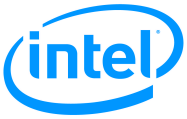
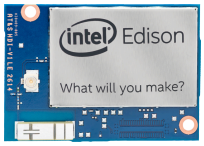
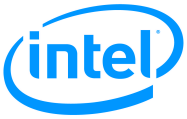




SparkFun® GPIO Block: Programming Guide



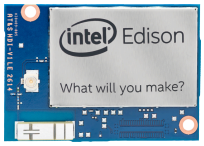
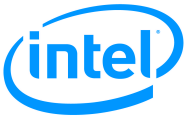
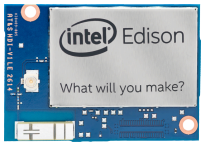


Table of Contents

Revision history		
Version	**Date**	**Comment**
1.0	1/26/2016	Initial release





Introduction

This document provides guidance on how to write a C program for Intel® Edison and SparkFun® GPIO Block. We have developed a C library to enable easy GPIO access on this breakout board. The programming instructions utilize this library. Please refer to https://learn.sparkfun.com/tutorials/sparkfun-blocks-for-intel-edison---gpio-block/all for more information about the device. In this tutorial, you will learn to

1. Assemble the hardware,
2. Turn on/off an LED,
3. Read an input signal from a button, and
4. Send PWM signals to control the LED brightness.

Things Needed

1. An Intel® Edison
2. A SparkFun® GPIO Block
3. A SparkFun® Battery Block or a Base Block
    a. If you use a base block, you need a micro USB cable supply power.
4. An LED,
5. A button,
6. Wires,
7. Resistors,
8. A breadboard,
9. Soldering tools, and
10. A PC or a Mac

Hardware Assembly

It is important to note that you must assemble the hardware BEFORE you supply power. Otherwise, you may damage the devices. The instruction below includes specific configurations only for the demos presented in this tutorial. Please follow the steps below.

1. Insert your Edison module into the GPIO block.

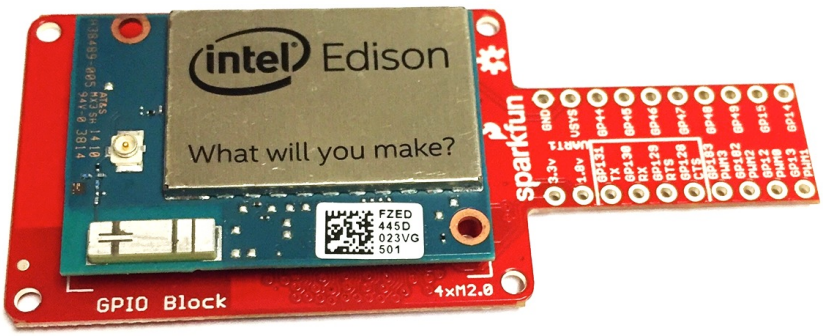


Figure 1 Edison and GPIO Block

2. Connect a battery block or a base block.

Figure 2 Edison, GPIO Block, and Battery Block

3. This is the general hardware installation. You can add more SparkFun® blocks. If you have a hardware pack (screws and nuts), you can use them to secure the connection.

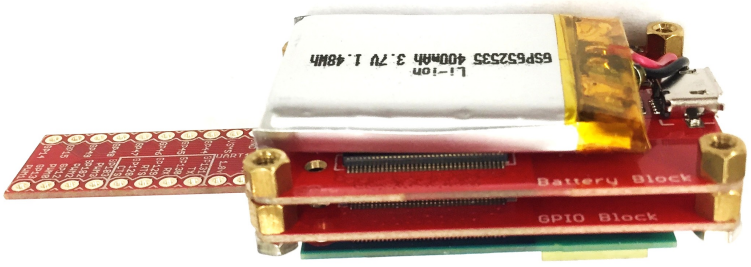


Figure 3 Hardware Pack

4. Now, we will configure the hardware specifically for the demos in this tutorial. Separate the GPIO block from the other devices.
5. Solder wires onto the block for the following pins: **3.3V**, **GND**, **GP44**, **GP48**, and **GP12**.

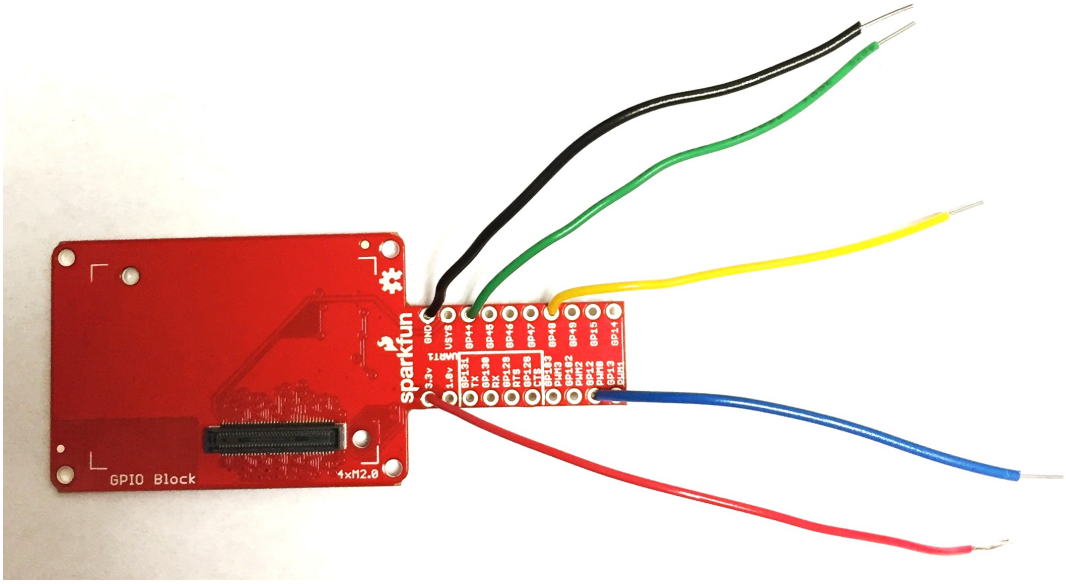


Figure 4 Wires

6. Reassemble the blocks.
7. Configure the hardware as shown in the figure below.
   (Note: The Edison module is omitted in the figure, but you should have it inserted)

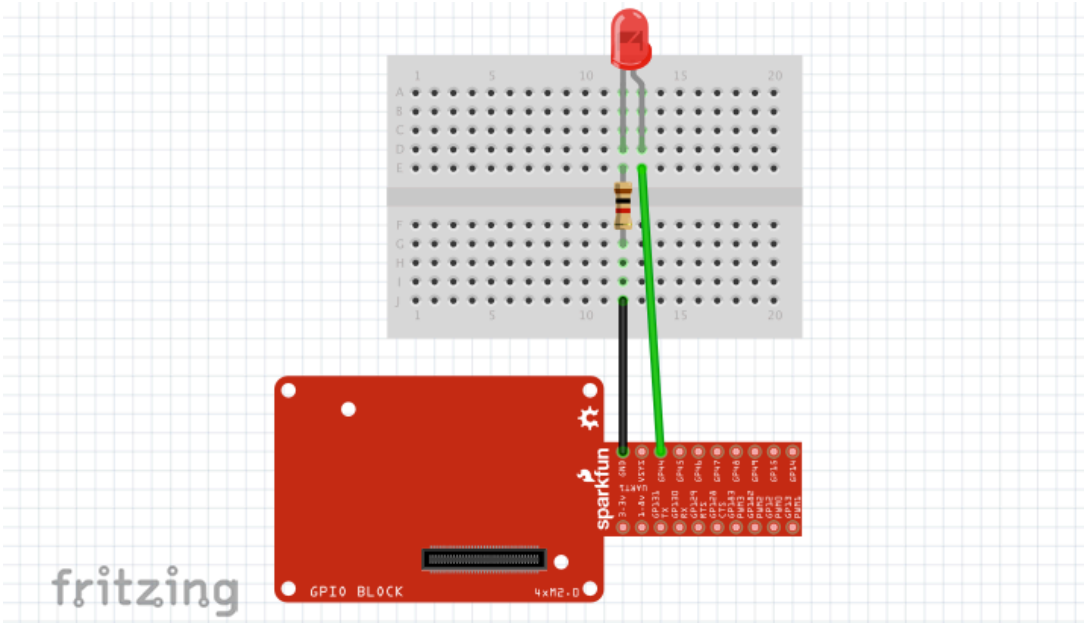
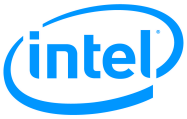
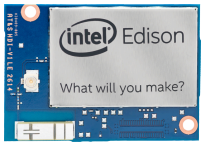


Figure 5 Hardware Configuration

8. We will change the configuration as we proceed.

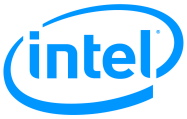


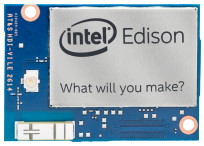


C Library Overview

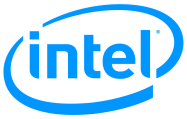
"GPIO and I2C Interfaces" tutorial introduces GPIO access on the Intel Edison. It requires an Arduino breakout board. GPIO access without the Arduino breakout may need a different procedure. For instance, the MRAA library may no longer work correctly. The MRAA library abstracts the Linux's GPIO interface and includes the configurations for the Arduino breakout. We may try to use the GPIO interface directly. However, as we know from "SPI, PWM, and More GPIO" tutorial, the Linux's GPIO interface may not be so convenient for direct use. Thus, we have developed a simple C library for easy GPIO access (available at: https://github.com/chrisIHbaek/gpio_library). The library provides useful functions as listed below:

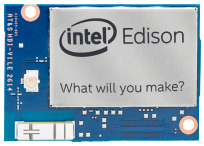
1. **gpio_init**
   - This function initializes a GPIO pin for a simple digital input/output.
   - It takes one argument to specify the GPIO pin number.
     - Ex) gpio_init(GPIO44);
   - Since the library is designed for the SparkFun block, the available pins are limited to the listed values below:
     - GPIO12
     - GPIO13
     - GPIO14
     - GPIO15
     - GPIO44
     - GPIO45
     - GPIO46
     - GPIO47
     - GPIO48
     - GPIO49
     - GPIO182
     - GPIO183
2. **gpio_close**
   - This function closes a GPIO pin.
   - It takes one argument to specify the GPIO number.
     - Ex) gpio_close(GPIO44);
3. **gpio_direction**
   - This function sets the direction of the GPIO pin.
   - It takes two arguments: one to specify the pin number and the other to specify the direction.
     - Ex) gpio_direction(GPIO44, OUTPUT);





4. **gpio_write**
    - This function writes a digital value (0 or 1) to the pin.
    - It takes two arguments: one to specify the pin number and the other to specify the digital value.
        - Ex) gpio_write(GPIO44, 1);
    - This function is available only if the GPIO pin is set as an output.
5. **gpio_read**
    - This function reads the value from a GPIO pin and returns the value as an integer.
    - It takes one argument to specify the pin number.
        - Ex) int value = gpio_read(GPIO48);
    - This function is available only if the GPIO pin is set as an input.





Programming Guide

GPIO Output

1. Make sure your hardware is configured as shown in Figure 5.
2. Power on the Edison.
3. Serial connect or SSH into the Edison.
4. **$ git clone https://github.com/chrisIHbaek/gpio_library.git**
5. **$ cd gpio_library**
6. **$ vi gpio_output.c**
7. Enter the following C code.

```
1 #include <stdio.h>
2 #include "gpio.h"
3
4 int main()
5 {
6         GPIO gpio = GPIO44;
7
8         gpio_init(gpio); // initialize the pin
9         gpio_direction(gpio, OUTPUT); // set it as an output
10
11        gpio_write(gpio, 1); // send HIGH
12        sleep(3); // wait for 3 seconds
13        gpio_write(gpio, 0); // send LOW
14
15        gpio_close(gpio); // close the pin
16        return 0;
17 }
```

Figure 6 gpio_output.c

8. **$ gcc -o gpio_output gpio_output.c gpio.c**
9. **$ ./gpio_output**
10. The LED will turn on for three seconds and turn off.

GPIO Input

1. Remove power from the Edison to avoid any damages.
2. Configure the hardware as shown in the figure below.

   You may use a Grove 4-pin cable to connect the button.

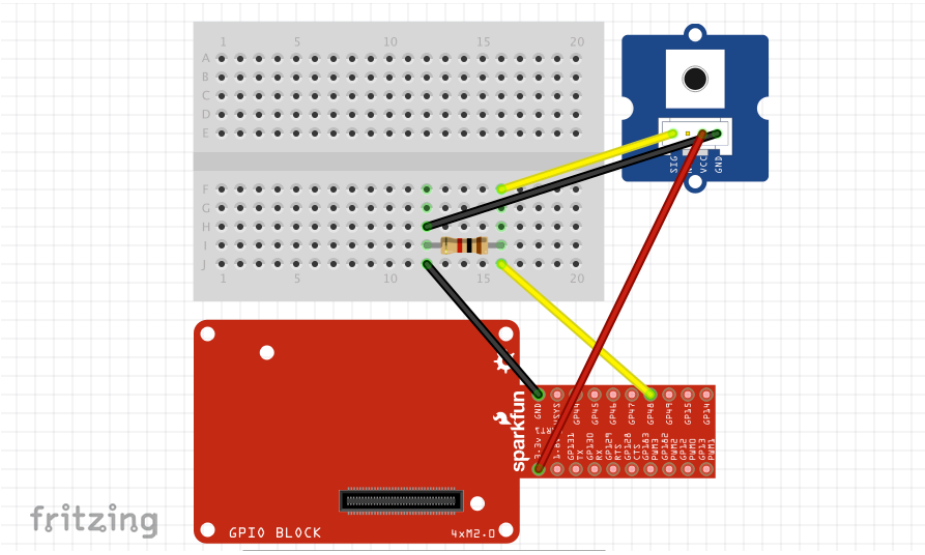


Figure 7 Hardware Configuration

3. Power on the Edison.
4. Serial Connect or SSH into the Edison.
5. **$ cd gpio_library**
6. **$ vi gpio_input.c**
7. Enter the following C code.

```c
#include <stdio.h>
#include "gpio.h"

int main()
{
        int value;
        GPIO gpio = GPIO48;

        gpio_init(gpio); // initialize the pin
        gpio_direction(gpio, INPUT); // set it as an input

        value = gpio_read(gpio); // read the value
        printf("Value: %d\n", value); // print the value

        gpio_close(gpio); // close the pin
        return 0;
}
```

Figure 8 gpio_input.c

8. **$ gcc -o gpio_input gpio_input.c gpio.c**
9. **$ ./gpio_input**
10. Now, press the button and execute the program again.

PWM

We can use the MRAA library to generate PWM signals. In this demo, we will generate a PWM signal to control the LED brightness.

1. Remove power from the Edison to avoid any damages.
2. Configure the hardware as shown in the figure below.

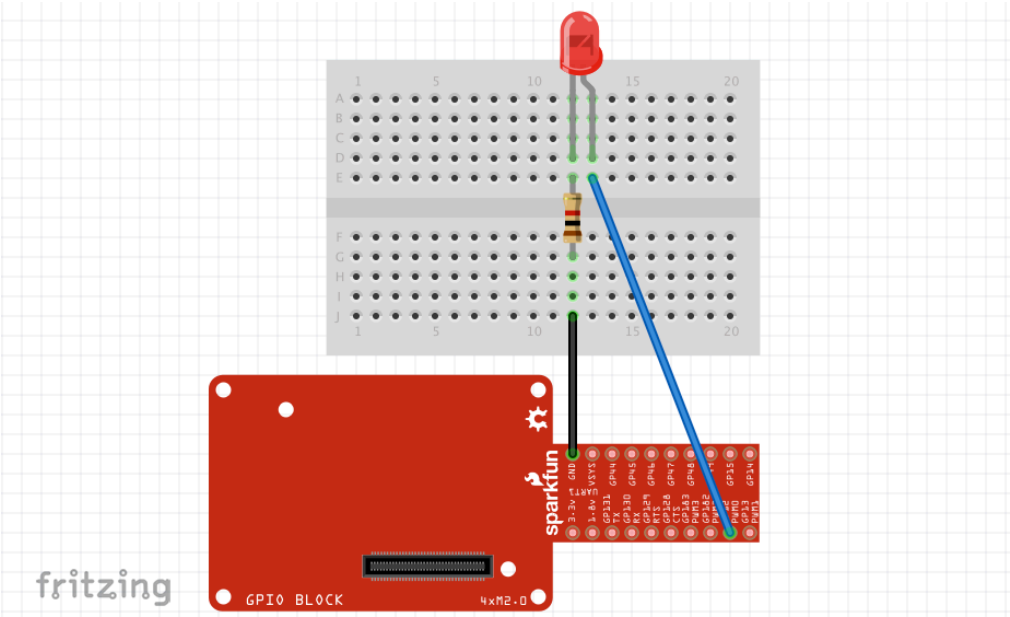


Figure 9 Hardware Configuration

3. Power on the Edison.
4. Serial Connect or SSH into the Edison.
5. **$ vi pwm.c**
6. Enter the following C code. This code is a modified version of the LED brightness control code from Tutorial 5. The pin value specified in line 14 is IO 20. However, the GPIO pin we soldered the wire onto is GPIO 12. The MRAA library expects pin numbers that are not the Linux pin numbers. The corresponding value for each pin is give below:

GPIO 12	IO 20
GPIO 13	IO 14
GPIO 182	IO 0
GPIO 183	IO 21

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <mraa/pwm.h>

#define MAXBUFSIZ 1024

int main() {
        float brightness;
        char user_input[MAXBUFSIZ];
        mraa_pwm_context pwm;

        pwm = mraa_pwm_init(20);

        if (pwm == NULL) {
                fprintf(stderr, "Failed to initialize.\n");
                return 1;
        }

        mraa_pwm_period_us(pwm,200);
        mraa_pwm_enable(pwm,1);

        while(1) {
                printf("Enter brightness value (0-100): ");
                scanf("%s", user_input);
                brightness = atof(user_input);

                if (brightness > 100 || brightness < 0)
                        printf("Error: Choose between 0 and 100\n");
                else {
                        brightness = brightness/100;
                        mraa_pwm_write(pwm, brightness);
                }
        }
        return 0;
}
```

Figure 10 pwm.c

7. **$ gcc -lmraa -o pwm pwm.c**
8. **$ ./pwm**