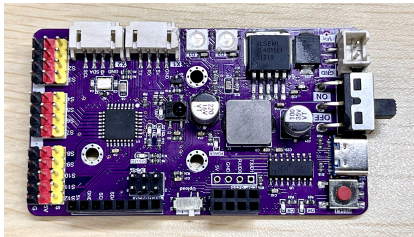


Lesson 9 Reading the Data of MPU6050

9.1 Overview

This course mainly introduces the relevant knowledge of MPU6050 modules, including their working principles, hardware connections, and programming applications on the Arduino platform.

9.2 Required Components

Components	Quantity	Picture
Adeept Pixie Board	1	
Type-C USB Cable	1	
MPU6050	1	
OLED Screen	1	

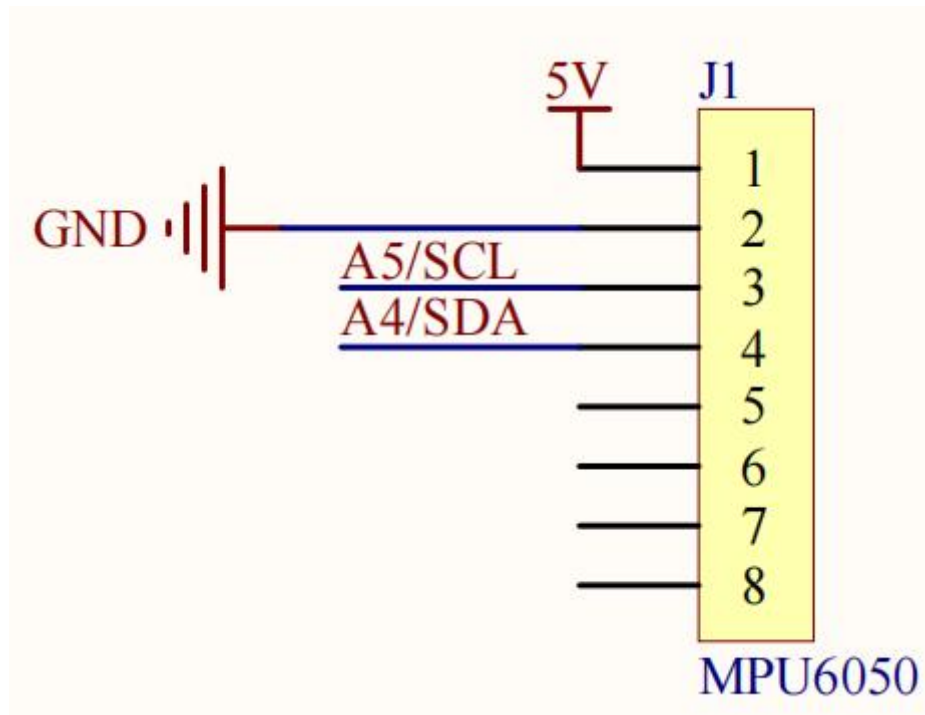
9.3 Principle Introduction

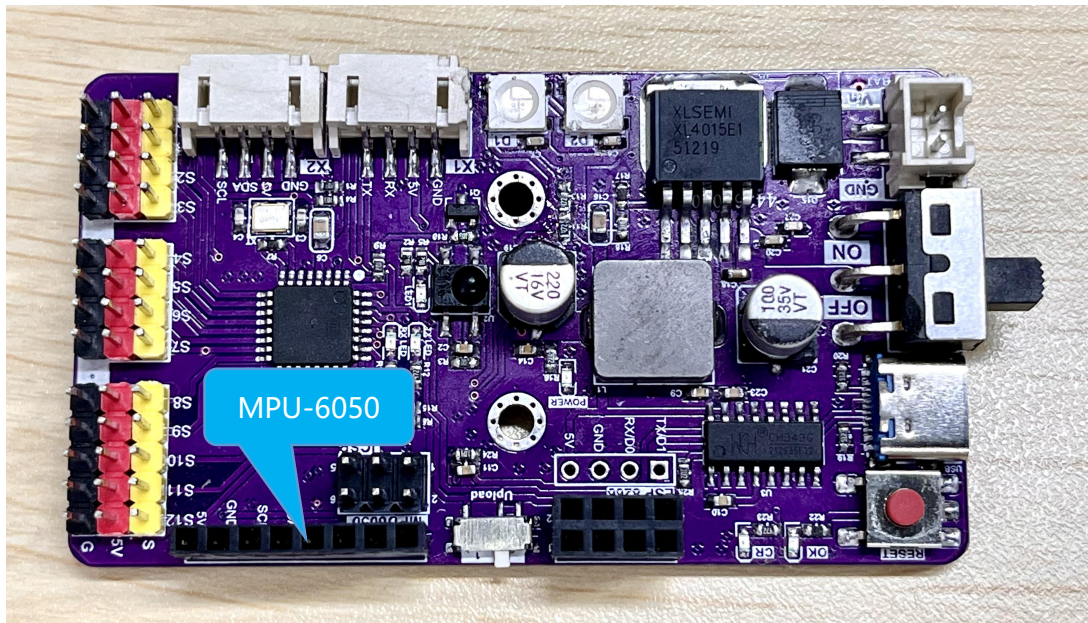
The MPU6050 consists of a three-axis accelerometer and a three-axis gyroscope, which measures the acceleration and angular velocity of an object in the x, y, and z directions. An accelerometer can detect the linear acceleration of an object, while a gyroscope can detect the angular velocity of an object. By combining measurements from the accelerometer and gyroscope, the direction and Angle of the object can be calculated.

The MPU6050 uses the I2C bus communication protocol and can be directly connected to a microcontroller or a single chip computer. Before using the MPU6050, calibration is required to ensure the accuracy of its measurement results. The calibration process can be carried out by software or hardware.

When the MPU6050 is working, it continuously reads the data of the accelerometer and gyroscope, and performs data processing and filtering. The processed data can be sent to the microcontroller or microcontroller via the I2C bus for subsequent processing and application. In practical applications, the measurement results of the MPU6050 can be converted into the direction and Angle of the object by the algorithm.

9.4 Wiring Diagram





PINS of Adept Pixie Board	MPU-6050
GND	GND
5V	VCC
A4	SDA
A5	SCL

9.5 Demonstration

1. Connect your computer and Adept Pixie Board (Arduino Board) with a USB cable.
2. Open **"09_MPU6050"** folder in **"/Code"** , double-click **"09_MPU6050.ino"** .



```

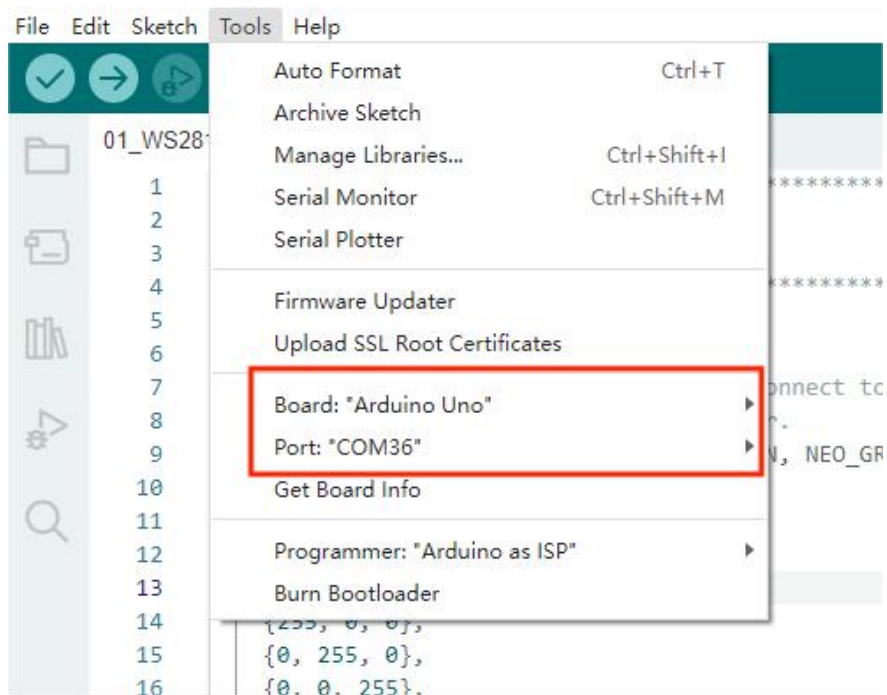
09_MPU6050 | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Arduino Uno
09_MPU6050.ino
1  /*****
2  File name:MPU6050.ino
3  Description: Get the X, Y and Z axis data of MPU6050.
4  Website: www.adeept.com
5  E-mail: support@adeept.com
6  *****/
7  #include<Wire.h>
8  #include <SSD1306AsciiWire.h>
9  #define MPU_ADDR 0x68 // I2C address of the MPU-6050
10
11  SSD1306AsciiWire display;
12
13  unsigned long lastTime = 0;
14  float agx = 0, agy = 0, agz = 0; //Angle variable
15  long axo = 0, ayo = 0, azo = 0; //Accelerometer offset
16  long gx0 = 0, gy0 = 0, gz0 = 0; //Gyroscope offset
17  int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
18
19  void setup()
20  {
21    Wire.begin();
22    Wire.beginTransmission(MPU_ADDR);
23    Wire.write(0x6B); // PWR_MGMT_1 register
24    Wire.write(0); // set to zero (wakes up the MPU-6050)
25    Wire.endTransmission(true);
  
```

3. Select development board and serial port.

Board: **Tools---**>**Board---**>**Arduino AVR Boards---**>**Arduino Uno**

Port: **Tools --->Port---**>**COMx**

Note: The port number will be different in different computers.





4. After opening, click  to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.

```

Output
Sketch uses 2952 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 82 bytes (4%) of dynamic memory, leaving 1966 bytes for local variables. Maximum is 2048 bytes.

```

5. After successfully uploading the code, run the program. You can see the MPU6050 information display in the OLED.

9.6 Code

```

001  /*****
002  File name:MPU6050.ino
003  Description: Get the X, Y and Z axis data of MPU6050.
004  Website: www.adeept.com
005  E-mail: support@adeept.com
006  *****/
007  #include<Wire.h>
008  #include <SSD1306AsciiWire.h>
009  #define MPU_ADDR 0x68 // I2C address of the MPU-6050
010
011  SSD1306AsciiWire display;
012
013  unsigned long lastTime = 0;
014  float agx = 0, agy = 0, agz = 0;    //Angle variable
015  long axo = 0, ayo = 0, azo = 0;    //Accelerometer offset
016  long gx0 = 0, gy0 = 0, gzo = 0;    //Gyroscope offset
017  int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
018
019  void setup()
020  {
021    Wire.begin();
022    Wire.beginTransmission(MPU_ADDR);
023    Wire.write(0x6B); // PWR_MGMT_1 register
024    Wire.write(0); // set to zero (wakes up the MPU-6050)
025    Wire.endTransmission(true);
026
027    for(int i=0;i<200;i++)
028    {
029      getMotion6(); // read the original value of six axes
030      axo += AcX; ayo += AcY; azo += AcZ;    //sampling interval
031      gx0 += GyX; gy0 += GyY; gzo += GyZ;
032    }
033    axo /= 200; ayo /= 200; azo /= 200; //Calculate the accelerometer offset

```

```

034   gxo /= 200; gyo /= 200; gzo /= 200; //Calculate the accelerometer offset
035
036   // Serial.begin(115200);
037   display.begin(&Adafruit128x64, 0x3C);
038   display.setFont(Adafruit5x7);
039   display.clear();
040   display.set2X();
041   display.setCursor(20, 0); // Center display
042   display.print("X:");
043   display.setCursor(20, 3); // Center display
044   display.print("Y:");
045   display.setCursor(20, 6); // Center display
046   display.print("Z:");
047 }
048
049 void loop()
050 {
051   count6Axle();
052 }
053
054 void getMotion6()
055 {
056   Wire.beginTransmission(MPU_ADDR);
057   Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
058   Wire.endTransmission(false);
059   Wire.requestFrom(MPU_ADDR,14,true); // request a total of 14 registers
060   AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
061   AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
062   AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
063   Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
064   GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
065   GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
066   GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
067 }
068
069 void count6Axle()
070 {
071   const float pi = 3.1415926;
072   const float AcceRatio = 16384.0; //Accelerometer scaling factor
073   const float GyroRatio = 131.0; //Gyroscope scale factor
074   static float aaxs[8] = {0}, aays[8] = {0}, aazs[8] = {0}; //Sample the queue on the x and y
075   axes
076   const uint8_t n_sample = 8; //The number of samples sampled by accelerometer
077   filtering algorithm
078
079   static float a_x[10]={0}, a_y[10]={0},a_z[10]={0} ,g_x[10]={0} ,g_y[10]={0},g_z[10]={0};
080   //Accelerometer covariance calculation queue
081   static float Px=1, Rx, Kx, Sx, Vx, Qx; //The kalman variable on the X-axis
082   static float Py=1, Ry, Ky, Sy, Vy, Qy; //The kalman variable on the Y-axis
083   static float Pz=1, Rz, Kz, Sz, Vz, Qz; //The kalman variable on the Z-axis
084
085   unsigned long now = millis(); // current time (ms)
086   float dt = (now - lastTime) / 1000.0; // differential time (s)
087   lastTime = now; // last sampling time (ms)
088   getMotion6();
089   float accx = AcX / AcceRatio; //x axis acceleration

```

```

090 float accy = AcY / AcceRatio;          //y axis acceleration
091 float accz = AcZ / AcceRatio;          ///Z axis acceleration
092
093 float aax = atan(accy / accz) * (-180) / pi;    // the Angle between the Y-axis and the z-axis
094 float aay = atan(accx / accz) * 180 / pi;      // the Angle between the X-axis and the z-axis
095 float aaz = atan(accz / accy) * 180 / pi;      // the Angle between the z axis and the y axis
096
097 long aax_sum = 0;                          // the sliding weighted filtering algorithm for the
098 original accelerometer data
099 long aay_sum = 0;
100 long aaz_sum = 0;
101 for(int i=1;i<n_sample;i++)
102 {
103     aaxs[i-1] = aaxs[i];
104     aax_sum += aaxs[i] * i;
105     aays[i-1] = aays[i];
106     aay_sum += aays[i] * i;
107     aazs[i-1] = aazs[i];
108     aaz_sum += aazs[i] * i;
109 }
110
111 aaxs[n_sample-1] = aax;
112 aax_sum += aax * n_sample;
113 aax = (aax_sum / (11*n_sample/2.0)) * 9 / 7.0; // Angle am to 0-90°
114 aays[n_sample-1] = aay;                      // here the appropriate coefficient is obtained by the
115 experimental method
116 aay_sum += aay * n_sample;                   // the coefficient in this example is 9/7
117 aay = (aay_sum / (11*n_sample/2.0)) * 9 / 7.0;
118 aazs[n_sample-1] = aaz;
119 aaz_sum += aaz * n_sample;
120 aaz = (aaz_sum / (11*n_sample/2.0)) * 9 / 7.0;
121
122 float gyroX = - (GyX-gxo) / GyroRatio * dt; // angular velocity in the X-axis
123 float gyroY = - (GyY-gyo) / GyroRatio * dt; // angular velocity in the Y-axis
124 float gyroZ = - (GyZ-gzo) / GyroRatio * dt; // angular velocity in the Z-axis
125 agx += gyroX;                               //Angular velocity integral
126 agy += gyroY;
127 agz += gyroZ;
128
129 /* kalman start */
130 Sx = 0; Rx = 0;
131 Sy = 0; Ry = 0;
132 Sz = 0; Rz = 0;
133
134 for(int i=1;i<10;i++)
135 {
136     // average calculation of measured values
137     a_x[i-1] = a_x[i];                      // is the average acceleration
138     Sx += a_x[i];
139     a_y[i-1] = a_y[i];
140     Sy += a_y[i];
141     a_z[i-1] = a_z[i];
142     Sz += a_z[i];
143 }
144
145 a_x[9] = aax;
146 Sx += aax;

```

```

146     Sx /= 10;                                // average acceleration of the X-axis
147     a_y[9] = aay;
148     Sy += aay;
149     Sy /= 10;                                // average acceleration of the Y-axis
150     a_z[9] = aaz;
151     Sz += aaz;
152     Sz /= 10;
153
154     for(int i=0;i<10;i++)
155     {
156         Rx += sq(a_x[i] - Sx);
157         Ry += sq(a_y[i] - Sy);
158         Rz += sq(a_z[i] - Sz);
159     }
160
161     Rx = Rx / 9;                                // to get the variance
162     Ry = Ry / 9;
163     Rz = Rz / 9;
164
165     Px = Px + 0.0025;
166     Kx = Px / (Px + Rx);                        // calculate the kalman gain
167     agx = agx + Kx * (aax - agx);                // the gyroscope Angle is superimposed with the
168     accelerometer velocity
169     Px = (1 - Kx) * Px;                        // update the p value
170
171     Py = Py + 0.0025;
172     Ky = Py / (Py + Ry);
173     agy = agy + Ky * (aay - agy);
174     Py = (1 - Ky) * Py;
175
176     Pz = Pz + 0.0025;
177     Kz = Pz / (Pz + Rz);
178     agz = agz + Kz * (aaz - agz);
179     Pz = (1 - Kz) * Pz;
180     /* kalman end */
181
182     display.setCursor(50, 0); // Center display
183     display.print(agx);
184     display.setCursor(50, 3); // Center display
185     display.print(agy);
186     display.setCursor(50, 6); // Center display
187     display.print(agz);
188     // Serial.print(agx);Serial.print(",");
189     // Serial.print(agy);Serial.print(",");
190     // Serial.print(agz);Serial.println();
191 }

```

Code explanation

Initialization Stage:

In the setup function, initialize I2C communication and wakes up the MPU-6050, then initialize OLED in order to view the mpu6050 results.

Loop Control Process:

Use count6Axle() function to get the data of MPU-6050 then display the data on OLED.