PewPew Documentation

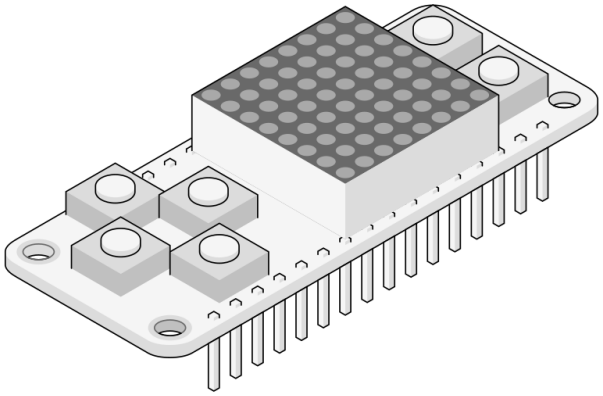
Release 1.0

Radomir Dopieralski

Jul 20, 2018

Contents

PewPew is a FeatherWing (an add-on board for the Adafruit Feather microcontroller boards) that lets you make simple games.



CHAPTER 1

Hardware

1.1 PewPew Lite FeatherWing

PewPew Lite is a shield for the Adafruit Feather series of development boards (a FeatherWing, as they call them). You put it on top of one of their boards, and then program that board (preferably in CircuitPython) to create games and other interactive programs.

Specification	
Display	8×8 4-color LED matrix
Input	6 buttons
Sound	no sound
Interface	2-wire I^2C at address 0x70
Controller	Holtek HT16K33
Battery	Optional 3.7V 100mAh LiPO (not included)

The shield contains six buttons and an eight-by-eight bi-color LED display, capable of displaying four different colors (black, green, red and yellow). It also includes a HT16K33 chip that handles the display and the buttons, so that you can focus on just writing the code for your game. Optionally, you can add a LiPO battery to it, to make the whole thing into a portable handheld game console. The shield includes a power switch for that battery.

1.2 Open Development

This hardware is developed in the open, and all the designs and related materials are publicly available under a permissive license. You are free to inspect how it is build, build your own, improve and extend the design, and even sell your own versions.

The materials are available in the project's repository.

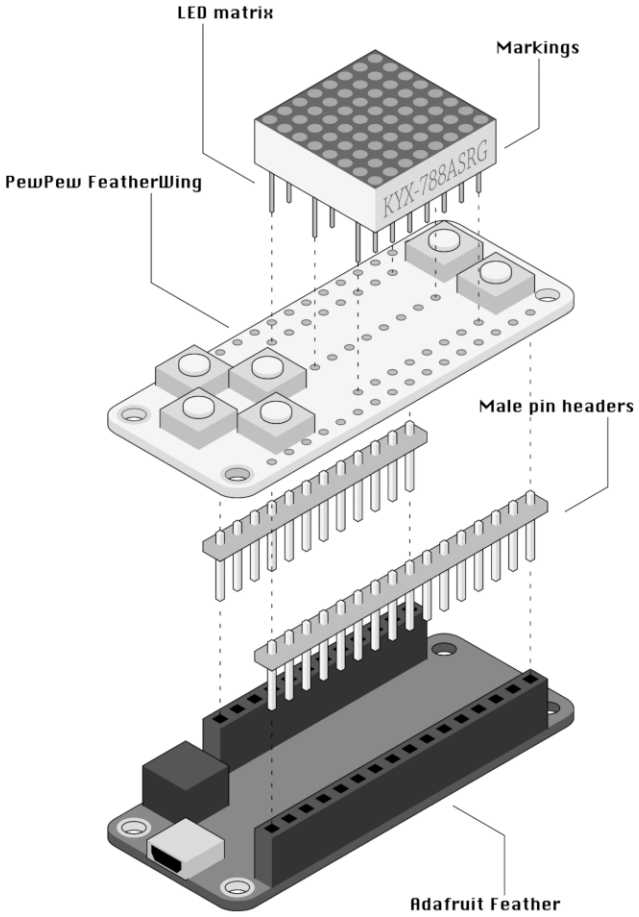
CHAPTER 2

Assembly and Setup

The PewPew FeatherWing comes without the pin headers and LED matrix attached, so that some simple soldering is required before using it.

2.1 Basic Assembly

The simplest way to assemble the PewPew board is to use an Adafruit Feather board that supports CircuitPython (the Feather M0 Express works great for this) with female pin headers, and to solder the male pin headers onto the PewPew FeatherWing. Afterwards, insert the LED Matrix into place, solder its pins from the bottom, and trim them with wire cutters.



Make sure to insert the LED Matrix in correct orientation – with the markings in the same direction as the "down" button. Otherwise the colors will be wrong, and the image will be flipped.

Note that you should solder the pin headers first, and then solder the LED matrix. Trying to do it in the opposite order may lead to damaging the matrix with your soldering iron.

Note: The Adafruit Feather board is not included in the kit.

2.2 Making it Flatter

The basic assembly is fine, but it's good 2cm thick. You can make the whole thing much thinner by applying a number of tricks. First, use the short female pin headers for the Adafruit Feather. Second, move the plastic on your male pin headers to the end of the header, insert the header from the top, and try it with the Feather to see how long the pins need to be. Then solder them, and trim the parts sticking out from the top with wire cutters. This way the PewPew FeatherWing can sit right on top of the battery plug of the Feather.

Note: Note that the short female headers are not included in the kit.

2.3 Optional Battery

To make the PewPew even more portable, you can add a small LiPO battery (the 100mAh batteries are about the right size). The FeatherWing even has a power switch built-in that you can use. Just solder the battery leads to the two holes marked "+" for the red wire, and "-" for the black wire. Be very careful to not short the battery wires while you are doing that. Then you can use two-sided tape to stick the battery on the underside of the FeatherWing, sandwiching it between the two boards. You may need to add some electric tape to prevent the metal body of the battery from shorting things on the boards.

Note: Note that the battery is not included in the kit.

2.4 Software Setup

All the example programs are written using CircuitPython, so you will need to get that running on your Feather board. Fortunately Adafruit provides a lot of tutorials about how to do it. Some of the boards (like the M0 Express) even come with CircuitPython already loaded on them. Please see

this guide.

Once you have CircuitPython running on the board, you need to copy the `pew.mpy` file to it. That's the main library that contains all the functions needed to use the PewPew hardware.

In addition, you need to copy your program onto the board. You have to name it `main.py` for it to be started automatically on power-up. Later on we will see how we can use a menu program that will let you choose what to run (you have to rename the `menu.py` to `main.py` to have it run at start).

You can get all the files from the project's repository, and the compiled files are available from the releases page.

CHAPTER 3

PewPew Library Reference

pew.**init**()
Initialize the module.

This function switches the display on and performs some basic setup.

pew.**brightness**(*level*)
Set the brightness of the display, from 0 to 15.

pew.**show**(*pix*)
Show the provided image on the display, starting at the top left corner. You will want to call this once for every frame.

pew.**keys**()
Return a number telling which keys have been pressed since the last check. The number can then be filtered with the `&` operator and the `K_X`, `K_DOWN`, `K_LEFT`, `K_RIGHT`, `K_UP`, and `K_O` constants to see whether any of the keys was pressed.

pew.**tick**(*delay*)
Wait until `delay` seconds have passed since the last call to this function. You can call it every frame to ensure a constant frame rate.

class pew.**Pix**(*width=8*, *height=8*, *buffer=None*)
Pix represents a drawing surface, `width` pixels wide and `height` pixels high.

If no `buffer` is specified for storing the data, a suitable one will be automatically created.

classmethod from_iter(*cls*, *lines*)
Creates a new Pix and initialzes its contents by iterating over `lines` and then over individual pixels in each line. All the lines have to be at least as long as the first one.

classmethod from_text(*cls*, *text*, *color=None*, *background=0*, *colors=None*)
Creates a new Pix and renders the specified text on it. It is exactly the size needed to fit the specified text. Newlines and other control characters are rendered as spaces.

If `color` is not specified, it will use yellow and red for the letters by default. Otherwise it will use the specified color, with `background` color as the background.

Alternatively, `colors` may be specified as a 4-tuple of colors, and then the `color` and `background` arguments are ignored, and the four specified colors are used for rendering the text.

pixel(*self*, *x*, *y*, *color=None*)
If `color` is specified, sets the pixel at location `x`, `y` to that color. If not, returns the color of the pixel at that location.

If the location is out of bounds of the drawing surface, returns 0.

box(*self*, *color*, *x=0*, *y=0*, *width=self.width*, *height=self.height*)
Draws a filled box with the specified `color` with its top left corner at the specified location and of the specified size. If no location and size are specified, fills the whole drawing surface.

blit(*self*, *source*, *dx=0*, *dy=0*, *x=0*, *y=0*, *width=None*, *height=None*, *key=None*)
Copied the `source` drawing surface onto this surface at location specified with `dx` and `dy`.

If `x`, `y`, `widht` and `height` are specified, only copies that fragment of the `source` image, otherwise copies it whole.

If `key` color is specified, that color is considered transparent on the source image, and is not copied onto this drawing surface.

Python Module Index

p

Index