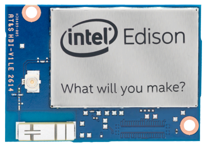


SparkFun® ADC Block: Programming Guide

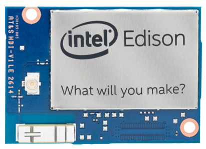
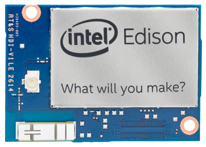


Table of Contents

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® ADC Block. SparkFun provides a library written in C++ and a simple example program using their library. We have ported their C++ library over to C. The programming instructions utilize this library. Please refer to https://learn.sparkfun.com/tutorials/sparkfun-blocks-for-intel-edison---adc-v20 for more information about the device. In this tutorial, you will

1. Assemble hardware,
2. Read an input signal from a potentiometer

Things Needed

1. An Intel® Edison
2. A SparkFun® ADC 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. A potentiometer,
5. Wires,
6. Soldering tools, and
7. 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 ADC block.

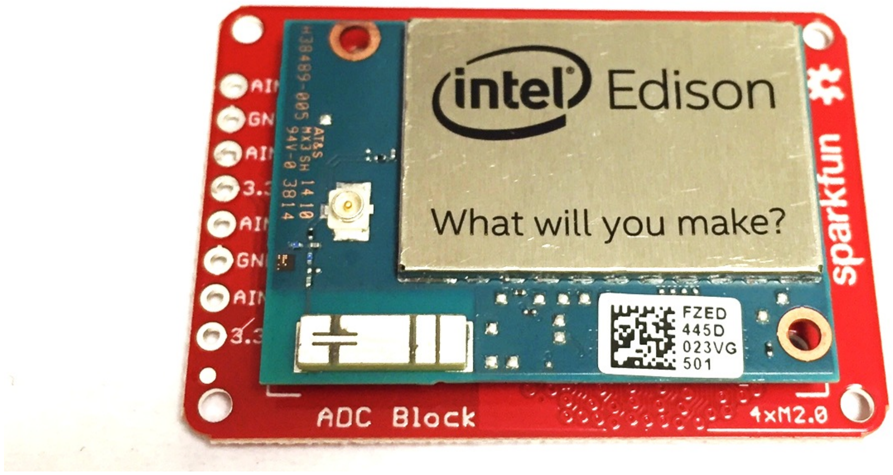


Figure 1 Edison and ADC Block

2. Connect a battery block or a base block.

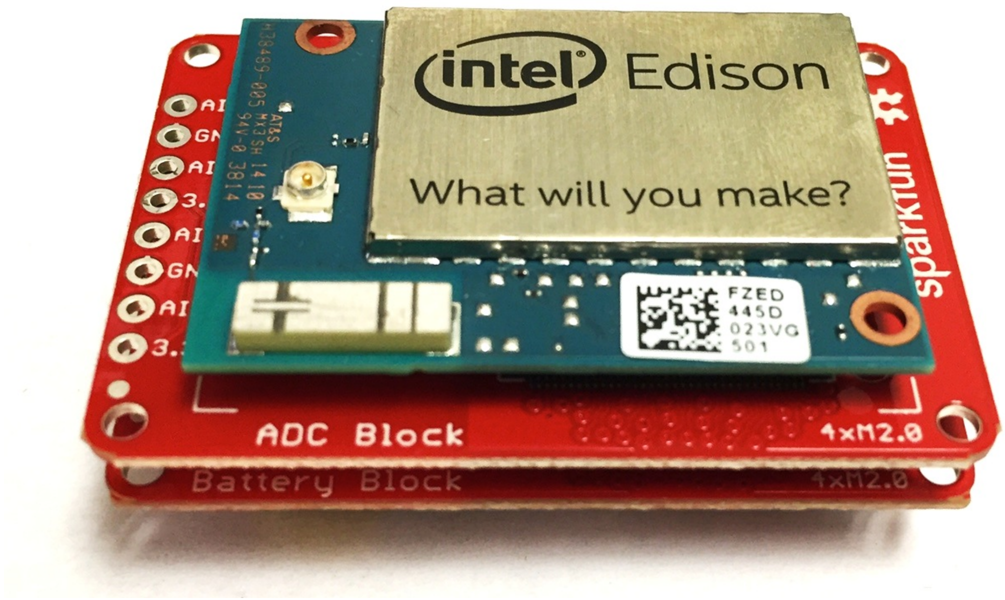


Figure 2 Edison, GPIO Block, and Battery Block

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.

4. Now, we will configure the hardware specifically for the demos in this tutorial. Separate the ADC block from the other devices.
5. Solder wires onto the block on the bottom 3 pins: **AIN0**, **GND**, and **3.3V**.

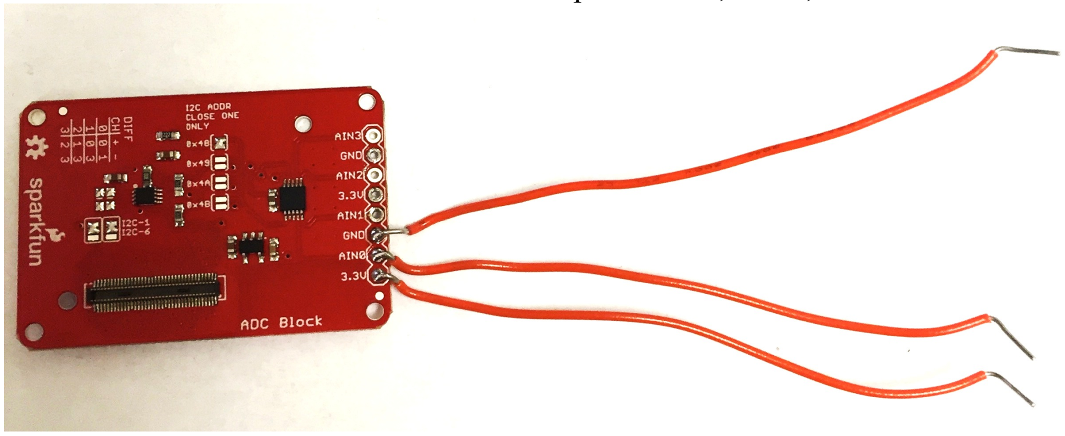


Figure 3 Wires

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)

   - Yellow – AIN0
   - Black – GND
   - Red – 3.3V

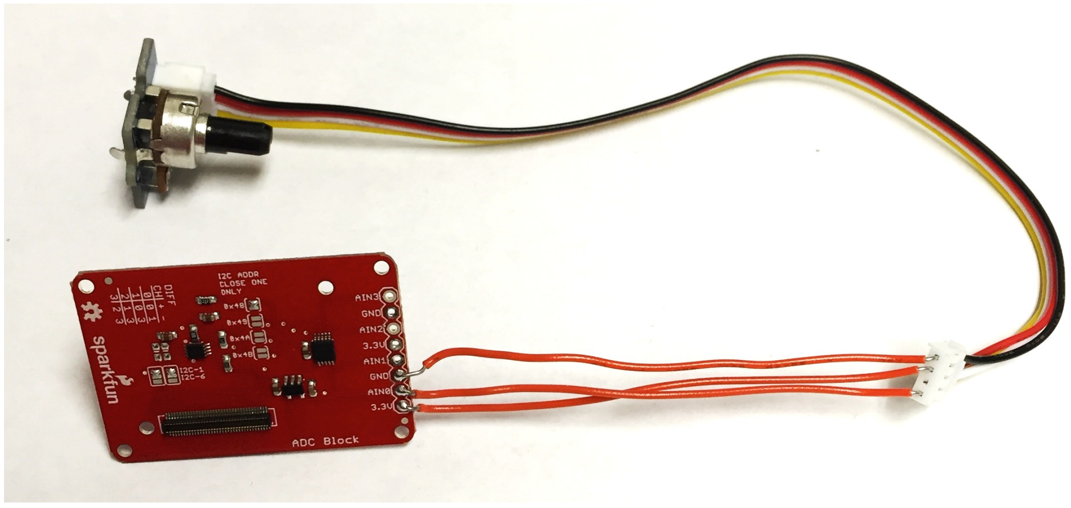
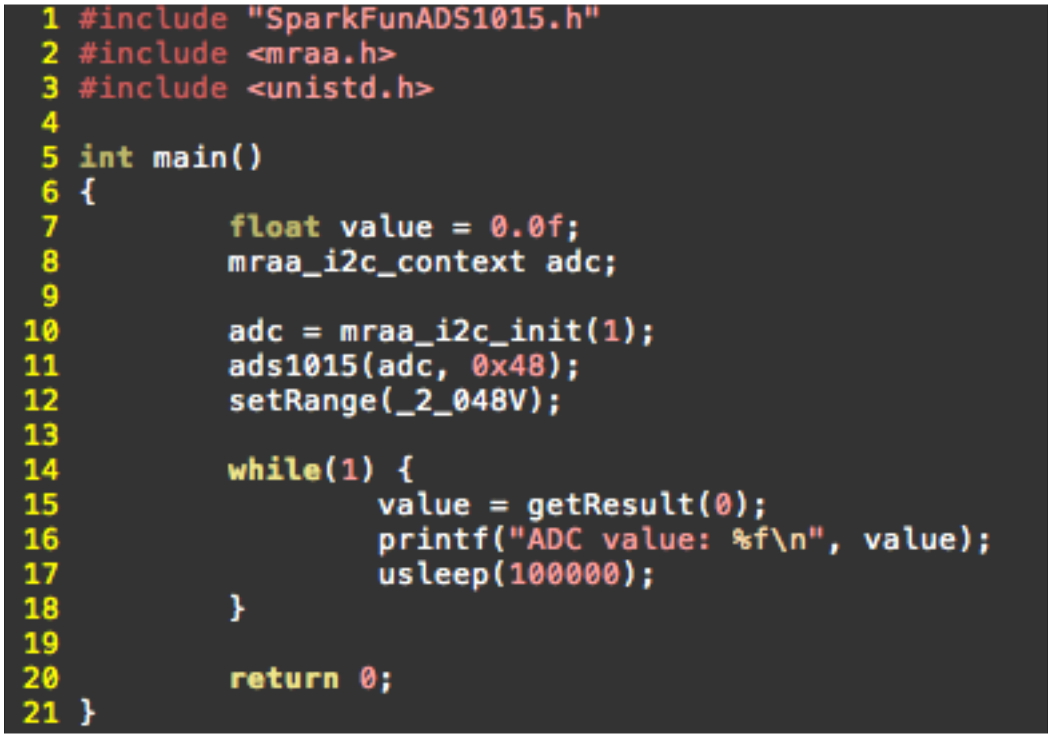


Figure 4 Hardware Configuration

Programing Guide

We will do a demo very similar to the analog input demo from "GPIO and I2C Interfaces" tutorial, which required an Arduino breakout. Since A potentiometer will be used as the input to the ADC.

1. Power on the Edison.
2. Serial connect or SSH into the Edison.
3. **$ mkdir adc_library**
4. **$ cd adc_library**
5. Transfer the two library files, **SparkFunADS1015.h** and **SparkFunADS1015.c**, from your personal laptop to the Intel Edison. One way this can be done is using sftp.
6. **$ vi adc.c**
7. Type the following C code.

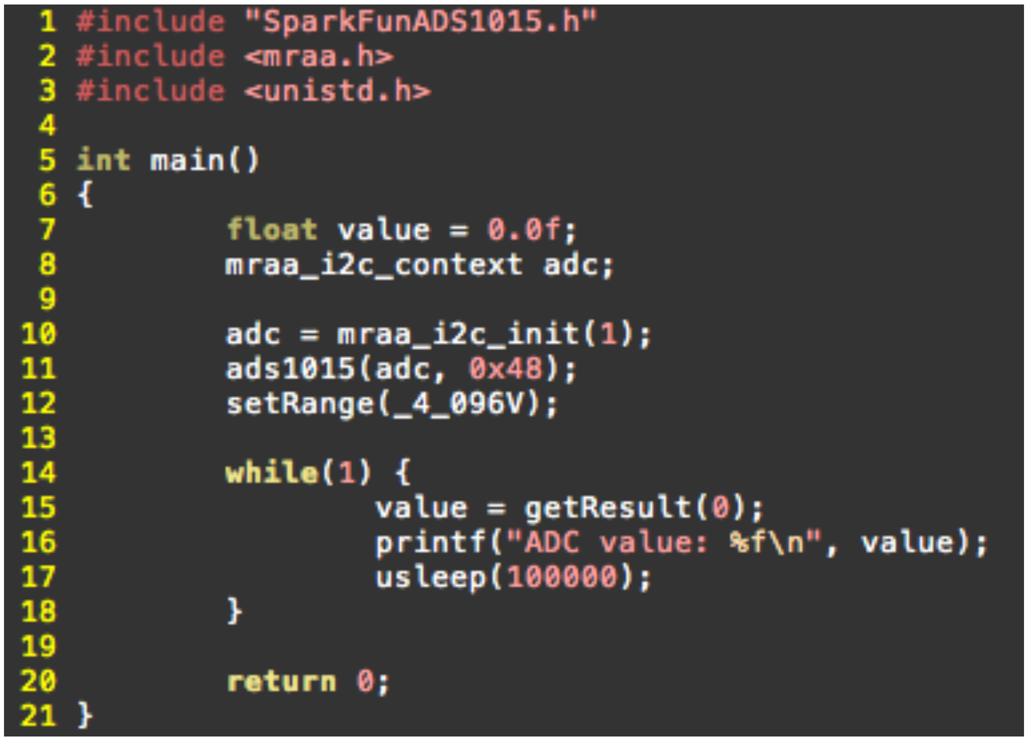


```c
#include "SparkFunADS1015.h"
#include <mraa.h>
#include <unistd.h>

int main()
{
        float value = 0.0f;
        mraa_i2c_context adc;

        adc = mraa_i2c_init(1);
        ads1015(adc, 0x48);
        setRange(_2_048V);

        while(1) {
                value = getResult(0);
                printf("ADC value: %f\n", value);
                usleep(100000);
        }

        return 0;
}
```

Figure 5 adc.c

8. **$ gcc -lmraa -o adc adc.c SparkFunADS1015.c**
9. **$ ./adc**
10. Slowly rotate the knob and observe the range of values displayed on the output terminal.
11. Press **Ctrl-C** to quit.

12. Return to your C code, but change the argument of **setRange** on line 12 to the new value: **_4_096V**



```
 1 #include "SparkFunADS1015.h"
 2 #include <mraa.h>
 3 #include <unistd.h>
 4
 5 int main()
 6 {
 7         float value = 0.0f;
 8         mraa_i2c_context adc;
 9
10         adc = mraa_i2c_init(1);
11         ads1015(adc, 0x48);
12         setRange(_4_096V);
13
14         while(1) {
15                 value = getResult(0);
16                 printf("ADC value: %f\n", value);
17                 usleep(100000);
18         }
19
20         return 0;
21 }
```

Figure 6 Modified adc.c

13. Recompile the program:

    $ gcc -lmraa -o adc2 adc.c SparkFunADS1015.c

14. **$ ./adc2**

15. Slowly rotate the knob and observe the range of values displayed on the output terminal. You should notice that the range of values observed is different than from the first version.

16. Press **Ctrl-C** to quit.

17. There are 6 predefined values in the library:

    a. **_0_256V** - Range is -0.256V to 0.255875V, and step size is 125uV.
    b. **_0_512V** - Range is -0.512V to 0.51175V, and step size is 250uV.
    c. **_1_024V** - Range is -1.024V to 1.0235V, and step size is 500uV.
    d. **_2_048V** - Range is -2.048V to 2.047V, and step size is 1mV.
    e. **_4_096V** - Range is -4.096V to 4.094V, and step size is 2mV.
    f. **_6_144V** - Range is -6.144V to 6.141V, and step size is 3mV.