simple-i2c-mpu-basics-sketch.ino

```
/*////// Reference Tutorials ///////

I suggest that you watch all these videos. They will help you understand this code.

MPU-6050 6dof IMU tutorial for auto-leveling quadcopters with Arduino source code (Part 1)
Joop Brokking
https://www.youtube.com/watch?v=4BoIE8YQwM8&t=0s

MPU-6050 6dof IMU tutorial for auto-leveling quadcopters with Arduino source code - Part 2
Joop Brokking
https://www.youtube.com/watch?v=j-kE0AMEWy4

Build a Digital Level with MPU-6050 and Arduino
DroneBot Workshop
https://dronebotworkshop.com/mpu-6050-level/

Ep. 57 Arduino Accelerometer & Gyroscope Tutorial MPU-6050 6DOF Module
EEEnthusiast
https://www.youtube.com/watch?v=M9lZ5Qy5S2s

How I2C Communication Works and How To Use It with Arduino
How To Mechatronics
https://www.youtube.com/watch?v=6IAkYpmA1DQ

*///////////////////////////////////

/*/// Datasheets and Manuals ///////

MPU-6000 and MPU-6050
Product Specification
Revision 3.4
(MPU-6000-and-MPU-6050-Product-Specification.pdf)
https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

MPU-6000 and MPU-6050
Register Map and Descriptions
Revision 4.2
(MPU-6000-and-MPU-6050-Register-Map-and-Descriptions.pdf)
https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

*//////////////////////////////////

/*
* Basics:
1. The gyro gives you the rotational speed around the x, y and z axis.
2. The accelerometer gives you the g force (1g = 9.81m/sec2) experienced by each axis.
When the MPU-6050 is lying flat and not moving, z = 1g, x = 0g, y = 0g.
*/



// Include the wire.h library for I2C
#include

// Define the MPU I2C address
// Manual: Section 9.2
// 0b1101000 in binary is the same as 0x68 in hexadecimal
// Both formats can be used here.
const int mpu_i2c_address = 0x68; //Binary: 0b1101000

long raw_acc_x, raw_acc_y, raw_acc_z;
int raw_temp;
long raw_gyro_x, raw_gyro_y, raw_gyro_z;

float gforce_acc_x, gforce_acc_y, gforce_acc_z;
float rot_speed_gyro_x, rot_speed_gyro_y, rot_speed_gyro_z;
float rpm_rot_speed_gyro_x, rpm_rot_speed_gyro_y, rpm_rot_speed_gyro_z;
```

```
//////////////////////////////////////////////////////////////////////
////////////////////////// DEFINE FUNCTIONS //////////////////////


void setup_mpu_6050() {

/*
* Establish communication with the MPU.
* Set up the registers that we will be using. (In other words, configure the MPU
parameters.)
* Refer to this video that shows how this is done:
* Ep. 57 Arduino Accelerometer & Gyroscope Tutorial MPU-6050 6DOF Module
EEEnthusiast
https://www.youtube.com/watch?v=M9lZ5Qy5S2s
*
* This is the datsheet referenced below:
* MPU-6000 and MPU-6050
Register Map and Descriptions
Revision 4.2
https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

* Registers are various locations in the MPU6050 memory containing information or data.
*/

// Wake up the MPU.
// By default the MPU is in sleep mode when powered up. (Ref: Register Map, Section 4)

// Start communication with the MPU-6050
Wire.beginTransmission(mpu_i2c_address);
// Select the register
// Register 6B - Power Management
Wire.write(0x6B);
// Write to the register
// Set the SLEEP register to 0
Wire.write(0x00); //Binary: 0b00000000
// End the transmission
Wire.endTransmission();


// Configure the gyro (set the full scale range to: +/- 250 degrees per sec)

/*
(Sec. 4.19 pg. 31)
FS_SEL, Full Scale Range, LSB Sensitivity
0, ± 250 °/s, 131 LSB/°/s
1, ± 500 °/s, 65.5 LSB/°/s
2, ± 1000 °/s, 32.8 LSB/°/s
3, ± 2000 °/s, 16.4 LSB/°/s

Notes:

Imagine a jet plane.

x - roll
y - pitch
z - yaw

"Nose up" is a positive increase in pitch angle.
"Left wing up" is a positive increase in roll angle.
"Nose rotated right" is a positive increase in yaw.

On the MP6050 there are two arrows shown. The tip of the Y arrow is
the nose. Therefore, lifting the nose up and down changes the pitch
angle. The tip and the end of the X arrow are the two wings.

*/

// Ref: Product Specification, Sect. 5.1
```

```
// Example: (250/360)*60 = 41.6rpm
// If we select 250 then we will only be able to detect
// up to 41.6rpm rotational speed.
// Keep in mind that the Gyro Sensitivity decreases as we increase the
// full scale range i.e. as we select 500deg/sec, 1000deg/sec or 2000deg/sec.

// Start communication with the MPU-6050
Wire.beginTransmission(mpu_i2c_address);
// Select the register
// Register 1B - GYRO_CONFIG
Wire.write(0x1B);
// Write to the register
// Set the FS_SEL register to 250 deg/sec
Wire.write(0b00000000); // this is for FS_SEL 0
// End the transmission
Wire.endTransmission();

// Example: If we wanted to select 2000 deg/sec
// On the table, page 14 FS_SEL is 3 for 2000 deg/sec.
// 3 in binary is 0b11
// Therefore we need to set bits 4 and 3 to 11. (See table at top of page 14).
// Start counting the bits from right to left, starting at 0.
// We would use: 0b00011000

// 500deg/sec, FS_SEL 1, 0b00001000, 0x08



// Configure the Accelerometer (set the full scale range to: +/-2g)

/*
(Sec. 4.17, pg.29)
AFS_SEL, Full Scale Range, LSB Sensitivity
0, ±2g, 16384 LSB/g
1, ±4g, 8192 LSB/g
2, ±8g, 4096 LSB/g
3, ±16g, 2048 LSB/g

Example:
Convert a raw accelerometer value to a value in g.

Full scale range = ±2g
LSB Sensitivity at ±2g = 16384 LSB/g
raw_acc_z = 20000 (along z axis)

z axis value in g = 20000/16384 = 1.22g (i.e. 1.22 * 9.81m/sec2)
*/


// Start communication with the MPU-6050
Wire.beginTransmission(mpu_i2c_address);
// Select the register
// Register 1C - ACCEL_CONFIG (Sect. 4.5, Page 15)
Wire.write(0x1C);
// Write to the register
// Set the AFS_SEL register to +/- 2g
Wire.write(0b00000000); // this is for AFS_SEL 0 (Page 15)
// End the transmission
Wire.endTransmission();

}

void get_raw_mpu6050_data() {

/*
* Get the raw gyro, accelerometer and tepmerature data from the
* MPU-6050 registers.
* 14 Registers: 3B to 48 (59 - 72). (Ref: Pg. 7)
* The values are stored in global variables. Therefore, this function
* does not return anything.
*
* These two tutorials will help you understand what the raw gyro and accel values mean:
```

```
MPU-6050 6dof IMU tutorial for auto-leveling quadcopters with Arduino source code (Part 1)
Joop Brokking
https://www.youtube.com/watch?v=4BoIE8YQwM8&t=0s

MPU-6050 6dof IMU tutorial for auto-leveling quadcopters with Arduino source code - Part 2
Joop Brokking
https://www.youtube.com/watch?v=j-kE0AMEWy4
*
*/

// Set the start register
// The data we want is stored in register 3B to register 48 (14 registers in total)

// Start communication with the MPU-6050
Wire.beginTransmission(mpu_i2c_address);
// Set the start register (0x3B).
Wire.write(0x3B);
// End the transmission
Wire.endTransmission();


// Request data from the 14 registers
// This will begin at the start register that was set above.

// Request registers 3B to 48
Wire.requestFrom(mpu_i2c_address, 14);
// Wait until all bytes are received
while(Wire.available() < 14);


// Store the data in variables

// These statements left shift 8 bits then bitwise OR.
// Turns two 8-bit values into one 16-bit value.
// These are global variables.
raw_acc_x = Wire.read()<<8|Wire.read(); // store first two bytes into raw_acc_x
raw_acc_y = Wire.read()<<8|Wire.read(); // store next two bytes into raw_acc_y
raw_acc_z = Wire.read()<<8|Wire.read(); // store next two bytes into raw_acc_z
raw_temp = Wire.read()<<8|Wire.read(); // store next two bytes into raw_temp
raw_gyro_x = Wire.read()<<8|Wire.read(); // store next two bytes into raw_gyro_x
raw_gyro_y = Wire.read()<<8|Wire.read(); // store next two bytes into raw_gyro_y
raw_gyro_z = Wire.read()<<8|Wire.read(); // store last two bytes into raw_gyro_z

}


void print_raw_data() {

Serial.print("raw_acc_x: ");
Serial.print(raw_acc_x);

Serial.print(" raw_acc_y: ");
Serial.print(raw_acc_y);

Serial.print(" raw_acc_z: ");
Serial.print(raw_acc_z);

Serial.print(" raw_gyro_x: ");
Serial.print(raw_gyro_x);

Serial.print(" raw_gyro_y: ");
Serial.print(raw_gyro_y);

Serial.print(" raw_gyro_z: ");
Serial.println(raw_gyro_z);

}

void calc_acc_gforces() {

/*
* Convert the raw acc values to values in g.
```

```
*/

/*
(Sec. 4.17, pg.29)
AFS_SEL, Full Scale Range, LSB Sensitivity
0, ±2g, 16384 LSB/g
1, ±4g, 8192 LSB/g
2, ±8g, 4096 LSB/g
3, ±16g, 2048 LSB/g

Example:
Convert a raw accelerometer value to a value in g.

Full scale range = ±2g
LSB Sensitivity at ±2g = 16384 LSB/g
raw_acc_z = 20000 (along z axis)

z axis value in g = 20000/16384 = 1.22g (i.e. 1.22 * 9.81m/sec2)
*/

gforce_acc_x = raw_acc_x/16384.0;
gforce_acc_y = raw_acc_y/16384.0;
gforce_acc_z = raw_acc_z/16384.0;
}


void print_acc_gforces() {

Serial.print("gforce_acc_x: ");
Serial.print(gforce_acc_x);

Serial.print(" gforce_acc_y: ");
Serial.print(gforce_acc_y);

Serial.print(" gforce_acc_z: ");
Serial.println(gforce_acc_z);

}


void calc_gyro_rot_speed() {

/*
* Convert the raw gyro values to rotational speed in degrees per second.
*/

/*
(Sec. 4.19 pg. 31)
FS_SEL, Full Scale Range, LSB Sensitivity
0, ± 250 °/s, 131 LSB/°/s
1, ± 500 °/s, 65.5 LSB/°/s
2, ± 1000 °/s, 32.8 LSB/°/s
3, ± 2000 °/s, 16.4 LSB/°/s

Example:
Convert a raw gyro value to degrees per second and rpm.

Full scale range = ± 250 °/s
LSB Sensitivity at ± 250 °/s = 131 LSB/°/s (When the gyro is rotating at 1 deg/sec the raw
output is 131)
raw_gyro_x = 20000

x axis value in degrees = 20000/131 = 15.27 deg/sec

x axis value in rpm = (20000/131) * (60/360) = 2.5 rpm

Notes:

Imagine a jet plane.

x - roll
y - pitch
```

```
z - yaw

"Nose up" is a positive increase in pitch angle.
"Left wing up" is a positive increase in roll angle.
"Nose rotated right" is a positive increase in yaw.

On the MP6050 there are two arrows shown. The tip of the Y arrow is
the nose. Therefore, lifting the nose up and down changes the pitch
angle. The tip and the end of the X arrow are the two wings.

*/

// rotational speed in deg/sec
rot_speed_gyro_x = raw_gyro_x/131.0;
rot_speed_gyro_y = raw_gyro_y/131.0;
rot_speed_gyro_z = raw_gyro_z/131.0;

// rotational speed in rpm
rpm_rot_speed_gyro_x = rot_speed_gyro_x * (60.0/360.0);
rpm_rot_speed_gyro_y = rot_speed_gyro_y * (60.0/360.0);
rpm_rot_speed_gyro_z = rot_speed_gyro_z * (60.0/360.0);
}



void print_gyro_rot_speed() {

Serial.print("rot_speed_gyro_x: ");
Serial.print(rot_speed_gyro_x);

Serial.print(" rot_speed_gyro_y: ");
Serial.print(rot_speed_gyro_y);

Serial.print(" rot_speed_gyro_z: ");
Serial.println(rot_speed_gyro_z);

}


void print_gyro_rpm_rot_speed() {

Serial.print("rpm_rot_speed_gyro_x: ");
Serial.print(rpm_rot_speed_gyro_x);

Serial.print(" rpm_rot_speed_gyro_y: ");
Serial.print(rpm_rot_speed_gyro_y);

Serial.print(" rpm_rot_speed_gyro_z: ");
Serial.println(rpm_rot_speed_gyro_z);

}

///////////////////////////// End of Functions //////////////////////////
//////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////
///////////////////////////// SETUP and MAIN LOOP ////////////////////

void setup() {

// Connect to the Arduino serial monitor
Serial.begin(9600);

// Start I2C
Wire.begin();

// Configure the MPU-6050 registers
setup_mpu_6050();

}
```

```
void loop() {


// Get the raw gyro and accelerometer data
get_raw_mpu6050_data();

// Print the raw data
//print_raw_data();

calc_acc_gforces();
print_acc_gforces();

//calc_gyro_rot_speed();
//print_gyro_rot_speed();
//print_gyro_rpm_rot_speed();


delay(1000);

}

//////////////////////////// End of Setup and Main Loop ////////////////
///////////////////////////////////////////////////////////////////////
```