AdafruitBundle Documentation

Release 1.0

Adabot

Feb 22, 2020

Contents

This repo bundles a bunch of useful CircuitPython libraries into an easy to download zip file. CircuitPython boards can ship with the contents of the zip to make it easy to provide a lot of libraries by default.

CHAPTER 1

Use

To use the bundle download the zip (not source zip) from the latest release, unzip it and copy over the subfolders, such as `lib`, into the root of your CircuitPython device. Make sure to indicate that it should be merged with the existing folder when it exists.

1.1 CPython

DO NOT use this to install libraries on a Linux computer, such as the Raspberry Pi, with regular Python (aka CPython). Instead, use the python3 version of `pip` to install the libraries you want to use. It will automatically install dependencies for you. For example:

```
pip3 install adafruit-circuitpython-lis3dh
```

CHAPTER 2

Development

After you clone this repository you must run `git submodule init` and then `git submodule update`.

2.1 Updating libraries

To update the libraries run `update-submodules.sh`. The script will fetch the latest code and update to the newest tag (not master).

To find libraries with commits that haven't been included in a release do:

```
git submodule foreach "git log --oneline HEAD...origin/master"
```

2.2 Adding a library

Determine the best location within `libraries` (`libraries/drivers/` or `libraries/helpers/`)for the new library and then run:

```
git submodule add <git url> libraries/<target directory>
```

The target directory should omit any CircuitPython specific prefixes such as `adafruit-circuitpython` to simplify the listing.

2.3 Removing a library

Only do this if you are replacing the module with an equivalent:

```
git submodule deinit libraries/<target directory>
git rm libraries/<target directory>
```

2.4 Building the bundle

To build this bundle locally you'll need to install the circuitpython-build-tools package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-bundle --library_
↪location libraries --library_depth 2
```

CHAPTER 3

Table of Contents

3.1 Adafruit Sponsored Libraries and Drivers on GitHub

These are libraries and drivers available in separate GitHub repos. They are designed for use with CircuitPython and may or may not work with MicroPython.

3.1.1 Foundational

These libraries provide critical functionality to many of the drivers below. It is recommended to always have them installed onto the CircuitPython file system in the `lib/` directory. Some drivers may not work without them.

3.1.2 Board-specific Helpers

These libraries tie lower-level libraries together to provide an easy, out-of-box experience for specific boards.

3.1.3 Helper Libraries

These libraries build on top of the low level APIs to simplify common tasks.

3.1.3.1 LED Helpers

Helpers for animating LEDs.

3.1.3.2 User Interface and GFX Helpers

Helpers for building graphical interfaces using the displayio core module and framebuf code module (framebuf is deprecated).

3.1.3.3 Motor Helpers

Helpers for driving motors, servos, and steppers.

3.1.3.4 Internet of Things Web Service Helpers

Helpers for connecting with hosted and self-hosted internet-of-things web services.

3.1.3.5 Internet/Internet-of-Things Helpers

Helpers for interfacing with the internet, including IoT protocols.

3.1.3.6 Bluetooth Low Energy Helpers

Helpers for Bluetooth Low Energy (BLE).

3.1.3.7 LoRa Wireless Helpers

Helpers for wireless communication via LoRa.

3.1.3.8 Cryptography Helpers

Helpers for secure communication.

3.1.3.9 CPython-module Helpers

Pure-Python implementations of standard CPython libraries. Some of these modules may have a CircuitPython Core API implementation too.

3.1.3.10 Audio Helpers

Music, noisemakers, and more.

3.1.3.11 Miscellaneous Helpers

3.1.4 Blinky

Multi-color LED drivers.

3.1.5 Displays

Drivers used to display information. Either pixel or segment based.

Pixel based displays are implemented in two different ways. The original method called "framebuf" uses a traditional frame buffer model where all pixels are stored in the microcontroller's ram. The newer method called "displayio" generates the pixels on the fly and relies on the display's ram to store the final pixels. "displayio" drivers will also work with CircuitPython to display error messages and other output to the display when the user code is not using it.

The "displayio" drivers are recommended.

3.1.5.1 Color TFT-LCD

3.1.5.2 OLED

3.1.5.3 E-Paper / E-Ink

3.1.5.4 Other

3.1.6 Real-time clocks

Chips that keep current calendar time with a backup battery. The current date and time is available through `datetime`.

3.1.7 Motion Sensors

Motion relating sensing including `acceleration`, `magnetic`, `gyro`, and `orientation`.

3.1.8 Environmental Sensors

Sense attributes of the environment including `temperature`, `relative_humidity`, `pressure`, equivalent carbon dioxide (`eco2` / `eCO2`), and total volatile organic compounds (`tvoc` / `TVOC`).

3.1.9 Light Sensors

These sensors detect light related attributes such as `color`, `light` (unit-less), and `lux` (light in SI lux).

3.1.10 Distance Sensors

These sensors measure the `distance` to another object and may also measure light level (`light` and `lux`).

3.1.11 Radio

These chips communicate to other's over radio.

3.1.12 IO Expansion

These provide functionality similar to `analogio`, `digitalio`, `pulseio`, and `touchio`.

3.1.13 Miscellaneous

CHAPTER 4

Indices and tables

- genindex
- modindex
- search