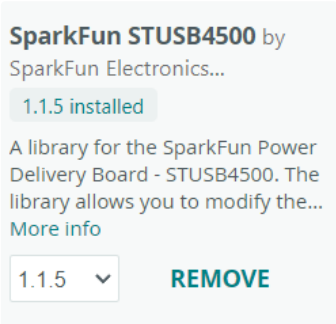
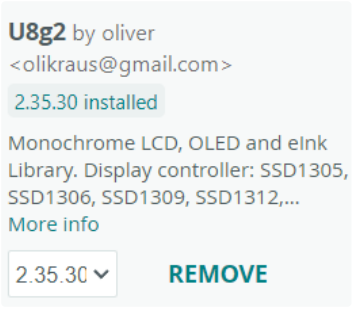
USB-PowerDelivery: Read

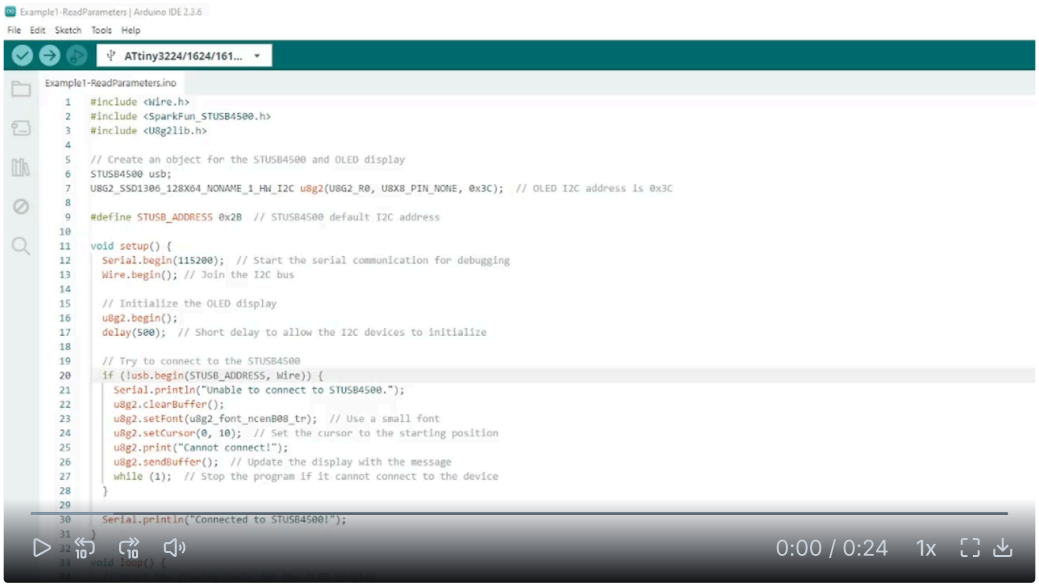
In this part of this project, a `STUSB4500` power delivery board was integrated with an `OLED` display to display the power settings of the `STUSB4500`. The `STUSB4500` is a `USB` power delivery controller, and it allows the configuration of multiple power data objects, which define the voltage and amperage that can be delivered through the `USB` port.

Used an `Arduino Nano` to communicate with the `STUSB4500` via `I2C` protocol. The `OLED` display, based on the `SSD1306` driver, also communicates over `I2C` to show the data retrieved from the `STUSB4500`.It's also necessary the `U8g2` library for the configuration of the display.





Once both libraries are installed, open the read example from the `STUSB4500` library (video below).

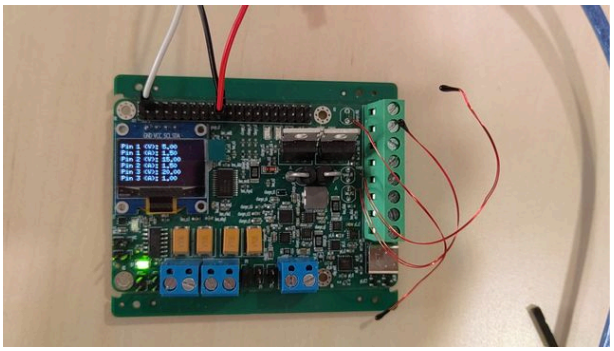


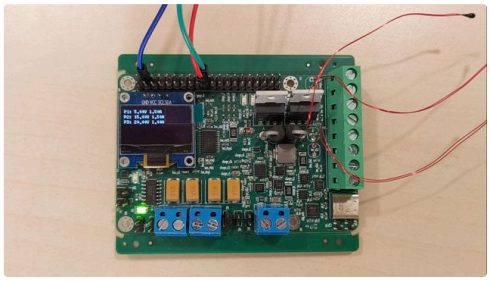
It's necessary to adapt the code to display voltage and amperage on the `OLED` screen using the library `U8g2` by making the following changes:

1. `STUSB4500` and `OLED` communication:
   - Use the `Wire` library to establish an `I2C` connection between the `Arduino` and both the `STUSB4500` and the `OLED` screen.
   - The `SparkFun STUSB4500` library provided functions to communicate with the `STUSB4500`, allowing us to read the configured `PDO` settings (voltage and current values).
2. `OLED` **display setup**:
   - The `U8g2` library was used to handle the `OLED` display, allowing us to draw text and update the screen.
   - Choose a small font (6x10) to display more information on the screen at once, given the limited space on a 128x64 `OLED` display.

3. **Power Data Objects (`PDOs`)**:
   - The program continuously reads the voltage and current values for three different `PDOs` defined on the `STUSB4500`.
   - Display these values on the `OLED` screen for each `PDO`, showing both voltage and current for each.
4. **Display update**:
   - The `firstPage()` and `nextPage()` functions were used to manage screen updates and ensure the display was properly refreshed.
   - Included a 1.5-second delay between updates to give users enough time to read the displayed values.

Options:





```ino
#include <Wire.h>
#include <SparkFun_STUSB4500.h>
#include <U8g2lib.h>

// Create an object for the STUSB4500 and OLED disp
STUSB4500 usb;
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, U

#define STUSB_ADDRESS 0x2B  // STUSB4500 default I2

void setup() {
  Serial.begin(115200);  // Start the serial commun
  Wire.begin(); // Join the I2C bus

  // Initialize the OLED display
  u8g2.begin();
  delay(500);  // Short delay to allow the I2C devi

  // Try to connect to the STUSB4500
  if (!usb.begin(STUSB_ADDRESS, Wire)) {
    Serial.println("Unable to connect to STUSB4500.
    u8g2.clearBuffer();
    u8g2.setFont(u8g2_font_ncenB08_tr);  // Use a s
    u8g2.setCursor(0, 10);  // Set the cursor to th
    u8g2.print("Cannot connect!");
    u8g2.sendBuffer();  // Update the display with
    while (1);  // Stop the program if it cannot co
  }

  Serial.println("Connected to STUSB4500!");
}

void loop() {
  // Start the drawing cycle for the OLED display
```

```ino
#include <Wire.h>
#include <SparkFun_STUSB4500.h>
#include <U8g2lib.h>

// OLED 128x32 via I2C (adjust if using a different
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /

STUSB4500 usb;

float volt1, curr1, volt2, curr2, volt3, curr3;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  delay(500);

  // Initialize OLED
  u8g2.begin();

  // Initialize STUSB4500 at I2C address 0x2B
  if(!usb.begin(0x2B, Wire)) {
    Serial.println("Unable to connect to STUSB4500.
    while(1);
  }

  Serial.println("Connected to STUSB4500.");

  // Read voltages and currents
  volt1 = usb.getVoltage(1);
  curr1 = usb.getCurrent(1);
  volt2 = usb.getVoltage(2);
  curr2 = usb.getCurrent(2);
  volt3 = usb.getVoltage(3);
  curr3 = usb.getCurrent(3);

  // Display on OLED
  u8g2.clearBuffer();
```

```
  u8g2.firstPage();  // Start the first page (neede
  do {
    // Clear the display before drawing new data
    u8g2.clearBuffer();

    // Use a small font to fit more text on the scr
    u8g2.setFont(u8g2_font_6x10_tr);  // Small font

    // Display voltage and current for PDO1 (Pin 1)
    u8g2.setCursor(0, 10);  // Set cursor to the fi
    u8g2.print("Pin 1 Voltage (V): ");
    u8g2.print(usb.getVoltage(1));  // Display the

    u8g2.setCursor(0, 20);  // Set cursor to the se
    u8g2.print("Pin 1 Current (A): ");
    u8g2.print(usb.getCurrent(1));  // Display the

    // Display voltage and current for PDO2 (Pin 2)
    u8g2.setCursor(0, 30);  // Set cursor to the th
    u8g2.print("Pin 2 Voltage (V): ");
    u8g2.print(usb.getVoltage(2));  // Display the

    u8g2.setCursor(0, 40);  // Set cursor to the fo
    u8g2.print("Pin 2 Current (A): ");
    u8g2.print(usb.getCurrent(2));  // Display the

    // Display voltage and current for PDO3 (Pin 3)
    u8g2.setCursor(0, 50);  // Set cursor to the fi
    u8g2.print("Pin 3 Voltage (V): ");
    u8g2.print(usb.getVoltage(3));  // Display the

    u8g2.setCursor(0, 60);  // Set cursor to the si
    u8g2.print("Pin 3 Current (A): ");
    u8g2.print(usb.getCurrent(3));  // Display the

    // Send the buffer to update the OLED screen wi
    u8g2.sendBuffer();

  } while (u8g2.nextPage());  // Proceed to the nex

  delay(1500);  // Wait for 1.5 seconds before upda
}

```

```
  u8g2.setFont(u8g2_font_5x8_tf);  // Smaller font
}

void loop() {
  u8g2.firstPage();  // Start a new page on the scr
  do {
    // Display values on the screen
    u8g2.setCursor(0, 10);
    u8g2.print("P1: ");
    u8g2.print(volt1);
    u8g2.print("V ");
    u8g2.print(curr1);
    u8g2.print("A");

    u8g2.setCursor(0, 20);
    u8g2.print("P2: ");
    u8g2.print(volt2);
    u8g2.print("V ");
    u8g2.print(curr2);
    u8g2.print("A");

    u8g2.setCursor(0, 30);
    u8g2.print("P3: ");
    u8g2.print(volt3);
    u8g2.print("V ");
    u8g2.print(curr3);
    u8g2.print("A");

    u8g2.sendBuffer();  // Send the content to the
  } while (u8g2.nextPage());  // Move to the next p
}
```