



---

# SparkFun Blocks for Intel® Edison - OLED Block

## Introduction

The Intel® Edison is a powerful single-board computer, but it lacks any form of display output. The OLED Block serves to provide at least some form of output capabilities to the powerful Edison module. The crisp blue-on-black OLED display is 64x48 pixels – about 1.6 inches across the diagonal. It's small, but that allows it to maintain the Edison's tiny form factor. And, it still leaves room for text or games.



Speaking of games, the board also includes a 4-way joystick (with a select switch) and two momentary push buttons. Someone start working on porting an NES emulator to the Edison!

## Suggested Reading

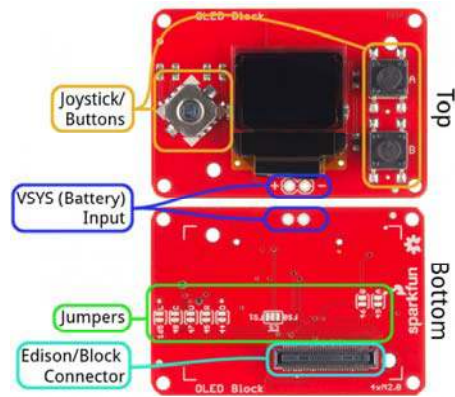
If you are unfamiliar with Blocks, take a look at the [General Guide to Sparkfun Blocks for Intel Edison](#).

Other tutorials that may help you on your Edison adventure include:

- [Powering Your Project](#)
- [Battery Technologies](#)
- [Connector Basics](#)

## Board Overview

Before we jump into using the OLED Block, here's a quick overview of what features and components the board includes:



## Button Pin Mapping

In total, the OLED block has seven button/joystick inputs, here's how they're mapped to the Edison's GPIO pins:

| Button | Edison GPIO Pin |
|--------|-----------------|
| Up     | 47              |
| Down   | 44              |
| Left   | 165             |
| Right  | 45              |
| Select | 48              |
| A      | 49              |
| B      | 46              |

**Note 1:** "Up" is towards the top of the screen (the side *without* the black ribbon connector).

**Note 2:** The "Select" button refers to a downward press on the joystick.

## Battery Supply Input

A pair of through-holes below the bottom of the screen, labeled "+" and "-", can be used to supply a battery input to the Edison.



This voltage is supplied directly to the Edison's VSYS pin, and should be **between 3.3 to 4.5V**. That voltage is *not* regulated, so be very careful applying voltage to this input. A single-cell Lithium Polymer battery is a perfect, mobile choice for this power supply.

## Jumpers

The bottom of the OLED block is filled with jumpers, which allow you to tailor the block to fit your project.



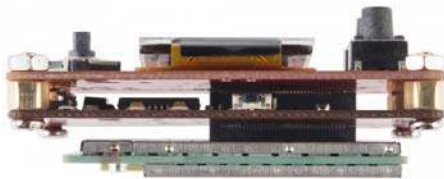
Seven of the eight jumpers allow you to disconnect any of the joystick or button inputs from the Edison, in case you need those GPIO for another purpose. Keep in mind, if you cut any of the jumpers, that button will serve no purpose unless you wire it to another pin or re-solder the jumper.

The "CS" jumper allows you to flip the OLED's chip-select pin from "FS0" (default) to "FS1". If you switch this pin, you'll need to modify the code accordingly.

## Using the OLED Block

To use the OLED Block, attach it to either an Edison or another SparkFun Block. This board, unlike most Edison blocks is only single-sided, so it must be at the top of your stack (you don't want to cover that beautiful display!).

The OLED block *can* supply power to the Edison, using the battery supply input, but we recommend using it in conjunction with a development block like the Console Block or Base Block.



*OLED Block in a stack with the Edison and a Console Block.*

Blocks can be stacked without hardware, but it leaves the expansion connectors unprotected from mechanical stress. We recommend adding a few screws from our Hardware Pack between the OLED and the next block.



*Intel Edison Hardware Pack*

## Programming for the OLED Block

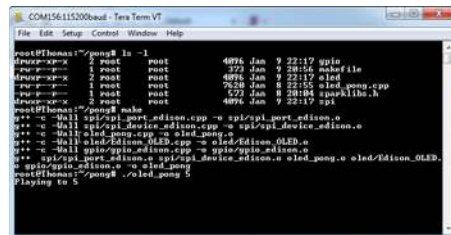
It'll take a bit of coding to get your display up-and-running. The Edison has all sorts of programming-language support, but we've decided to stick with C++ to control the display.

We've written a simple library to help get you started. You can download the latest version from our GitHub repository, or click the button below to get it in a zip folder.

**DOWNLOAD THE C++ LIBRARY AND EXAMPLE**

The example code includes a simple makefile that should compile directly on the Edison. But first, you need to get these files onto the Edison. There are a few ways to do that. For example, you could get the Edison connected to a WiFi network and SSH it over – check out the SSH how-to section of our Getting Started with Edison tutorial for help with that. Even better than that, you could follow along with our Programming the Intel® Edison: Beyond the Arduino IDE and set up Eclipse to remotely upload the code – even remotely compile it on your development computer.

Once you've loaded the code, navigate to the "pong" folder and type `make`. The dependencies (spi, gpio, and oled libraries) will build, then the main example code ("oled\_pong.cpp") will build to create an "oled\_pong" executable.

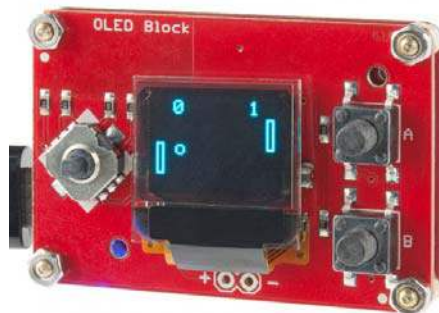


```

root@edison:~/pong# ls -l
lsroot@edison:~/pong# ls -l
-rw-r--r-- 1 root root 4896 Jan 9 22:17 gpio
-rw-r--r-- 1 root root 1222 Jan 9 22:16 makefile
lsroot@edison:~/pong# ls -l
-rw-r--r-- 1 root root 4896 Jan 9 22:17 oled
-rw-r--r-- 1 root root 7538 Jan 8 22:15 oled_pong.cpp
-rw-r--r-- 1 root root 1222 Jan 8 22:16 sparklib.h
lsroot@edison:~/pong# make
make
*** -Wall spi/spi_uart_edison.cpp -o spi/spi_uart_edison.o
*** -Wall spi/spi_device_edison.cpp -o spi/spi_device_edison.o
*** -Wall oled_pong.cpp -o oled_pong.o
*** -Wall oled/Edison_OLED.cpp -o oled/Edison_OLED.o
*** -Wall spi/gpio_edison.cpp -o spi/gpio_edison.o
*** spi/spi_uart_edison.o spi/spi_device_edison.o oled_pong.o oled/Edison_OLED.o
g++ spi/spi_uart_edison.o spi/spi_device_edison.o oled_pong.o oled/Edison_OLED.o -o oled_pong
root@edison:~/pong# ./oled_pong
Playing to 5

```

After the code has successfully been built, type `./oled_pong` to play. This should trigger the OLED to light up and start a game of pong. Have fun!



*Edison Pong in action!*

## Using the OLED Library

The OLED library, included in the download above, allows you to draw anything from pixels and lines to shapes and text on the little display.

The "pong" example code should serve as an excellent teaching tool. To begin, create an instance of the `edOLED` class, then initialize the display like this:

```

edOLED oled;
...
oled.begin()
oled.clear(ALL);
oled.display();

```

Then, continue to use the `oled` object to draw pixels, lines, and other shapes like this:

```
oled.pixel(x, y); // Draw a pixel at x,y
oled.line(x0, y0, x1, y1); // Draw a line from x0,y0 to x1,y1
oled.rect(x, y, width, height); // Draw a rectangle, beginning
at x,y with a set width and height
oled.circle(x, y, radius); // Draw a circle, centered at x,y
with a set radius
```

Or you can draw text on the screen with function calls like:

```
oled.setCursor(x, y); // Set the text cursor to x,y
oled.setFontType([0:3]); // Set the font to one of four typ
es.
oled.print(char); // Draw a character
oled.write(char *); // Draw an array of characters (strin
g)
oled.write(int); // Draw an integer value
```

Don't forget: the OLED's display will not update until you call `oled.display()`.

## Using the GPIO Library

The "gpio" library, also included with the example code, can be used to listen for button presses. Again, the pong example should serve as a good place to start learning how to use the library.

First, you'll need to initialize the pins:

```
gpio BUTTON_UP(47, INPUT); // UP button is tied to GPIO 47
gpio BUTTON_DOWN(44, INPUT); // Down is tied to GPIO 44
gpio BUTTON_LEFT(165, INPUT); // Left is tied to GPIO 165
gpio BUTTON_RIGHT(45, INPUT); // Right is tied to GPIO 45
gpio BUTTON_SELECT(48, INPUT); // Select (pushing the joystic
k down) is GPIO 48
gpio BUTTON_A(49, INPUT); // Button A is GPIO 49
gpio BUTTON_B(46, INPUT); // Button B is GPIO 46
```

Then you can use the `readPin()` function of the `gpio` class to see if they're HIGH or LOW. The buttons are all pulled high, so they'll read as LOW if they're pressed down.

```
if (BUTTON_UP.pinRead() == LOW)
{
    printf("You're pressing down");
}
if (BUTTON_A.pinRead() == LOW)
{
    printf("You're pressing the A button");
}
```

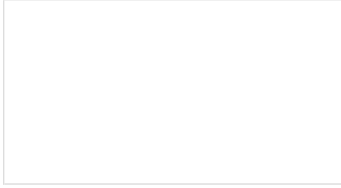
Hopefully that brief primer will get you ready to begin developing for the Edison OLED Block. If you haven't already, we really recommend following our Programming the Intel® Edison tutorial for help setting up your programming toolchain.

## Resources and Going Further

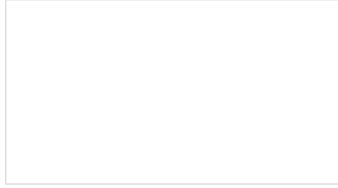
Now that you have had a brief overview of the OLED Block, take a look at some of these other tutorials. These tutorials cover programming, Block stacking, and interfacing with the Intel Edison ecosystems.

- General Guide to Sparkfun Blocks for Intel Edison
- Edison Getting Started Guide
- Loading Debian (Ubilinux) on the Edison

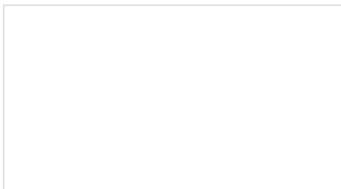
Check out these other Edison related tutorials from SparkFun:



**SparkFun Blocks for Intel® Edison - microSD Block**  
A quick overview of the features of the microSD Block.



**SparkFun Blocks for Intel® Edison - Console Block**  
A quick overview of the features of the Console Block.



**Programming the Intel® Edison: Beyond the Arduino IDE**  
Intel's Edison module goes beyond being just another Arduino clone. Check this tutorial for advice on how to get the most out of your Edison by writing code in C++!



**SparkFun Blocks for Intel® Edison - ADC V20**  
A quick overview of the features of the ADC Block.