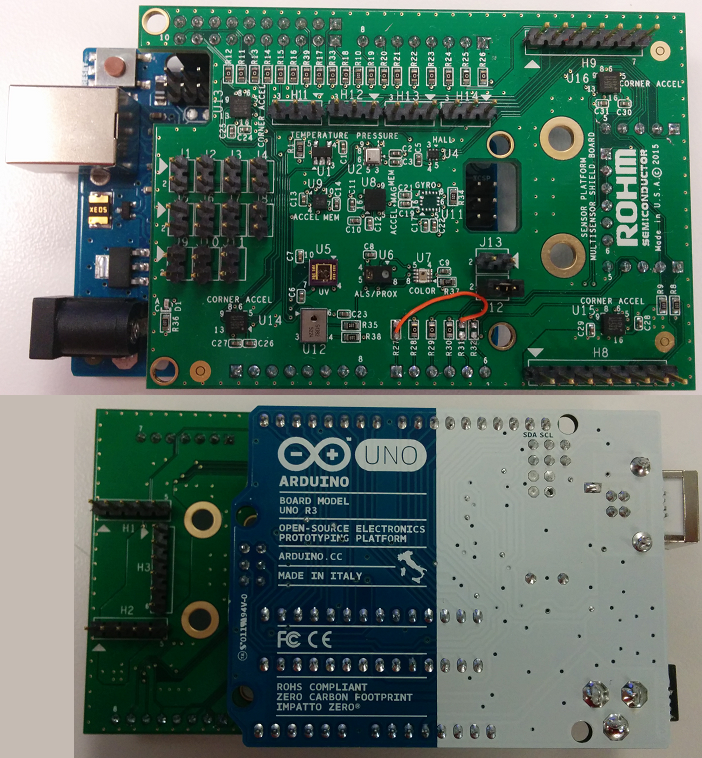
Connecting the Multi-Sensor Shield to the Arduino UNO Platform



Above: ROHM Sensor Platform Shield directly connected to the Arduino Uno

15 January 2015, Revision A

Table of Contents

[Copyright and License 2](#_Toc422155682)

[Revision History 3](#_Toc422155683)

[Introduction 3](#_Toc422155684)

[Getting Started 4](#_Toc422155685)

[Connecting ROHM Multi-Sensor Shield to the Arduino UNO 4](#_Toc422155686)

[Required HW Rework 4](#_Toc422155687)

[Hardware Connection for ROHM BDE0600G Temp Sensor to the Arduino Uno 5](#_Toc422155688)

[Software Explanation for ROHM BDE0600G Temp Sensor to the Arduino Uno 6](#_Toc422155689)

[Hardware Connection for LAPIS ML8511 UV Sensor to the Arduino Uno 7](#_Toc422155690)

[Software Explanation for LAPIS ML8511 UV Sensor to the Arduino Uno 8](#_Toc422155691)

[Hardware Connection for ROHM BU52011 Hall Sensor to the Arduino Uno 9](#_Toc422155692)

[Software Explanation for ROHM BU52011 Hall Sensor to the Arduino Uno 9](#_Toc422155693)

[Hardware Connection for Kionix KMX61 Accel+Mag Sensor to the Arduino Uno 10](#_Toc422155694)

[Software Explanation for Kionix KMX061 Accel+Mag Sensor to the Arduino Uno 11](#_Toc422155695)

# Copyright and License

The following except is copied directly from the Arduino FAQ (<http://arduino.cc/en/Main/FAQ>)

**“What do you mean by open-source hardware?”** - Open-source hardware shares much of the principles and approach of free and open-source software. In particular, we believe that people should be able to study our hardware to understand how it works, make changes to it, and share those changes. To facilitate this, we release all of the original design files (Eagle CAD) for the Arduino hardware. These files are licensed under a Creative Commons Attribution Share-Alike license, which allows for both personal and commercial derivative works, as long as they credit Arduino and release their designs under the same license.

The Arduino software is also open-source. The source code for the Java environment is released under the GPL and the C/C++ microcontroller libraries are under the LGPL.

Please note that all references to ROHM’s Multi-Sensor Shield are also shared under open source guidelines under the GNU General Public License, Version 3. Details can be found at the following link: <https://github.com/ROHMUSDC/ROHM_SensorPlatform_Multi-Sensor-Shield>.

# Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Description** | **Revision ID** |
| 15 June 2015 | First Draft | A |
|  |  |  |
|  |  |  |

# Introduction

The following document was written to provide a brief connection guide and starting point for using ROHM’s Multi-sensor shield with the Arduino Uno. This guide assumes that the user has basic functional knowledge of both the ROHM Multi-Sensor Shield and the Arduino itself. If this is not correct, please see the following links for other guides and information on these products.

ROHM’s Multi-Sensor Shield GitHub Repository Page: <https://github.com/ROHMUSDC/ROHM_SensorPlatform_Multi-Sensor-Shield>

Arduino: <http://arduino.cc/>

Please note that the Arduino platform is a microcontroller platform; thus, this document will explain and show examples of how to connect to and convert values for the ROHM Sensor Kit Sensors. This includes the following:

* ROHM BDE0600G – Analog Temperature Sensor
* LAPIS ML8511 – Analog UV Sensor
* ROHM BU52011HFV – Hall Switch Sensor
* KIONIX KMX62 – Digital Accelerometer and Magnetometer
* ROHM BM1383GLV – Digital Barometric Pressure Sensor
* ROHM RPR-0521 – Digital Ambient Light Sensor and Proximity Sensor
* ROHM BH1745 – Digital Color Sensor
* KIONIX KX022 – Digital Accelerometer

# Getting Started

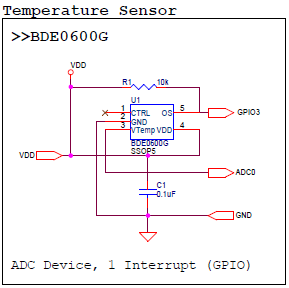
1. Initial Setup
   1. ROHM Multi-Sensor Shield Board
      1. For this guide we will connect ROHM’s Multi-Sensor Shield board directly to the Arduino UNO using the standard Arduino Shield Headers
   2. The following are recommended for using the Arduino Uno for this guide
      1. PC with Arduino IDE
      2. USB Cable to power and monitor serial communication
      3. Download the example file “**ROHM\_SensorKitBreakoutBoardConnect\_06-04-2015.ino**” from <https://github.com/ROHMUSDC/ROHM_SensorPlatform_Multi-Sensor-Shield>. This document will refer to this code to help explain how to connect and properly calculate sensor data.

# Connecting ROHM Multi-Sensor Shield to the Arduino UNO

## Required HW Rework

* On the Arduino UNO board, please note that the I2C pins connected to the top left header are actually routed to pins A4 and A5 on the bottom right connector. This conflicts with an ADC output and the KX122 INT pin already existing on the board. Thus, in order to reroute this on our board, we suggest the following rework…
  + Remove R27, R31, R32
  + Tie the top pad of R31 to the bottom pad of R27
* Picture Reference
  + 
* Please note that multi-sensor shield board schematics can also be found at the following site:
  + <https://github.com/ROHMUSDC/ROHM_SensorPlatform_Multi-Sensor-Shield>

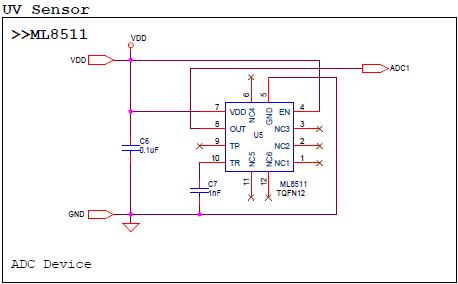
## Hardware Connection for ROHM BDE0600G Temp Sensor to the Arduino Uno

* ROHM BDE0600G – Analog Temperature Sensor
  + 
* As seen in the schematic above, this device connects 4 pins.
  + VDD – Connect to Arduino 3.3V output pin on power connector
  + GND – Connect to Arduino GND pin on the power connector
  + ADC0 – ADC Output for Temperature Readings, Connected to pin A2 on Arduino Board
  + GPIO0 – This pin is used to trigger the thermostat output function. This pin is not used in the example application.

## Software Explanation for ROHM BDE0600G Temp Sensor to the Arduino Uno

* Code Segments pertaining to this sensor can be found by defining and seeing code within the “***AnalogTemp***” #ifDef statements
* Pseudo-Code Explanation
  1. Define Relevant Variables
  2. Begin Loop()
     1. Read back the Analog Sensor Output from Pin A1
        + Note: Default Arduino Reference voltage is 5V; however we are supplying 3.3V to the sensor. Take these values into account when performing your conversions!
     2. Convert value to V, then to Temperature reading
        + Known Values
          - Temperature Sensitivity = -10.68mV/degC
          - Temperature Sensitivity = -0.01068V/degC
          - Temp Known Point = 1.753V @ 30 degC
        + Calculation
          - ADC\_Voltage = (sensorValue / 670) \* 3.3V
          - ADC\_Voltage = sensorValue \* (3.3V/670)
          - ADC\_Voltage = sensorValue \* 0.004925
          - Temperature (in deg C) = (ADC\_Voltage - 1.753)/(-0.01068) + 30
     3. Format Serial Output and Return Information

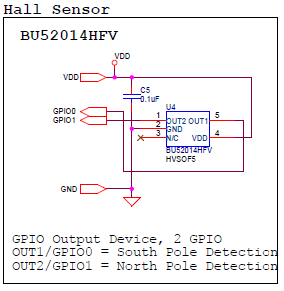
## Hardware Connection for LAPIS ML8511 UV Sensor to the Arduino Uno

* LAPIS ML8511 – Analog UV Sensor
  + 
* As seen in the schematic above, this device connects 3 pins.
  + VDD – Connect to Arduino 3.3V output pin on power connector
  + GND – Connect to Arduino GND pin on the power connector
  + ADC0 – ADC Output for UV Sensor Output, After HW Rework, this should be connected to Arduino Pin A0

## Software Explanation for LAPIS ML8511 UV Sensor to the Arduino Uno

* Code Segments pertaining to this sensor can be found by defining and seeing code within the “***AnalogUV***” #ifDef statements
* Pseudo-Code Explanation
  1. Define Relevant Variables
  2. Begin Loop()
     1. Read back the Analog Sensor Output from Pin A2
        + Note: Default Arduino Reference voltage is 5V; however we are supplying 3.3V to the sensor. Take these values into account when performing your conversions!
     2. Convert value to V, then to UV Intensity
        + Known Values
          - UV Sensitivity = 0.129 V/(mW/cm2)
          - Temp Known Point = 2.2V @ 10mW/cm2
        + Calculation
          - ADC\_Voltage = (sensorValue / 670) \* 3.3V
          - ADC\_Voltage = sensorValue \* (3.3V/670)
          - ADC\_Voltage = sensorValue \* 0.004925
          - UV Intensity (in mW/cm2)= (ADC\_Voltage - 2.2)/(0.129) + 10
     3. Format Serial Output and Return Information

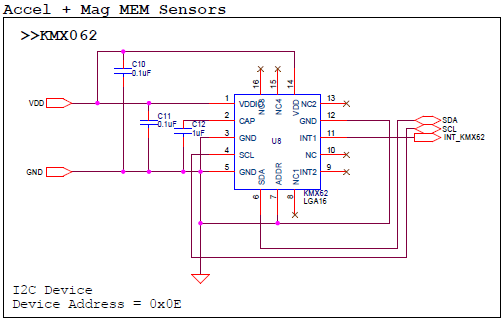
## Hardware Connection for ROHM BU52011 Hall Sensor to the Arduino Uno

* ROHM BU52011HFV – Hall Switch Sensor
  + 
* As seen in the schematic above, this device connects 4 pins.
  + VDD – Connect to Arduino 3.3V output pin on power connector
  + GND – Connect to Arduino GND pin on the power connector
  + GPIO0, GPIO1 – Indicates Hall Switch Output
    - GPIO0 – pin 3
    - GPIO1 – pin 4

## Software Explanation for ROHM BU52011 Hall Sensor to the Arduino Uno

* Code Segments pertaining to this sensor can be found by defining and seeing code within the “***HallSen***” #ifDef statements
* Pseudo-Code Explanation
  1. Define Relevant Variables
  2. Begin setup()
     1. Setup Arduino Pins 3 and 4 and Input pins
  3. Begin loop()
     1. Perform digital reads on pins 3 and 4
        + Output will be either 1 or 0 and will indicate presence of north or south magnetic fields
     2. Format Serial Output and Return Information

## Hardware Connection for Kionix KMX62 Accel+Mag Sensor to the Arduino Uno

* Kionix KMX62 – Digital Accelerometer and Magnetometer
  + 
* As seen in the schematic above, this device connects 6 pins.
  + VDD – Connect to Arduino 3.3V output pin on power connector
  + GND – Connect to Arduino GND pin on the power connector
  + SDA/SCL – I2C Connection, connected to A4 and A5
  + INT\_KMX62 - This pin is used monitor the interrupt pins on the KMX61. This pin is not used in the example application.

## Software Explanation for Kionix KMX62 Accel+Mag Sensor to the Arduino Uno

* Code Segments pertaining to this sensor can be found by defining and seeing code within the “***KMX62***” #ifDef statements
* Pseudo-Code Explanation
  1. Define Relevant Variables
  2. Begin setup()
     1. Initialize the I2C output
        + Note: this sensor uses the “SoftI2CMaster” library to read and write send the I2C commands (<http://arduino.cc/en/reference/wire>). This was required because the “wire” library does not support the “repeated start” condition which is required for I2C reads from the KMX061
        + This Library is not built into the Arduino IDE. Resources for this can be found at the following address:
          - <http://playground.arduino.cc/Main/SoftwareI2CLibrary>
          - <https://github.com/felias-fogg/SoftI2CMaster>
     2. Configure the Accel/Mag Sensor once by performing the following reads
        + 1. Standby Register (0x29), write 0x03 (Turn Off)
        + 2. Self Test Register (0x60), write 0x00
        + 3. Control Register 1 (0x2A), write 0x13
        + 4. Control Register 2 (0x2B), write 0x00
        + 5. ODCNTL Register (0x2C), write 0x00
        + 6. Temp EN Register (0x4C), write 0x01
        + 7. Buffer CTRL Register 1 (0x78), write 0x00
        + 8. Buffer CTRL Register 2 (0x79), write 0x00
        + 9. Standby Register (0x29), write 0x0u
        + Note: Please see the KMX061 Datasheet for additional information on these registers
  3. Begin loop()
     1. Read back the Accelerometer output by reading 6 Bytes starting from address 0x0A. [0][1]...[5]
     2. Format Each of the X, Y, and Z axis acceleration
        + Xout = ([1]<<6) | ([0]>>2)
        + Yout = ([3]<<6) | ([2]>>2)
        + Zout = ([5]<<6) | ([4]>>2)
        + *Note:* to save some time with conversions, please note that these outputs are meant to be unsigned 14bit values. Thus, to make it easier to convert we recommend the following…
          - Xout = (float)(([1]<<8) | ([0]))/4
          - Yout = (float)(([3]<<8) | ([2]))/4
          - Zout = (float)(([5]<<8) | ([4]]))/4
     3. Convert Returned Value to G
        + Axis\_ValueInG = MEMS\_Accel\_axis / 1024
     4. Read back the Mag Sensor output by reading 6 Bytes starting from address 0x12. [0][1]...[5]
     5. Format Each of the X, Y, and Z axis mag data
        + Xout = ([1]<<6) | ([0]>>2)
        + Yout = ([3]<<6) | ([2]>>2)
        + Zout = ([5]<<6) | ([4]>>2)
        + *Note:* to save some time with conversions, please note that these outputs are meant to be unsigned 14bit values. Thus, to make it easier to convert we recommend the following…
          - Xout = (float)(([1]<<8) | ([0]))/4
          - Yout = (float)(([3]<<8) | ([2]))/4
          - Zout = (float)(([5]<<8) | ([4]]))/4
     6. Convert Returned Value to uT
        + Axis\_ValueInuT = MEMS\_Mag\_axis / 0.146
     7. Format Serial Output and Return Information