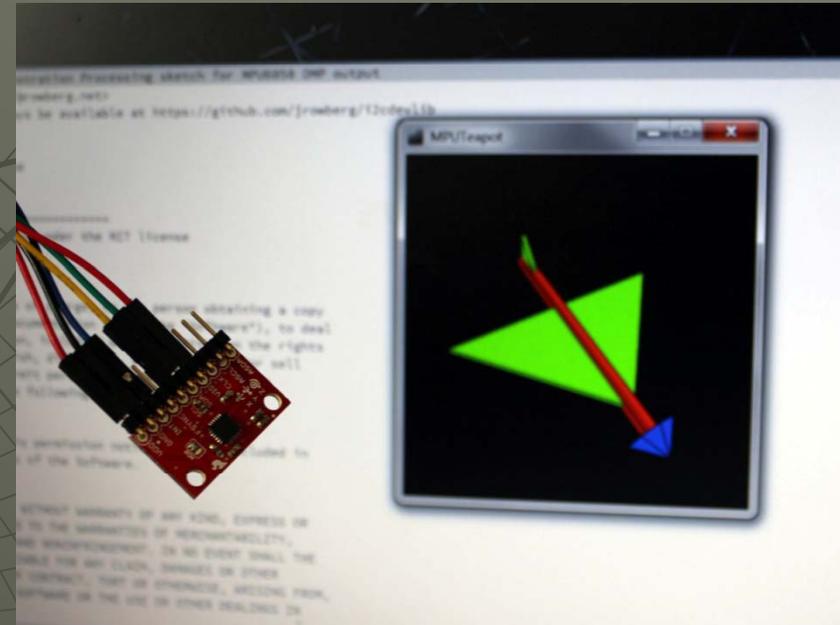
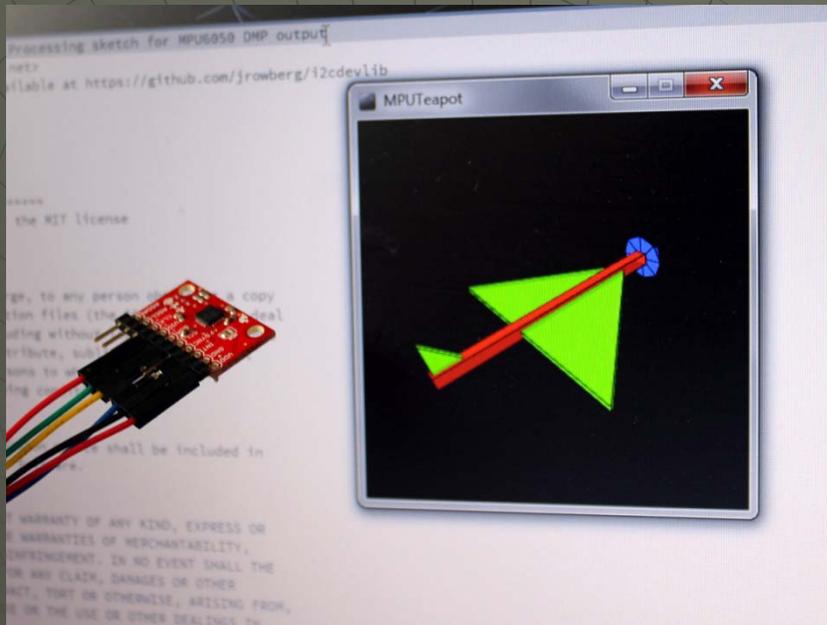


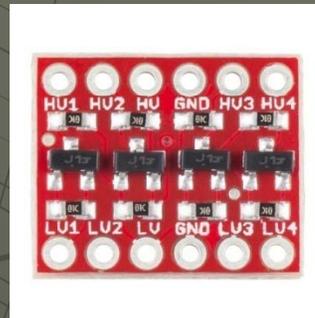
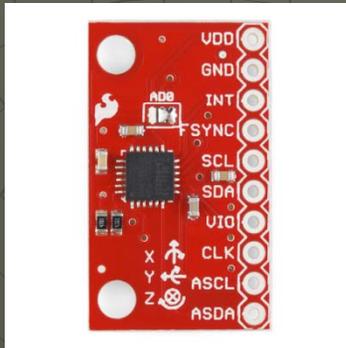


Triple Axis Accelerometer-Gyro with Arduino Uno



This tutorial uses the MPU-6050 triple axis accelerometer-gyro (6 degrees of freedom) connected to the Arduino Uno to continuously measure the attitude of a vehicle (pitch, yaw, and roll of your rocket). This data can be recorded on an SD card or radioed back to a ground station

Hardware Required



- Three Axis MPU-6050 Accelerometer-Gyro: From Sparkfun, Sainsmart, Robotshop and many more
 - Purchase: <https://www.sparkfun.com/products/11028>
 - Schematic: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/IMU/Triple_Axis_Accelerometer_-_Gyro_Breakout_-_MPU-6050-v12.pdf
- Bidirectional Level Shifter (converter) 3.3V to 5V from Sparkfun, Adafruit, and others
 - Purchase: <https://www.sparkfun.com/products/12009>
 - Schematic: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf
- Arduino Uno: From Arduino, Amazon, Sparkfun, MP3Cars, many more:
 - Purchase: <http://www.amazon.com/Arduino-UNO-board-DIP-ATmega328P/dp/B006H06TVG>
 - Schematic: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf
- A PC or laptop



Data Sheets

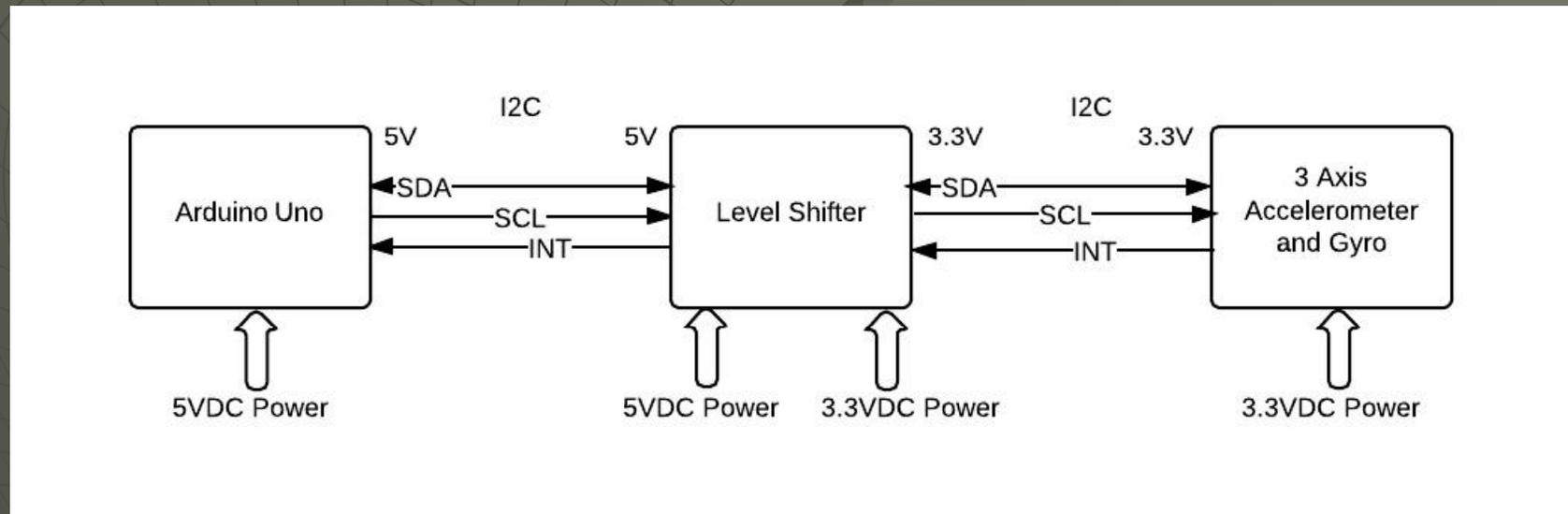
- ◆ Arduino Uno uses the ATmega328 microcontroller
 - Datasheet: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet.pdf
 - ◆ Although the ATmega328 can run at lower voltage, the Arduino Uno uses $V_{CC} =$ at 5V
 - ◆ From the data sheet, at $V_{CC} = 5VDC$, V_{OH} is a minimum of 4.2V
 - ◆ From the data sheet, V_{IH} is a minimum of $0.7V_{CC} = 3.5V$
- ◆ The Triple Axis Accelerometer and Gyro uses an InvenSense MPU-6050
 - Datasheet: <http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf>
 - ◆ V_{DD} is 2.375V – 3.46V – since we have 3V available on the Uno we'll use that
 - ◆ VLOGIC input is a maximum of $V_{DD} + 0.5V$ or 3.5V

Since the output from the Arduino Uno is higher than the allowed input to the accelerometer, and the minimum High input to the Arduino Uno is less than the maximum HIGH output from the accelerometer, we will need a level shifter. It may work for a while without it, but not reliably and the accelerometer will probably be damaged from overvoltage

- Datasheet: <http://www.fairchildsemi.com/ds/BS/BSS138.pdf>

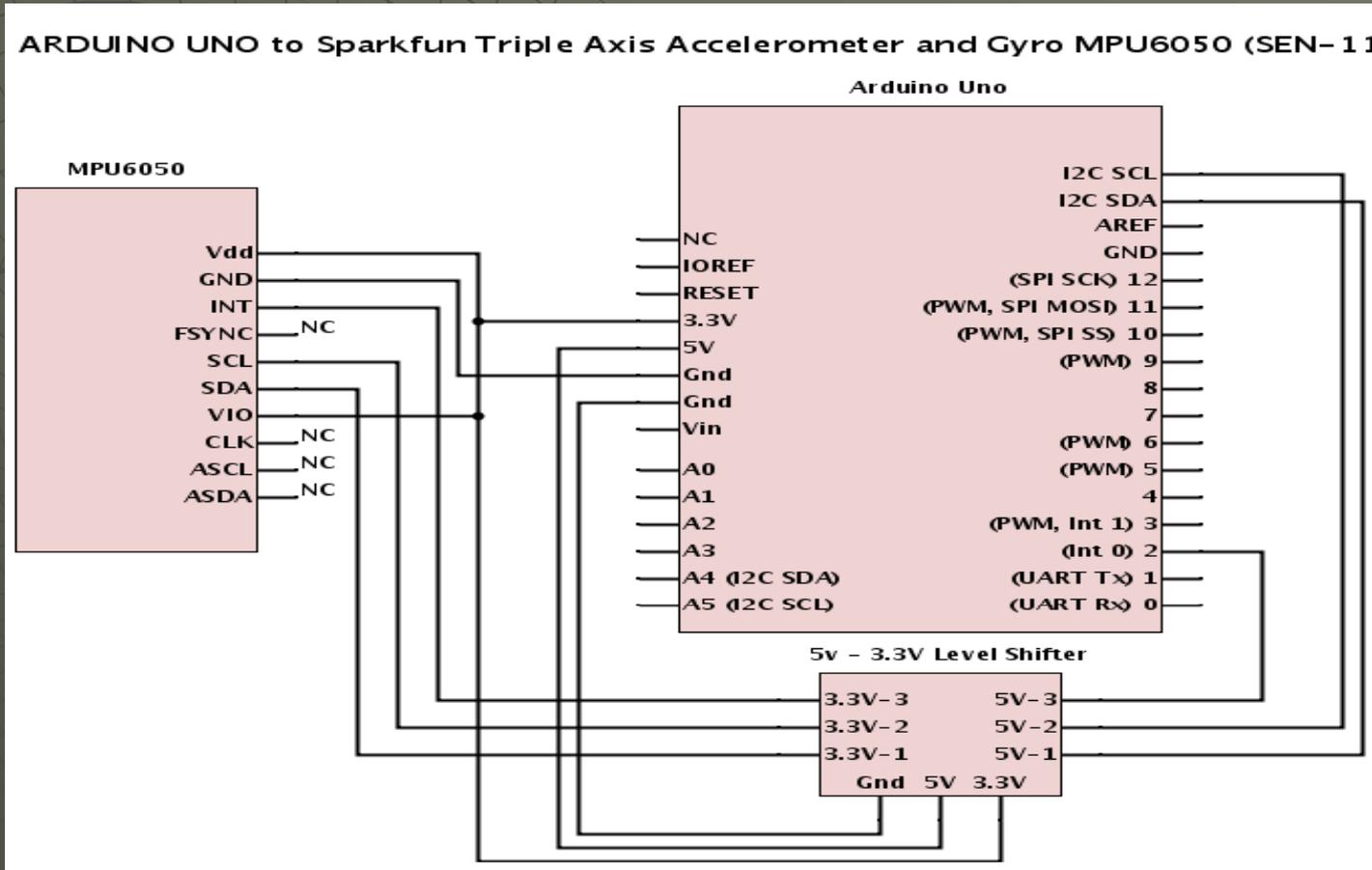


Connecting the Uno and 3-Axis Accel-Gyro Block Diagram



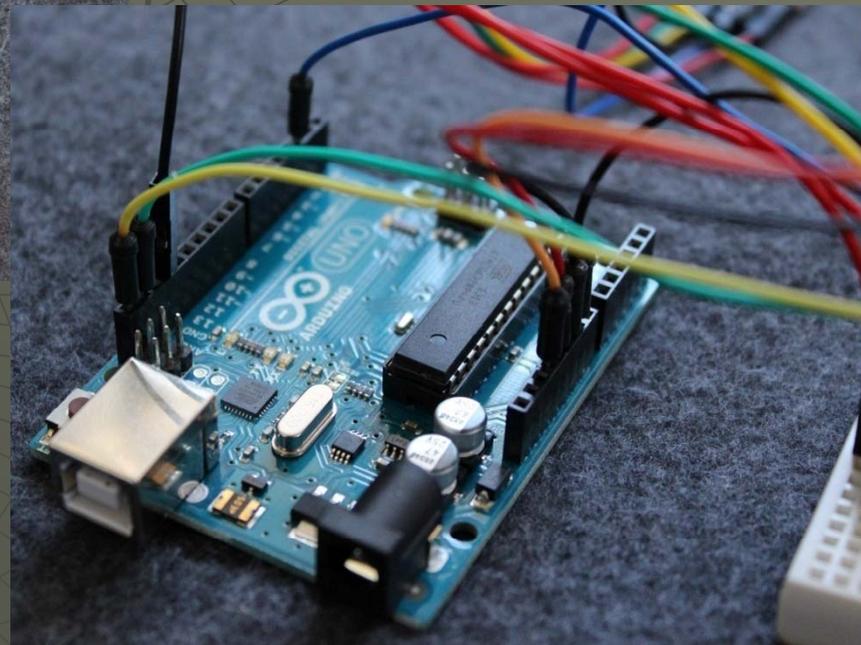
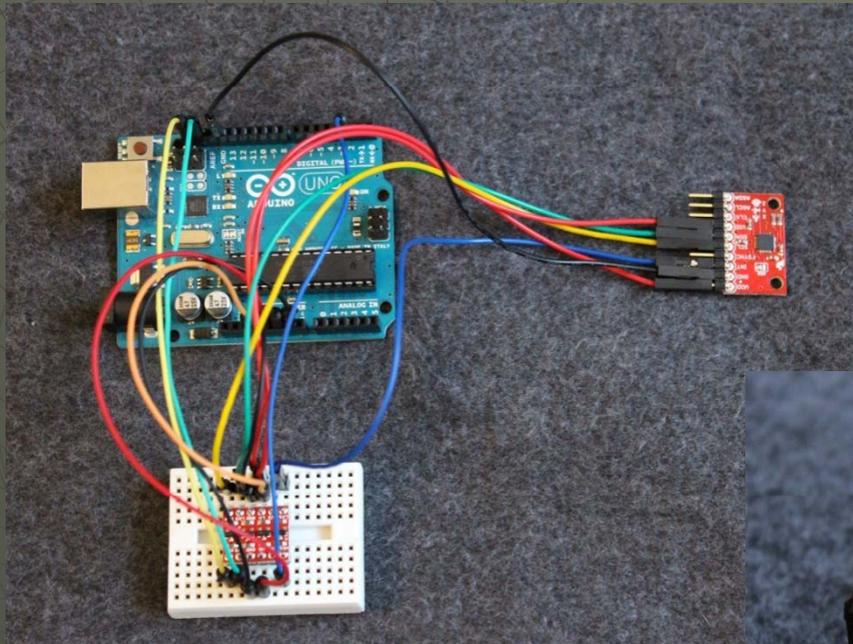


Connecting the Uno and 3-Axis Accel-Gyro Schematic



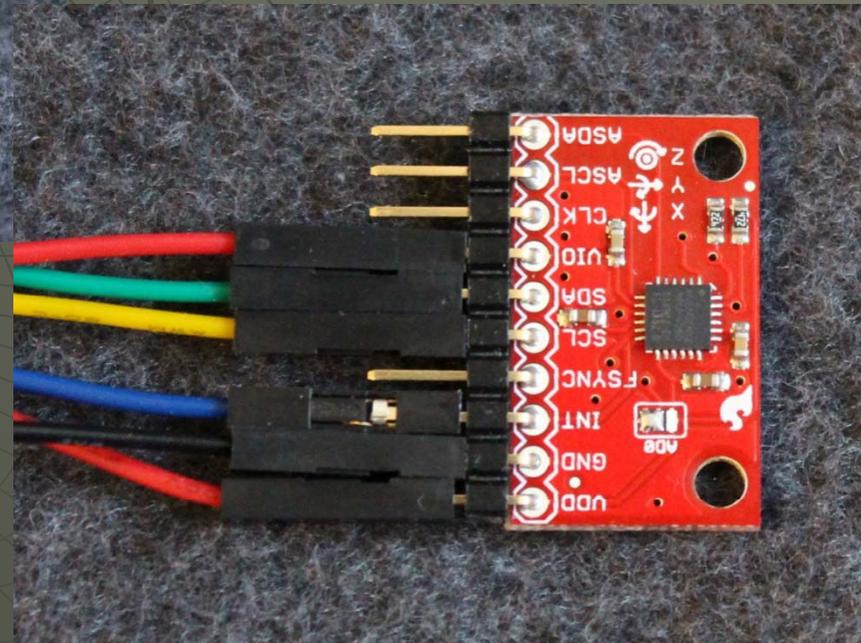
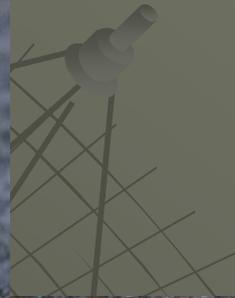
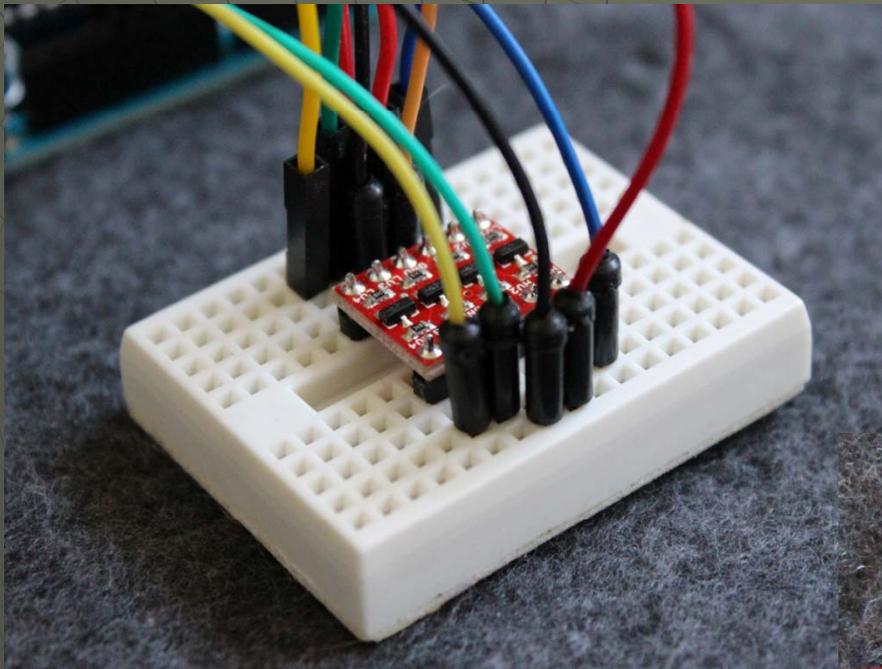


Connecting the Uno and 3-Axis Accel-Gyro





Connecting the Uno and 3-Axis Accel-Gyro





Arduino Software Required

```
MPU6050_DMP6 | Arduino 1.0.5
File Edit Sketch Tools Help

MPU6050_DMP6
#include <helper_9math.h>
#include <MPU6050.h>
#include <MPU6050_6axis_MotionApps20.h>
#include <MPU6050_6axis_MotionApps41.h>

// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2.0)
// 8/21/2012 by Jeff Rowberg <jrowberg@me.com>
// Updates should (hopefully) always be available at <https://github.com/jrowberg/i2cdevlib>
//
// Changelog:
// 2013-05-08 - added seamless Fastwire support
//             - added note about gyro calibration
// 2012-06-21 - added note about Arduino 1.0.1 + Leonardo compatibility error
// 2012-06-20 - improved FIFO overflow handling and simplified read process
// 2012-06-19 - completely rearranged DMP initialization code and simplification
// 2012-06-13 - pull gyros and accel data from FIFO packet instead of reading directly
// 2012-06-09 - fix broken FIFO read sequence and change interrupt detection to RISING
// 2012-06-05 - add gravity-compensated initial reference frame acceleration output
//             - add 3D math helper file to DMP6 example sketch
//             - add Euler output and Yaw/Pitch/Roll output formats
// 2012-06-04 - remove accel offset clearing for better results (thanks Shuang Lee)
// 2012-06-01 - fixed gyro sensitivity to be 2000 deg/sec instead of 250
// 2012-05-30 - basic DMP initialization working

/* *****
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg
*/
```

Arduino Integrated Development Environment (IDE):

http://arduino.cc/en/main/software#_Uy4WgU1DUUpA

Arduino Sketch MPU6050_DMP.ino and I2Cdev Library and MPU6050 Library:

<https://github.com/jrowberg/i2cdevlib> (select "Download .zip" on the right below the list of files)

- MPU6050_DMP.ino is in i2cdevlib-master->Arduino->mpu6050->Examples->MPU6050_DMP6
- I2Cdev library is in i2cdevlib-master->Arduino->I2Cdev
- MPU6050 library is in i2cdevlib-master->Arduino->MPU6050

Arduino Calibration Sketch MPU6050_calibration.ino:

<http://www.i2cdevlib.com/forums/topic/112-another-auto-offset-calibration-sketch/?hl=calibration>

MPU6050 Library Documentation (optional download):

http://www.i2cdevlib.com/docs/html/class_m_p_u6050.html



Install Arduino IDE and Libraries

```
Working Directory Tree:  
(My) Documents  
  Arduino  
    libraries  
      I2CDev  
      MPU6050  
  sketchbook  
    MPU6050
```

Note: You can skip this step if you have already installed the IDE

STEP 1: INSTALL THE ARDUINO IDE

- Download the IDE (currently named "arduino-1.0.5-r2-windows.exe") and click on it to install
- It will be installed in the Program Files directory
- Your sketches and libraries will go in the Arduino folder in (my) Documents
- Also allow the installation of the device driver software

STEP 2: INSTALL THE TWO LIBRARIES

- Library install instructions are here:
<http://arduino.cc/en/Guide/Libraries#.Uy4bYU1OUpA>
- To automatically install a downloaded .zip file library, start the Arduino IDE and click on: SKETCH->IMPORT LIBRARY->ADD LIBRARY then navigate to where the libraries were downloaded and click on the library (.zip file or folder) – check that the library has been added under "contributed" in the list under SKETCH->IMPORT LIBRARY->ADD LIBRARY
- To manually install a downloaded library, unzip it and move the folder containing all files into the "libraries" folder in the (My) Documents\Arduino\libraries, then restart the IDE



```
MPU6050_DMP6 [Arduino 1.0.5]
File Edit Sketch Tools Help

MPU6050_DMP6
#include <I2Cdev.h>
#include <MPU6050.h>
#include <MPU6050_6Axis_MotionApps20.h>
#include <MPU6050_6Axis_MotionApps41.h>

// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2.0)
// 8/21/2012 by Jeff Rowberg <jeffrowberg@gmail.com>
// Updates should (hopefully) always be available at https://github.com/rowberg/i2cdevlib
//
// Changelog:
// 2012-05-08 - added seamless FastWrite support
//           - added note about gyro calibration
// 2012-06-21 - added note about Arduino 1.0.1 + Leonardo compatibility error
// 2012-06-20 - improved FIFO overflow handling and simplified read process
// 2012-06-19 - completely reorganized DMP initialization code and simplification
// 2012-06-13 - pull gyro and accel data from FIFO packet instead of reading directly
// 2012-06-09 - fix broken FIFO read sequence and change interrupt detection to FIFO
// 2012-06-05 - add gravity-compensated initial reference frame acceleration output
//           - add 3D math helper file to DMP example sketch
//           - add Euler output and Two/Fitch/Poll output formats
// 2012-06-04 - remove accel offset clearing for better results (thanks Mungun Lee)
// 2012-06-01 - fixed gyro sensitivity to be 2500 degrees instead of 250
// 2012-05-30 - basic DMP initialization working

/* *****
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg
*/
```

Arduino Sketch Required

1. Find the folder containing the Arduino Sketch named "MPU6050_DMP6.ino" . You will find it in the i2cdevlib-master .zip file you previously downloaded at:
i2cdevlib-master->Arduino->mpu6050->Examples->MPU6050_DMP6
2. Copy the MPU6050 FOLDER into the "sketchbook" folder at (My)Documents->Arduino->Sketchbook (the folder contains the MPU6050_DMP6.ino sketch)
3. Important: You may need to make two changes to the sketch as downloaded (more instructions on this later):
 1. The sketch defaults to an I2c address of 0x68 – the default for Sparkfun. You may need 0x69 for other devices, or if you have modified the board
 2. There is a section in the code with several output options: "#define OUTPUT_READABLE_YAWPITCHROLL" is the default. To use with the teapot sketch comment (add "//" at the beginning of that line) and uncomment (remove the "//") on the line that reads "#define OUTPUT_TEAPOT"

AIAA OC Rocketry

AIAA OC Section – NAR #718



PC Software Required

```
MPU6050 (Processing 2.1.1)
File Edit Sketch Tools Help

MPU6050
// I2C device (class I2Cdev) demonstration Processing sketch for MPU6050 DMP output
// v1.0/2012 by Jeff Rowberg (jeff@rowberg.net)
// Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
//
// Changelog:
//   2012-06-20 - Initial release

/* =====
 * ATTENTION: Official Arduino code is placed under the MIT License
 * Copyright (c) 2011 Jeff Rowberg
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 */
```

Any decent Serial Communications Program or terminal emulator OR use the Serial Monitor built into the Arduino IDE

Processing Integrated Development Environment (Processing is a programming environment much like the one we have for Arduino – and was developed originally for the visual arts):

<https://processing.org/download/>

MPU Teapot Sketch (called the teapot sketch as earlier versions showed a teapot):

https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050/Examples/MPU6050_DMP6/Processing

Requires Toxi Library:

<http://hg.postspectacular.com/toxiclibs/downloads/>

More information about this project:

<http://www.i2cdevlib.com/>

Version 1.1 May 19, 2014



Install the Processing IDE and Libraries

Working Directory Tree:
(My) Documents
 Processing
 libraries
 toxiclibs_p5
 toxiclibscore
 sketchbook
 MPUTEapot

STEP 1: INSTALL THE PROCESSING IDE

- Download the IDE (currently named “processing-2.2-windows64.zip or processing-2.2-windows32.zip” depending upon your system)
- Unzip the file
- Start processing by clicking on the processing.exe file – it needs no formal installation
- It is more convenient if you manually add a link on your desktop

STEP 2: INSTALL THE TOXI LIBRARIES

- Library install instructions are here:

http://wiki.processing.org/w/How_to_Install_a_Contributed_Library

- To manually install a downloaded library, unzip it and move the folder containing all files into the “libraries” folder in the (My) Documents\Processing\libraries, then restart the Processing IDE

Running the Arduino Sketch

- 1 - Use the Arduino IDE to open the MPU6050_DMP6.ino sketch
- 2 – Make certain you have selected the Arduino Uno (Tools->Board)
- 3 – Make certain you have selected the correct serial port (Tools->Serial Port)
- 4 - Verify that your MPU 6050 breakout board I2C address is set to 0x68
- 5 - If the ADO jumper looks like the picture you are set to 0x68

```
// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// ADO low = 0x68 (default for SparkFun breakout board)
// ADO high = 0x69
```

```
MPU6050 mpu;
```

```
//MPU6050 mpu(0x69); // <-- use for ADO high
```

- 6 – Find the section that changes the form of the output:

```
"uncomment "OUTPUT_READABLE_QUATERNION" if you want to see the actual quaternion..."
```

```
"//#define OUTPUT_READABLE_QUATERNION"
```

```
"uncomment "OUTPUT_READABLE_EULER" if you want to see Euler angles (in degrees)"
```

```
"//#define OUTPUT_READABLE_EULER"
```

```
etc....
```

- 7 - The only one of these that should be uncommented (remove the "//" at the beginning of the line with the #define) is the one that reads:

```
// uncomment "OUTPUT_TEAPOT" if you want output that matches the
```

```
// format used for the InvenSense teapot demo
```

```
#define OUTPUT_TEAPOT
```

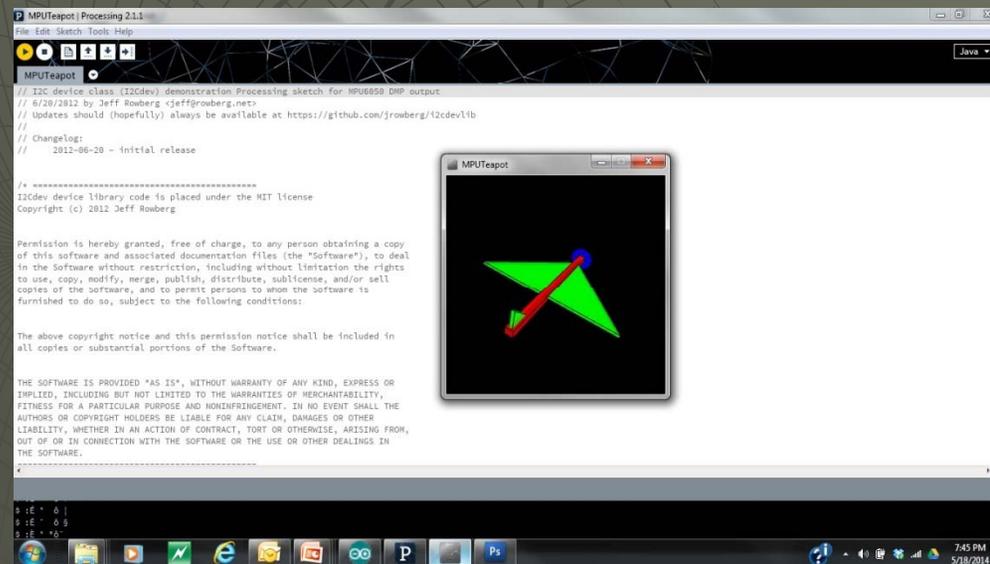
- 8 - Connect your circuit to your PC via the USB cable

- 9 - Compile and upload the sketch (click the right facing arrow at the top)



Running the Processing Sketch

- 1 – Move the MPUteapot folder containing the MPUteapot.pde sketch into the sketchbook folder under Processing ((my)documents->Processing->sketchbook)
- 2 - Start the Processing (processing.exe in the folder where you unzipped the downloaded Processing program) program to open the MPUteapot.pde sketch
- 3 - Compile and start the sketch (the right arrow in the upper left)
- 4 – You should see the small screen with a simple image of an airplane (there is no teapot – there was a previous version that did show a teapot)
- 5 – The airplane image will probably slowly wander in one direction – you will need to calibrate your MPU6050



Calibrating the MPU 6050

- 1 - Use the Arduino IDE to open the MPU6050_calibration.ino sketch
- 2 - Verify that your MPU 6050 breakout board I2C address is set to 0x68 (you should have already done this in a previous step)
- 3 - If the ADO jumper looks like the picture you are set to 0x68

```
// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// ADO low = 0x68 (default for SparkFun breakout board)
// ADO high = 0x69
MPU6050 mpu;
//MPU6050 mpu(0x69); // <-- use for ADO high
```



- 4 - Connect your circuit to your PC via the USB cable
- 5 - Start your RS-232 communications program (or the Arduino Serial Monitor) and set to 38.4K Baud and the correct port
- 6 - Compile and upload the sketch (click the right facing arrow at the top)
- 7 - The program will start to run after it is loaded and find the proper values to configure YOUR MPU 6050 (these values will be different for every chip)
- 8 - Record the values listed in the "Calculated Offsets" when the program stops (see next slide- x gyro = -75, y gyro = -56, z gyro = 21, z accel = 2155)



Calibrating the MPU 6050

```
See It....
File Edit Options Tools Quick Comm Settings Select String Set View Strings Help
Port Number 13
Baud Rate 38400
Parity None
Data Bits 8
Stop Bits 1
Hand Shaking 0-none
Port Active
RTS Idle
CTS ?
Clear Screen
Data Purge
[LF ]Average reading of 115 with z gyro offset of 49[CR ]
[LF ]Average reading of 112 with z gyro offset of 48[CR ]
[LF ]Average reading of 108 with z gyro offset of 47[CR ]
[LF ]Average reading of 103 with z gyro offset of 46[CR ]
[LF ]Average reading of 99 with z gyro offset of 45[CR ]
[LF ]Average reading of 96 with z gyro offset of 44[CR ]
[LF ]Average reading of 93 with z gyro offset of 43[CR ]
[LF ]Average reading of 88 with z gyro offset of 42[CR ]
[LF ]Average reading of 83 with z gyro offset of 41[CR ]
[LF ]Average reading of 80 with z gyro offset of 40[CR ]
[LF ]Average reading of 75 with z gyro offset of 39[CR ]
[LF ]Average reading of 72 with z gyro offset of 38[CR ]
[LF ]Average reading of 68 with z gyro offset of 37[CR ]
[LF ]Average reading of 63 with z gyro offset of 36[CR ]
[LF ]Average reading of 60 with z gyro offset of 35[CR ]
[LF ]Average reading of 56 with z gyro offset of 34[CR ]
[LF ]Average reading of 52 with z gyro offset of 33[CR ]
[LF ]Average reading of 48 with z gyro offset of 32[CR ]
[LF ]Average reading of 44 with z gyro offset of 31[CR ]
[LF ]Average reading of 39 with z gyro offset of 30[CR ]
[LF ]Average reading of 34 with z gyro offset of 29[CR ]
[LF ]Average reading of 32 with z gyro offset of 28[CR ]
[LF ]Average reading of 26 with z gyro offset of 27[CR ]
[LF ]Average reading of 23 with z gyro offset of 26[CR ]
[LF ]Average reading of 20 with z gyro offset of 25[CR ]
[LF ]Average reading of 13 with z gyro offset of 24[CR ]
[LF ]Average reading of 11 with z gyro offset of 23[CR ]
[LF ]Average reading of 9 with z gyro offset of 22[CR ]
[LF ]Average reading of 5 with z gyro offset of 21[CR ]
[LF ]--> Converged z gyro at: 21 with average reading of: 5[CR ]
[LF ] [CR ]
[LF ]Calculated offsets:[CR ]
[LF ] x accel: -1550[CR ]
[LF ] y accel: -397[CR ]
[LF ] z accel: 2155[CR ]
[LF ] x gyro: -75[CR ]
[LF ] y gyro: -56[CR ]
[LF ] z gyro: 21[CR ]
[LF ]
```



Calibrating the MPU 6050

1 - Use the Arduino IDE to open the MPU6050_DMP6.ino sketch

2 - Find the section labeled "supply your own gyro offsets here..":

```
// supply your own gyro offsets here, scaled for min sensitivity  
mpu.setXGyroOffset(220);  
mpu.setYGyroOffset(76);  
mpu.setZGyroOffset(-85);  
mpu.setZAccelOffset(1788); // 1688 factory default for my test chip
```

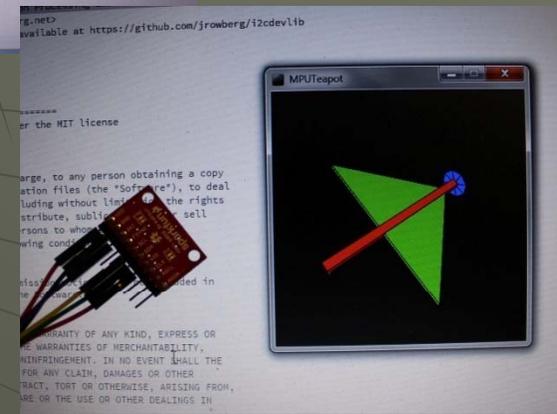
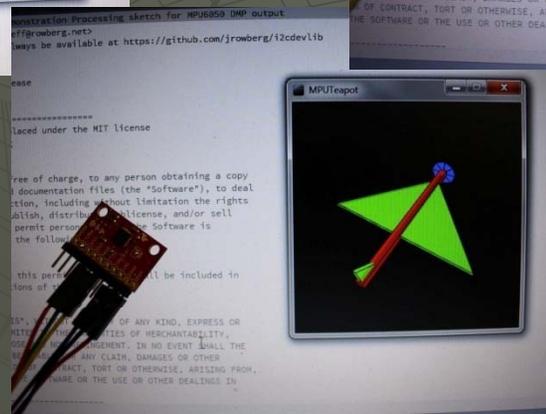
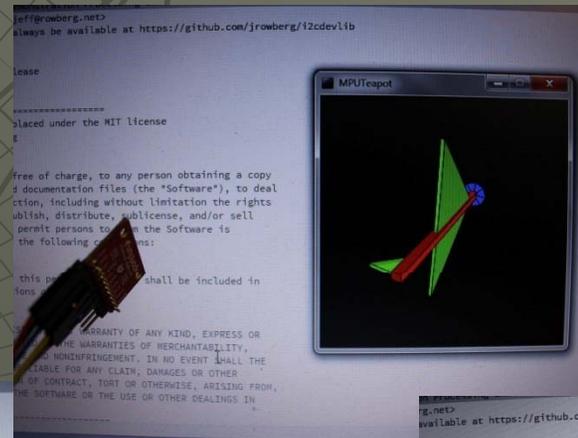
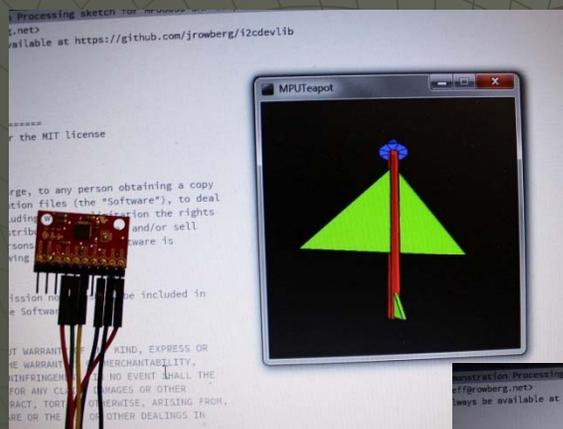
3 - And change them to match the values that you recorded from the configuration program (x and y acceleration values are not used)

```
// supply your own gyro offsets here, scaled for min sensitivity  
mpu.setXGyroOffset(-75);  
mpu.setYGyroOffset(-56);  
mpu.setZGyroOffset(21);  
mpu.setZAccelOffset(2155); // 1688 factory default for my test chip
```



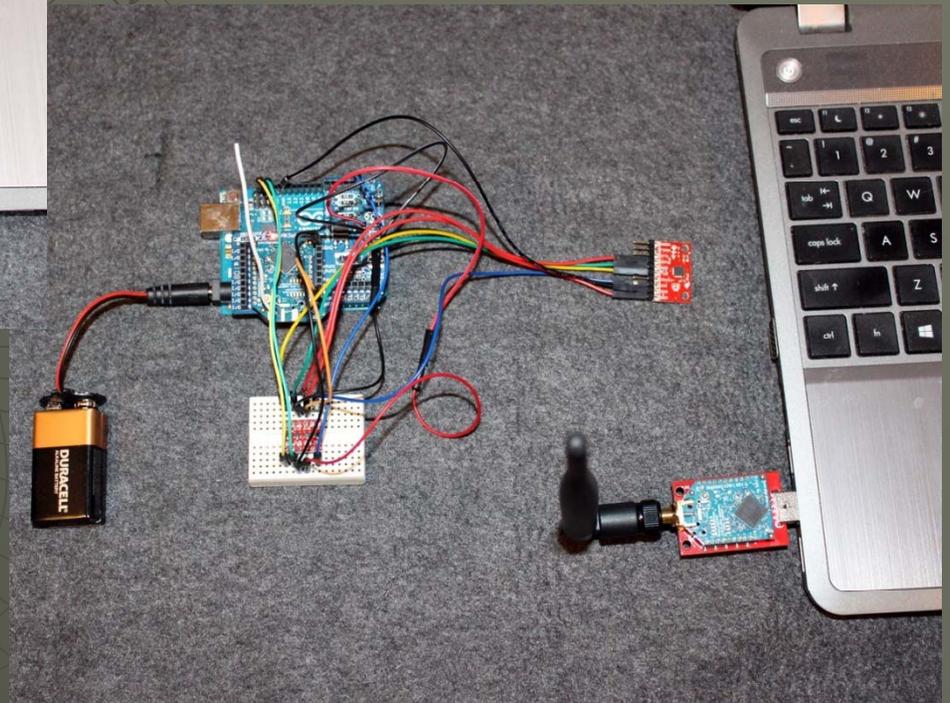
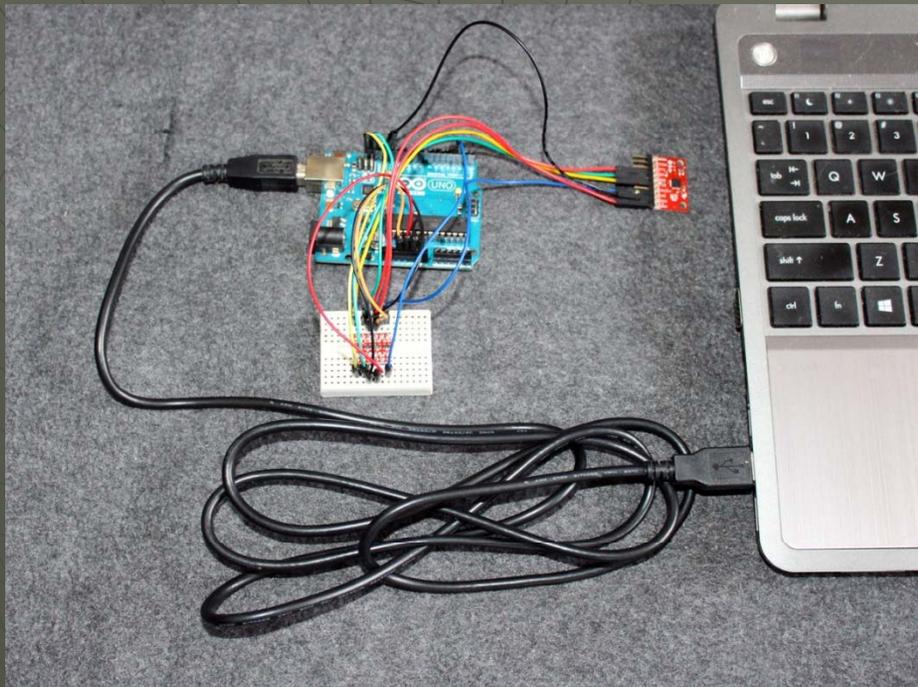
Rerun both Sketches

- 1 - With the changes in place and saved, recompile and run the Arduino Sketch
- 2 – Recompile and run the Processing sketch
- 3 – The image should now be stable and follow the movement of the MCU6050

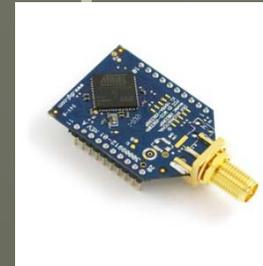
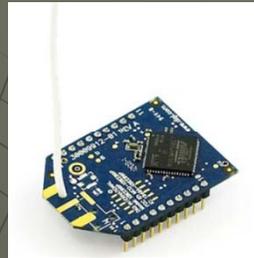
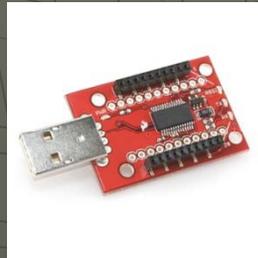




Wired or Wireless



Hardware Required



-XBee Shield for Arduino Uno

- Purchase: <http://www.sainsmart.com/sainsmart-xbee-shield-module-for-zigbee-arduino-uno-duemilanove-mega-1280-2560.html>
- Schematic: <http://www.sainsmart.com/zen/documents/20-011-902/Libellum-Xbee-Shield.pdf>

-XBee USB Explorer Dongle

- Purchase: <https://www.sparkfun.com/products/9819>
- Schematic: <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/Zigbee/XBee-Explorer-DongleV12.pdf>

-XBee Pro 60mW Wire Antenna Series 1

- 900 MHz Purchase: <https://www.sparkfun.com/products/9097>
- 2.4GHz Purchase: <https://www.sparkfun.com/products/8742>

-XBee Pro 60mW RPSMA connector

- 900MHz Purchase: <https://www.sparkfun.com/products/9099>
- 2.4GHz Purchase (U.FL Connector): <https://www.sparkfun.com/products/8710>

-Rubber Duck Antenna

- Purchase: <https://www.sparkfun.com/products/9143>
- Purchase (Cable U.FL to RPSMA connector): <https://www.sparkfun.com/products/662>



XBee Series 1 Setup

- 1 – Download the XCTU Next Generation Software and install
<http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities>
- 2 – Plug one of the XBee modules into the USB Explorer dongle and plug that into the PC
- 3 - Set up the XBee modules for 115.2K Baud
 - A – Start the XCTU Software
 - B – Click on the search Icon to find your module (you will need to specify the COM port and BAUD rate)
 - C – Click on “Add Selected Device”
 - D – After the module is found click on the module name (the configuration tab should already be selected)
 - E – Set the module BAUD rate to 115.2K Baud
 - F – Click on the write icon to write to the module
 - G – Verify that the BAUD rate changed in the description block that originally appeared when you discovered the module
- 4 - Repeat for the second module

XBee XCTU Configuration

The screenshot displays the XBee XCTU software interface. On the left, the 'Radio Modules' pane shows a module with the following details:

- Name: (blank)
- Function: XBee-PRO 900
- Port: COM9 - 115200/8/N/1/N - AT
- MAC: 0013A200408A5E84

The main 'Radio Configuration' pane is titled 'Radio Configuration [- 0013A200408A5E84]'. It is divided into two sections:

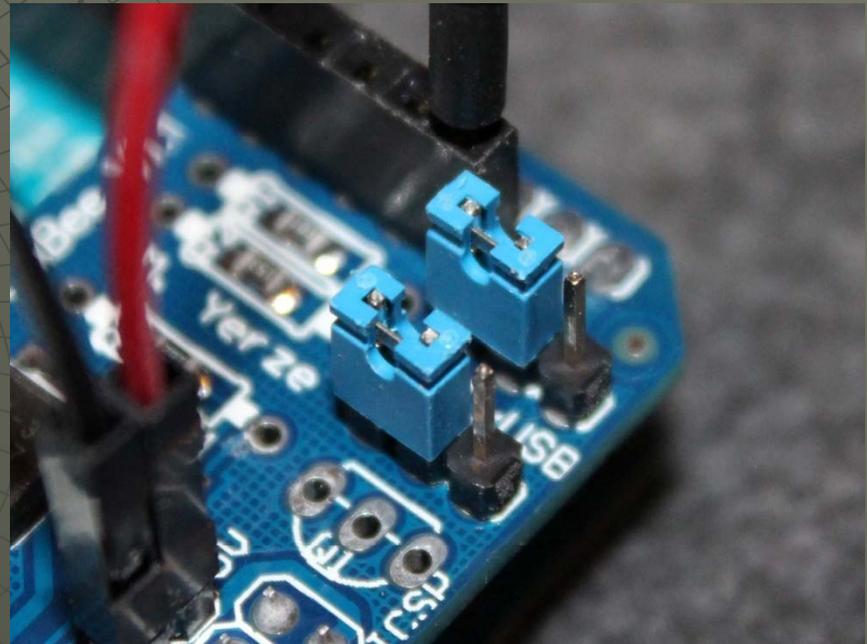
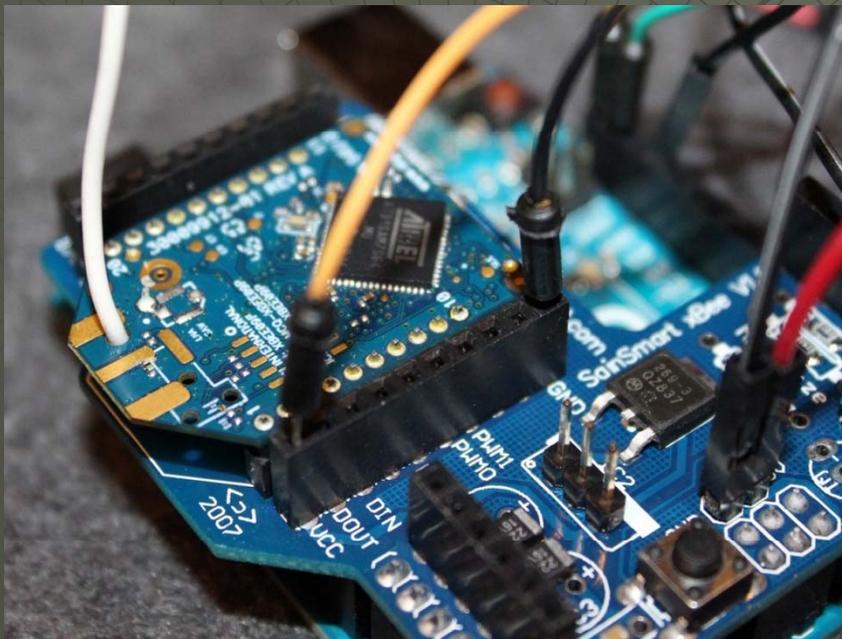
- Serial Interfacing**: Change modem interfacing options. This section contains a list of parameters with dropdown menus and status icons (green checkmarks and red pencils):
 - BD Baud Rate: 115200 [7]
 - NB Parity: No Parity [0]
 - SB Stop Bits: One stop bit [0]
 - RO Packetization Timeout: 3 x character times
 - D7 DIO7 Configuration: CTS flow control [1]
 - D6 DIO6 Configuration: Disable [0]
 - FT Flow Control Threshold: 13F Bytes
 - AP API Enable: API off [0]
 - AO API Options: XBee DigiMesh - 0x90 [0]
- I/O Settings**: Modify DIO and ADC options. This section contains a list of parameters with dropdown menus and status icons:
 - D0 AD0/DIO0 Configuration: Commissioning Button [1]
 - D1 AD1/DIO1 Configuration: Disabled [0]
 - D2 AD2/DIO2 Configuration: Disabled [0]
 - D3 AD3/DIO3 Configuration: Disabled [0]
 - D4 AD4/DIO4 Configuration: Disabled [0]

Xbee/Arduino Hardware Setup

The schematic remains the same as for the wired version – the XBee replaces the USB wire with an equivalent wireless RF connection

You can still access all of the pins on the Arduino Uno, but most of them will be connected to the XBee shield instead of directly to the Uno. It helps to solder a female header to the shield on either side of the XBee socket

The two jumpers should be in the XBEE position (not the USB position)



Xbee/Arduino Operation

- 1 - Apply power to the Arduino Uno with the XBee Shield with the same program loaded as with USB connected (MPU6050_DMP.ino)
- 2 – Start the Processing IDE and load and compile the same program as with USB connected (MPUTEapot.pde)
- 3 – The program should function the same as before, but you will notice a slight delay in response on the PC screen

