An Arduino-Based Handheld UVI meter

Chapter 20 in my 2019 book, *Exploring Your Environment with Arduino Microcontrollers*, discusses measuring the UV Index (UVI) using a Vishay VEML6075 two-band UVA and UVB sensor breakout board. That chapter even included a breadboard test layout for a UVI meter intended for outdoor measurements. It used an UNO, a TFT display that I happened to have on hand, and a 9 V battery for power – not at all a practical solution for a usable instrument!

This document describes a much more practical solution. It assumes that you already have some familiarity with Arduinos and the Integrated Development Environment (IDE). The project also serves as an introduction to using one of adafruit.com's "Feather" development boards. The project is easily modified for measuring other things, such as indoor or outdoor visible light intensity, simply by using one or more different sensors. The parts for the UVI meter include:

Adafruit Feather 32u4 Basic Proto board, ID 2771
12- and 16-pin headers, Adafruit header ID 2886 or similar
VEML6075 UV breakout board, SparkFun SEN-15089 or DFRobot SEN0303
IZOKEE 0.96″ monochrome OLED display or similar, from Amazon or other online sources
LiPo battery, 500 mAh (Adafruit ID 1578) or smaller, or similar
Mini breadboard (170 contacts), various online sources
ABS screw-lid project box, Hammond 1591 (4.4″×2.4″ ×1.2″) or similar
on/off pushbutton (*e.g.* adafruit.com ID 3870) or small SPST or SPDT toggle or slide switch
(*e.g.*, allelectronics.com CAT# MTS-4 or SSW-86)
#22 AWG solid wire, a few inches in various colors (red, black, yellow, blue)
Superglue, silicon caulk, or hot glue gun

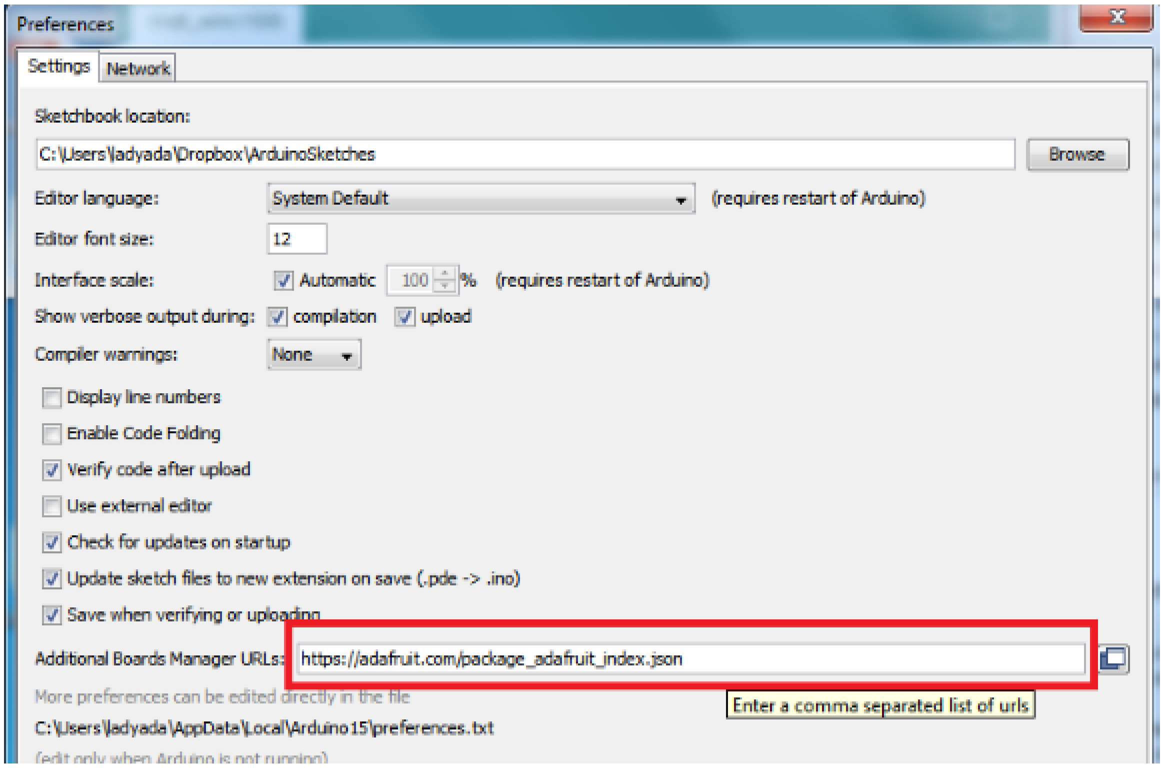
As of late 2020, Vishay discontinued production of the VEML6075 due to a reported "lack of demand," although at the time this document was being written breakout boards with this sensor were still available from the vendors listed above. Adafruit's VEML6075 board (the one used in Brooks' book) is no longer available. The VEML6070 UVA sensor, which does not provide even an estimate of the UV index, is also discontinued. The Si1145 sensor, available on a breakout board from Adafruit, provides an estimate of the UV index based on measurements of visible and IR radiation rather than on actual UV measurements, so its accuracy is questionable. Replacing the VEML6075 sensor with an Si1145 requires installing the relevant software library and rewriting the code using examples provided; this should not be a difficult task and, in fact, the Si1145 code will be simpler than the SparkFun code for its VEML6075 breakout board. As of this writing, I'm unaware of any direct replacement for the VEML6075.

The heart of this project is Adafruit's Feather 32u4 Basic Proto development board, which is ideal for battery-powered applications. It has built-in USB connectivity for programming through the Arduino IDE (unlike a Pro Mini, for example). But the significant advantage for a project like this is that it has an on-board LiPo battery charger that automatically switches back and forth between USB power when it's connected to a computer and battery power when it's used offline. Under USB power the battery charges automatically so you don't need to remove it from the system and use a separate charger.

For the data display, I used an inexpensive IZOKEE I2C 128×64 pixel 0.96″ monochrome white-on-black OLED – $13 through Amazon for a set of three at the time I wrote this document. Software libraries for these small SSD1306 OLEDs are widely available. Graphics capabilities are available, but for this project I used just a subset of a library to display text and numerical values. The code shown below may not work for other OLEDs without changing the board setup; code examples provided with software libraries usually include a long list of possible board specifications.

Using the 32u4 with the Arduino IDE

You must install the required files before you can use the 32u4 Basic Proto board, as it's not part of a standard Arduino IDE installation. First, go to File→Preferences and add the URL shown below. If there are already links in the Additional Boards Manager URLs, add the one shown after a comma at the end of the list or followed by a comma at the beginning of the list.


Preferences

Settings | Network

Sketchbook location: C:\Users\adyada\Dropbox\ArduinoSketches [Browse]

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ✓ compilation ✓ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

✓ Verify code after upload

☐ Use external editor

✓ Check for updates on startup

✓ Update sketch files to new extension on save (.pde -> .ino)

✓ Save when verifying or uploading

Additional Boards Manager URLs: https://adafruit.com/package_adafruit_index.json

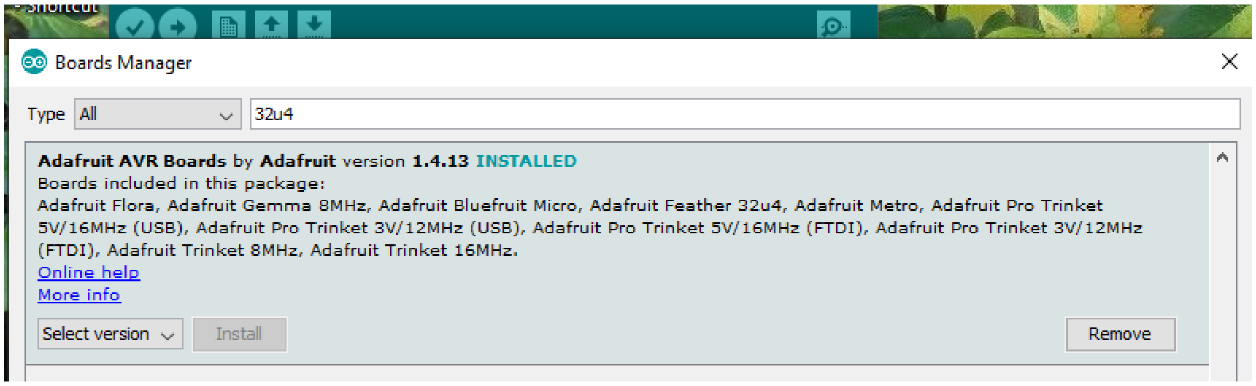
Enter a comma separated list of urls

More preferences can be edited directly in the file

C:\Users\adyada\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

Then, go to Tools→Boards→Board Manager. Type in 32u4 and install the package. On my system, the files are already installed.


Boards Manager

Type All 32u4

Adafruit AVR Boards by **Adafruit** version **1.4.13** INSTALLED

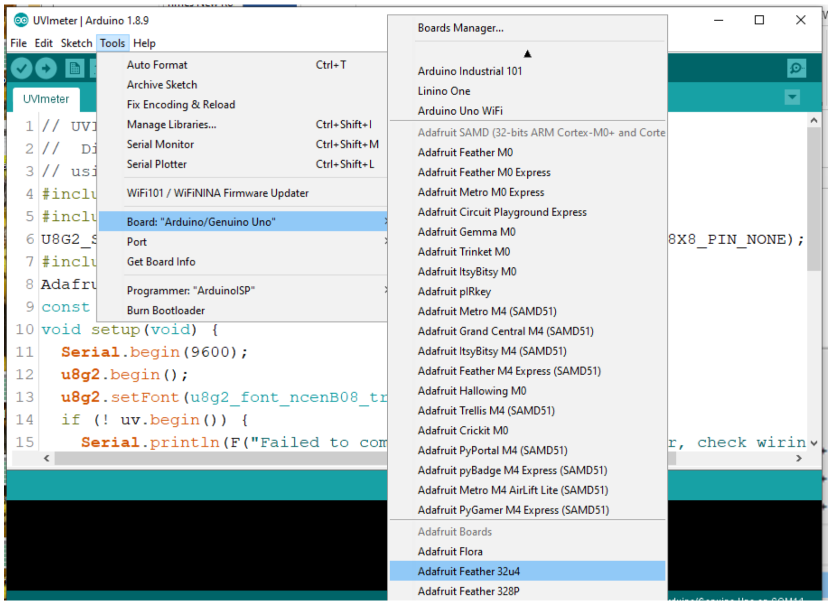
Boards included in this package:

Adafruit Flora, Adafruit Gemma 8MHz, Adafruit Bluefruit Micro, Adafruit Feather 32u4, Adafruit Metro, Adafruit Pro Trinket 5V/16MHz (USB), Adafruit Pro Trinket 3V/12MHz (USB), Adafruit Pro Trinket 5V/16MHz (FTDI), Adafruit Pro Trinket 3V/12MHz (FTDI), Adafruit Trinket 8MHz, Adafruit Trinket 16MHz.

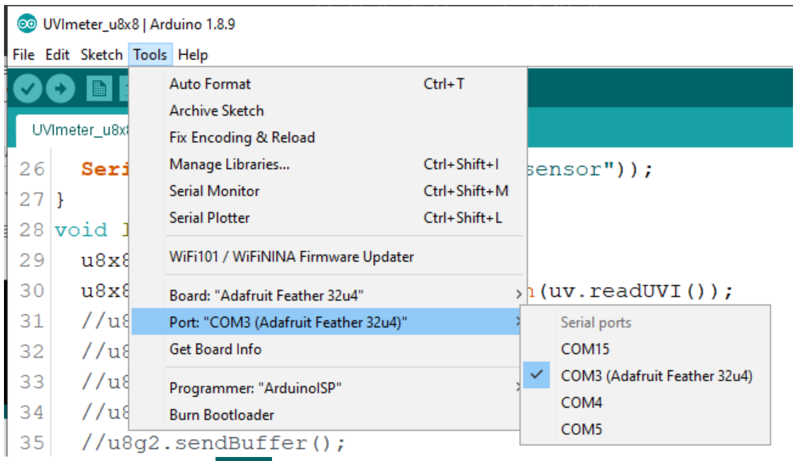
Online help

More info

When you write your sketch, scroll down through the supported boards until you find Adafruit Feather 32u4. Do not compile your sketch with any other board choice! (If you try to upload a sketch compiled for another board, Adafruit says you may "brick" the Feather 32u4.) The image below shows how to change the board from an UNO, used for a previous project, to the Feather 32u4.


UVImeter | Arduino 1.8.9
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
Get Board Info
Programmer: "ArduinoISP"
Burn Bootloader
Boards Manager...
Arduino Industrial 101
Linino One
Arduino Uno WiFi
Adafruit SAMD (32-bits ARM Cortex-M0+ and Corte
Adafruit Feather M0
Adafruit Feather M0 Express
Adafruit Metro M0 Express
Adafruit Circuit Playground Express
Adafruit Gemma M0
Adafruit Trinket M0
Adafruit ItsyBitsy M0
Adafruit pIRkey
Adafruit Metro M4 (SAMD51)
Adafruit Grand Central M4 (SAMD51)
Adafruit ItsyBitsy M4 (SAMD51)
Adafruit Feather M4 Express (SAMD51)
Adafruit Hallowing M0
Adafruit Trellis M4 (SAMD51)
Adafruit Crickit M0
Adafruit PyPortal M4 (SAMD51)
Adafruit pyBadge M4 Express (SAMD51)
Adafruit Metro M4 AirLift Lite (SAMD51)
Adafruit PyGamer M4 Express (SAMD51)
Adafruit Boards
Adafruit Flora
Adafruit Feather 32u4
Adafruit Feather 328P

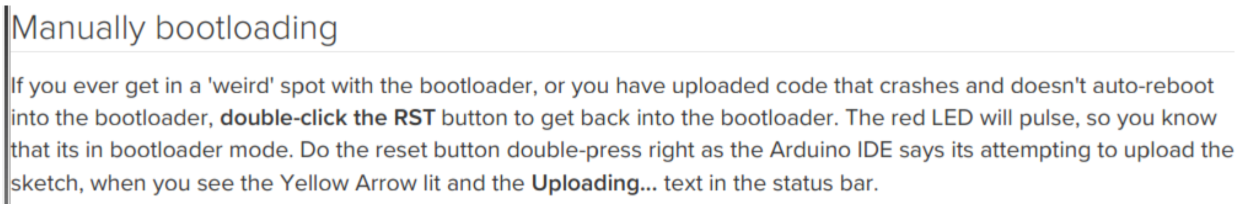
To check that the board is working, you can upload a "blink" sketch. Make sure you've chosen the 32u4 board from the list of available boards. Connect the USB cable and select an appropriate COM port, which depends on your computer's operating system; this image is from my Windows 10 desktop.


UVImeter_u8x8 | Arduino 1.8.9
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: "Adafruit Feather 32u4"
Port: "COM3 (Adafruit Feather 32u4)"
Get Board Info
Programmer: "ArduinoISP"
Burn Bootloader
Serial ports
COM15
✓ COM3 (Adafruit Feather 32u4)
COM4
COM5

Then click on the button to compile and upload the sketch.

```
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);
}
void loop() {
  digitalWrite(13,HIGH);
  Serial.println("ON");
  delay(2000);
  digitalWrite(13,LOW);
  Serial.println("OFF");
  delay(1000);
}
```

A problem with Adafruit's Feather boards (and not just the 32u4) is that they can be ***very*** finicky about uploading sketches even when the correct board and port have been selected – a problem never seen with boards such as the UNO. This appears to be especially true if the sketch currently in memory is trying to write to the COM port window (*i.e.,* with `Serial.print()` commands). Here's what Adafruit's documentation says about their Feather boards:



Manually bootloading

If you ever get in a 'weird' spot with the bootloader, or you have uploaded code that crashes and doesn't auto-reboot into the bootloader, **double-click the RST** button to get back into the bootloader. The red LED will pulse, so you know that its in bootloader mode. Do the reset button double-press right as the Arduino IDE says its attempting to upload the sketch, when you see the Yellow Arrow lit and the **Uploading...** text in the status bar.

On the 32u4 board you must press the (very tiny!) reset button twice, rapidly, at just the appropriate time. In practice, I've found that it's often necessary to repeat this operation *several* times, restarting the entire compile/upload process, to get the timing right before a new sketch will upload. It's frustrating, but eventually it will work.

I've also found that some sketches requiring access to connected hardware won't reliably compile at all. One way to get around these problems, for reasons I don't understand, is to disconnect power to all peripherals from the board before compiling and uploading the sketch; this is easy to do with breadboard layouts. The code won't run, of course, if the peripherals – sensors, display devices, etc. – that the code expects to see aren't there, but once uploaded, you can disconnect the USB cable, re-connect everything, and then connect the USB cable or battery.

Designing the meter

Figure 1 shows a breadboard layout for testing the UVI meter with a SparkFun VEML6075 breakout board, running from a LiPo battery. (As noted above, the 32u4 board automatically switches power between a USB cable and a battery when it's connected.) Following Adafruit's documentation, do ***not*** connect any battery other than a 3.7-4.2 V LiPo to the board! (You might be tempted because, for example, Adafruit sells a 3-battery AAA holder with an on/off switch and 2-pin JST connector just like the one found on a LiPo battery.)

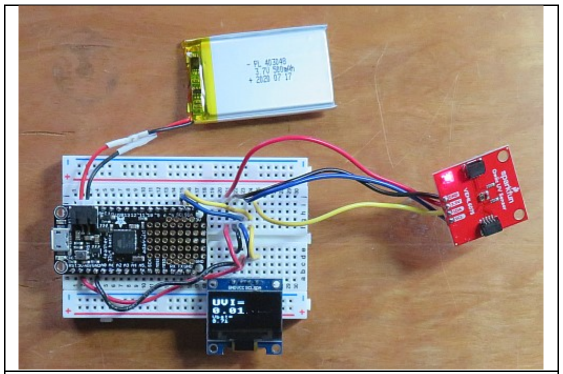


Figure 1. Breadboard layout for UVI meter.

Also, and **very importantly(!)**, note that the red and black wires from this LiPo battery have been cut, reversed, and re-soldered. To its credit, Adafruit warns specifically against using other than their own LiPo batteries if for no other reason than the positive and negative leads from other batteries may be reversed! By design, JST connectors can be mated in only one orientation. The positive lead, conventionally red, *must* be the lead closest to the end of the board with the USB connector, as shown. The wires on the "offshore" 500 mAh LiPo battery I used for this project were, in fact, reversed for the wiring on the 3u4. (I checked polarity with a voltmeter and a cable that terminates the JST connections with jumper wires for use with a breadboard. You should do the same if you use a non-Adafruit LiPo battery!)

By the way, although it's true that powering this layout with a USB cable will simultaneously charge a connected battery, this can take a long time! For this typically low-power project layout, the LiPo battery I used started out at 3.79 V (near the "nominal" voltage for these batteries). It took almost 3 hours to "top off" at 4.2 V. The fact that the 32u4 isn't a particularly fast LiPo charger isn't necessarily a problem, but just something to keep in mind. The project will continue to run with a battery voltage as low as roughly 3.5 V or so. A fully charged 500 mAh battery will run the project continuously (not its intended mode of operation) for *several* hours (I haven't tested the limits), even with the extraneous "power on" LED on the SparkFun VEML6075 board.

For mounting later in a project box, I soldered four wire connections to the VEML6075 board rather than using a Qwiic connector; the two black rectangles on the board are for Qwiic cables.[1] There is a red LED power-on light in the upper left-hand corner.

Hookup to the Feather board is straightforward with the I2C UV sensor and OLED devices. The 32u4 is a 3.3 V board. The SparkFun VEML6075 board specifies a 3.3 V operating voltage and the DFRobot version specifies 3.3-5 V. (Many other Arduino boards like the 5 V UNO have a 3.3 V power output pin.)

Finally, there are some software matters to deal with. The code for this project uses the U8x8lib text-only subset of the U8g2lib OLED-compatible graphics library, as shown in Figure 2. The easiest way to install this library is to select Sketch→Include Library→Manage Libraries from the IDE and type u8g2 in the search box.

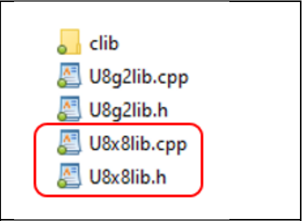
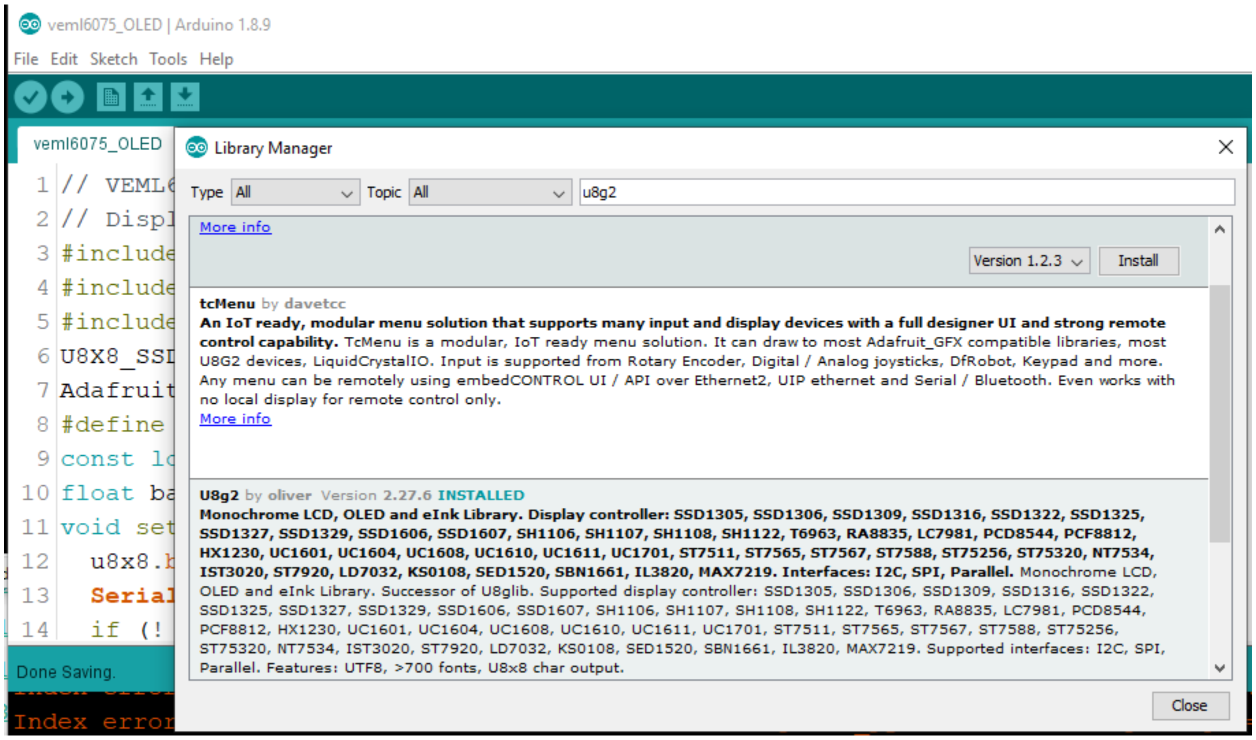


Figure 2. Libraries for the UVI meter's text-only OLED display.

The default 8×8 font is pretty small, so a 16×16 pixel font (no more than 8 characters per line) will be used to display the UVI value. Then (just to show how to do it) the code will switch back to an 8×8 font (no more than 16 characters per line) to display the LiPo battery voltage when it's connected, to let you know when the battery needs recharging by switching to power from a USB source.



```
// VEML6
// Displ
#include
#include
#include
U8X8_SSI
Adafruit
#define
const lo
float ba
void set
  u8x8.b
  Serial
  if (!
```

Done Saving.

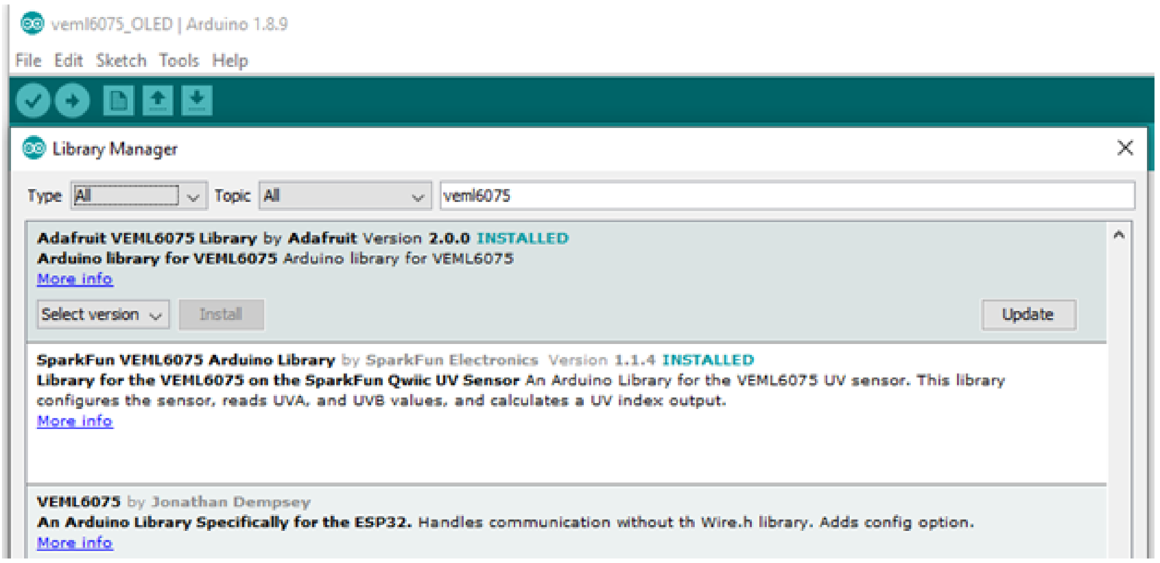
Index error

tcMenu by davetcc
An IoT ready, modular menu solution that supports many input and display devices with a full designer UI and strong remote control capability. TcMenu is a modular, IoT ready menu solution. It can draw to most Adafruit_GFX compatible libraries, most U8G2 devices, LiquidCrystalIO. Input is supported from Rotary Encoder, Digital / Analog joysticks, DfRobot, Keypad and more. Any menu can be remotely using embedCONTROL UI / API over Ethernet2, UIP ethernet and Serial / Bluetooth. Even works with no local display for remote control only.
More info

U8g2 by oliver Version 2.27.6 **INSTALLED**
Monochrome LCD, OLED and eInk Library. Display controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1322, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1701, ST7511, ST7565, ST7567, ST7588, ST75256, ST75320, NT7534, IST3020, ST7920, LD7032, KS0108, SED1520, SBN1661, IL3820, MAX7219. Interfaces: I2C, SPI, Parallel. Monochrome LCD, OLED and eInk Library. Successor of U8glib. Supported display controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1322, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1701, ST7511, ST7565, ST7567, ST7588, ST75256, ST75320, NT7534, IST3020, ST7920, LD7032, KS0108, SED1520, SBN1661, IL3820, MAX7219. Supported interfaces: I2C, SPI, Parallel. Features: UTF8, >700 fonts, U8x8 char output.

You will also need to install a VEML6075 software library for whichever board you use. (Although the sensor itself is the same on all the boards, the library written by one vendor won't work with a board from another vendor.) At the time this document was being written, the search from the IDE found libraries for Adafruit's discontinued VEML6075 board and SparkFun's VEML6075 board, which is what I used for this project. If you use DFRobot's VEML6075 board, you will have to download and install their library manually. Shut down your IDE, download the library, unzip it, and copy its folder into your Arduino's libraries folder.

[1] The SDA and SCL I2C wire colors on SparkFun's Qwiic connect cables are different from the colors I use for my projects, yellow for SDA and blue for SCL, which I chose long before SparkFun developed their Qwiic system.



Adafruit VEML6075 Library by **Adafruit** Version **2.0.0** INSTALLED
Arduino library for VEML6075 Arduino library for VEML6075
More info

SparkFun VEML6075 Arduino Library by SparkFun Electronics Version 1.1.4 INSTALLED
Library for the VEML6075 on the SparkFun Qwiic UV Sensor An Arduino Library for the VEML6075 UV sensor. This library configures the sensor, reads UVA, and UVB values, and calculates a UV index output.
More info

VEML6075 by Jonathan Dempsey
An Arduino Library Specifically for the ESP32. Handles communication without th Wire.h library. Adds config option.
More info

Here's the code for this project. If you use an OLED from a source other than IZOKEE you may need to change the board definition, the `U8X8…` line in the code.

```
// VEML6075_SparkFun_OLED.ino, D. Brooks, November 2020
// Displays SparkFun VEML6075 UV Index on 128x64 px IZOKEE 0.96" OLED.
// For explanation of preset values see
// https://cdn.sparkfun.com/assets/3/9/d/4/1/designingveml6075.pdf
// IMPORTANT NOTE: As of late 2020, Vishay's VEML6075 is discontinued,
// but SparkFun's board is still available.
#include <Wire.h>
#include "U8x8lib.h"
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
#include <SparkFun_VEML6075_Arduino_Library.h>
VEML6075 uv; // Create a VEML6075 object
#define VBATPIN A9 // Pick up LiPo voltage from pin 9.
const long int delayTime=2000L;
float batV;
void setup(void) {
  Wire.begin();
  u8x8.begin();
  Serial.begin(9600);
  if (uv.begin() == false) {
    Serial.println("Unable to communicate with VEML6075.");
    while (1) ;
  }
  uv.setIntegrationTime(VEML6075::IT_100MS);
  uv.setHighDynamic(VEML6075::DYNAMIC_NORMAL);
}
void loop() {
  u8x8.setFont(u8x8_font_px437wyse700a_2x2_r);
  u8x8.setCursor(0,1);
  u8x8.println(" UVI="); u8x8.print(' '); u8x8.println(getUVI());
  u8x8.setFont(u8x8_font_chroma48medium8_r);
  batV=3.3*(analogRead(VBATPIN)*2)/1023;  // See Adafruit docs.
  u8x8.print(" Vbat = ");u8x8.println(batV);
  delay(delayTime);
}
float getUVI() {
// Calibration constants (lab calibration bypassed):
const float CAL_ALPHA_VIS = 1.0; // UVA / UVAgolden
const float CAL_BETA_VIS = 1.0;  // UVB / UVBgolden
const float CAL_GAMMA_IR = 1.0;  // UVcomp1 / UVcomp1golden
const float CAL_DELTA_IR = 1.0;  // UVcomp2 / UVcomp2golden
const float UVA_RESPONSIVITY = 0.00110; // UVAresponsivity
const float UVB_RESPONSIVITY = 0.00125; // UVBresponsivity
// UV coefficients:
const float UVA_VIS_COEF_A = 2.22; // a
const float UVA_IR_COEF_B = 1.33;  // b
const float UVB_VIS_COEF_C = 2.95; // c
const float UVB_IR_COEF_D = 1.75;  // d
uint16_t rawA, rawB, visibleComp, irComp;
float uviaCalc, uvibCalc, uvia, uvib, uvi;
  // Read raw and compensation data from the sensor
  rawA = uv.rawUva();
  rawB = uv.rawUvb();
  visibleComp = uv.visibleCompensation();
  irComp = uv.irCompensation();
  // Calculate the simple UVIA and UVIB to calculate the UVI signal.
  uviaCalc=(float)rawA-((UVA_VIS_COEF_A*CAL_ALPHA_VIS*visibleComp)/CAL_GAMMA_IR);
  uviaCalc-=((UVA_IR_COEF_B*CAL_ALPHA_VIS*irComp)/CAL_DELTA_IR);
  uvibCalc=(float)rawB-((UVB_VIS_COEF_C*CAL_BETA_VIS*visibleComp)/CAL_GAMMA_IR);
  uvibCalc-=((UVB_IR_COEF_D*CAL_BETA_VIS*irComp)/CAL_DELTA_IR);
  // Convert raw UVIA and UVIB to values scaled by the sensor responsivity
  uvia = uviaCalc * (1.0 / CAL_ALPHA_VIS) * UVA_RESPONSIVITY;
  uvib = uvibCalc * (1.0 / CAL_BETA_VIS) * UVB_RESPONSIVITY;
  // Use UVIA and UVIB to calculate the average UVI:
  return uvi = (uvia + uvib) / 2.0;
}
```

You can find code examples with more information about the preset values used to interpret raw data and access other VEML6075 features in the SparkFun software library.

Building the meter

Figure 3 shows the completed meter, built in an ABS project box from allelectronics.com, CAT# MB-132, The top image in the upper left-hand cell of the table shows some headers soldered onto the 32u4 board. The upper and lower headers use the 3.3V and ground buses provided on the board for connecting power to other devices. The pins for each of the other two headers are soldered together on the underside of the prototyping area of the board, to provide connection points for I2C devices.

The other image in the upper left-hand table cell shows connections inside the project box. The LiPo battery is held in place with a dot of hot glue and its positive lead goes through the small toggle switch.

The front of the box, roughly 2"×4" (5cm×10cm), is shown in the upper right-hand cell. (The blue plastic rectangle holding the OLED display, attached to the project box with superglue, is there only due to the fact that I had to cover over a cutout for that display that I made in the wrong place.) The OLED is attached to the back of the plastic rectangle with hot glue.

The other two cells show the side of the box, with the switch and USB port, and the SparkFun VEML6075 on the top, with its four leads inserted through a slot cut in the top of the box and fastened to the project box with hot glue. The two small black rectangles on the sensor are for chaining mutitiple I2C devices through SparkFun's Qwiic connecting cables – not needed for this project. With the sensor at right angles to the display, the UV readings can be taken pointed up at the sky or directly at the sun while the display remains in shadow, to make it easier to read.

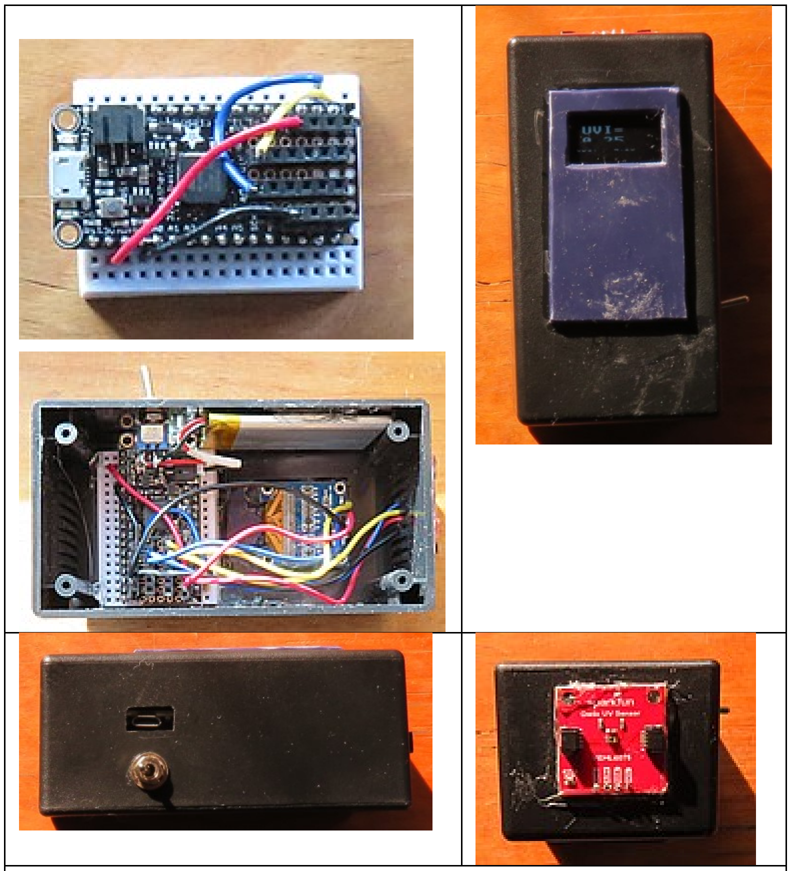


Figure 3. Completed UVI project.

How Does the U.S. EPA Calculate the UV Index?

It's important to understand that the EPA's reported UV index, UVI, for a particular day and location is based primarily on mathematical models, calculated at local solar noon, rather than on-site measurements. That model takes into account solar elevation angle, Earth-Sun distance, stratospheric ozone, cloud conditions, air pollutants, surface albedo, and surface elevation. All these factors determine the total amount of UV radiation reaching Earth's surface. The amount of UV radiation as a function of wavelength is then weighted by the sensitivity of Caucasian skin over those wavelengths. This weighting is called the McKinlay-Diffey erythemal action spectrum. The weighted incident UV radiation reaching Earth's surface, typically around 250 mW/m^2 under a cloud-free summer sky at midday in temperate latitudes, is then (arbitrarily) divided by 25 to give the UVI – 10 when the weighted UV radiation is 250 mW/m^2.

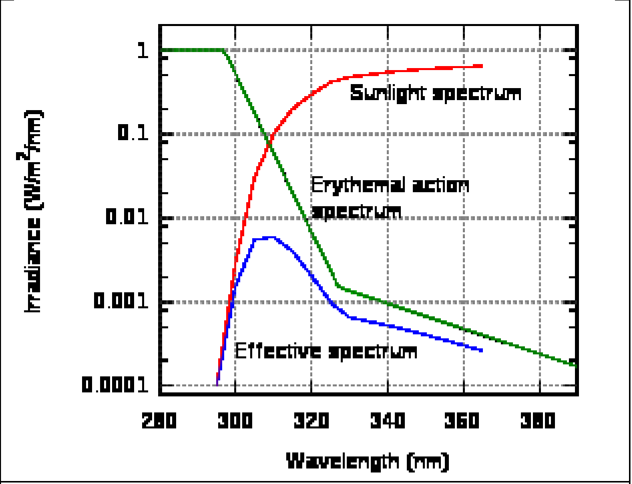


Figure 4. McKinlay-Diffey weighted UV action spectrum.

Thus, to accurately reproduce the calculations required for calculating the UVI, it's necessary to measure UV radiation across a wide range of wavelengths. Inexpensive sensors like the VEML6075 (~$7 on a breakout board) use two or even one UV sensors with a relatively small range of wavelength responsivity – see Figure 5 – and model the sensor's response to the rest of the erythemal action spectrum. So, it's necessary to view the results from such sensors with some skepticism. Having said that, even if these inexpensive devices don't produce UVI values that agree with EPA's values, they are at least based on measurements rather than models and will still produce interesting and very usable results.

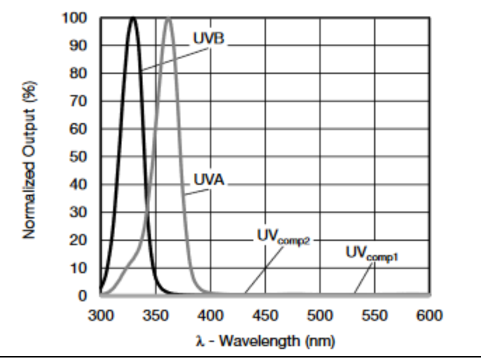


Figure 5. Wavelength response of VEML6075 UV sensors.

Some resources:
https://www.epa.gov/sunsafety/calculating-uv-index-0
https://en.wikipedia.org/wiki/Ultraviolet_index
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5086780/