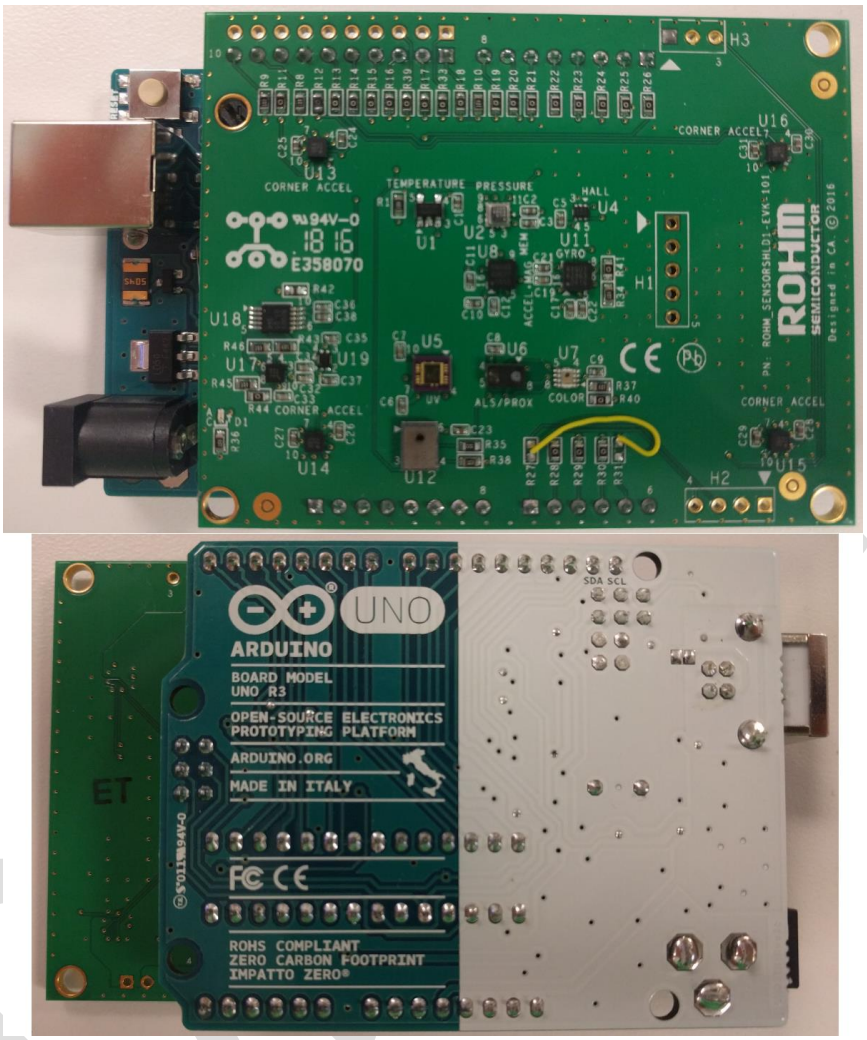
Connecting the Multi-Sensor Shield to the Arduino UNO Platform



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

Table of Contents

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
25 June 2015	Added Additional Sensor Information	A
17 November 2015	Added Additional Sensor Information	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:

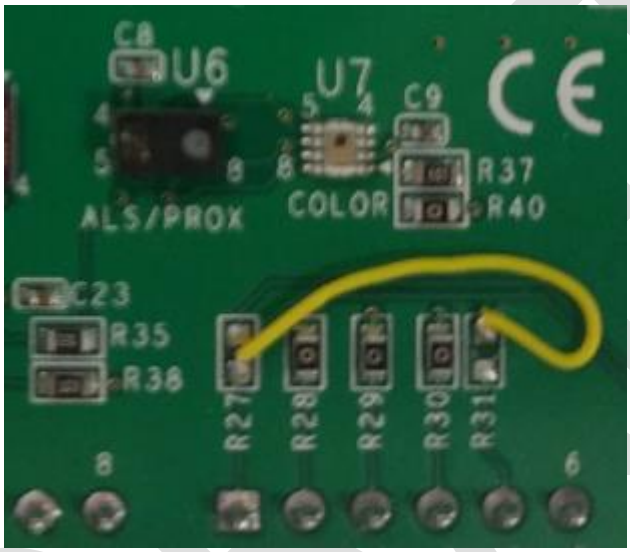
- ROHM BDE0600G – Analog Temperature Sensor
- LAPIS ML8511 – Analog UV Sensor
- ROHM BU52014HFV – Hall Switch Sensor
- KIONIX KMX62 – Digital Accelerometer and Magnetometer
- ROHM BM1383GLV – Digital Barometric Pressure Sensor
- ROHM RPR-0521 – Digital Ambient Light Sensor and Proximity Sensor
- ROHM BH1745 – Digital Color Sensor
- KIONIX KX122 – Digital Accelerometer
- KIONIX KXG03 – Digital Gyroscopic Sensor

Getting Started

1. Initial Setup
    a. ROHM Multi-Sensor Shield Board
        i. For this guide we will connect ROHM's Multi-Sensor Shield board directly to the Arduino UNO using the standard Arduino Shield Headers
    b. The following are recommended for using the Arduino Uno for this guide
        i. PC with Arduino IDE
        ii. USB Cable to power and monitor serial communication
        iii. 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.

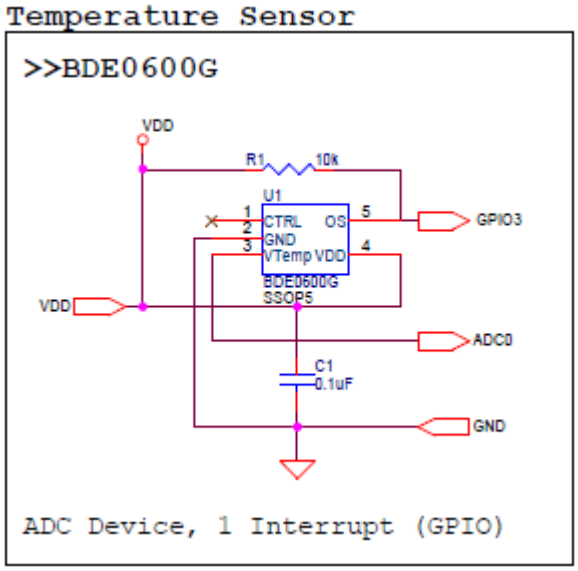
Connecting ROHM Multi-Sensor Shield to the Arduino UNO

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 the UV sensor's ADC output already existing on the board. Thus, in order to reroute this on our board, we suggest the following rework…
    - Disconnect existing nets by removing R27, R31
    - Reconnect UV sensor ADC by connecting 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
  - GPIO3 – This pin is used to trigger the thermostat output function. This pin is not used in the example application.

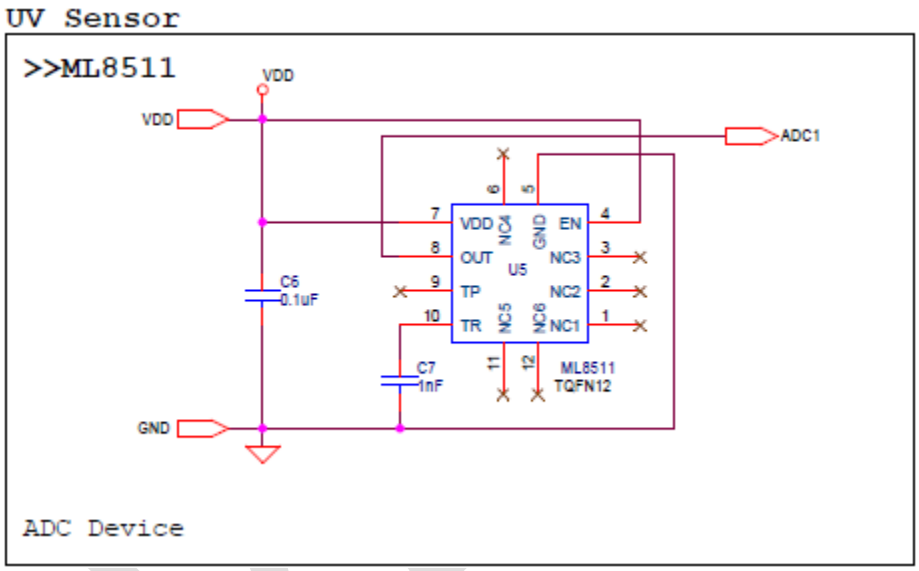
Software Explanation for ROHM BDE0600G Temp Sensor to the Arduino Uno

- 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 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 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
  - ADC1 – 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

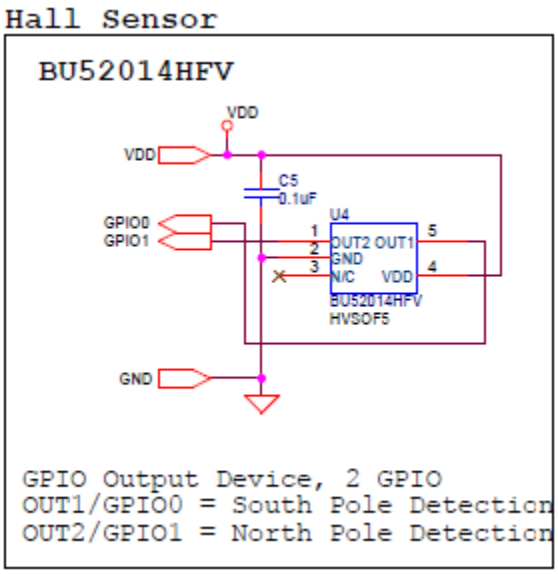
- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***AnalogUV***" #ifDef statements
- Pseudo-Code Explanation
  1. Define Relevant Variables
  2. Begin Loop()
     1. Read back the Analog Sensor Output from Pin A0
        - Note: Default Arduino Reference 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/cm^2)
  - Temp Known Point = 2.2V @ 10mW/cm^2
- Calculation
  - ADC_Voltage = (sensorValue / 670) * 3.3V
  - ADC_Voltage = sensorValue * (3.3V/670)
  - ADC_Voltage = sensorValue * 0.004925
  - UV Intensity (in mW/cm^2)= (ADC_Voltage - 2.2)/(0.129) + 10

3. Format Serial Output and Return Information

Hardware Connection for ROHM BU52014 Hall Sensor to the Arduino Uno

- ROHM BU52014HFV – 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: Indicates South Pole Detection
    - GPIO1: Indicates North Pole Detection

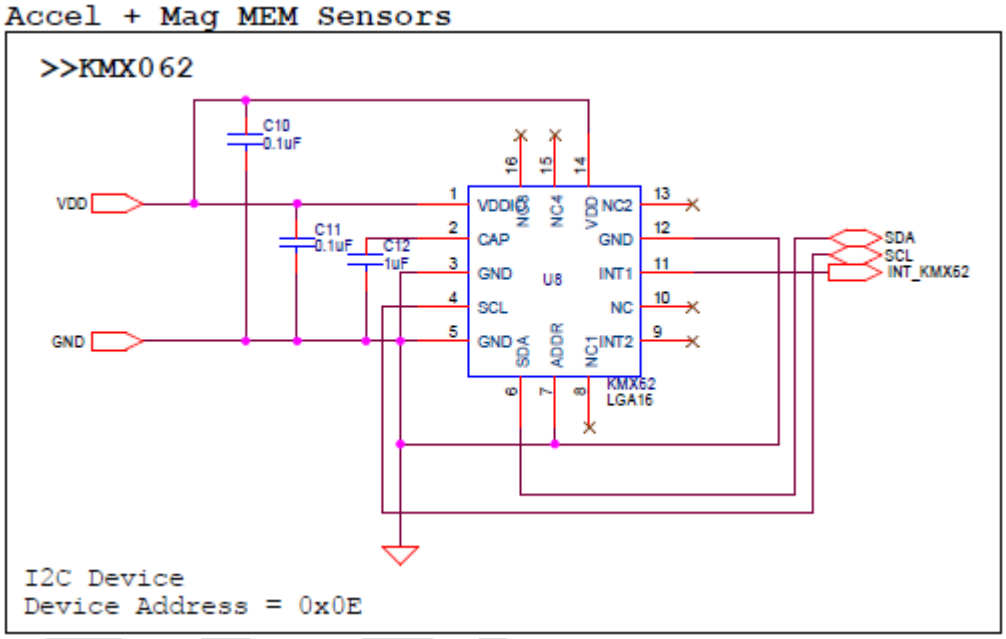
Software Explanation for ROHM BU52014 Hall Sensor to the Arduino Uno

- 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 2 and 3 and Input pins

3. Begin loop()
    1. Perform digital reads on pins 2 and 3
        - 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 KMX62
- 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

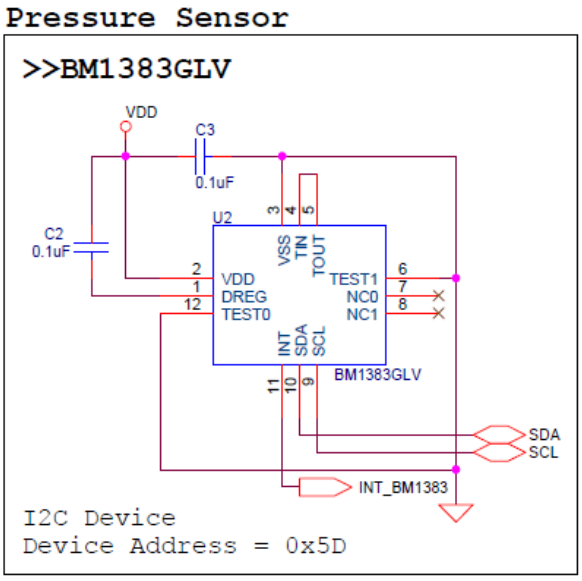
2. Configure the Accel/Mag Sensor once by performing the following reads
   - 1. CNTL2 Register (0x3A), write 0x5F (4g Sensor Resolution, Max RES, Enable Temp/Mag/Accel)
     - Note: Please see the KMX62 Datasheet for additional information on these registers

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 / 2048
   4. Read back the Magnetometer output by reading 6 Bytes starting from address 0x10. [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

Hardware Connection for ROHM BM1383GLV Barometric Pressure Sensor to the Arduino Uno

- ROHM BM1383GLV Barometric Pressure Sensor
  - 

- 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_BM1383 - This pin is used monitor the interrupt pins on the BM1383. This pin is not used in the example application.

Software Explanation for ROHM BM1383GLV Barometric Pressure Sensor to the Arduino Uno

- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***Pressure***" #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 BM1382 pressure sensor

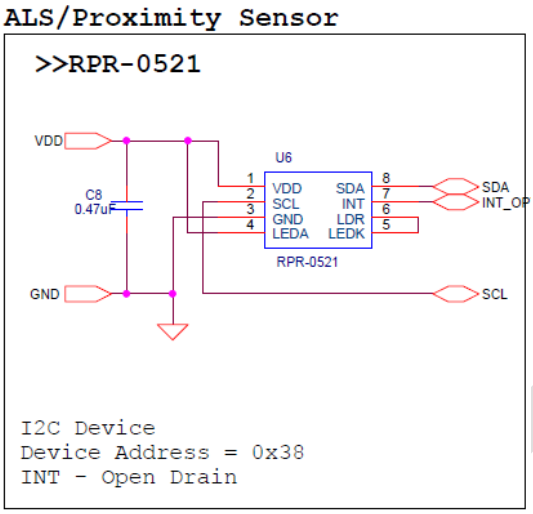
- This Library is not built into the Arduino IDE. Resources for this can be found at the following address:
  - http://playground.arduino.cc/Main/SoftwareI2CLibrary
  - https://github.com/felias-fogg/SoftI2CMaster

2. Configure the Pressure Sensor once by performing the following writes:
   - 1. PWR_DOWN Register (0x12), write (0x01)
   - 2. SLEEP Register (0x13), write (0x01)
   - 3. Mode Control Register (0x14), write (0xC4)
   - Note: Please see the BM1383 datasheet for additional information on these registers

3. Begin loop()
   1. Read back the pressure and temperature output by reading 5 Bytes starting from address 0x1A. [0][1]...[4]
   2. Format raw temperature and pressure
      - Raw ADC Pressure (Integer Portion) = ([0]<<3) | ([1]>>5)
      - Raw ADC Pressure (Decimal Portion) = (([1] & 0x1f) << 6 | (([2] >> 2)));
      - Raw ADC Temp = ([3]<<8) | ([4])
   3. Convert Returned Raw Value to Real Values
      - Temp in degC = Raw ADC Temp / 32
      - Pressure in hPa =
        - (Raw ADC Pressure(int)) + (Raw ADC Pressure(dec)*2^{-11})
   4. Format Serial Output and Return Information

Hardware Connection for ROHM RPR-0521 3-in-1 Ambient Light Sensor, Proximity Sensor, and IR LED Combo Package for the Arduino Uno

- ROHM RPR-0521 ALS/PROX Sensor



- 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_Optical - This pin is used monitor the interrupt pins on the RPR-0521. This pin is not used in the example application.

Software Explanation for ROHM RPR-0521 3-in-1 Ambient Light Sensor, Proximity Sensor, and IR LED Combo Package to the Arduino Uno

- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***ALSProx***" #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 RPR-0521 ALS/PROX Sensor
        - 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 Ambient Light Sensor and Proximity Sensor once by performing the following writes:

- 1. Mode Control Register (0x41), write (0xC6)
- 2. ALS_PS_Control Register (0x42), write (0x03)
- 3. Persist Register (0x43), write (0x20)
- Note: Please see the RPR-0521 datasheet for additional information on these registers

3. Begin loop()
    1. Read back the Ambient Light Sensor and Proximity Sensor output by reading 6 Bytes starting from address 0x44. [0][1]...[5]
    2. Format Raw ambient light and proximity
        - RAW PROX ADC = ([1]<<8) | ([0])
        - RAW ALS DATA0 = ([3]<<8) | ([2])
        - RAW ALS DATA1 = ([5]<<8) | ([4])
    3. Convert Raw Returned Value to Real Values
        - ALS can be returned by using the following code block to return ALS value in lx (RPR0521_ALS_OUT)

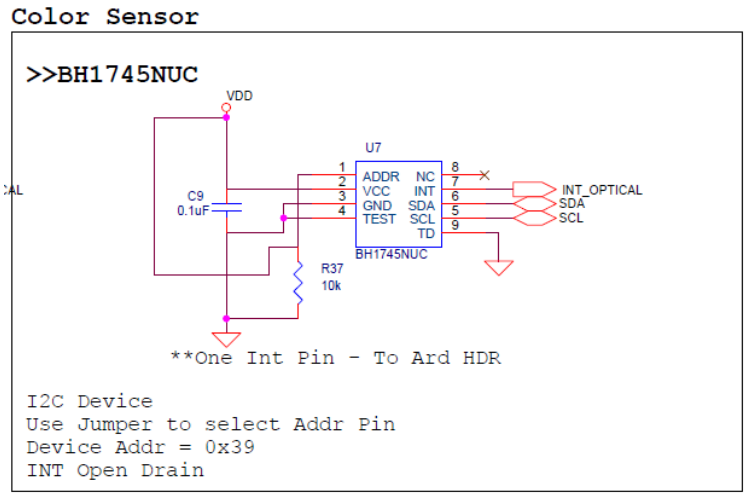
```
RPR0521_ALS_DataRatio = (float) RPR0521_ALS_D1_RAWOUT /
(float)RPR0521_ALS_D0_RAWOUT;

if(RPR0521_ALS_DataRatio < 0.595){
      RPR0521_ALS_OUT = (1.682*(float) RPR0521_ALS_D0_RAWOUT - 1.877*(float)
      RPR0521_ALS_D1_RAWOUT);
}
else if(RPR0521_ALS_DataRatio < 1.015){
      RPR0521_ALS_OUT = (0.644*(float) RPR0521_ALS_D0_RAWOUT - 0.132*(float)
      RPR0521_ALS_D1_RAWOUT);
}
else if(RPR0521_ALS_DataRatio < 1.352){
      RPR0521_ALS_OUT = (0.756*(float) RPR0521_ALS_D0_RAWOUT - 0.243*(float)
      RPR0521_ALS_D1_RAWOUT);
}
else if(RPR0521_ALS_DataRatio < 3.053){
      RPR0521_ALS_OUT = (0.766*(float) RPR0521_ALS_D0_RAWOUT - 0.25*(float)
      RPR0521_ALS_D1_RAWOUT);
}
else{
      RPR0521_ALS_OUT = 0;
}
```

    4. Format Serial Output and Return Information

Hardware Connection for ROHM BH1745NUC Color Sensor for the Arduino Uno

- ROHM RPR-0521 ALS/PROX Sensor



- 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_Optical - This pin is used monitor the interrupt pins on the RPR-0521. This pin is not used in the example application.

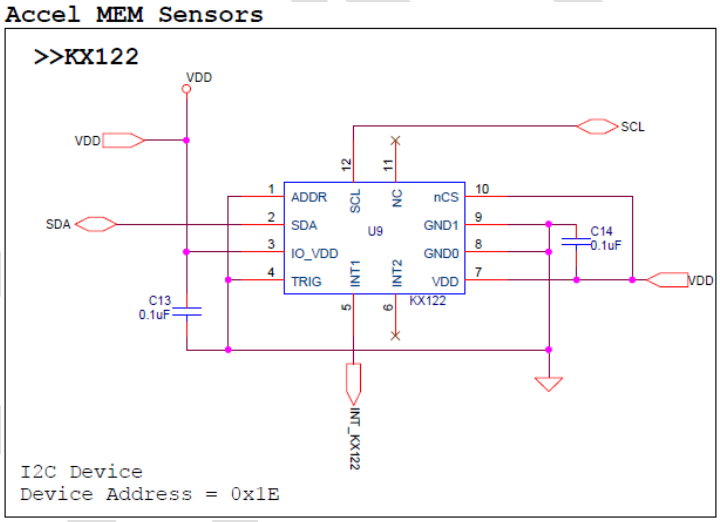
Software Explanation for ROHM BH1745NUC Color Sensor to the Arduino Uno

- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***Color***" #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 BH1745 Color Sensor
        - 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 Color Sensor once by performing the following writes:
        - 1. Persistence Register (0x61), write (0x03)
        - 2. Mode Control 1 Register (0x41), write (0x00)
        - 3. Mode Control 2 Register (0x42), write (0x92)

- 4. Mode Control 3 Register (0x43), write (0x02)
- Note: Please see the color sensor datasheet for additional information on these registers

3. Begin loop()
    1. Read back the Color Sensor output by reading 6 Bytes starting from address 0x50. [0][1]...[5]
    2. Format Raw color
        - RED ADC = ([1]<<8) | ([0])
        - GREEN ADC = ([3]<<8) | ([2])
        - BLUE ADC = ([5]<<8) | ([4])
    3. Convert Raw Returned Value to Real Values
        - Coming Soon!
    4. Format Serial Output and Return Information

Hardware Connection for Kionix KX122 Accelerometer Sensor for the Arduino Uno

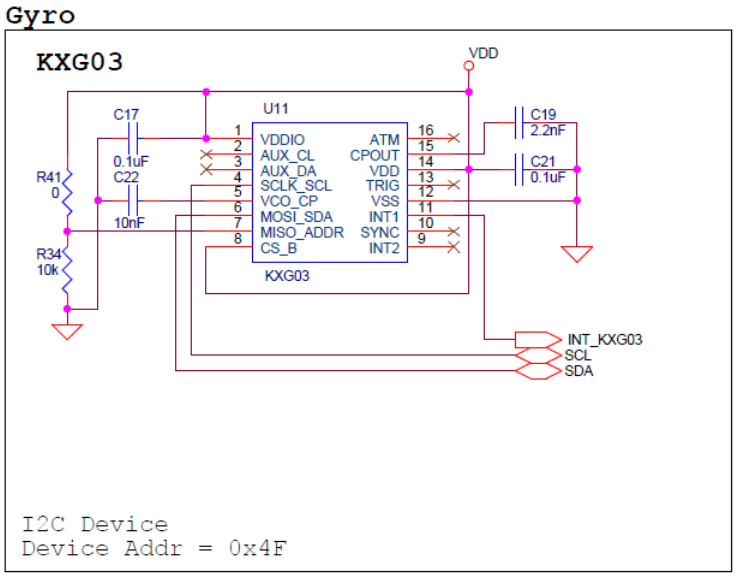
- Kionix KX122 Accelerometer Sensor
    - 

- 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_KX122 - This pin is used monitor the interrupt pins on the KX122. This pin is not used in the example application.

Software Explanation for Kionix KX122 Accelerometer Sensor to the Arduino Uno

- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***KX122***" #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 KX122 accelerometer sensor
            - 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 Accelerometer Sensor once by performing the following writes:
            - 1. CNTL1 Register (0x18), write (0x41)
            - 2. ODCNTL Register (0x1B), write (0x02)
            - 3. CNTL3 Register (0x1A), write (0xD8)
            - 4. TILT_TIMER Register (0x22), write (0x01)
            - 5. CNTL1 Register (0x18), write (0xC1) (Enable bit to ON)
            - Note: Please see the KX122 datasheet for additional information on these registers
    3. Begin loop()
        1. Read back the Accelerometer output by reading 6 Bytes starting from address 0x06. [0][1]...[5]
        2. Format Raw acceleration
            - ACCEL X Axis Raw = ([1]<<8) | ([0])
            - ACCEL Y Axis Raw = ([3]<<8) | ([2])
            - ACCEL Z Axis Raw = ([5]<<8) | ([4])
        3. Convert Raw Returned Value to g
            - ACCEL X in g = ACCEL X Axis Raw / 16384
            - ACCEL X in g = ACCEL Y Axis Raw / 16384
            - ACCEL X in g = ACCEL Z Axis Raw / 16384
        4. Format Serial Output and Return Information

Hardware Connection for Kionix KXG03 Gyroscopic Sensor for the Arduino Uno

- Kionix KXG03 Gyroscopic Sensor
  - 

- 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_KXG03 - This pin is used monitor the interrupt pins on the KXG03. This pin is not used in the example application.

Software Explanation for Kionix KXG03 Gyroscopic Sensor to the Arduino Uno

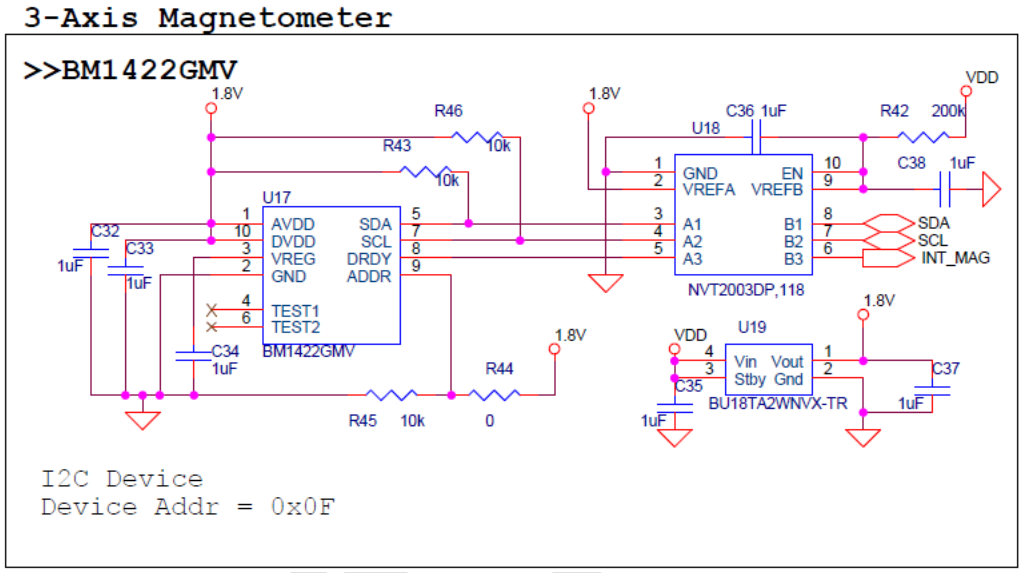
- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***KXG03***" #ifDef statements
- Pseudo-Code Explanation
  1. Define Relevant Variables
  2. Begin setup()
     1. Initialize the I2C output
        - Note: this sensor uses the "SoftI2CMaster" library to read 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 KXG03 Gyro sensor
        - 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 Gyroscopic Sensor once by performing the following writes:
    - 1. STDBY Register (0x43), write 0x00
    - Note: Please see the KXG03 datasheet for additional information on these registers

3. Begin loop()
    1. Read back the Gyroscopic Sensor output by reading 6 Bytes starting from address 0x02. [0][1]...[5]
    2. Format Raw Gyroscopic data
        - Gyro_X_RawOUT = ([1]<<8) | ([0])
        - Gyro_Y_RawOUT = ([3]<<8) | ([2])
        - Gyro_Z_RawOUT = ([5]<<8) | ([4])
    3. Cancel for offset value
        - Measure RawOUT data 10 times in order to proofread offset value
        - Calculate average offset value. This data is defined as proofread data.
        - Subtract proofread data from each RawOUT data.
    4. Convert Returned Value to deg/sec
        - Gyro_X = (float)KXG03_Gyro_X_RawOUT2 * 0.007813 + 0.000004
        - Gyro_Y = (float)KXG03_Gyro_Y_RawOUT2 * 0.007813 + 0.000004
        - Gyro_Z = (float)KXG03_Gyro_Z_RawOUT2 * 0.007813 + 0.000004
    5. Format Serial Output and Return Information

Hardware Connection for ROHM BM1422 Magnetometer Sensor for the Arduino Uno

- ROHM BM1422Magnetometer Sensor
  - 

- As seen in the schematic above, this device is slightly more complicated than the other devices on this shield.
  - Part Explanation:
    - U17 = BM1422GMV Magnetometer
      - 1.8V max input voltage… therefore, we can't use the 3.3V rail native to the system. U18 and U19 are added to protect the magnetometer from the 3.3V system voltage.
    - U18 = NVT2003DP Level Shifter for I2C and Interrupt lines
      - Used to reduce I2C interface and interrupt pin connection's 3.3V signals from the Arduino board to 1.8V to the sensor.
    - U19 = BU18TA2WNVX LDO Regulator
      - Used to source power to the magnetometer and the 1.8V logic level side of the level shifter U18
- When looking at this system together, we can see than ultimately, there are 5 connections
  - 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_MAG - This pin is used monitor the interrupt pins on the BM1422GMV. This pin is not used in the example application.

Software Explanation for ROHM BM1422 Magnetometer Sensor to the Arduino Uno

- Code Segments pertaining to this sensor can be found by defining and seeing code within the "***MagField***" #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 sensor
            - 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 Magnetometer Sensor once by performing the following writes:
            - 1. CNTL1 Register (0x1B), write 0xC0
            - 2. CNTL4 Register (0x5C), write 0x00
            - 3. CNTL4 Register (0x5D), write 0x00
            - 4. CNTL3 Register (0x1D), write 0x40
            - Note: Please see the BM1422GMV datasheet for additional information on these registers
    3. Begin loop()
        1. Read back the Magnetometer Sensor output by reading 6 Bytes starting from address 0x10. [0][1]...[5]
        2. Format Raw Magnetometer data
            - Mag_X_RawOUT = ([1]<<8) | ([0])
            - Mag_Y_RawOUT = ([3]<<8) | ([2])
            - Mag_Z_RawOUT = ([5]<<8) | ([4])
        3. Format Serial Output and Return Information