

AZ-Delivery

Benvenuto!

Grazie per aver acquistato il nostro *schermo OLED I2C da 0,96 pollici AZ-Delivery*. Nelle pagine seguenti, ti illustreremo come utilizzare e configurare questo pratico dispositivo.

Buon divertimento!



Az-Delivery

Indice dei Contenuti

Introduzione.....	3
Specifiche.....	4
L'indirizzo I2C dello schermo OLED.....	5
La piedinatura.....	6
Come configurare l'Arduino IDE.....	7
Come configurare il Raspberry Pi e il Python.....	11
Collegamento dello schermo con Atmega328P Board.....	12
Libreria per Arduino IDE.....	13
Esempio di sketch.....	14
Collegamento dello schermo con Raspberry Pi.....	25
Abilitazione dell'interfaccia I2C.....	26
Librerie e strumenti per Python.....	27
Script Python.....	31



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 0,96 ha una risoluzione di 128x32 pixel (LED). Per controllare questi LED abbiamo bisogno di un circuito pilota o di un chip. Lo schermo ha un chip del driver chiamato *SSD1306*. Il chip del driver ha un'interfaccia I2C per la comunicazione con il microcontrollore principale. L'indirizzo I2C di un chip del driver è predefinito con valore *0x3C*.

Lo schermo OLED e il chip del driver SSD1306 funzionano nel range 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.

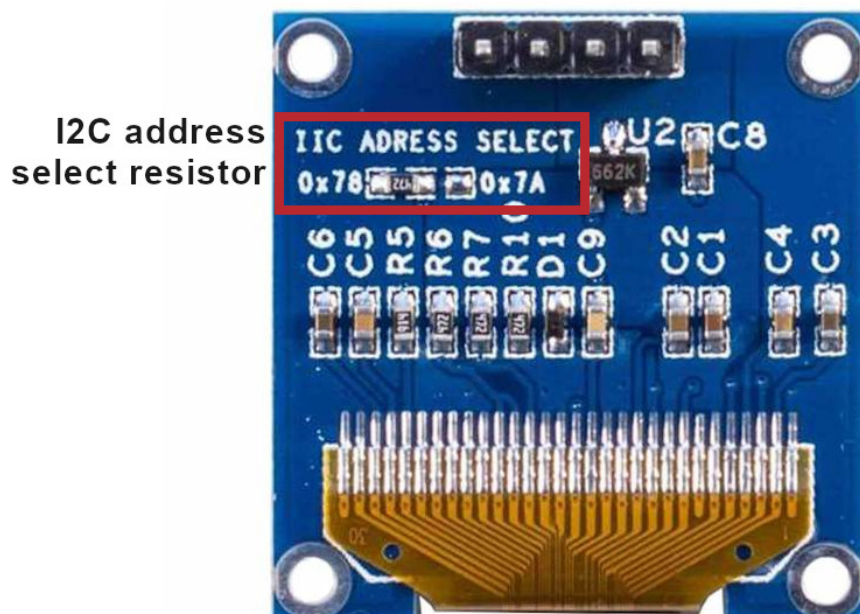
Specifiche

» Tensione di alimentazione:	da 3.3V a 5V DC
» Interfaccia di comunicazione:	I2C
» Colore pixel:	Bianco
» Temperatura di esercizio:	da -20 a 70 °C
» Basso consumo energetico	meno di 11mA
» Dimensioni	28 x 33 x 4mm [1.1 x 1.3 x 0.13 in]

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

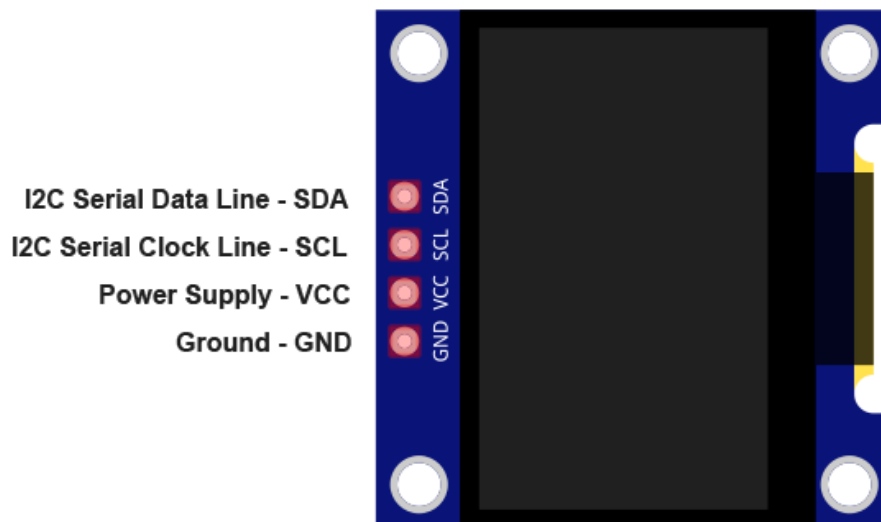
L'indirizzo I2C dello schermo OLED

Tutti gli schermi OLED, che AZ-Delivery offre, hanno lo stesso indirizzo seriale, $0x3C$. Lo schermo da 0,96 pollici offre la possibilità di saldare nuovamente un resistore sul retro della scheda schermo per un altro indirizzo, $0x3D$, ma ciò non è consigliabile. Nel caso in cui si debbano utilizzare più schermi con un'interfaccia I2C, si raccomanda di utilizzare un dispositivo multiplexer I2C o semplicemente uno schermo più grande.



La piedinatura

Lo schermo OLED da 0.96 pollici ha quattro pin. La piedinatura è mostrata nell'immagine seguente:



Lo schermo ha a bordo un regolatore di tensione a 3,3V. I pin dello schermo OLED da 0,96 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:

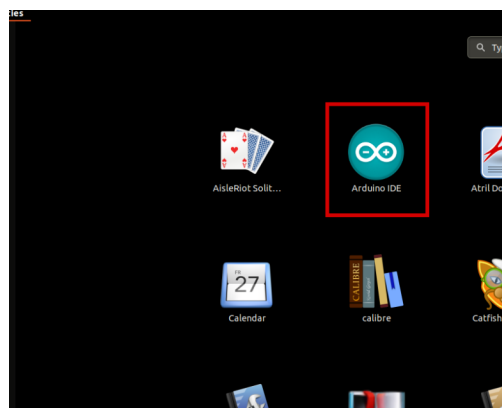
sh arduino-linux-setup.sh user_name

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

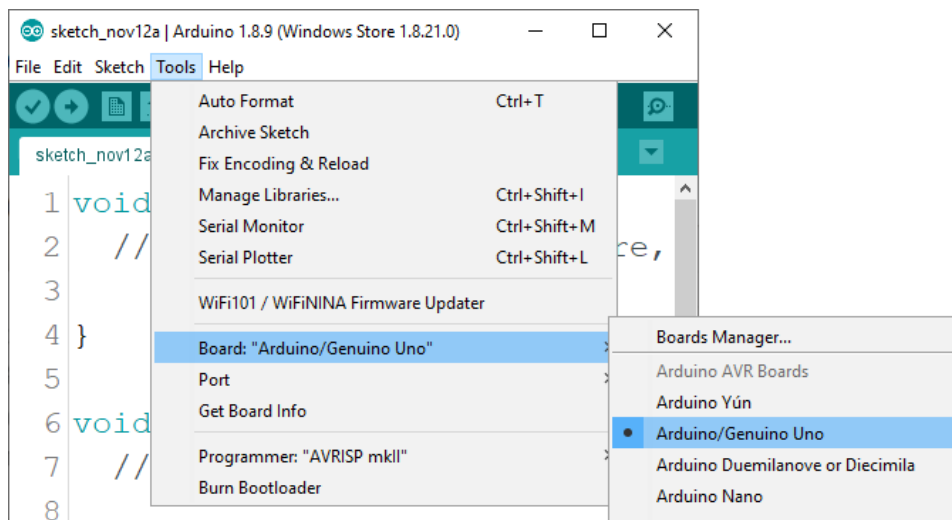
Dopo l'installazione di questi script, andare su *Tutte le App*, dove troverai l'Arduino IDE installato.



Az-Delivery

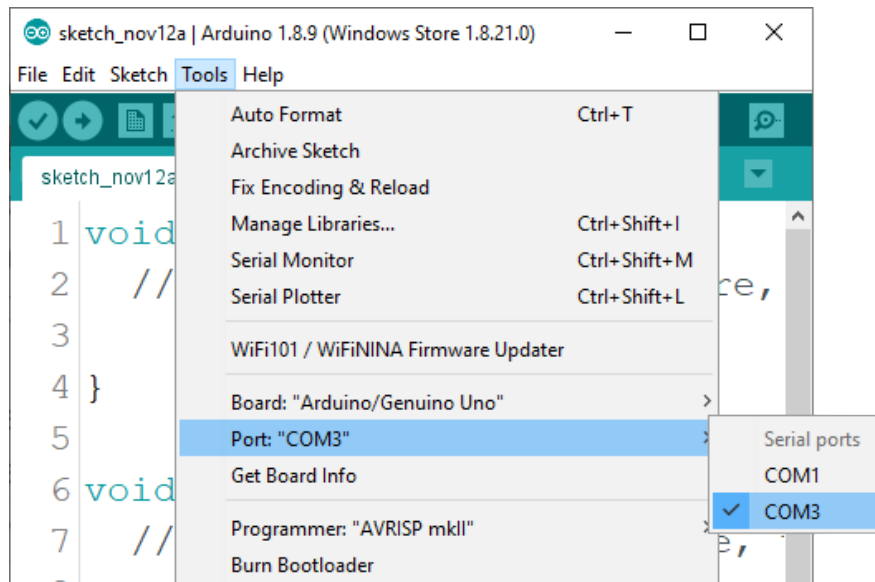
Quasi tutti i sistemi operativi sono dotati di un editor di testo preinstallato (ad esempio *Windows* viene fornito con *Notepad*, *Linux Ubuntu* viene fornito con *Gedit*, *Linux Raspbian* viene fornito con *Leafpad*, ecc..). Tutti questi editor di testo sono perfettamente adatti allo scopo dell'eBook.

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.



Come configurare il Raspberry Pi e il Python

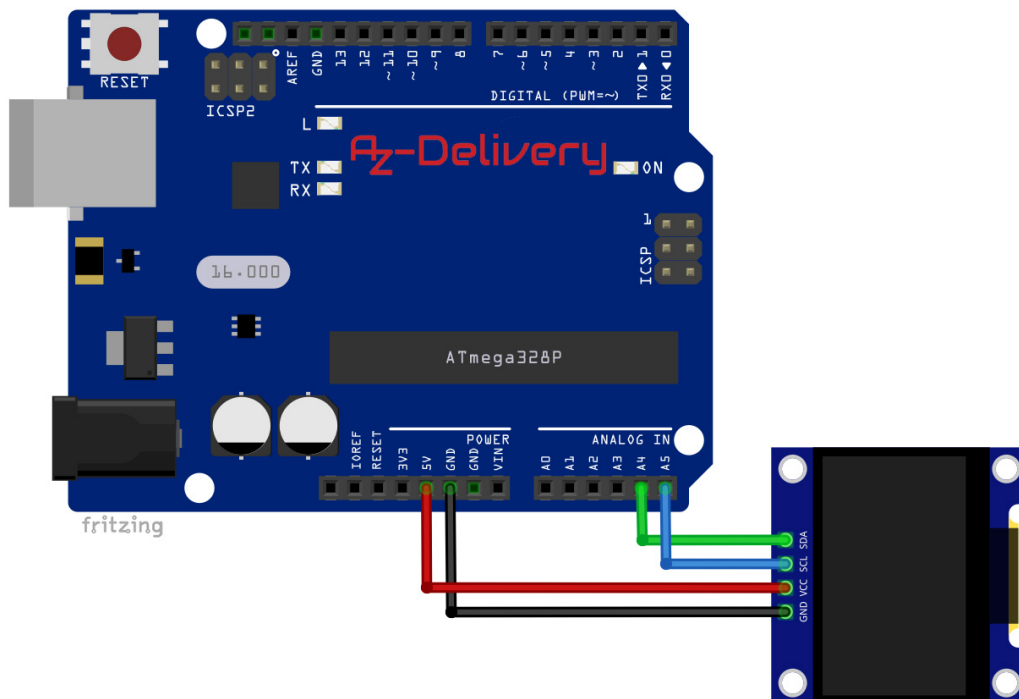
Per il Raspberry Pi, prima deve essere installato il sistema operativo, tutto deve essere impostato in modo da poter essere utilizzato in modalità Headless. La modalità Headless consente la connessione remota al Raspberry Pi, senza la necessità di uno schermo di un PC, mouse o tastiera. Le uniche cose 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 Atmega328P Board

Collegare lo schermo da 0.96 pollici OLED con Atmega328P Board come indicato nel seguente schema di collegamento:

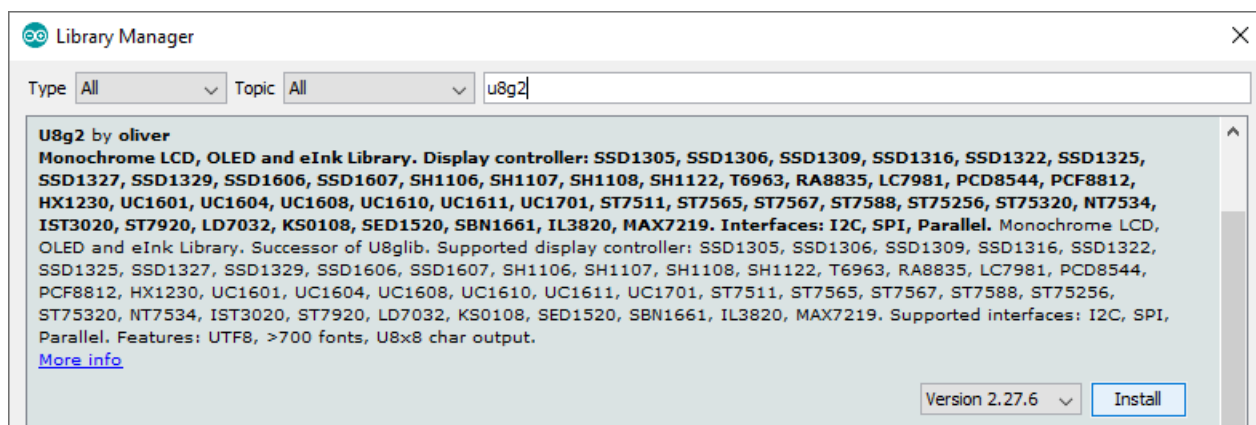


Pin schermo	Pin	Colore filo
SDA	A4	Filo blu
SCL	A5	Filo verde
VCC	5V	Filo rosso
GND	GND	Filo nero

Libreria per Arduino IDE

Per utilizzare lo schermo con Atmega328P Board, 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 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_SSD1306_128X64_UNIVISION_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, 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_SSD1306_128X32_UNIVISION_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
```

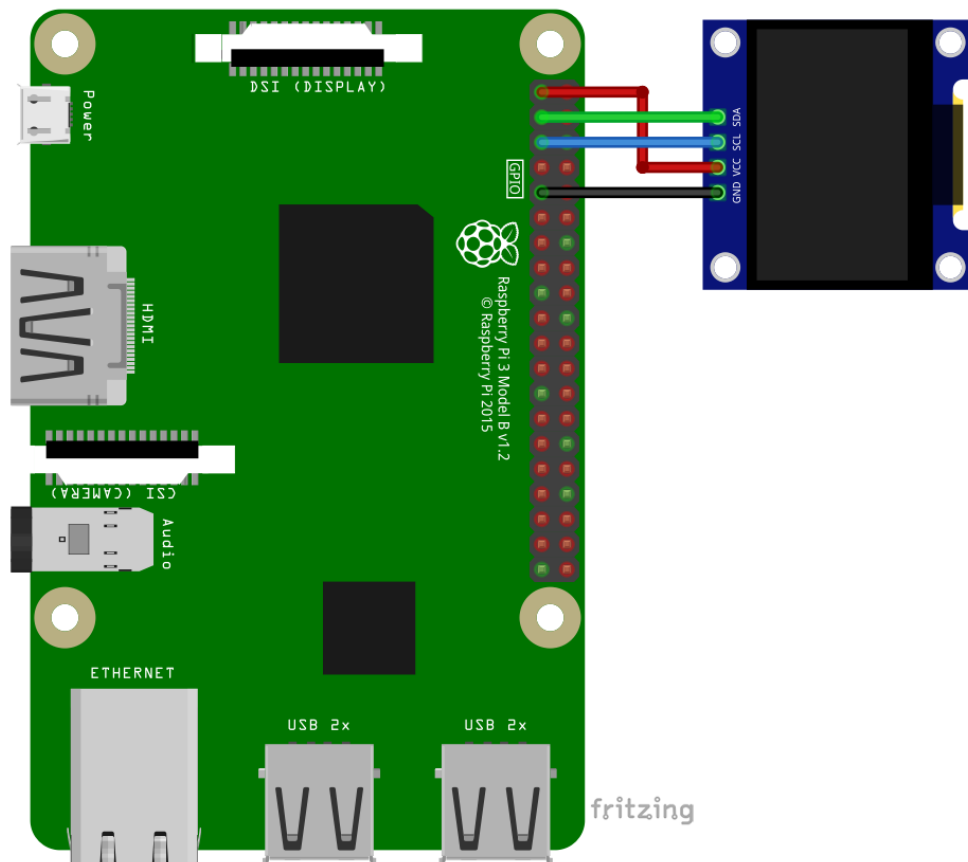
La funzione restituisce un valore, un numero intero che rappresenta la larghezza del testo (stringa).

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:



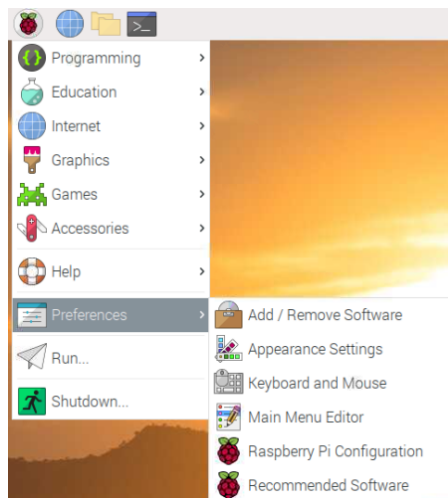
Az-Delivery

Pin schermo	Pin Raspberry Pi	Pin fisico No.	Colore filo
SDA	GPIO2	3	Filo verde
SCL	GPIO3	5	Filo blu
VCC	3V3	1	Filo rosso
GND	GND	9	Filo nero

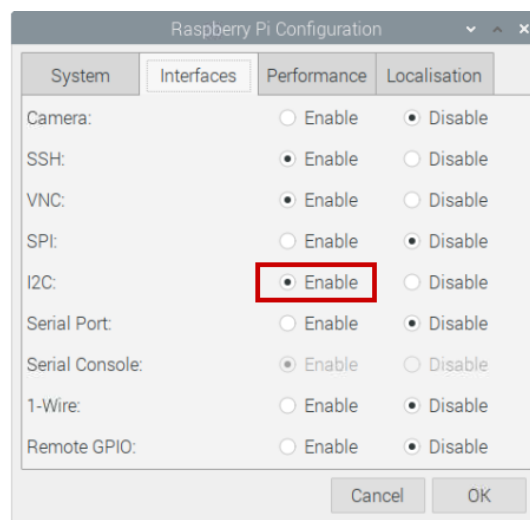
Abilitazione dell'interfaccia I2C

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 con il Raspberry Pi, si consiglia di scaricare e installare una libreria esterna. La libreria che verrà utilizzata si chiama *Adafruit_Python_SSD1306*.

Prima di installare la libreria principale, è necessario installare prima diversi strumenti e librerie.

Per prima cosa, è stato necessario installare la libreria *python-smbus* and *i2c-tools*.

Aprire il terminale ed eseguire i seguenti comandi:

```
sudo apt-get update
```

```
sudo apt-get install -y python-smbus i2c-tools
```

La seguente lista è un elenco di strumenti necessari che devono essere presenti sul sistema:

```
python3-dev, python-imaging, python3-pil, python3-pip,  
python3-setuptools, python3-rpi.gpio
```

Se gli strumenti della lista non sono presenti, possono essere installati eseguendo i seguenti comandi dalla finestra del terminale:

```
sudo apt install -y python3-dev python-imaging python3-  
pil python3-pip python3-setuptools python3-rpi.gpio
```

Az-Delivery

Per poter scaricare la libreria il git deve essere installato. Se il git non è presente nel sistema, l'installazione può essere effettuata con i seguenti comandi:

```
sudo apt install -y git
```

Il repository della libreria può essere clonato eseguendo il seguente comando:

```
git clone https://github.com/adafruit/Adafruit_Python_SSD1306.git
```

La directory deve essere cambiata in *Adafruit_Python_SSD1306*, eseguendo il seguente comando: `cd Adafruit_Python_SSD1306`

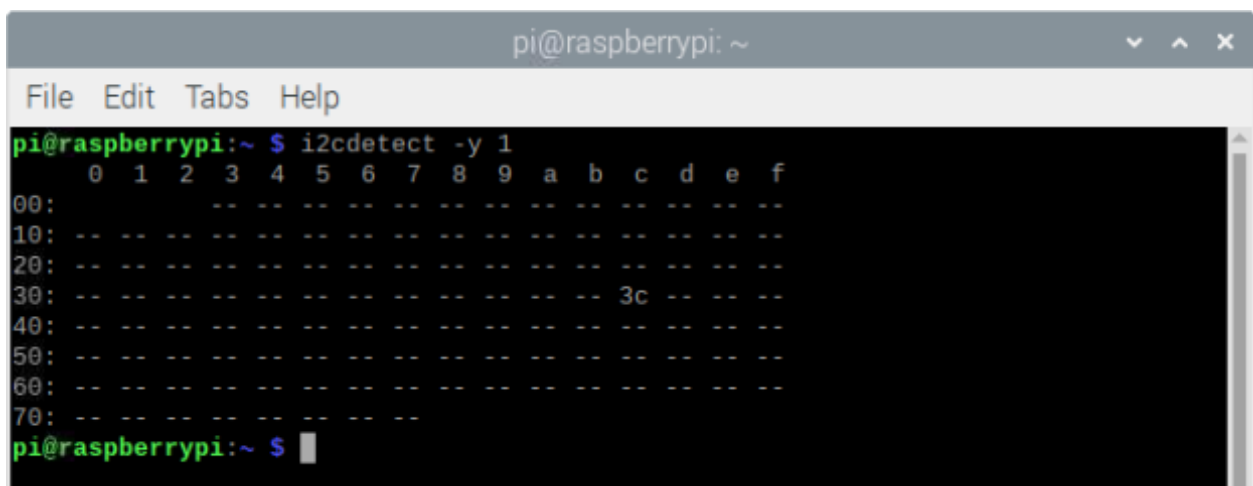
Quindi, la libreria verrà installata eseguendo il seguente comando:

```
sudo python3 setup.py install
```

Az-Delivery

Prima di utilizzare qualsiasi dispositivo collegato all'interfaccia I2C, l'indirizzo I2C deve essere rilevato prima. Per rilevare l'indirizzo I2C della schermata, nel terminale deve essere eseguito il seguente comando:
`i2cdetect -y 1`


Il risultato dovrebbe assomigliare all'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.

Se l'interfaccia I2C non è abilitata con il comando precedente, apparirà un errore come nell'immagine seguente:



```
pi@raspberrypi: ~/Scripts  
File Edit Tabs Help  
pi@raspberrypi:~/Scripts $ i2cdetect -y 1  
Error: Could not open file `/dev/i2c-1' or `/dev/i2c/1': No such file or directory  
pi@raspberrypi:~/Scripts $
```

Az-Delivery

Ci sono diversi esempi di script che vengono forniti con la libreria, navigare nella directory: */Adafruit_Python_SSD1306/examples*

eseguendo il seguente comando:

```
cd ~/Adafruit_Python_SSD1306/examples
```

Questa directory contiene diversi esempi di script, tra cui:

shapes.py,

image.py,

stats.py

e altri.

Focalizzatevi sullo script *shapes.py*. Per eseguire lo script, aprire il terminale nella directory in cui è stato salvato lo script ed eseguire il seguente comando:

```
python3 shapes.py
```

Az-Delivery

Script Python

```
import time
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

disp = Adafruit_SSD1306.SSD1306_128_64(rst=None)
disp.begin()
disp.clear()
disp.display()
image = Image.new('1', (disp.width, disp.height))
draw = ImageDraw.Draw(image)

print('[Press CTRL + C to end the script!]\n')
try:
    while True:
        draw.rectangle((0, 0, disp.width, disp.height),
                       outline=0, fill=0)

        padding = 2
        shape_width = 20
        top = padding
        bottom = disp.height - padding

        print('Drawing a ellipse')
        x = padding
        draw.ellipse((x, top, x + shape_width, bottom),
                    outline=255, fill=0)
        time.sleep(0.2)
```


Az-Delivery

```
# two tabs
print('Drawing a rectangle')
x += shape_width + padding
draw.rectangle((x, top, x + shape_width, bottom),
               outline=255, fill=0)
time.sleep(0.2)

print('Drawing a triangle')
x += shape_width + padding
draw.polygon([(x, bottom), (x + shape_width / 2, top),
              (x + shape_width, bottom)], outline=255, fill=0)
time.sleep(0.2)

print('Drawing two lines')
x += shape_width + padding
draw.line((x, bottom, x + shape_width, top), fill=255)
draw.line((x, top, x + shape_width, bottom), fill=255)
time.sleep(0.2)

print('Printing text')
x += shape_width + padding
my_font = ImageFont.load_default() # Load default font.
draw.text((x, top), 'AZ', font=my_font,
          fill=255)
draw.text((x, top + 20), 'DLVRY', font=my_font,
          fill=255)
time.sleep(0.2)

disp.image(image)
disp.display()
time.sleep(1)
```

Az-Delivery

```
# two tabs
print()
disp.clear()
disp.display()

except KeyboardInterrupt:
    print('\nScript end!')

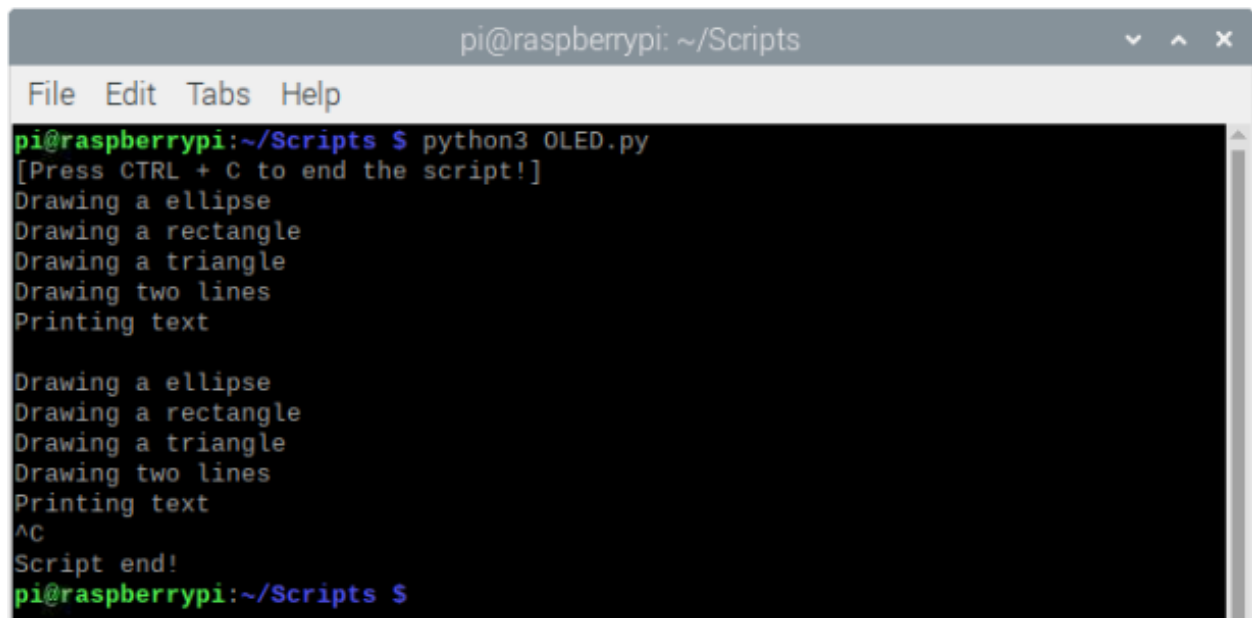
finally:
    disp.clear()
    disp.display()
```

Az-Delivery

Salvare lo script con il nome *OLED.py*. Per eseguire lo script, aprire il terminale nella directory in cui è stato salvato lo script ed eseguire il seguente comando:

python3 OLED.py

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



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 OLED.py
[Press CTRL + C to end the script!]
Drawing a ellipse
Drawing a rectangle
Drawing a triangle
Drawing two lines
Printing text

Drawing a ellipse
Drawing a rectangle
Drawing a triangle
Drawing two lines
Printing text
^C
Script end!
pi@raspberrypi:~/Scripts $
```

Per fermare lo script premere CTRL + C sulla tastiera.

Az-Delivery

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

Successivamente, l'oggetto chiamato `disp` viene creato con la seguente riga di codice:

```
disp = Adafruit_SSD1306.SSD1306_128_32(rst=None)
```

Dove viene usato *where* `rst=None`. Questo rappresenta il pin di reset, che lo schermo OLED da 0,91 pollici non ha. La libreria *Adafruit_SSD1306* può essere utilizzata per molte altre schermate OLED, per questo motivo esiste un'opzione per questo pin.

L'oggetto `disp` rappresenta lo schermo stesso e questo oggetto viene utilizzato per inviare comandi allo schermo.

Successivamente, l'oggetto `screen` viene inizializzato, il buffer dei dati del sinottico è stato cancellato. Dopo di che vengono create le immagini.

Per creare un'immagine, prima si deve creare un'immagine vuota con le dimensioni dello schermo. Si esegue con la seguente riga di codice:

```
image = Image.new('1', (disp.width, disp.height))
```

Successivamente, con questo oggetto immagine viene creato l'oggetto disegno, che viene utilizzato per disegnare forme: `draw = ImageDraw.Draw(image)`

In seguito, viene creato il blocco di codice *try-except-finally*. Nel blocco di codice *try* viene creato un loop indefinito (*while True:*). Nel blocco di codice indefinito c'è un algoritmo per controllare lo schermo.

Az-Delivery

Il blocco di codice `except` viene eseguito quando si preme CTRL + C sulla tastiera. Questo viene chiamato interruzione da tastiera e viene usato per terminare lo script. Quando si esegue questo blocco di codice, il messaggio *Script end!* Viene visualizzato nel terminale.

Il blocco di codice *finally* viene eseguito alla fine dell'esecuzione dello script. Viene utilizzato per cancellare il buffer di dati dello schermo e per disabilitare tutte le modalità di pin GPIO e/o interfacce utilizzate.

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 sono la posizione *X* e *Y* dell'angolo in alto a destra del rettangolo. Il terzo elemento è la larghezza del rettangolo e il quarto è l'altezza 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).

Az-Delivery

L'argomento del contorno rappresenta il colore del bordo della forma e l'argomento del riempimento rappresenta il colore della forma stessa. Poiché vengono utilizzati schermi OLED, i colori sono in bianco e nero, il nero - il pixel è spento, e il bianco - il pixel è acceso. Quando il valore di zero viene salvato nell'argomento contorno o riempimento, significa che si tratta di un colore nero. Quando viene salvato qualsiasi altro valore superiore a zero, ad esempio 255, questo rappresenta il colore bianco.

Per disegnare ellissi o cerchi, viene utilizzata 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 destra del rettangolo che contiene l'ellisse. Il terzo elemento è la larghezza del rettangolo e il quarto elemento è l'altezza del rettangolo. Se la larghezza è uguale all'altezza la forma che si disegna è il cerchio. 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 *poligon()*. 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 e rappresentano un punto d'angolo. Gli elementi nella tupla rappresentano la posizione X e Y del punto d'angolo della forma. Il numero di tuple nella lista è arbitrario, tre o più tuple, che dipende da quale forma viene disegnata. Il secondo argomento è l'argomento del contorno e il terzo argomento è l'argomento del riempimento.

Az-Delivery

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.

Per visualizzare il testo, viene utilizzata la funzione `text()`. La funzione accetta quattro argomenti e non restituisce alcun valore. Il primo argomento è una tupla di due elementi, che rappresenta le posizioni *X* e *Y* del cursore dove viene visualizzato il testo. Il secondo argomento rappresenta il testo stesso, (un valore di stringa). Il terzo argomento è l'argomento *font*, e il quarto argomento è l'argomento di riempimento. L'argomento *font* rappresenta il font usato. Per impostare il valore dell'argomento font si usa la seguente riga di codice con un font di libreria predefinito: `font = ImageFont.load_default()`

C'è un'opzione per utilizzare diversi tipi di carattere, ma non viene trattata in questo eBook.



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 e Raspberry Pi, 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>