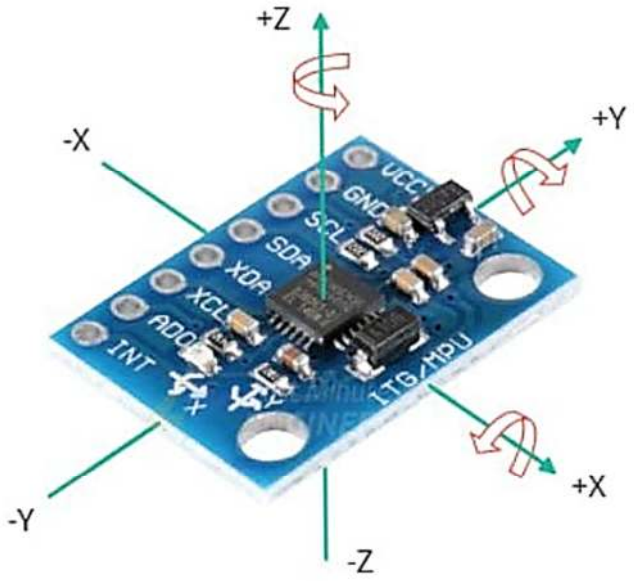
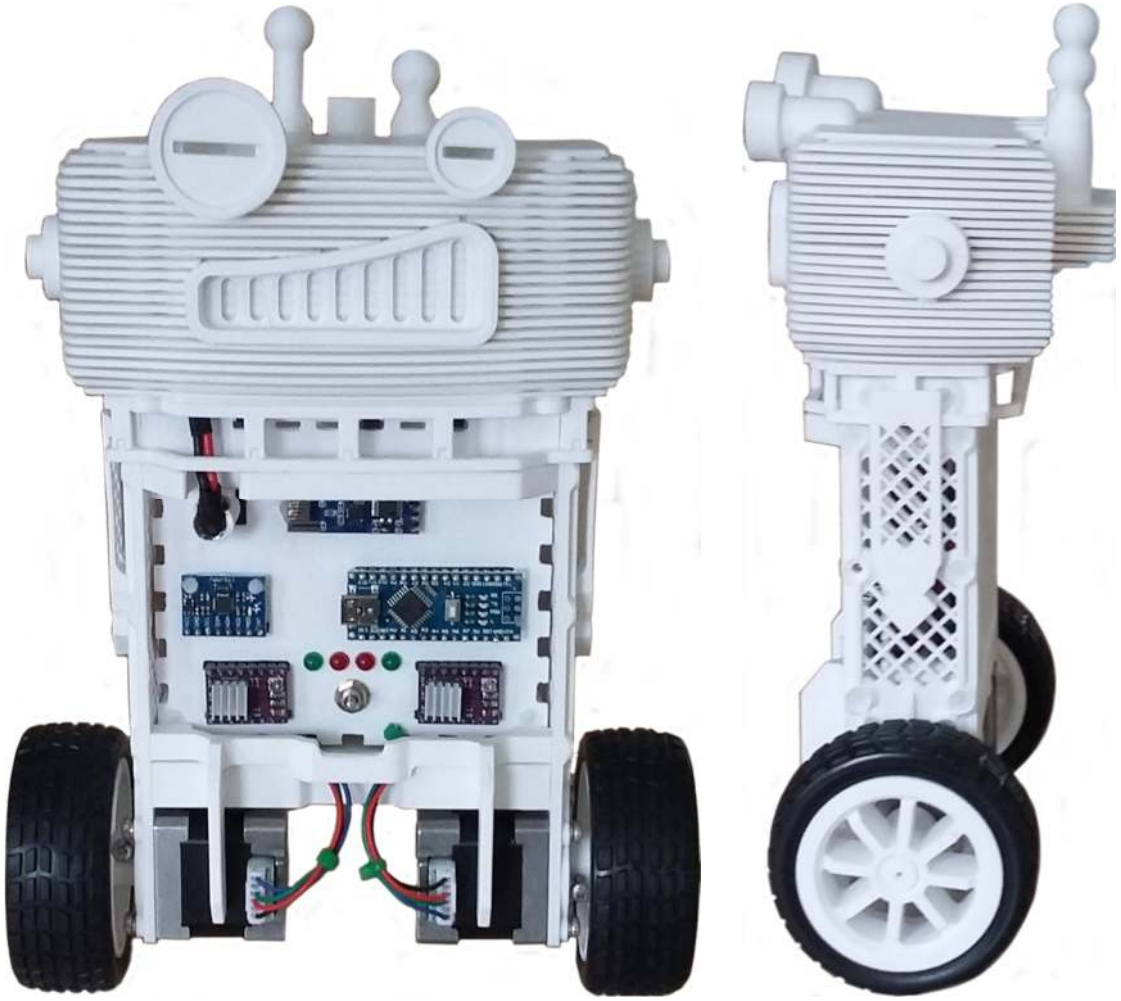
BalanceBot

Calibration

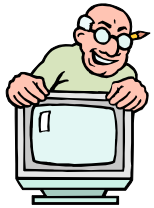


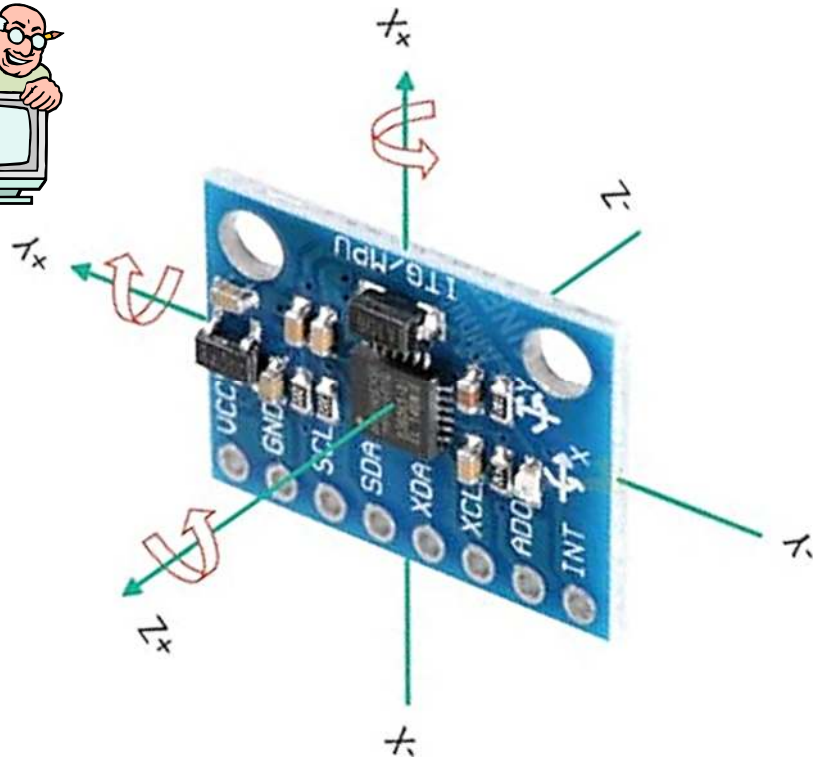
MPU6050
3-axis
Motion sensor



Understanding the role of the MPU6050 sensor

MPU 6050A Orientation

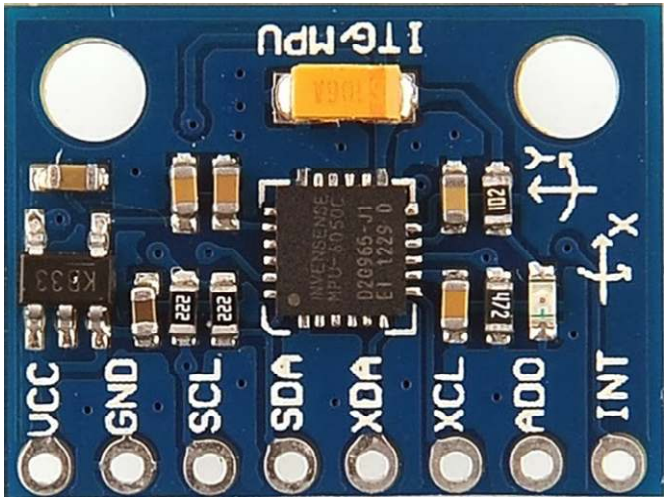


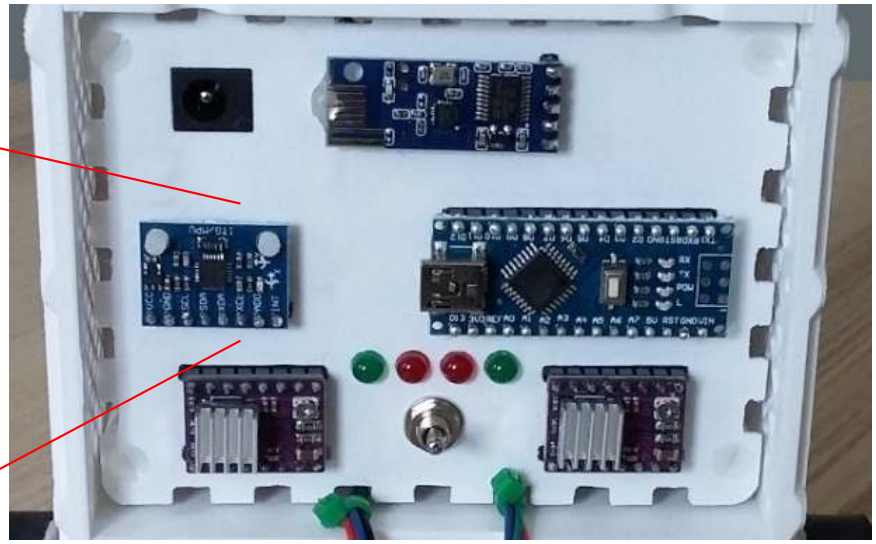


The MPU6050 is a great device for sensing motion, given its integrated accelerometers and gyroscopes, one for each of the 3 axis.

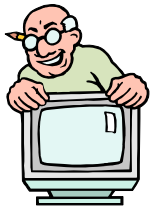
It's low cost means however that it often comes with limitations; one is that some devices don't work well at 5v, and can lock up or not function at all with Arduino. The other limitation is that most of the sensors have offsets, which add errors to your readings, and need to be measured and eliminated in your code. The good news is that once they are measured, and accounted for, they tend to stay pretty constant over time.

The mounting of the MPU sensor in your robot <u>will</u> affect which axis you use for measurements. In the BalanceBot the MPU is mounted, with its components facing forwards. To detect motion affecting balance, we use the output of the Y-axis gyroscope (rotational pitch), and the Z-axis accelerometer. The other sensor values are ignored here, but may be useful in other projects.





MPU 6050A Offsets



So how do we determine those nasty offsets, and eliminate them?

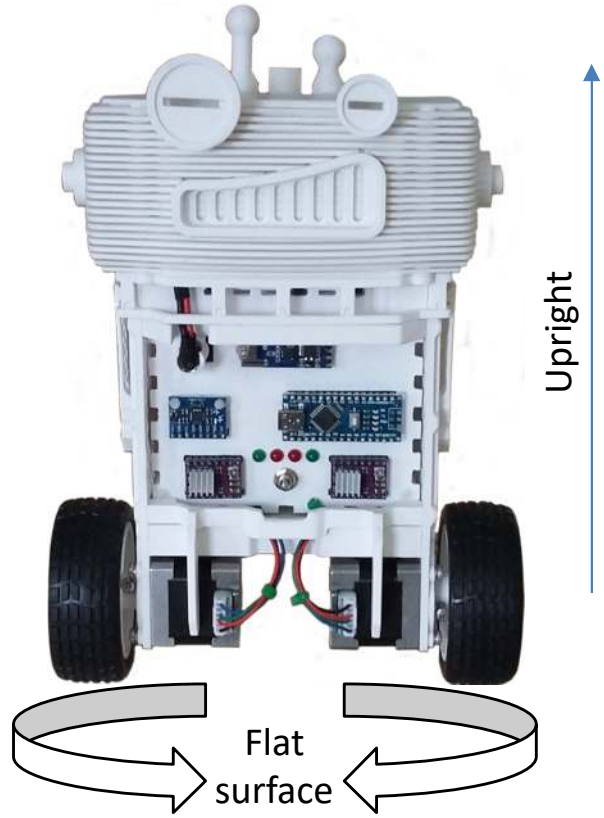
The simple answer is that we use code, to repeatedly read the MPU sensor values over the I2C bus, once it is mounted in the robot as you want it, and we take an average of many readings to determine the best value to use.

This is done with the robot held in a stand, to keep it vertical and steady, as the accelerometer sensors are very sensitive. The surface you are actually standing the stand on, may not be as perfectly flat as you think, so it can be useful to rotate the robot on its vertical axis, between a sequence of readings, to ensure that high and low offset limits can be detected, and we can then use their mid-point values.

The code provided in the **Cal_Offsets** sketch, will take 1,000 readings from the MPU, for each of the sensors, and list them to the IDE serial port as a stream of data. Whilst it does this, it also cumulates the values and presents an average for each at the end of the list. To run it again simply press RESET or modify the code as indicated.

Compile and load the sketch into the robot, using the IDE, and switch on the serial monitor. Note that the code sets the baud rate to 115200, which is around 11,520 characters per second, so that the serial port does not get bottlenecked and hold up the process.

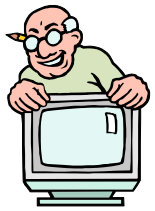
Note that the offsets for the gyroscopes are not affected by gravity, and the orientation of the MPU, where as the accelerometers are. So lets look at some results:



MPU 6050A Offset data

The IDE serial port will list the results from reading the MPU sensor registers, as shown on the right. There will be 1,000 readings, presented as six columns, with averages for each column presented at the end. In an ideal world, with the orientation of my MPU, and its scaling settings defined when it was initialised, it should present AccX as 8192 for 1g of gravitational force. There should also be a value of zero for AccY and AccZ, as they are measuring gravity horizontally. I'm getting this with the robot vertical:

AccX	AccY	AccZ	GyrX	GyrY	GyrZ
8449	-113	2510	-638	352	-114



and this with my robot, laid on its **back**, horizontal:

AccX	AccY	AccZ	GyrX	GyrY	GyrZ
454	-140	10779	-641	354	-116

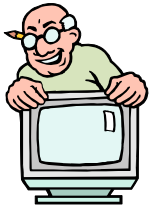
These results confirm that the MPU sensor has offsets in each axis. As expected, the gyros are not affected by the vertical/horizontal orientation of the robot. Note the Y-axis gyro offset is removed by the robots code on power-up. It also shows that the accelerometer sensors are working as expected, with AccX reducing and AccZ increasing as the robot is re-orientated. Where as Accy remains much the same.

You could easily change the Cal_Offsets sketch to take more readings if you wish, or run it several times to get a better average from the results. In this project we simply take the value for AccZ in the vertical position, and add it in the code, such that it is subtracted from all AccZ readings taken by the robot. My code contains the figure 2414, not the 2510 shown here. But that offset was taken some years back on a different table top, so I'm OK with that discrepancy.

8462	-110	2524	-640	348	-115
8458	-116	2529	-640	345	-112
8458	-141	2516	-643	351	-120
8457	-134	251	-641	347	-124
8454	-122	252	-634	336	-132
8457	-110	2529	-632	342	-130
8447	-108	2534	-633		-117
8462	-130		-644		-119
8	-117		-638		-118
8				346	-129
8450				345	-126
8444	-87			343	-117
8430	-97			361	-126
8448	-110			350	-121
8444	-11	2		355	
8432		2504		368	
8449		2500	-6	354	-102
8464		2488	-648	352	-105
8431	-95	250	-648	354	-106
8445	-114	247	-640	350	-111
8463	-118	2509	-639	363	-102
8457	-118	2508	-639	367	-101
8469	-138	2516	-642	359	-99
8447	-123	2541	-638	356	-104
8443	-124	2523	-638	360	-102
8443	-129	2493	-653	357	-110
8430	-112	2503	-655	354	-107
8433	-123	2514	-640	359	-110
8423	-109	2514	-639	352	-110
8454	-115	2488	-641	349	-112
8444	-114	2476	-639	355	-113
8442	-100	2459	-635	355	-115
8450	-123	2471	-640	359	-111
8457	-125	2486	-633	350	-116
8458	-110	2527	-640	342	-115
8448	-117	2524	-640	352	-113
8458	-116	2509	-629	348	-112
8448	-111	2525	-630	345	-118
8454	-121	2506	-640	348	-119
8444	-120	2475	-634	352	-113

AccX	AccY	AccZ	GyrX	GyrY	GyrZ
8449	-113	2510	-638	352	-114

Adding the offset to your code (Arduino code ref: Balance_Bot_16_Q.ino)



So in my case, when I did this process some time ago, I measured an average offset value of 2414 for the AccZ accelerometer, when the robot was stood upright, vertically on its stand. For improved accuracy, this value needs to be subtracted from every AccZ value you read from the MPU6050. In the code, at line 52, you will see the following definition:

```
50
51   // Define constants
52   #define acc_calibration_value -2414;   // Enter the accelerometer calibration value, default 1000
```

The value 2414 is substituted with the value you have measured, changing the sign of the number, as it will later be added to the measurement. So 2414 was entered here as -2414.

Later in the code at line 85, we assign this number to a variable as an initial value. This allows the value, as a variable, to be trimmed during robot movements, if needed.

```
83
84   // Declare and initialise global variables
85   int acc_cal_value = acc_calibration_value;   // Enter the accelerometer calibration value,
```

Later, in line 262 the code adds this offset to the value read from the MPU, in each 4ms cycle.

```
260   Wire.requestFrom(gyro_address, 2);                    // Request 2 bytes from the gyro
261   accelerometer_data_raw = Wire.read()<<8|Wire.read();  // Combine the two bytes to make one integer
262   accelerometer_data_raw += acc_cal_value;              // Add the accelerometer calibration value
```

Note that the main code for the robot performs a self-calibration test coming out of reset for the GyrY gyro. Since writing the original code however, I now believe that if you have a fairly stable MPU6050, you could also measure and define that offset, rather than performing the self-calibration test each time.