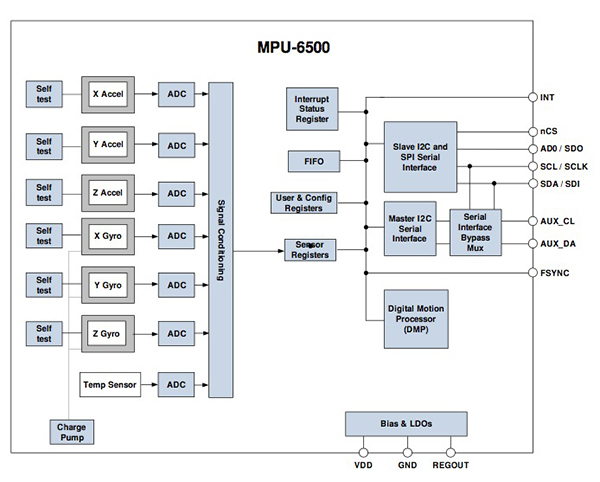
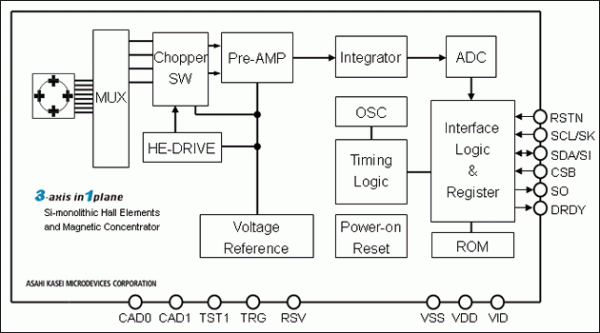
**THE SENSOR**

InvenSense®[MPU-9250](http://www.invensense.com/products/motion-tracking/9-axis/mpu-9250/) is smallest motion tracking MEMS (Micro Electro Mechanical System) device. A SiP (System in a Package) combining following two components.

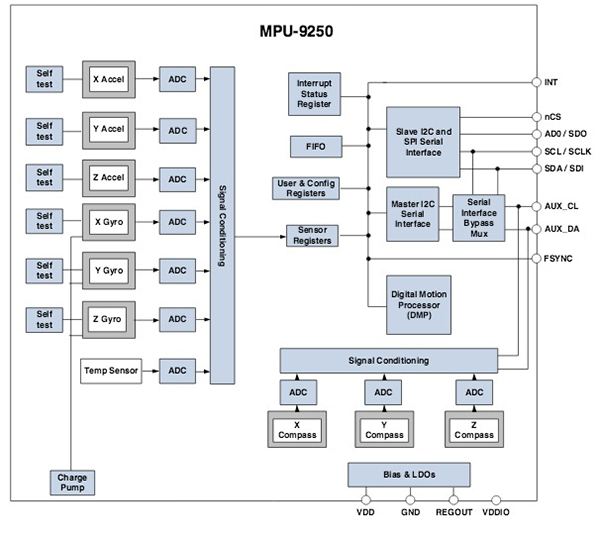
1. [MPU-6500](http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/), which contains 3-axis gyroscope, 3-axis accelerometer, and an onboard Digital Motion Processor™ (DMP™) capable of processing complex sensor fusion algorithms from InvenSense.
2. [AK8963](http://www.akm.com/akm/en/product/datasheet1/?partno=AK8963), the 3-axis magnetometer (compass) from Asahi Kasei Microdevices (AKM) Corporation.

[](https://stuka.files.wordpress.com/2015/10/article-2014september-sensors-a-key-fig3.jpg)

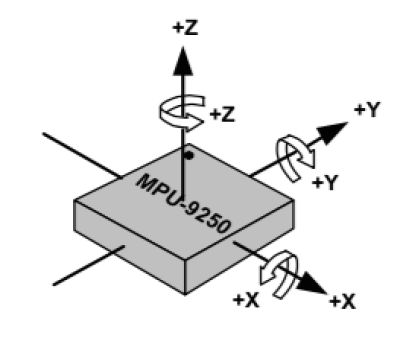
**Fig 1.0** MPU6500 component diagram

[](https://stuka.files.wordpress.com/2015/10/ak8963.gif)

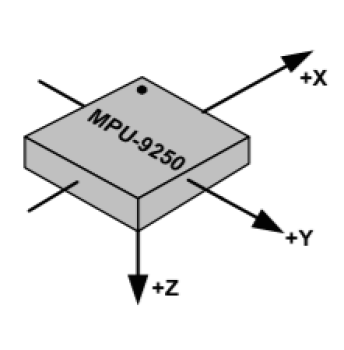
**Fig 2.0** AK8963 component diagram

[](https://stuka.files.wordpress.com/2015/10/article-2015september-sensors-play-a-key-fig2.jpg)

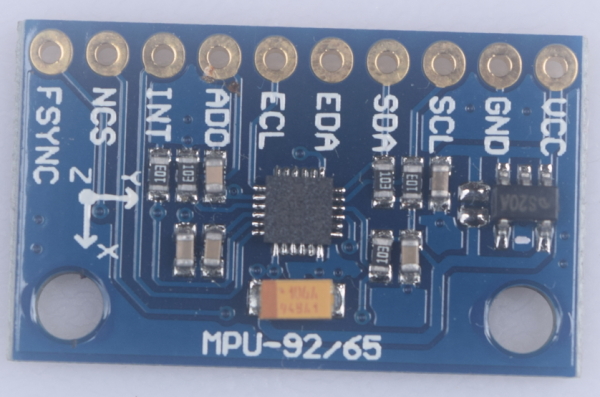
**Fig 3.0** Component diagram of MPU9250 which combine above 2 chips.

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-26-at-11-37-04-am.png)

**Fig 4.0**Orientation of axes of sensitivity and polarity of rotation for accelerometer and gyroscope.

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-26-at-11-37-21-am.png)

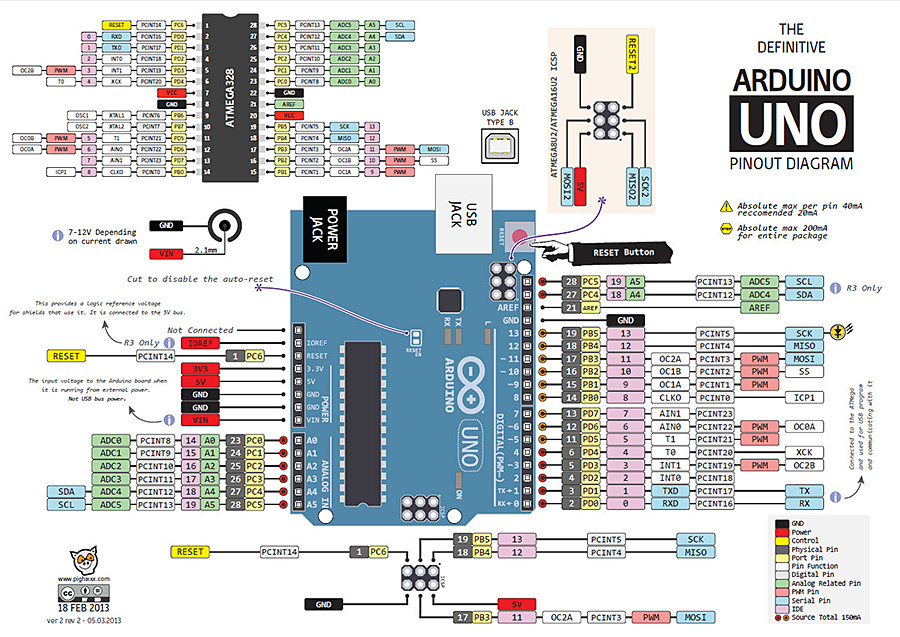
**Fig 5.0**Orientation of axes of sensitivity for magnetometer.

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-10-37-06-am.png)

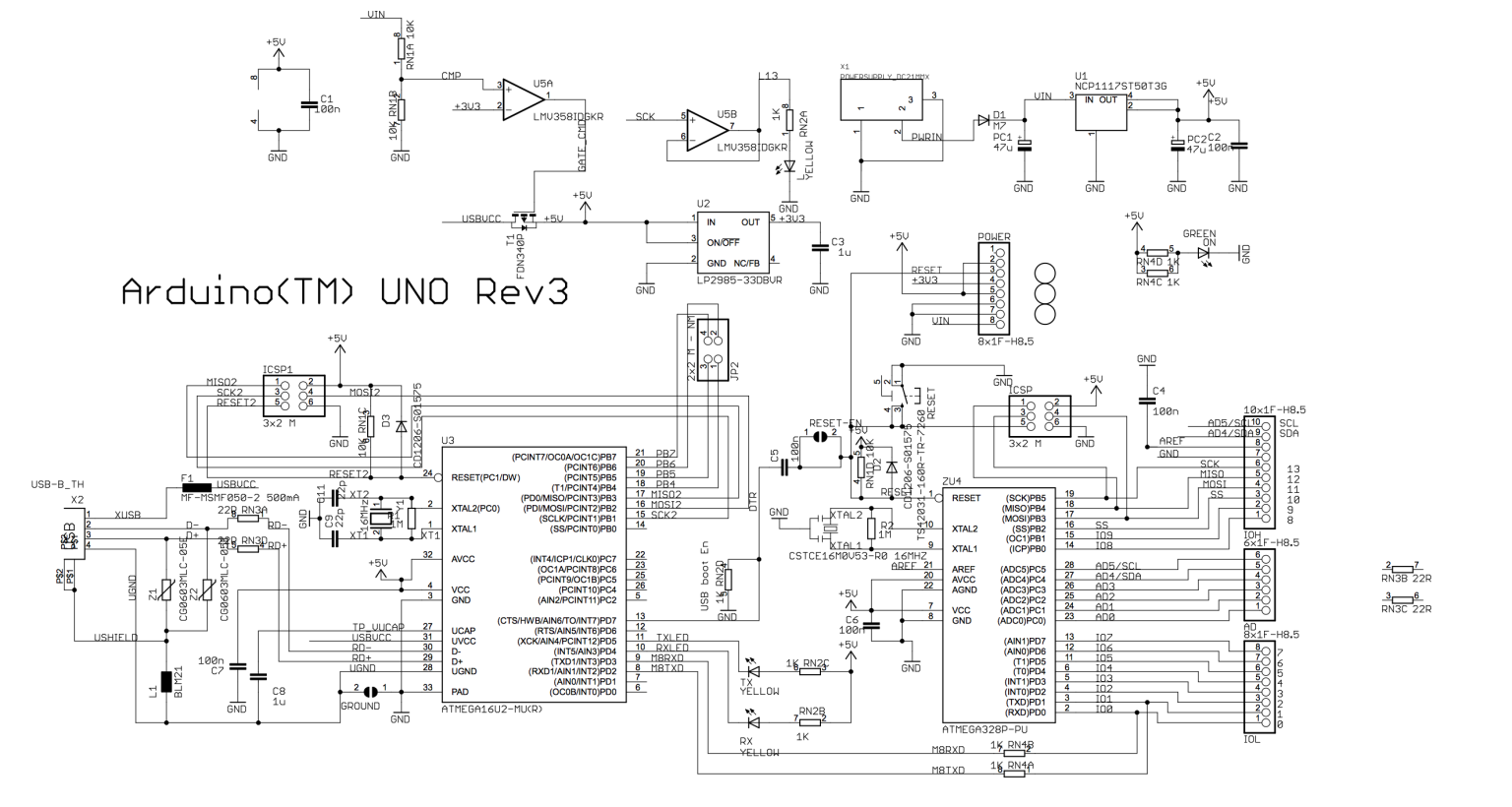
**Pic 1.0 :**MPU-9250 Breakout board

**THE MICROCONTROLLER**

The Arduino [Uno](https://www.arduino.cc/en/Main/ArduinoBoardUno) R3 is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.

[](https://stuka.files.wordpress.com/2015/10/arduinouno-900.jpg)

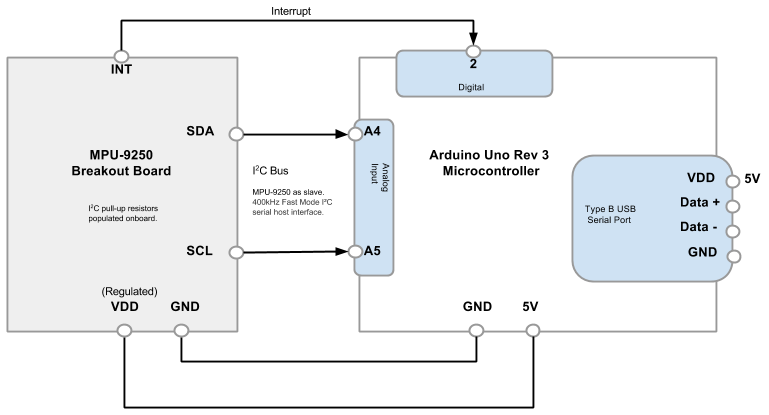
**Fig 6.0** Arduino Uno Rev3 (revision 3) pinout diagram

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-26-at-12-04-37-pm.png)

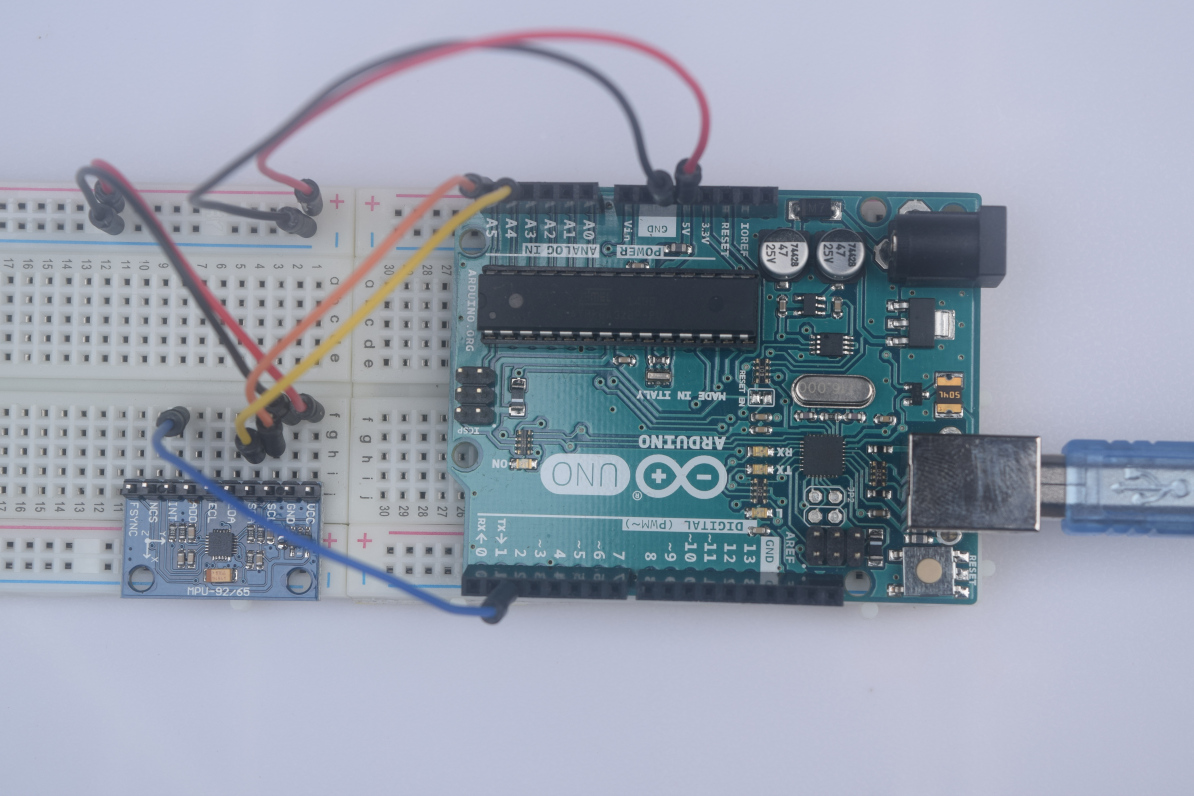
**Fig 7.0** Arduino Uno Rev3 (revision 3) schematic diagram

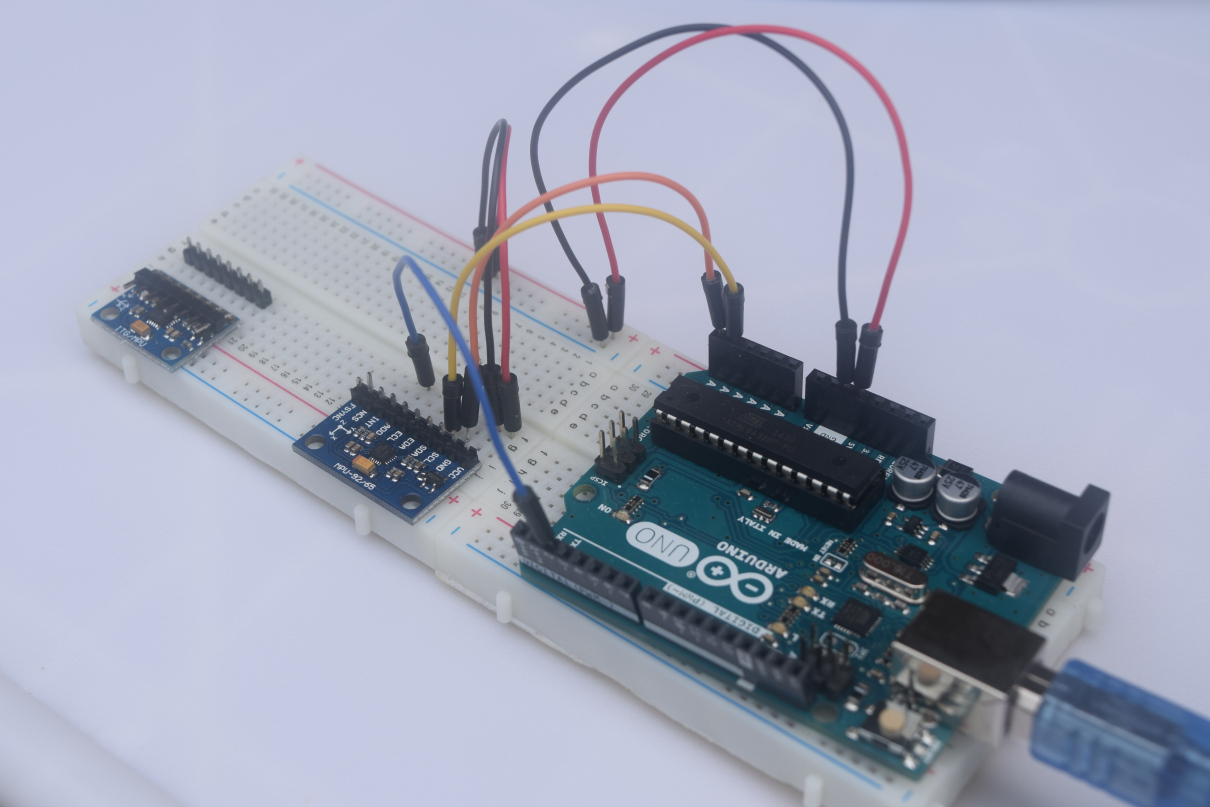
**THE CIRCUIT**

MPU 9250 sensor contains a on-chip 1024 byte FIFO buffer. The sensor values are programmed to be placed in the FIFO buffer. Buffer is read by Arduino. The FIFO buffer is used together with the interrupt signal. If the MPU-9250 places data in the FIFO buffer, it signals the Arduino with the interrupt signal so the Arduino knows that there is data in the FIFO buffer waiting to be read.

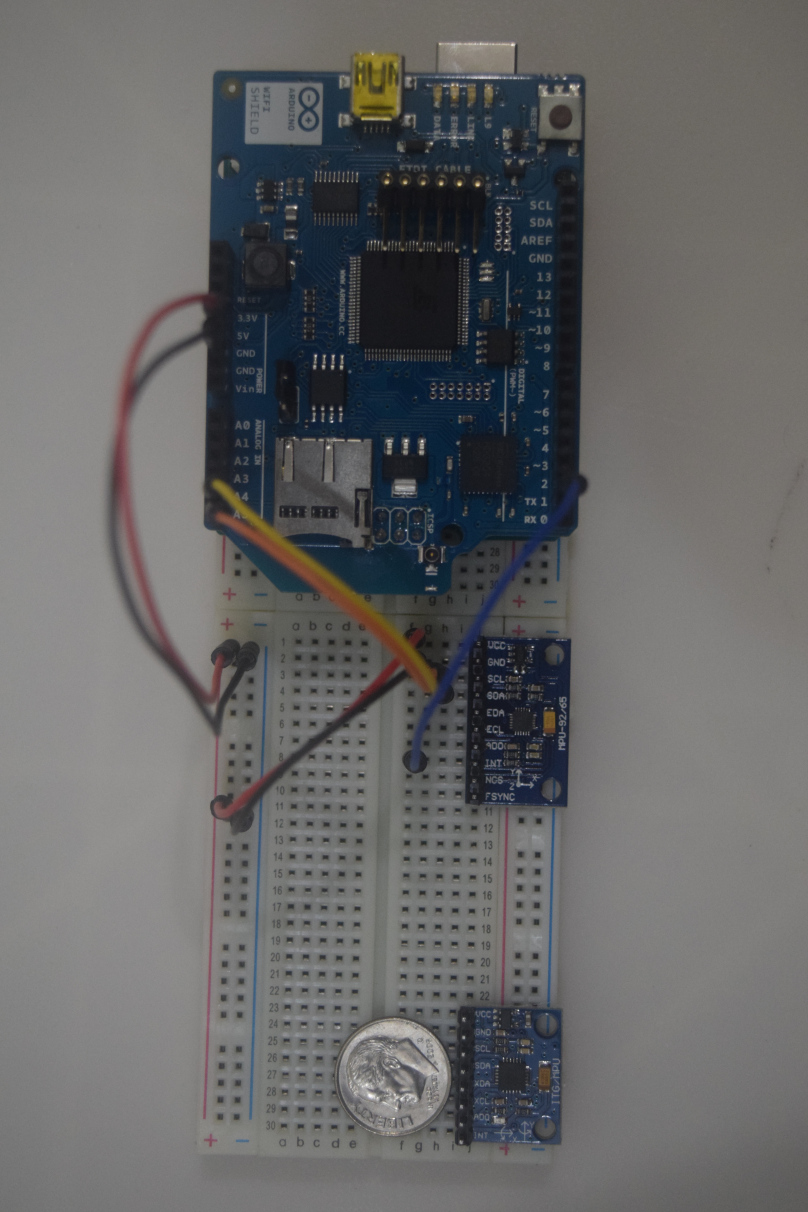


**Fig 8.0** Wiring diagram for pairing MPU 9250 with Arduino Uno Rev3 (revision 3)

[](https://stuka.files.wordpress.com/2015/10/dsc_0315.jpg)

[](https://stuka.files.wordpress.com/2015/10/dsc_0316.jpg)

**Pic 2.0** Two images of the circuit on the bread board. Arduino Uno is paired with MPU-9250 sensor module.

[](https://stuka.files.wordpress.com/2015/10/dsc_0306.jpg)

**Pic 3.0** Image of the circuit on the bread board. A Arduino wifi shield is mounted on Arduino Uno. Additional MPU-6050 (GY-521) 6 DOF sensor module is shown with a penny.

[](https://stuka.files.wordpress.com/2015/10/dsc_0309.jpg)

**Pic 4.0** My [Bell 212](https://en.wikipedia.org/wiki/Bell_212) “Twin Huey” replica model next to the circuit.

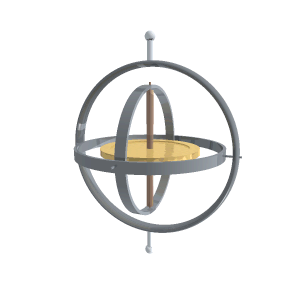
**SENSOR FUSION**

Sensor Fusion is a process by which data from several different sensors are “fused” to compute something more than could be determined by any one sensor alone or improve accuracy and reliability. An example is computing or estimating position and orientation of a device in three dimensional space. A prerequisite for sensor fusion, is that of calibration: the sensors themselves have to be calibrated and provide measurement in known units. Furthermore, whenever multiple sensors are combined additional calibration issues arise, since the measurements are seldom acquired in the same physical location and expressed in a common coordinate frame.

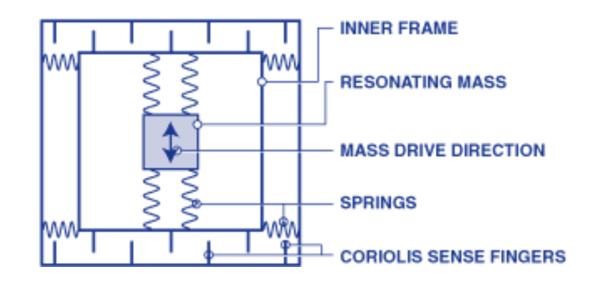
In this demo we are using 3 types of sensors. Each of these sensor types provides unique functionality, but also has limitations:

* Accelerometer: x-, y- and z-axis linear motion sensing, but sensitive to vibration
* Gyroscope: pitch, roll and yaw rotational sensing, gyroscope drift
* Magnetometer: x-, y- and z-axis magnetic field sensing, but sensitive to magnetic interference

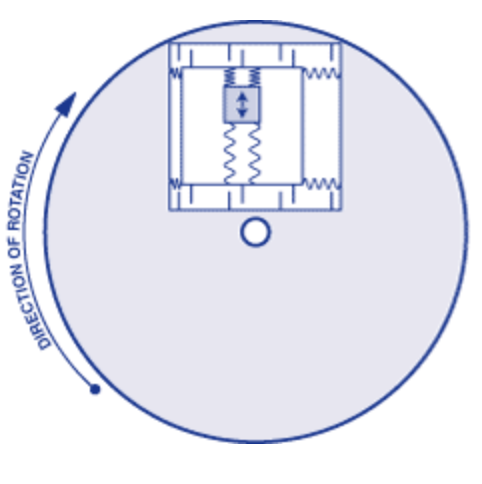
Gyroscopes sense orientation through angular velocity changes and therefore find orientation, but they have a tendency to drift over time because they only sense changes and have no fixed frame of reference.

[](https://stuka.files.wordpress.com/2015/10/gyroscope_operation.gif)

**Pic 5.0** Mechanical gyroscope in operation.

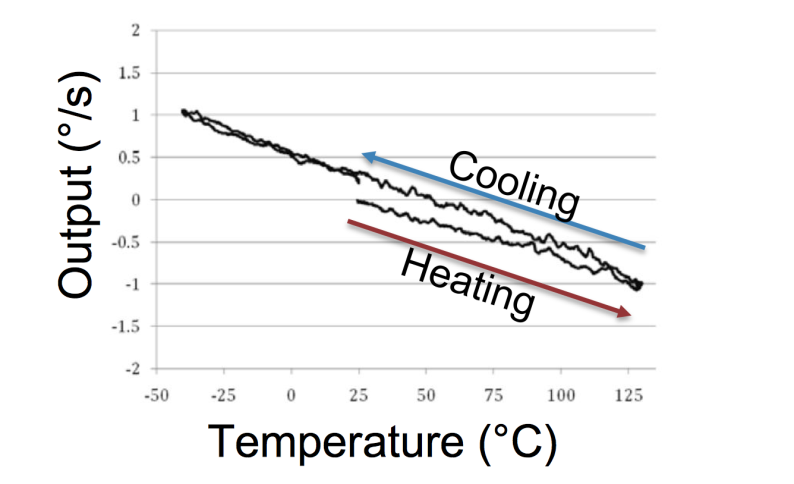
[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-7-21-23-pm.png)

**Pic 6.0** Micro electro mechanical (MEMS) gyroscope assembly.

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-7-22-04-pm.png)

**Pic 7.0** Micro electro mechanical (MEMS) gyroscope operation.

What is meant by Gyroscopic Drift (a.k.a Null Drift/Bias) in simple terms is (Angular Velocity is represented as K \* Vout) ω ≠ 0 when stationary. This drift function changes with temperature and shows hysteresis.

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-10-04-06-pm.png)

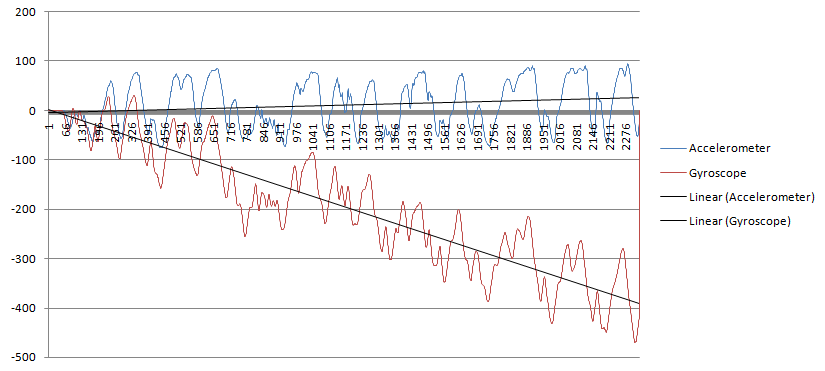
**Pic 8.0** Hysteresis creates null drift on MEMS gyroscope.

Gyroscope is sensitive to gravity and vibration.

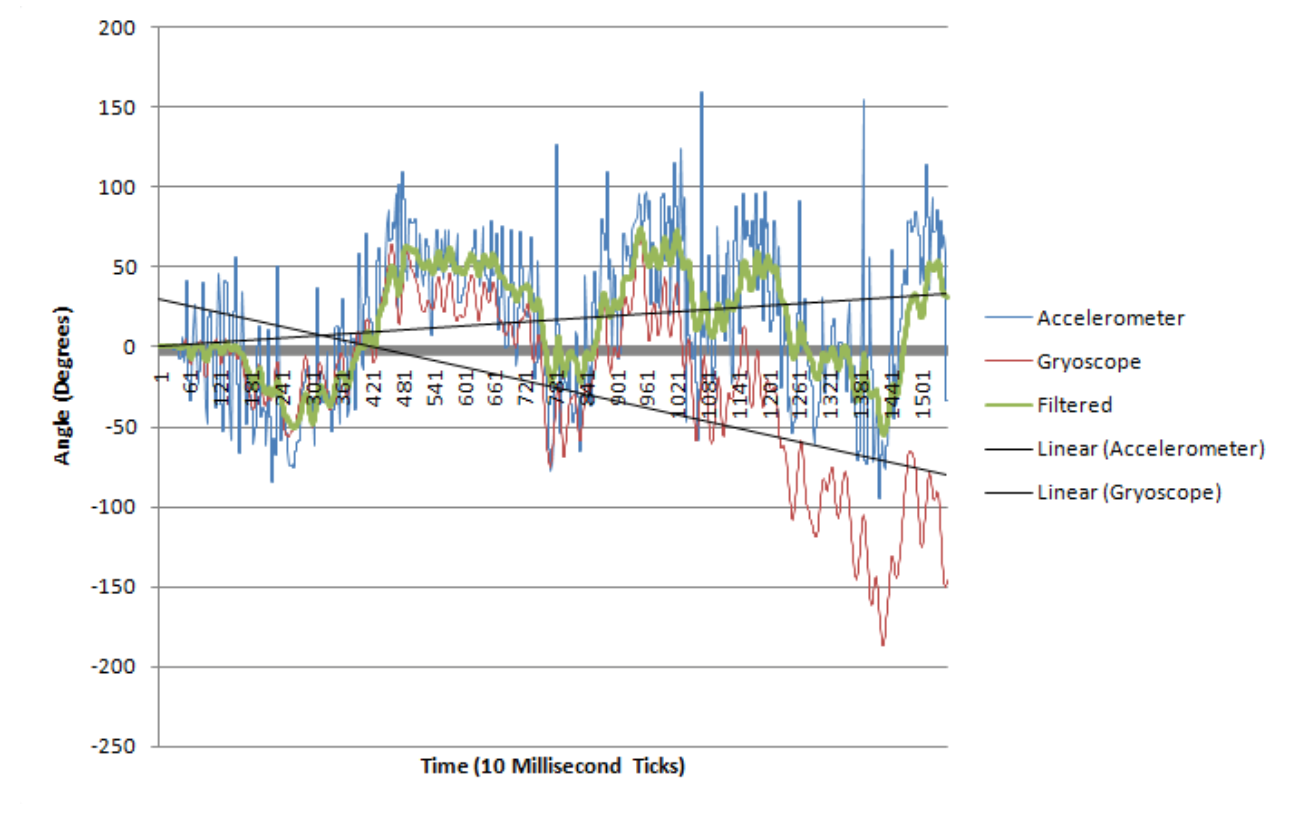
[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-10-09-56-pm.png)

**Pic 9.0** Vibration noise of a MEMS gyroscope.

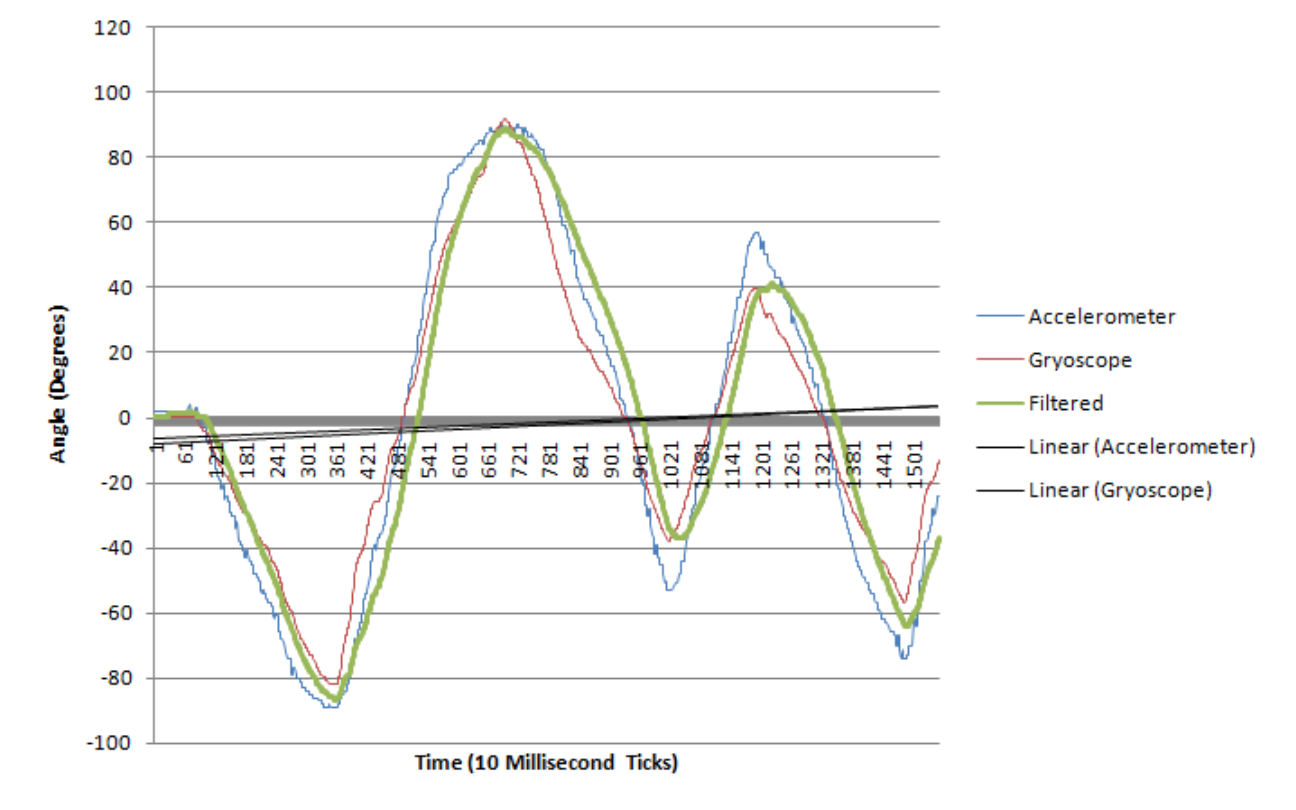
Accelerometers sense changes in direction with respect to gravity which can orient a gyroscope to a more exact angular displacement. However, accelerometers are more accurate in static calculations, when the system is closer to its fixed reference point whereas gyroscopes are better at detecting orientation when the system is already in motion. Accelerometers tend to distort accelerations due to external forces as gravitational forces in motion; which accumulates as noise in the system and erroneous spikes in resulting outputs. With the addition of the long-term accuracy of a gyroscope combined with the short-term accuracy of the accelerometer, these sensors can be combined to obtain more accurate orientation readings by utilizing the benefits of each sensor.

[](https://stuka.files.wordpress.com/2015/10/ahx4r.png)

**Pic 10.0** Gyroscope and accelerometer output with drift.

[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-10-23-13-pm.png)

**Pic 11.0** Gyroscope and accelerometer output after filtering applied.

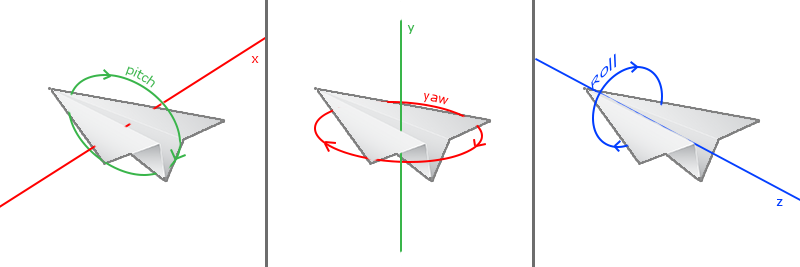
[](https://stuka.files.wordpress.com/2015/10/screen-shot-2015-10-27-at-10-27-41-pm.png)

**Pic 12.0** Gyroscope and accelerometer signals combined and smoothened.

This can be further improved using special algorithms and filtering techniques, introducing more types of sensors and using them to correct error from one input to another to reduce propagation error. This creates an output that is greater than the sum of its parts—similarly to how the human body functions.

**Video 1.0** Demonstrating fused sensory data through a visualisation.

Above visualisation shows how Yaw, Pitch and Roll is calculated using 6DOF sensory data with sensor fusion via on board DMP.

[](https://stuka.files.wordpress.com/2015/10/camera_pitch_yaw_roll.png)

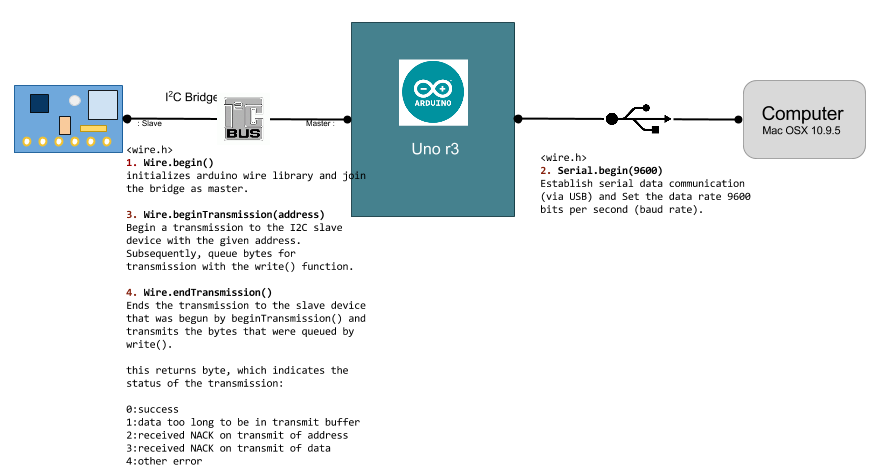
**THE SOLUTION**

[](https://stuka.files.wordpress.com/2015/10/the-circuit-3.png)

**Fig 9.0**The data analytics using WSO2 Data analytics Server

**THE CODE**

* Arduino Uno R3 works as the I2C (Inter-Integrated Circuit) master and MPU-6050 device becomes the slave. Each device on the I2C bus is having an unique address (for this exercise its ‘Øx68′[HEX], 1101000 [Binary], 104 [Decimal]).
* I’m using “**Wire.h**” Arduino library to manage communication between devices. Code block from Arduino sketch for initialising southbound and northbound communication is explained below.



**Pic 13.0**Communication model explained in code

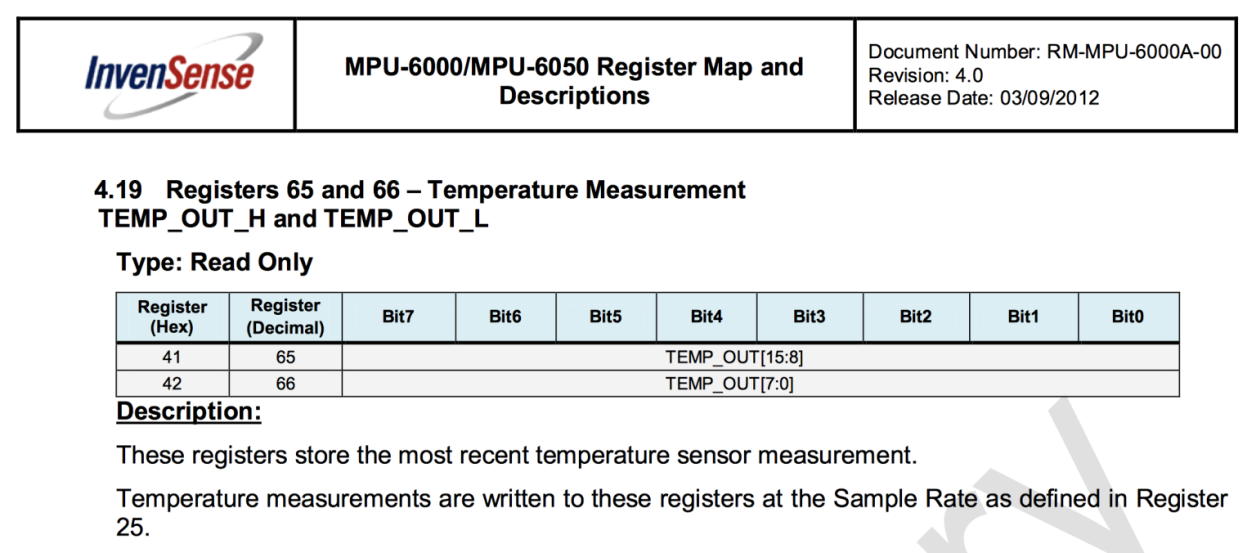
* ***Receiving***data from an I2C device into our Arduino requires two things: the unique device address (we need this in hexadecimal) and the number of bytes of data to accept from the device. Receiving data at this point is a two stage process. Most I2C devices will have multiple registers.  A register is a one or more bytes of memory that can be read, written, or both. If you review the table below from the MPU-6050 data sheet, note that there is eight registers, or bytes of data in there. The first thing we need to do is have the I2C device start reading from the first register, which is done by sending a zero to the device.

Wire.beginTransmission(0x68);

// Points to the TEMP\_OUT\_H Register.

Wire.write(0x41);

Wire.endTransmission();



* Now the I2C device will send data from the first register when requested. We now need to ask the device for the data, and how many bits we want. For example, if a device held three bytes of data, we would ask for three, and store each byte in its own variable (for example, we have three variables of type *bit: a, b, and c*. The first function to execute is:

// Receives data from TEMP\_OUT\_H and TEMP\_OUT\_L Registers.

Wire.requestFrom(0x68,2);

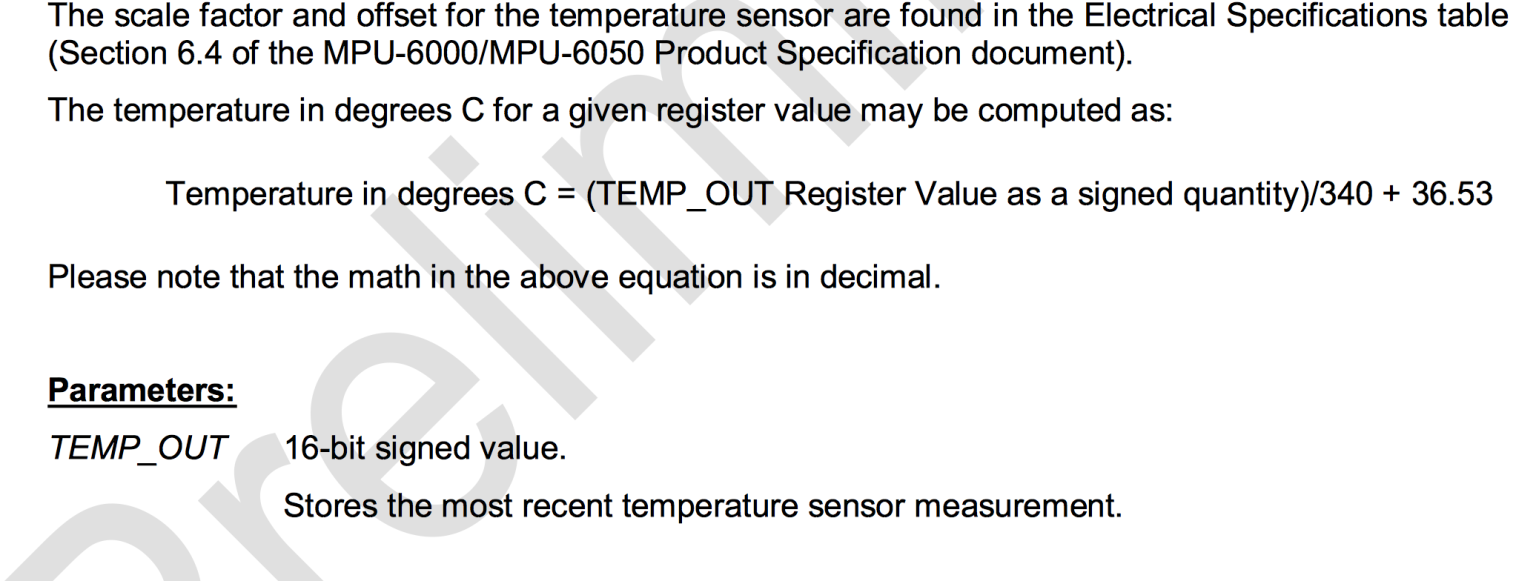
* The two bytes are now stored in a buffer in the micro controller. We then immediately follow this with code below. The **Wire.read()** function reads one byte and loads it into the variable temp (16 bit int).

temp = Wire.read();

temp = tempreg << 8;

temp |= Wire.read();

* The first byte includes the 8 left-most (highest) bits of the temperature value.  We need to move these all the way to the left-hand side of the bits. Then we use the or operator, |, to load the remaining bits.



* Now we have to apply the equation provided by data sheet above to get the true value of the temperature.

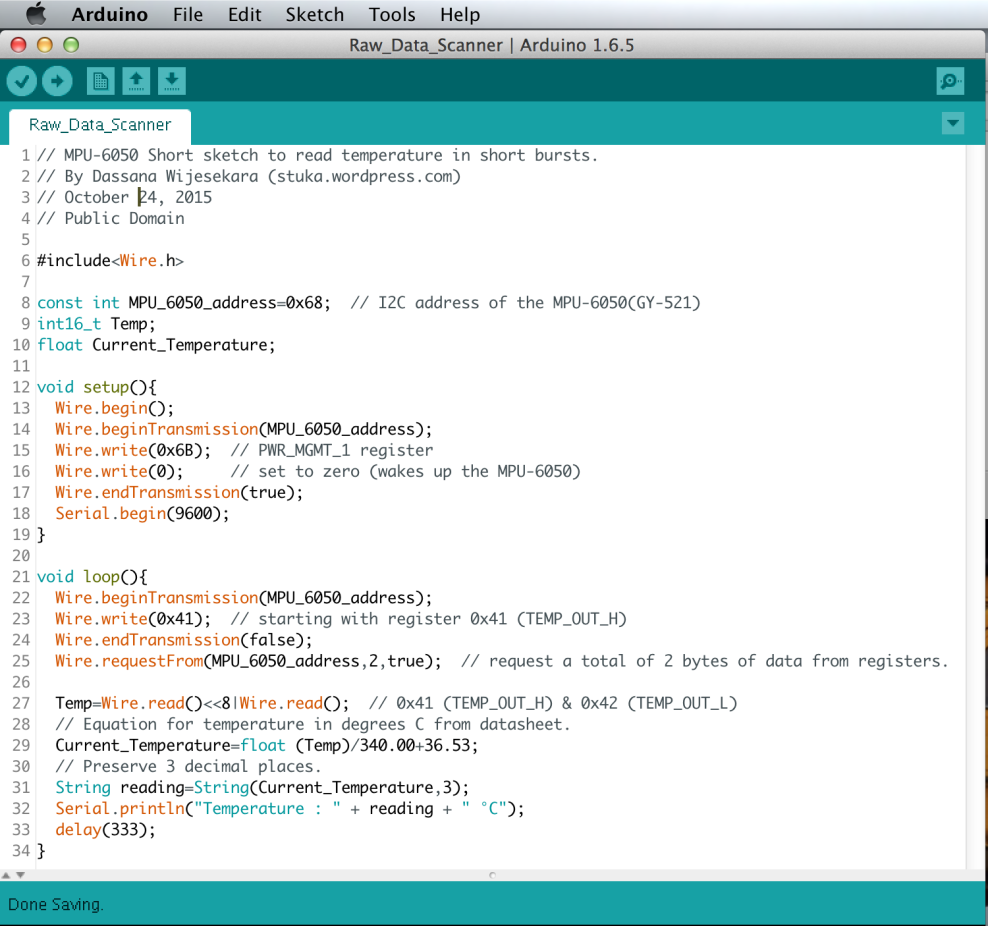
CurrentTemperature = float(temp)/340.00+36.53;

* Now we need to send temperature reading over serial USB communication bus to computer.

Serial.print("Temperature= ");

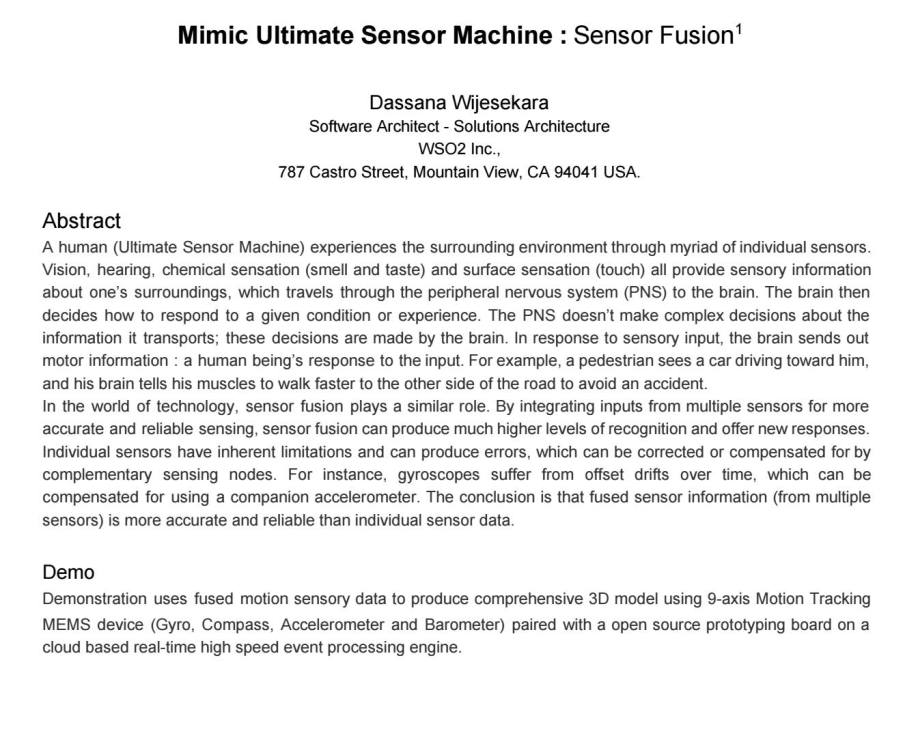
Serial.print(CurrentTemperature);

* Complete Arduino sketch is shown below.



**THE EVENT**

I spoke at the 4th Colombo IoT Meetup on 28th October 2015. Abstract of the talk is found below.



**Pic 14.0** Presenting at Colombo IoT 4th Meetup.

[Full video of the talk.](https://www.youtube.com/watch?v=8ZFd3IbEIbw&feature=youtu.be)

**Pic 15.0** Advert of Colombo IoT 4th Meetup.

**REFERENCES**

1. [MPU-6050 register map](http://www6.in.tum.de/pub/Main/TeachingWs2015SeminarAuonomousFahren/RM-MPU-6000A.pdf)
2. [MPU-6050 product description](http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/)
3. [Arduino Playground – MPU 6050](http://playground.arduino.cc/Main/MPU-6050)