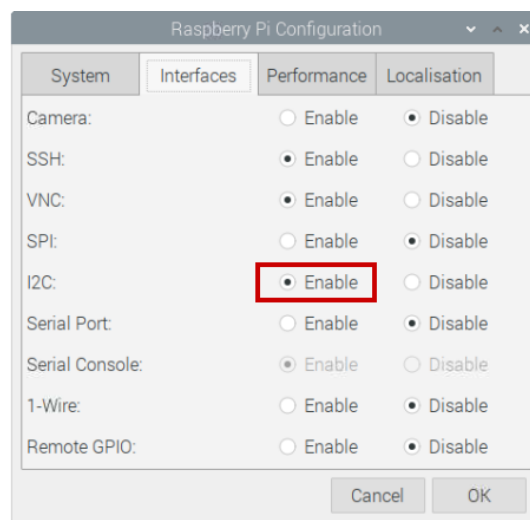
Abilitazione dell'interfaccia I2C

Per poter utilizzare lo schermo con Raspberry Pi, l'interfaccia I2C deve essere abilitata. Aprite il seguente menu:

Menu Applicazione > Preferiti > Configurazione Raspberry Pi



Nella nuova finestra, sotto la scheda Interfacce, abilitare il radio button I2C, come nell'immagine seguente:



Librerie e strumenti per Python

Per utilizzare lo schermo OLED da 1.3 pollici con il Raspberry Pi, si consiglia di scaricare e installare una libreria esterna. La libreria utilizzata in questo eBook si chiama luma.oled. Per utilizzarla è necessario installare alcuni strumenti per Python.

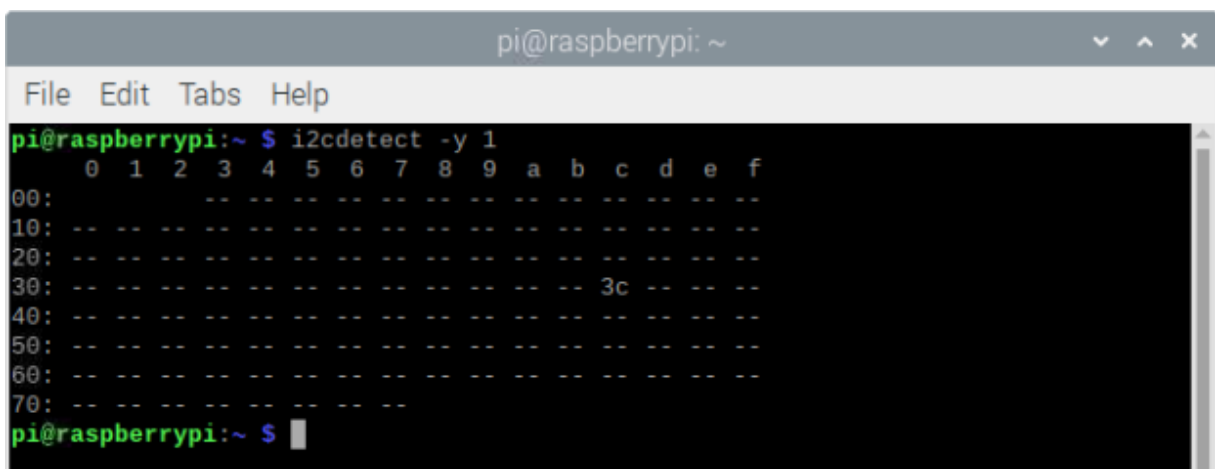
Aprire il terminale ed eseguire il seguente comando:

```
sudo apt install i2c-tools python3-dev python3-pip  
libfreetype6-dev libjpeg-dev build-essential libopenjp2-7  
libtiff5 git
```

Successivamente deve essere rilevato l'indirizzo I2C dello schermo. Apri il terminale e segui il seguente comando:

```
i2c detect -y 1
```

Il risultato dovrebbe assomigliare all'uscita mostrata nell'immagine seguente:



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

dove 0x3c è l'indirizzo I2C dello schermo.

Az-Delivery

Per installare la libreria *luma.oled*, aprire il terminale ed eseguire il seguente comando:

```
sudo -H pip3 install -upgrade luma.oled
```

L'ultima cosa da fare è scaricare esempi di script da GitHub. Aprire il terminale ed eseguire il seguente comando:

```
git clone https://github.com/rm-hull/luma.examples.git
```

Quindi, cambiare la directory in *luma.examples*, con il seguente comando:

```
cd luma.examples
```

e per installare la libreria eseguire il seguente comando:

```
sudo -H pip3 install -e .
```

(con un punto alla fine)

Az-Delivery

Script Python

Il seguente codice script è un codice modificato da due script:

`/luma.examples/demo.py` e

`/luma.examples/demo_opts.py`.

```
import time
import sys
from luma.core.interface.serial import i2c
from luma.oled.device import sh1106
from luma.core.render import canvas
from PIL import ImageFont

font_path = 'ChiKareGo.ttf'

def changing_var(device):
    size = 40
    nf = ImageFont.truetype(font_path, size) # new font

    for i in range(100):
        with canvas(device) as draw:
            draw.text((28, 7), 'Changing var.', fill=1)
            if i < 10:
                draw.text((50, 22), '0{}'.format(str(i)), font=nf, fill=1)
            else:
                draw.text((50, 22), str(i), font=nf, fill=1)

        time.sleep(0.001)
```

AZ-Delivery

```
def primitives(device):  
    with canvas(device) as draw:  
        # Draw a rectangle.  
        draw.rectangle((4, 4, 40, 10), outline=1, fill=0)  
  
        # Draw an ellipse.  
        draw.ellipse((4, 20, 18, 34), outline=1, fill=1)  
  
        # Draw a triangle.  
        draw.polygon([(10, 44), (40, 20), (40, 44)], outline=1, fill=0)  
  
        # Draw an X.  
        draw.line((4, 48, 126, 62), fill=1)  
        draw.line((4, 62, 126, 48), fill=1)  
  
        # Write two lines of text.  
        draw.text((45, 20), 'AZ-Delivery', fill=1)  
  
        size = 10  
        nf = ImageFont.truetype(font_path, size)  
        draw.text((45, 4), 'AZ-Delivery', font=nf, fill=1)
```

Az-Delivery

```
try:
    serial = i2c(port=1, address=0x3c)
    device = sh1106(serial, rotate=0, width=128, height=64)

    print('[Press CTRL + C to end the script!]\n')
    while(True):
        print('Testing printing variable.\n')
        changing_var(device)
        time.sleep(2)

        print('Testing basic graphics.\n')
        primitives(device)
        time.sleep(3)

        print('Testing display ON/OFF.\n')
        for _ in range(5):
            time.sleep(0.5)
            device.hide()
            time.sleep(0.5)
            device.show()

        print('Testing clearing display.\n')
        device.clear()
        time.sleep(2)

except KeyboardInterrupt:
    print('Script end!')
```

Az-Delivery

Salvare lo script con il nome *OLED.py*. Nella stessa directory dove viene salvato lo script, deve essere salvato anche il file di font. Il font utilizzato in questo script si chiama *ChikareGo*. Questo font viene salvato nella directory degli esempi, in posizione:

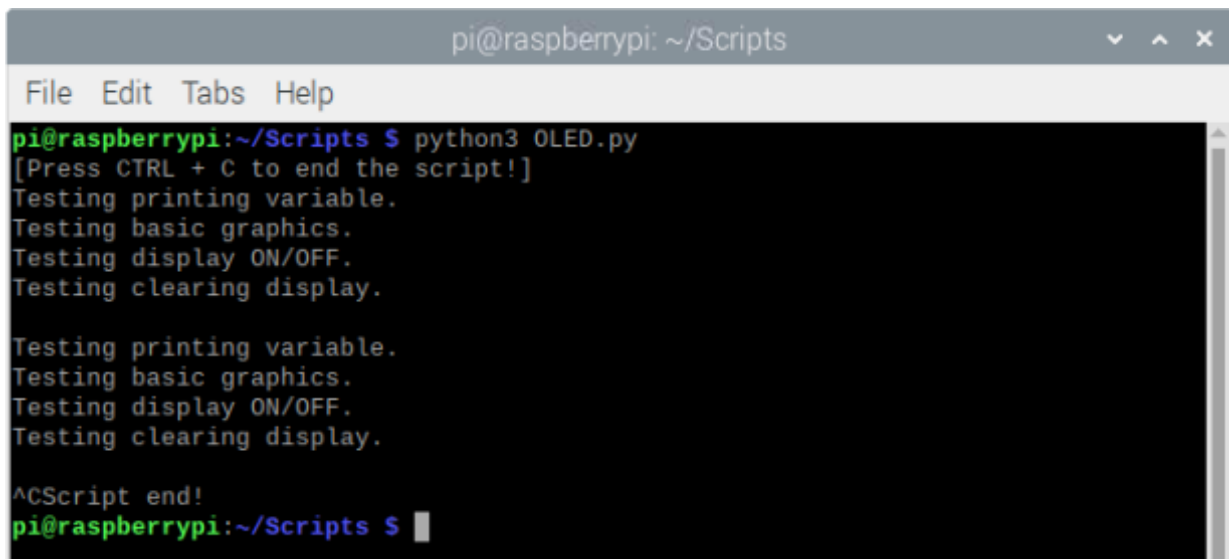
```
luma.examples > examples > fonts > ChikareGo.ttf
```

Basta copiare e incollare questo file nella stessa directory in cui è salvato lo script *OLED.py*. Se questo non viene fatto, quando viene eseguito il comando *python3 OLED.py*, viene generato l'errore che indica che lo script non riesce a trovare il file di font. Per eseguire lo script, aprire il terminale nella directory dove lo script viene salvato ed eseguire il seguente comando:

python3 OLED.py

Il risultato dovrebbe assomigliare all'uscita mostrata nell'immagine seguente:

Az-Delivery



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 OLED.py
[Press CTRL + C to end the script!]
Testing printing variable.
Testing basic graphics.
Testing display ON/OFF.
Testing clearing display.

Testing printing variable.
Testing basic graphics.
Testing display ON/OFF.
Testing clearing display.

^CScript end!
pi@raspberrypi:~/Scripts $
```

Per fermare lo script premere CTRL + C sulla tastiera.

Az-Delivery

Lo script inizia con l'importazione di librerie e funzioni.

Poi il percorso del file di font viene salvato nella variabile *font_path*. Il font utilizzato si chiama ChiKareGo.ttf, che si trova in:

```
luma.examples > examples > fonts > ChiKareGo.ttf
```

Il file di font deve essere copiato nella stessa directory dove viene salvato lo script!

Poi, vengono create due funzioni, la prima si chiama *changing_var()* e la seconda si chiama *primitive()*.

La funzione *changing_var()* viene utilizzata per visualizzare una variabile che cambia il suo stato ogni millisecondo. La funzione accetta un argomento e non restituisce alcun valore. L'argomento è chiamato *argomento del dispositivo*, e sarà spiegato più avanti nel testo. All'inizio della funzione *changing_var()*, la variabile di dimensione viene definita e inizializzata con il numero *40*. Questo valore rappresenta la dimensione del carattere. Quindi, viene creata una variabile chiamata *nf* (*new_font*), che rappresenta il font utilizzato. Le variabili *size* e *font_path* sono utilizzate per creare la variabile *nf*. Successivamente, il ciclo *for* viene creato per far passare in loop i primi cento numeri interi. Questi valori sono usati come valori per la variabile che viene visualizzata sullo schermo. Nel loop *for*, l'immagine viene creata con la seguente riga del codice:

```
with canvas(device) as draw:
```

dove viene creato un oggetto chiamato *draw*. L'oggetto *draw* rappresenta l'immagine stessa.

Az-Delivery

Per visualizzare il testo viene utilizzata la funzione `text()`. La funzione `text()` accetta quattro argomenti, dove uno è opzionale, e non restituisce alcun valore. Il primo argomento è una tupla che ha due elementi, dove gli elementi rappresentano la posizione *X* e *Y* del cursore. Il secondo argomento rappresenta il testo stesso, un valore di stringa. Il terzo argomento è quello *fill*. Il quarto argomento è opzionale, si chiama argomento *font*. Se l'argomento non viene utilizzato, il font è impostato su font e dimensioni predefinite. Ma se la variabile *nf* è memorizzata all'interno dell'argomento del font, si può usare il font desiderato e si cambia anche la dimensione del font.

L'argomento *fill* rappresenta il colore del testo. Se *fill* = 0 il colore è nero (nulla visualizzato sullo schermo) e se *fill* è uguale a qualsiasi altro valore, ad esempio *fill* = 1, il colore del testo è bianco (perché gli schermi OLED hanno solo colori in bianco e nero). Questa libreria può essere utilizzata anche per molti altri schermi, quindi all'interno dell'argomento di riempimento può essere memorizzato qualsiasi altro colore, cosa che non è possibile per lo schermo OLED da 1,3 pollici.

Il resto del codice all'interno del ciclo *for* è un algoritmo per visualizzare testo e variabile con numeri a due cifre a partire da 0 a 99.

La seconda funzione è chiamata *primitives()* ed è usata per disegnare forme sullo schermo. Anch'essa accetta un argomento, l'argomento del dispositivo, e non restituisce alcun valore. All'inizio della funzione, viene creato l'oggetto immagine chiamato *draw*. Questo oggetto viene utilizzato per disegnare forme.

Az-Delivery

Per disegnare il rettangolo si utilizza la funzione `rectangle()`. La funzione accetta tre argomenti e non restituisce alcun valore. Il primo argomento è una tupla di quattro elementi, dove i primi due elementi rappresentano la posizione X e Y dell'angolo in alto a destra del rettangolo. Il terzo e il quarto elemento rappresentano la posizione X e Y dell'angolo in basso a destra del rettangolo. Il secondo argomento è l'argomento del contorno e il terzo argomento è l'argomento del riempimento.

Il valore della posizione X inizia dal lato sinistro dello schermo (valore 0) e finisce dal lato destro dello schermo (valore 127). Il valore della posizione Y inizia dal lato superiore dello schermo (valore di 0) e finisce dal lato inferiore dello schermo (valore di 63).

L'argomento del contorno rappresenta il colore del bordo della forma e l'argomento del riempimento rappresenta il colore della forma stessa. Poiché si utilizzano schermi OLED, i colori sono bianco e nero, nero - il pixel è spento, bianco - il pixel è acceso. Quando il valore di zero viene salvato nell'argomento contorno o riempimento, significa che è un colore nero. Quando viene salvato qualsiasi altro valore superiore a zero, ad esempio 1, questo rappresenta il colore bianco.

Az-Delivery

Per disegnare ellissi o cerchi si usa la funzione *ellipse()*. La funzione accetta tre argomenti e non restituisce alcun valore. Il primo argomento è una tupla di quattro elementi. I primi due elementi rappresentano le posizioni *X* e *Y* dell'angolo in alto a sinistra del rettangolo che contiene l'ellisse. I secondi due elementi rappresentano le posizioni *X* e *Y* dell'angolo in basso a destra del rettangolo che contiene l'ellisse. Il secondo argomento è l'argomento del contorno e il terzo argomento è l'argomento del riempimento.

Per disegnare un poligono si usa la funzione *polygon()*. Un poligono è un triangolo, un rettangolo o qualsiasi altra forma con 3 o più angoli. La funzione accetta tre argomenti. Il primo argomento è una lista di tre o più tuple. Le tuple hanno due elementi, che rappresentano le posizioni *X* e *Y* del punto d'angolo della forma. Il numero di tuple nella lista è arbitrario, tre o più tuple, che dipende dalla forma che si vuole disegnare. Il secondo argomento è l'argomento del contorno e il terzo argomento è l'argomento del riempimento.

Per disegnare una linea si usa la funzione *line()*. La funzione accetta due argomenti e non restituisce alcun valore. Il primo argomento è una tupla di quattro elementi, dove i primi due elementi rappresentano la posizione *X* e *Y* del punto iniziale di una retta e i secondi due elementi rappresentano la posizione *X* e *Y* del punto finale di una retta. Il secondo argomento è l'argomento di riempimento.

Az-Delivery

Alla fine della funzione `primitives()`, la funzione `text()` viene utilizzata con e senza argomento `font` per mostrare la differenza.

La funzione `text()` ha quattro argomenti e non restituisce alcun valore. La funzione visualizza il testo sullo schermo. Il primo argomento è una tupla con due elementi. Gli elementi rappresentano la posizione `X` e `Y` del cursore dove viene visualizzato il testo. Il terzo e il quarto argomento sono facoltativi. Se questi due argomenti non vengono utilizzati, la funzione `text()` visualizza il testo con font e dimensioni predefinite. Se il testo deve essere visualizzato con dimensioni e/o caratteri diversi, utilizzare il terzo e il quarto argomento. Il terzo è l'argomento `font` e il quarto è l'argomento `size`. L'argomento `font` rappresenta il percorso del file font. L'argomento della dimensione è un numero intero senza segno, con valori compresi tra 1 e 100. Più alto è il valore più grande, più grande è il carattere visualizzato sullo schermo.

Dopo queste due funzioni viene creato il blocco di codice `try-except`. Nel blocco `try` del codice vengono creati due oggetti e il loop indefinito.

Il primo oggetto si chiama `serial` e serve per impostare l'interfaccia I2C e l'indirizzo per lo schermo. Per inizializzare questo oggetto si usa la seguente riga del codice:

```
serial = i2c(port=1, address=0x3c)
```

Az-Delivery

Il secondo oggetto è chiamato `device`. Questo oggetto rappresenta il chip del driver stesso, ed è utilizzato per controllare il chip del driver. Per inizializzare l'oggetto `device` vengono utilizzati gli argomenti oggetto `serial`, `width` e `height`, nella seguente riga di codice:

```
device = sh1106(serial, width=128, height=64)
```

Successivamente, viene creato un loop infinito (`while True:`). All'interno del loop infinito, sono state escluse due funzioni e sono state testate due proprietà dello schermo. Prima viene eseguita la funzione `changing_var()`, poi viene eseguita la funzione `primitives()`. Dopo di che, `device.hide()` viene utilizzato per spegnere lo schermo e `device.show()` per accenderlo di nuovo. Alla fine del loop infinito, `device.clear()` viene utilizzato per cancellare il buffer di dati dello schermo, che cancella l'immagine dello schermo.

Il blocco di codice `except` viene eseguito quando si preme CTRL + C sulla tastiera. Questo viene chiamato interruzione da tastiera. Quando viene eseguito il blocco di codice `except`, il messaggio *Script end!* viene visualizzato nel terminale.



E ora è tempo di imparare e di creare dei Progetti da solo. Lo puoi fare con l'aiuto di molti script di esempio e altri tutorial, che puoi trovare in internet.

Se stai cercando dei prodotti di alta qualità per il tuo microelettronica e accessori di alta qualità, AZ-Delivery Vertriebs GmbH è l'azienda giusta dove potrai trovarli. Ti forniremo numerosi esempi di applicazioni, guide di installazione complete, e-book, librerie e l'assistenza dei nostri esperti tecnici.

<https://az-delivery.de>

Buon divertimento!

Impressum

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