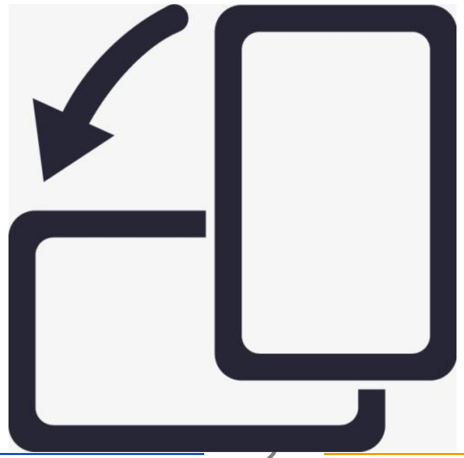
Acceleration

Acceleration

- When you tilt your smartphone, the screen probably turns from portrait to landscape.
- How did the phone know?
- Smartphones have a built-in accelerometer, and gravity is indistinguishable from acceleration downward, so the downward pull of gravity helps tell the phone which way it's oriented.



Acceleration

- Some games are controlled by tilting a phone in the air, using the accelerometer. The popular Wii game console uses an accelerometer for its controllers.
- Hard disks found in laptops and desktops can now take a lot of punishment. Many are rated to take a shock of 150 g (when off), an acceleration that would immediately kill any human.
- To brace for impact, some drives will turn themselves off: when the hard disk accelerometer detects it's in free fall, it automatically moves the actuator arm away from the sensitive plates.

Acceleration

- Have you ever tried to ride a self-balancing device, like a Segway or a Solowheel? After perhaps a shaky start, it almost feels like a miracle that the device stays upright.
- When a self-balancing device detects that it’s about to fall over forward, it quickly moves its wheels forward, turning itself upright again.
- Self-balancers measure angular velocity with a gyroscope. (An accelerometer would gather too many cumulative errors to work in a self-balancing device.)

Contents

Acceleration vs. Angular Velocity

- **Acceleration** is the rate at which an object's velocity changes (when it's slowing down or speeding up).
  - Acceleration is measured in g, as a multiple of the acceleration caused by Earth's gravity.
  - Another commonly used unit of acceleration is meters per second squared (m/s^2).
  - Free-fall acceleration (1 g) is 9.81 m/s^2.
- **Angular velocity** measures the rotational speed of an object, as well as the axis that it's rotating around.
- Depending on your project, you might need acceleration, angular velocity, or even both.

Acceleration vs. Angular Velocity

- Gyroscopes measure angular velocity, or how fast the sensor is rotating around an axis.
- For example, a gyroscope might report it's rotating at 10 degrees per second. They are used in self-balancers and airplane gyrocompasses.

Table 8-1. Accelerometer vs. gyroscope

传感器	测量	说明	单位	与重力相关
加速度传感器	加速度	速度的变化量，加速或减速	$m/s/s=m/s^2$	是，向下1g
陀螺仪	角速度	角度的变化量，旋转	Rad/s	否

Contents

Experiment: Accelerate with MX2125

- MX2125 is a simple two-axis acceleration sensor (see Figure 8-1).
- It reports acceleration as a pulse length, making the interface and code simple.
- The real, physical world is three dimensional. Objects can go
  - up and down (y),
  - left and right (x),
  - and back and forth (z).
- A two-axis sensor only measures two of these axes.

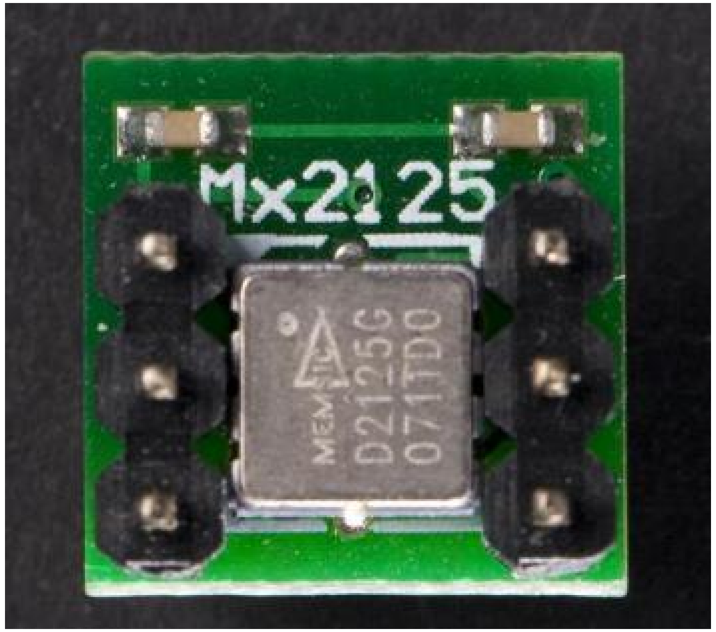


Figure 8-1. MX2125 sensor

Experiment: Accelerate with MX2125

- The MX2125 only measures up to 3 g per axis. But some sensors can measure extreme acceleration.
- For example, the maximum measured acceleration of the ADXL377 (200 g), is much more than would kill any human.
- Thus, it’s more than is experienced in a shuttle launch or high-g maneuvers in fighter jets. It could measure an object accelerating faster than a bullet fired from a pistol.

Experiment: Accelerate with MX2125

- When we made an early prototype for an Aalto-1 satellite sun sensor, even the satellite spec did not require acceleration this tough.
- It's unlikely that you would need to measure such an extreme acceleration, and it would probably not be possible with a breadboard setup (because the acceleration needed to test would shake your project apart !).
- The cost is quite minimal, though. However, the wider the area of measured acceleration (from -250 g to +250 g), the less precise the device is.

Accelerometer Code and Connection for Arduino

- Figure 8-2 shows the circuit diagram for Arduino. Wire it up as shown, and load the code from Example 8-1.

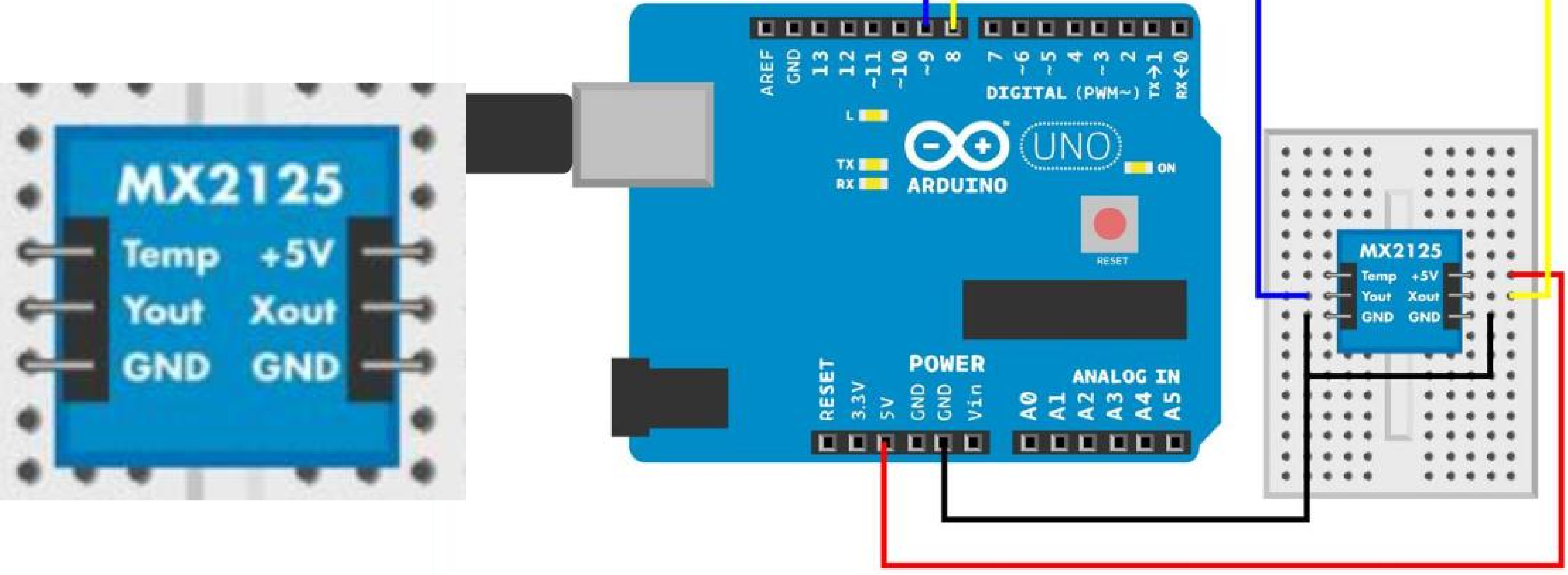


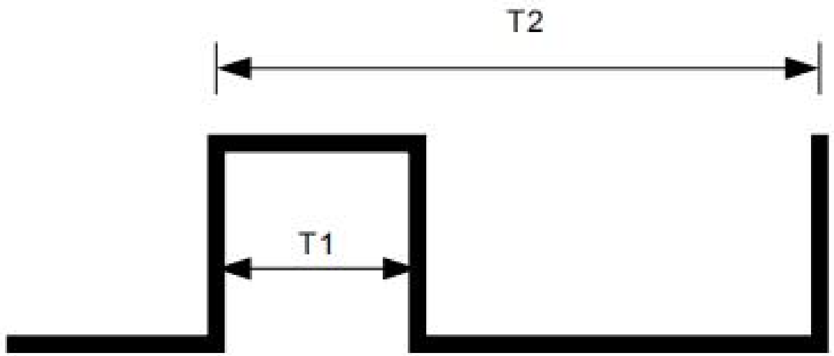
Figure 8-2. MX2125 dual axis accelerometer circuit for Arduino

Example 8-1. mx2125.ino

```
// mx2125.ino - measure acceleration on two axes using MX2125 and print to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int xPin = 8;
const int yPin = 9;
void setup() {
	Serial.begin(115200);
	pinMode(xPin, INPUT);
	pinMode(yPin, INPUT);
}
void loop() {
	int x = pulseIn(xPin, HIGH);	// ❶ 知道脉冲长度就能计算加速度，返回脉冲的长度，单位为us
	int y = pulseIn(yPin, HIGH);
	int x_mg = ((x / 10) - 500) * 8;	// ❷
	int y_mg = ((y / 10) - 500) * 8;
	Serial.print("Axels x: ");
	Serial.print(x_mg);
	Serial.print(" y: ");
	Serial.println(y_mg);
	delay(10);
}
```

Decoding MX2125 Pulse Length

- MX2125 works by heating a bubble of gas inside the device, and then measuring how the air bubble moves.
- Usually, an accelerometer's conversion formulas are found on data sheets.

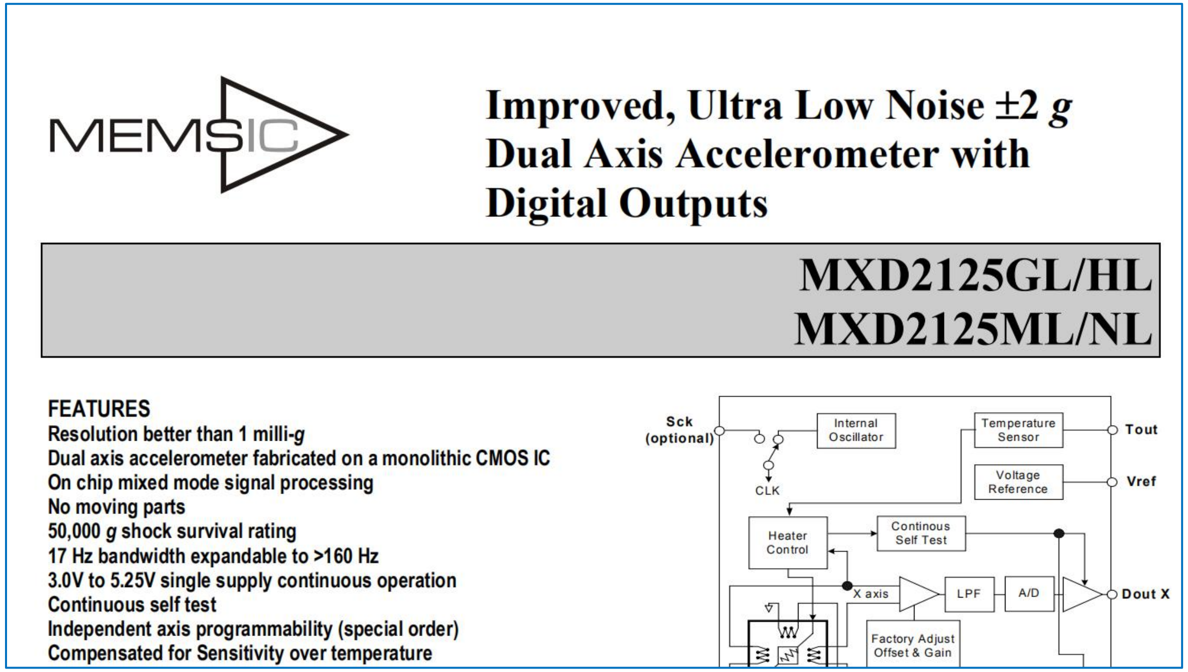


A (g)= (T1/T2 - 0.5)/20%

0g = 50% Duty Cycle

T2= 2.5ms or 10ms (factory programmable)

Figure 3: Typical output Duty Cycle



MEMSIC

Improved, Ultra Low Noise ±2 *g* Dual Axis Accelerometer with Digital Outputs

MXD2125GL/HL
MXD2125ML/NL

FEATURES

Resolution better than 1 milli-*g*
Dual axis accelerometer fabricated on a monolithic CMOS IC
On chip mixed mode signal processing
No moving parts
50,000 *g* shock survival rating
17 Hz bandwidth expandable to >160 Hz
3.0V to 5.25V single supply continuous operation
Continuous self test
Independent axis programmability (special order)
Compensated for Sensitivity over temperature

Decoding MX2125 Pulse Length

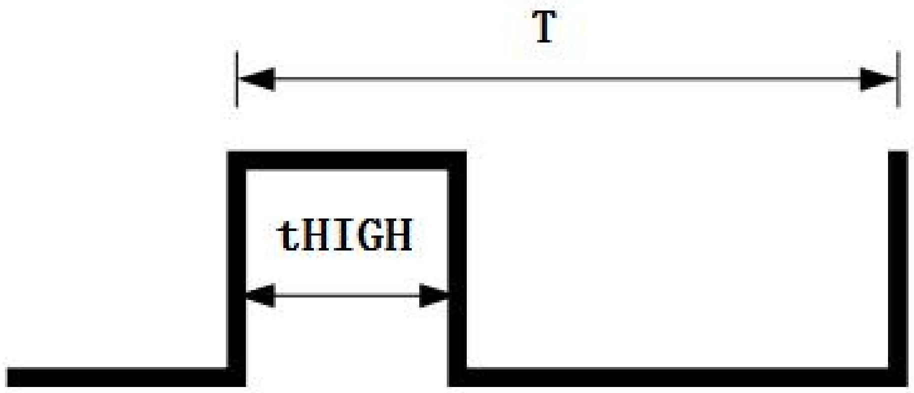
- Finding Data Sheets
  - Search for the data sheet. The obvious search query is the code name of the component and the word “datasheet.”
  - You may also find the data sheet on the website where you bought the component from (it’s usually on the product detail page for the part).
  - The actual data sheet contained a lot of information not found in the other documents.

Decoding MX2125 Pulse Length

- When power is on, the MX2125 reports the acceleration on each axis, 100 pulses a second.
- Consecutive HIGH and LOW signals form a 100 Hz square wave.
- The more acceleration there is, the more time the wave spends in the HIGH portion, and the less time in the LOW portion.
- You can read these pulses to determine acceleration.

Decoding MX2125 Pulse Length

- One full wave contains one HIGH and one LOW.
- The time taken by one wave (HIGH+LOW) is called period (T).
- Let's call the time of HIGH part tHIGH (time of HIGH).
- The duty cycle tells you how much of the wave is HIGH. The duty cycle is a percentage, for example 50% (0.50) or 80% (0.80).
  - dutyCycle = tHIGH / T



Decoding MX2125 Pulse Length

- According to the data sheet and other documents, the period T is set to 10 ms by default: dutyCycle = tHIGH / 10 ms

DUTY CYCLE DEFINITION

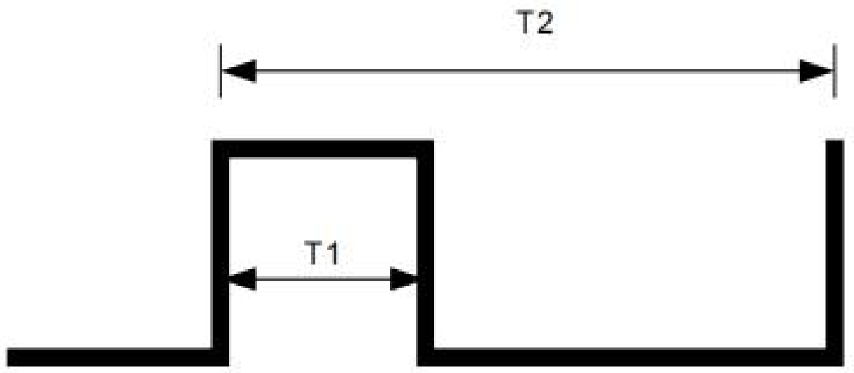
The MXD2125GL/HL/ML/NL has two PWM duty cycle outputs (x,y). The acceleration is proportional to the ratio T1/T2. The zero *g* output is set to 50% duty cycle and the sensitivity scale factor is set to 20% duty cycle change per *g*. These nominal values are affected by the initial tolerance of the device including zero *g* offset error and sensitivity error. This device is offered from the factory programmed to either a 10ms period (100 Hz) or a 2.5ms period (400Hz).

T1	Length of the "on" portion of the cycle.
T2 (Period)	Length of the total cycle.
Duty Cycle	Ratio of the "0n" time (T1) of the cycle to the total cycle (T2). Defined as T1/T2.

Decoding MX2125 Pulse Length

- Here's the acceleration formula from the data sheet:
  - A = (tHIGH/T-0.50)/20%
- Replacing tHIGH/T with dutyCycle and 20% with 0.2:
  - A = (dutyCycle-0.50)/0.2
- Now it can be written as follows:
  - A = 5*(dutyCycle-0.50)

or: A = 5*(tHIGH/T-0.50)



A (g)= (T1/T2 - 0.5)/20%

0g = 50% Duty Cycle

T2= 2.5ms or 10ms (factory programmable)

Figure 3: Typical output Duty Cycle

Decoding MX2125 Pulse Length

- When there is no acceleration (0 g), the duty cycle is 50%:
  - 0 = 5*(dutyCycle-0.50)
  - 0/5 = dutyCycle-0.50
  - 0 = dutyCycle-0.50
  - 0.50 = dutyCycle

Decoding MX2125 Pulse Length

- At the time we originally checked, the Parallax and Memsic documentation conflicted on the multiplier:
  - the Memsic documentation： A = (tHIGH/T-0.50) / 20%
  - the Parallax documentation： A = (tHIGH/T-0.50) / 12.5%

- In our experiments, we found 1/12.5% (8) to give proper readings with the breakout board from Parallax.

Decoding MX2125 Pulse Length

- This is why you need to be careful with data sheets you find online: always verify the values with experimentation.
- So we will use 8 as the multiplier:
  - A = 8*(tHIGH/T-0.50)

Example 8-1. mx2125.ino

```
// mx2125.ino - measure acceleration on two axes using MX2125 and print to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int xPin = 8;
const int yPin = 9;
void setup() {
		Serial.begin(115200);
		pinMode(xPin, INPUT);
		pinMode(yPin, INPUT);
}
void loop() {
		int x = pulseIn(xPin, HIGH);		// ❶ 知道脉冲长度就能计算加速度，返回脉冲的长度，单位为us
		int y = pulseIn(yPin, HIGH);
		int x_mg = ((x / 10) - 500) * 8;	// ❷
		int y_mg = ((y / 10) - 500) * 8;
		Serial.print("Axels x: ");
		Serial.print(x_mg);
		Serial.print(" y: ");
		Serial.println(y_mg);
		delay(10);
}
```

Accelerometer Code and Connection for Raspberry Pi

- Figure 8-3 shows the wiring diagram for the Raspberry Pi.
- Hook everything up as shown, and then run the code shown in Example 8-2.

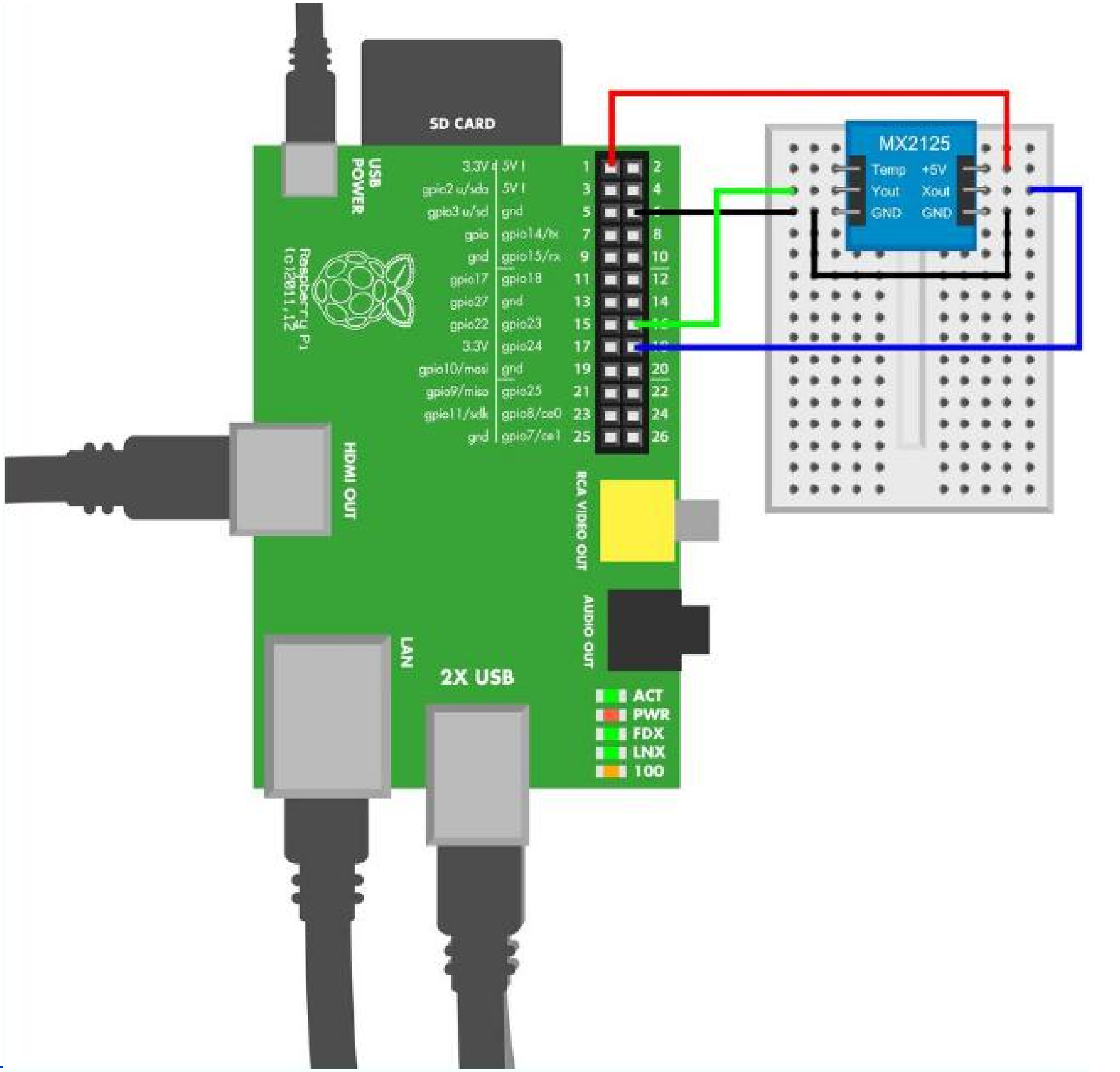


Figure 8-3. MX2125 dual axis accelerometer circuit for Raspberry Pi

Example 8-2. mx2125.py

```
# mx2125.py - print acceleration axel values.
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_gpio as gpio

xPin = 24
yPin = 23

def readAxel(pin):
	gpio.mode(pin, "in")
	gpio.interruptMode(pin, "both")
	return gpio.pulseInHigh(pin)    # ❶
```

```
def main():
	x_g = 0
	y_g = 0
	while True:
		x = readAxel(xPin) / 1000     # 转换成ms
		y = readAxel(yPin) / 1000
		if(x < 10):                   # ❷
			x_g = ((x / 10) - 0.5) * 8    # ❸
		if(y < 10):
			y_g = ((y / 10) - 0.5) * 8
		print ("Axels x: %fg, y: %fg" % (x_g, y_g))
		time.sleep(0.5)

if __name__ == "__main__":
	main()
```

Contents

Experiment: Accelerometer and Gyro Together

- When an accelerometer is not moving, it detects gravity and can tell where down is.
- A gyroscope can tell the orientation reliably, even if you spin it around and around. A gyroscope ignores gravity, though.

- Could we combine an accelerometer and gyroscope to get both benefits?
  - Yes.

Experiment: Accelerometer and Gyro Together

- An IMU (inertial measurement unit) combines multiple sensors and (optionally) some logic to get more precise and reliable motion information.
- In this experiment, you'll work hands-on with the basic features of the MPU 6050.
- In general, IMUs are more expensive and more precise than plain accelerometers and gyros. They also use more advanced protocols to communicate, such as I2C, instead of a simple pulse width signaling protocol.

Experiment: Accelerometer and Gyro Together

- The MPU 6050 (Figure 8-4) has an accelerometer, gyro, and microcontroller on the same chip.
- Even though space isn't a premium when you're in the breadboard prototyping stage, it's nice to know that all this functionality fits in a tiny surface-mounted component, just in case you ever run short of circuit real estate.

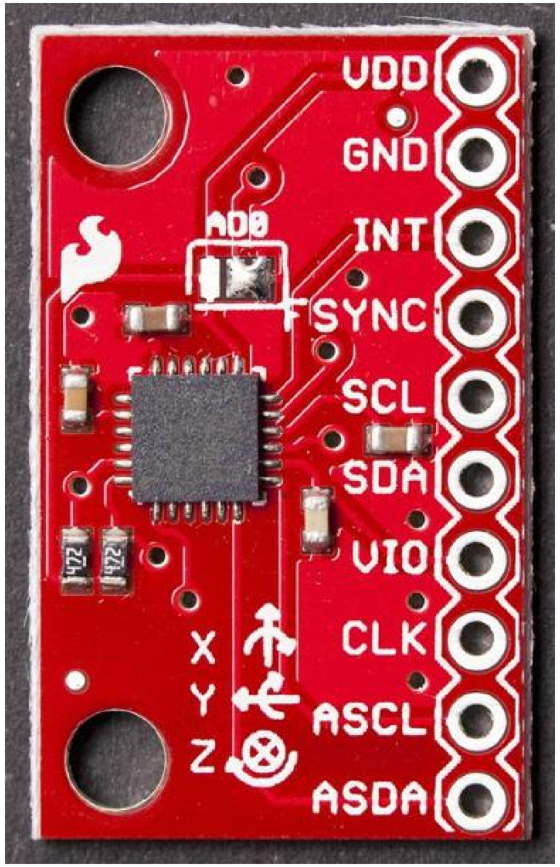


Figure 8-4. The MPU 6050

Experiment: Accelerometer and Gyro Together

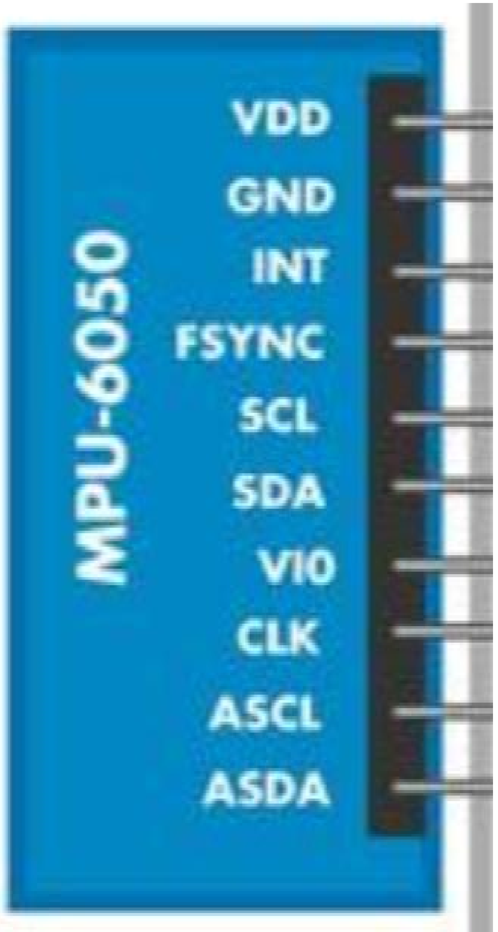
- The MPU 6050 uses the I2C protocol.
- Thanks to the python-smbus library, Raspberry Pi code is much simpler and easier than the equivalent Arduino code.
- In general, Raspberry Pi handles complicated protocols in less code than Arduino.

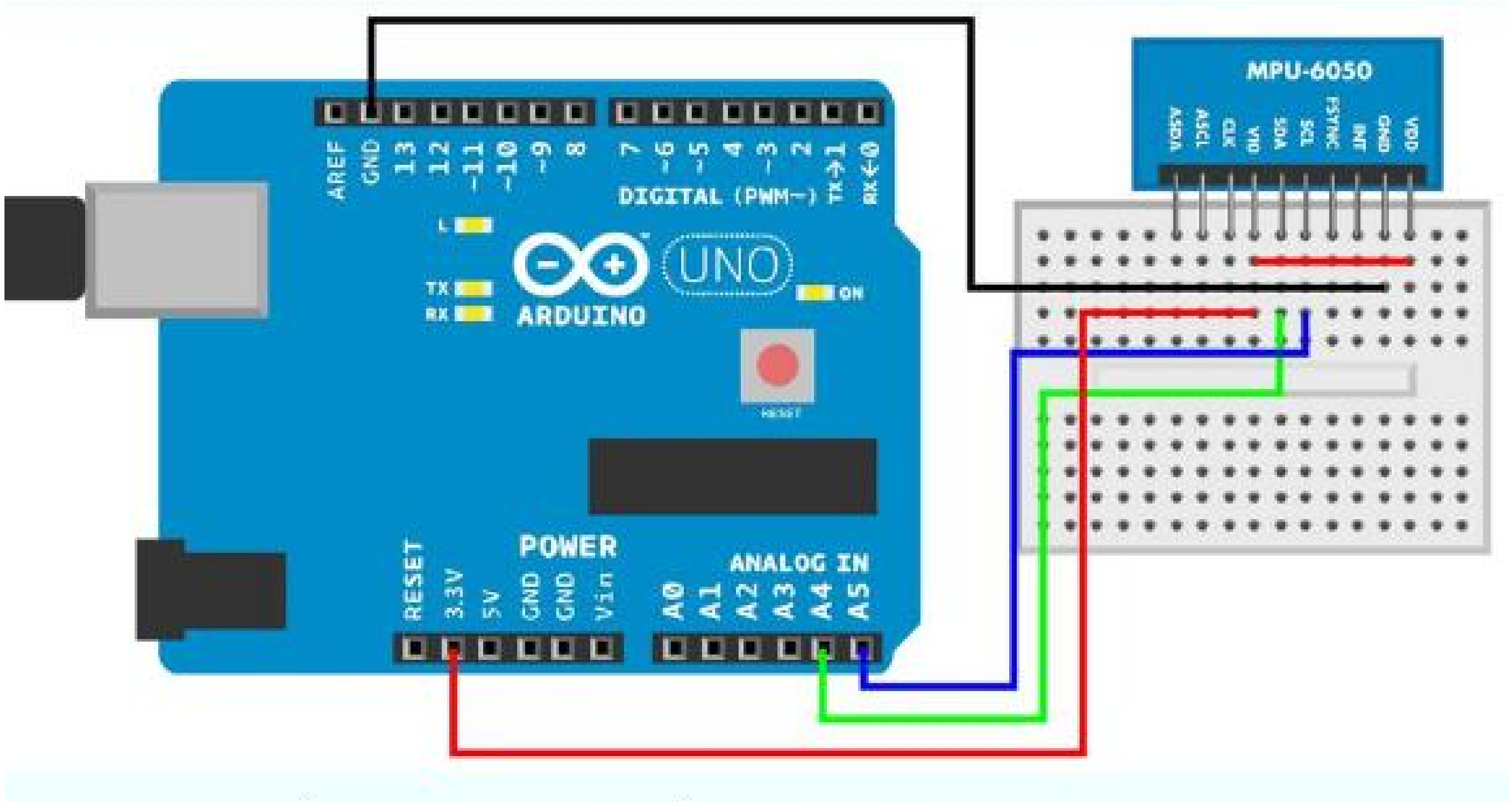
Experiment: Accelerometer and Gyro Together

- **Industry Standard Protocols：Most devices use one of the industry standard protocols instead of inventing their own.**
  - I2C is one of the easiest industry standard protocols.
  - SPI is also a common industry standard protocol.
  - Serial is often found in disguise: serial over USB, serial over Bluetooth, serial over some jumper wires.

MPU 6050 Code and Connection for Arduino

- Figure 8-5 shows the wiring diagram for Arduino. Hook everything up, and then run the code in Example 8-3.





-5. MPU 6050 (accelerometer+gyro) circuit for Arduino

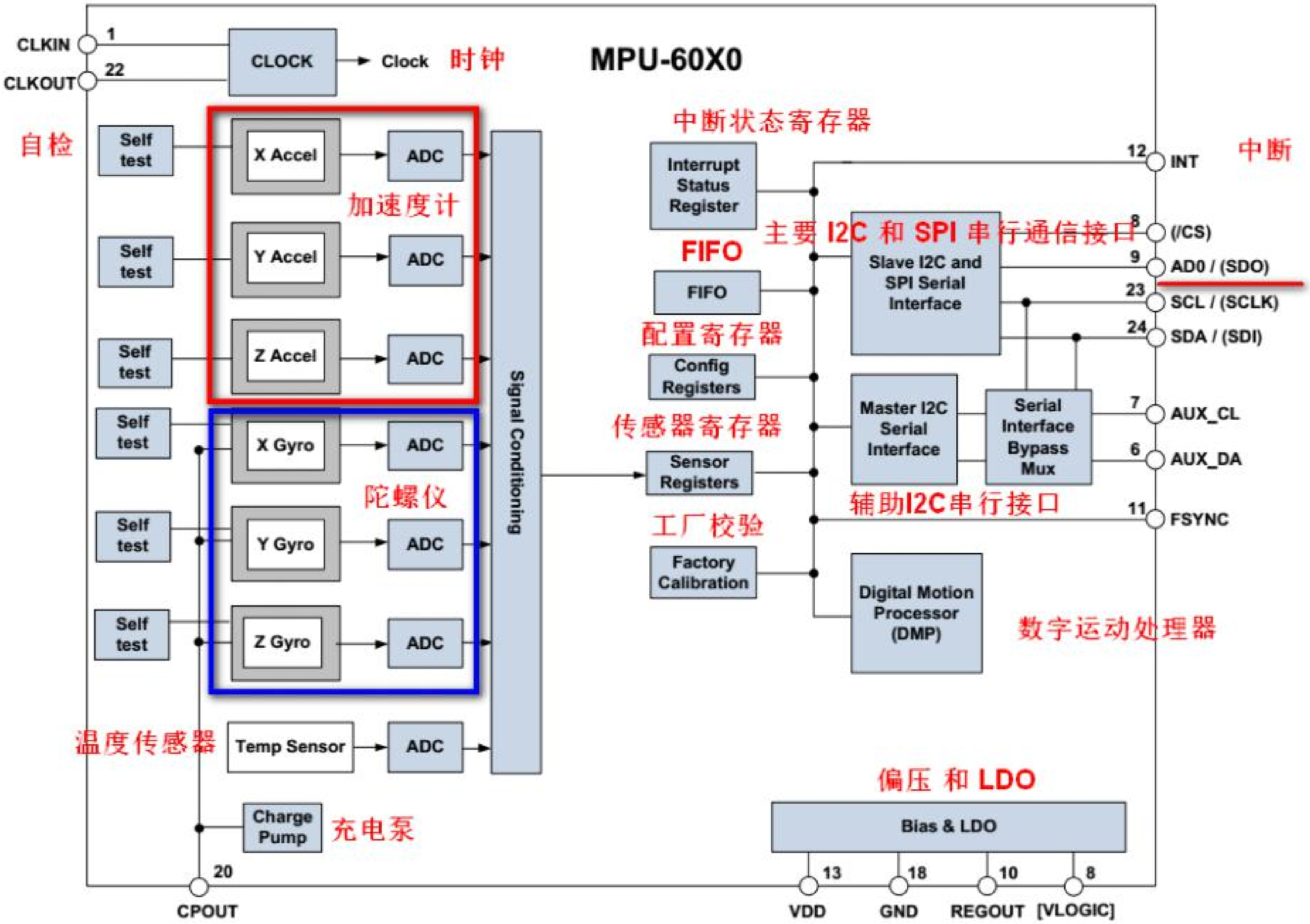
MPU 6050 Code and Connection for Arduino

- Difficult code! The code for MPU 6050 contains more difficult programming concepts than most other code examples in this book.
- If you find endianness, bit shifting, and structs difficult, you can simply use the code and play with the values. You don’t need to deeply understand the code to use it.
- If you want to understand the code, see the explanations after the code, such as Hexadecimal, Binary, and Other Numbering Systems and Bitwise Operations.

Example 8-3. mpu_6050.ino

```
// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>                                   // ❶

const char i2c_address =  0x68;                     // ❷

const unsigned char sleep_mgmt = 0x6B;              // ❸
const unsigned char accel_x_out = 0x3B;

struct data_pdu // {                                // ❹
        int16_t x_accel;                            // ❺
        int16_t y_accel;
        int16_t z_accel;
        int16_t temperature;                        // ❻
        int16_t x_gyro;                             // ❼
        int16_t y_gyro;
        int16_t z_gyro;
};
```

MPU-6050简介



MPU-60X0
CLKIN
1
CLKOUT
22
CLOCK
Clock
时钟
自检
Self test
X Accel
Y Accel
Z Accel
ADC
加速度计
X Gyro
Y Gyro
Z Gyro
陀螺仪
温度传感器
Temp Sensor
Charge Pump
充电泵
20
CPOUT
Signal Conditioning
中断状态寄存器
Interrupt Status Register
FIFO
配置寄存器
Config Registers
传感器寄存器
Sensor Registers
工厂校验
Factory Calibration
主要 I2C 和 SPI 串行通信接口
Slave I2C and SPI Serial Interface
Master I2C Serial Interface
Serial Interface Bypass Mux
辅助I2C串行接口
Digital Motion Processor (DMP)
数字运动处理器
12 INT
中断
8 (/CS)
9 AD0 / (SDO)
23 SCL / (SCLK)
24 SDA / (SDI)
7 AUX_CL
6 AUX_DA
11 FSYNC
偏压 和 LDO
Bias & LDO
13 VDD
18 GND
10 REGOUT
8 [VLOGIC]

MPU-6050简介

- **MPU-60X0 是全球首例9 轴运动处理传感器**
  - 它集成了3 轴MEMS 陀螺仪，3 轴MEMS加速度计，以及一个可扩展的数字运动处理器DMP(Digital Motion Processor)。
  - 可用I2C接口连接一个第三方的数字传感器，比如磁力计。
  - 扩展之后就可以通过其 I2C 或 SPI 接口输出一个9 轴的信号（SPI 接口仅在MPU-6000 可用）。
  - MPU-60X0 也可以通过其I2C 接口连接非惯性的数字传感器，比如压力传感器。

MPU-6050简介

- MPU-60X0 对陀螺仪和加速度计分别用了三个16 位的ADC，将其测量的模拟量转化为可输出的数字量。
- 为了精确跟踪快速和慢速的运动，传感器的测量范围都是用户可控的
  - 陀螺仪可测范围为： ±250，±500，±1000，±2000°/秒（dps）
  - 加速度计可测范围为：±2，±4，±8，±16g

MPU-6050简介

- **数字运动处理器（DMP）：**
  - DMP 从陀螺仪、加速度计以及外接的传感器接收并处理数据
  - 处理结果可以从 DMP 寄存器读出，或通过 FIFO 缓冲。
  - DMP 有权使用 MPU 的一个外部引脚产生中断。

MPU-6050简介

- 一个片上1024 字节的FIFO，有助于降低系统功耗。
- 和所有设备寄存器之间的通信采用400kHz 的I2C 接口或1MHz 的SPI 接口（SPI 仅MPU-6000 可用）。
- 对于需要高速传输的应用，对寄存器的读取和中断可用20MHz 的SPI。
- 另外，片上还内嵌了一个温度传感器和在工作环境下仅有±1%变动的振荡器。
- 芯片尺寸4×4×0.9mm，采用QFN 封装（无引线方形封装），可承受最大10000g 的冲击，并有可编程的低通滤波器。

MPU-6050简介

- 关于电源，MPU-60X0 可支持VDD 范围2.5V±5%，3.0V±5%，或3.3V±5%。
- 另外MPU-6050 还有一个VLOGIC 引脚，用来为I2C 输出提供逻辑电平。VLOGIC 电压可取1.8±5%或者VDD

MPU-6050应用领域

- AirSign™技术(安全/身份验证)
- TouchAnywhere™技术("不接触" UI应用程序控制/导航)
- MotionCommand™技术(手势捷径)
- Motion-enabled游戏和应用程序框架
- InstantGesture™iG™手势识别
- 基于位置服务的兴趣点、航迹推算
- 手机和便携式游戏
- 各自游戏控制器
- 3d网络连接遥控器,机顶盒,3 d小鼠
- 可穿戴传感器对健康、健身和体育
- 玩具

MPU-6050简介

➤ 重要寄存器

□ 电源管理寄存器1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

- **DEVICE_RESET 位**用来控制复位，设置为 1，复位 MPU6050，复位结束后， MPU硬件自动清零该位

MPU-6050简介

➢ 重要寄存器

☐ 电源管理寄存器1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

- **SLEEEP 位**用于控制 MPU6050 的工作模式，复位后，该位为 1，即进入了睡眠模式（低功耗），所以我们要清零该位，以进入正常工作模式

MPU-6050简介

➢ 重要寄存器

◻ 电源管理寄存器1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

- TEMP_DIS 用于设置是否使能温度传感器，设置为 0，则使能
- CLKSEL[2:0] 用于选择系统时钟源（内部晶振、外部晶振、PLL（X-Y-Z陀螺仪参考）等）

MPU-6050简介

➢ 重要寄存器

◻ 加速度传感器数据输出寄存器

Type: Read Only

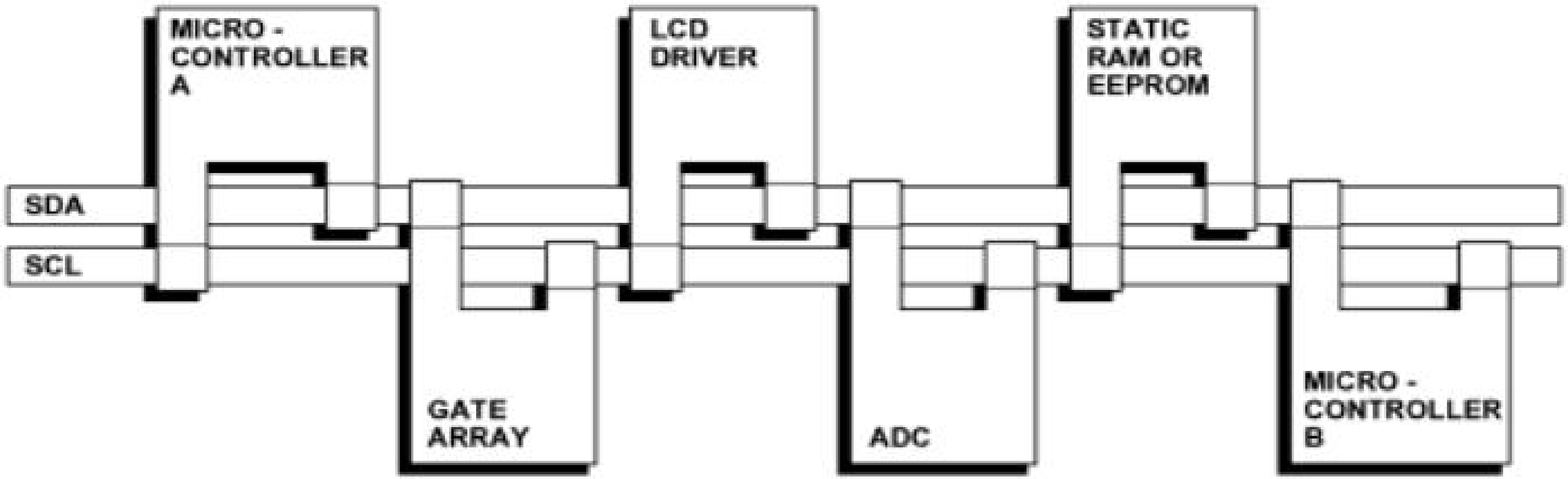
Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

- 通过读取这6个寄存器，就可以读到加速度传感器 x/y/z 轴的值，比如读 x 轴的数据，可以通过读取 0X3B（高 8 位）和0X3C（低8位）寄存器得到，其他轴以此类推。

补充：I²C总线原理及应用（1）

➢ **什么是I²C总线**（Inter Integrated Circuit **内部集成**）

- 是PHILIPS公司开发的一种简单、双向、二线制、同步串行总线，作为专利的控制总线，已成为世界性的工业标准
- 只需两根线（串行时钟线和串行数据线）即可在连接于总线上的器件之间传送信息。



补充：I²C总线原理及应用（2）

➢ I²C总线的特点

- I2C总线的优点是简单和有效性。
  - 由于接口直接在组件之上，因此I2C总线占用的空间非常小，减少了电路板的空间和芯片管脚的数量，降低了互联成本。能够以10Kbps的最大传输速率支持40个组件
- 支持多主控，其中任何能够进行发送和接收的设备都可以成为主总线。当然，任何时间点上只能有一个主控。
- 一个主控能够控制信号的传输和时钟频率。

补充:I^{2}C总线原理及应用（3）

➢I2C总线工作原理

□总线的构成

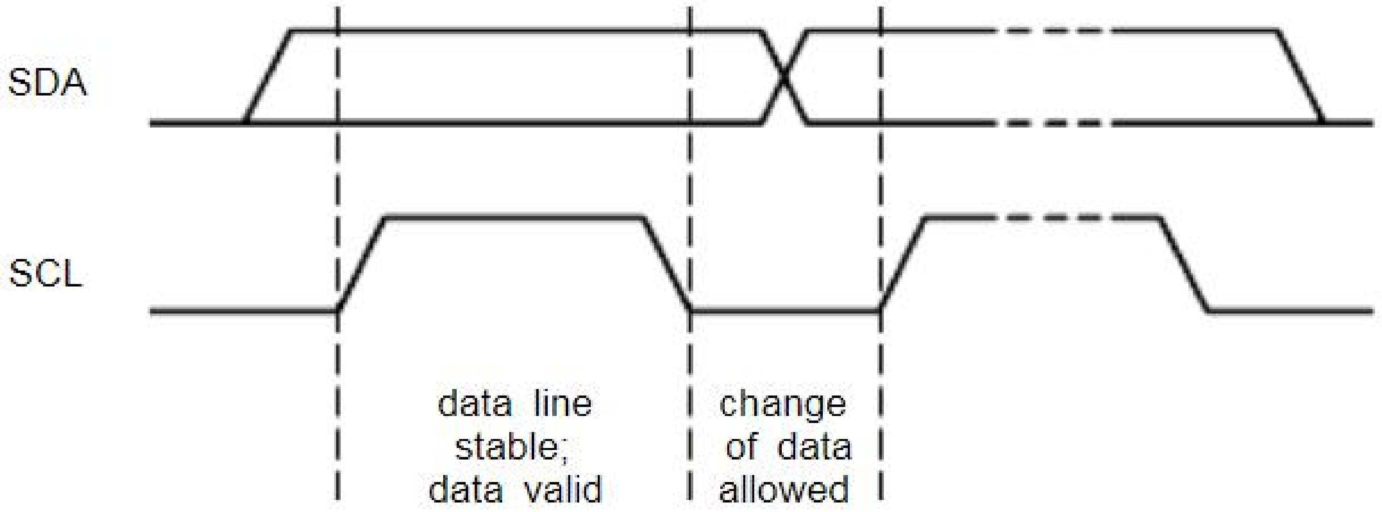
- I2C总线是由数据线SDA和时钟SCL构成的串行总线。
- 在CPU与被控IC之间、IC与IC之间进行双向传送，最高传送速率100kbps。
- 各种被控制电路均并联在这条总线上，但就像电话机一样只有拨通各自的号码才能工作，所以每个电路和模块都有唯一的地址。
- 在信息的传输过程中，I2C总线上并接的每一模块电路既是主控器（或被控器），又是发送器（或接收器），这取决于它所要完成的功能。
- CPU发出的控制信号分为地址码和控制量两部分。

补充:I²C总线原理及应用（3）

➢I2C总线工作原理

❑数据的有效性

- SDA线上的数据必须在时钟的高电平周期保持稳定
- 数据线的高或低电平状态只有在SCL线的时钟信号是低电平时才能改变



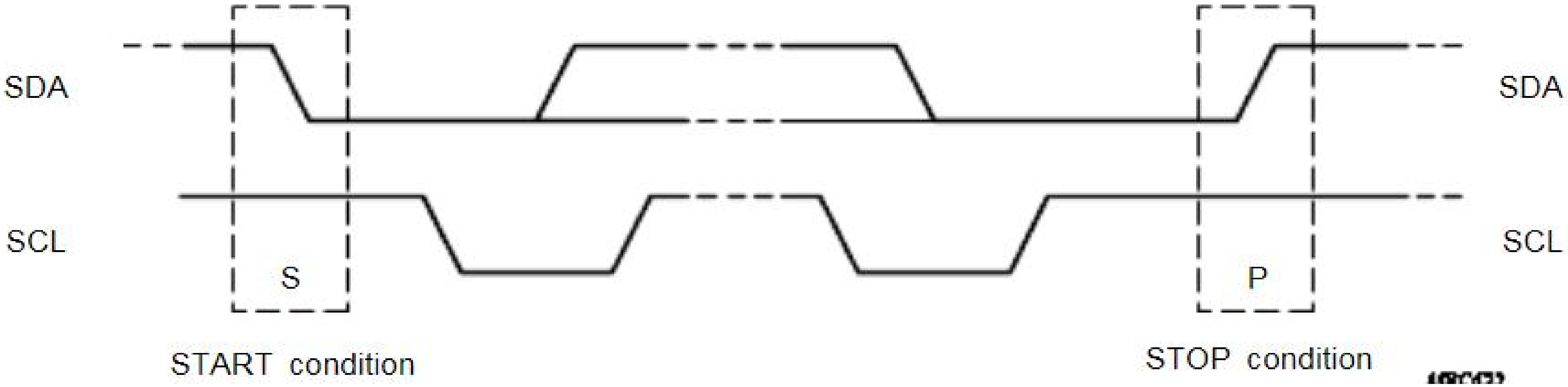
补充：I²C总线原理及应用（4）

➢I2C总线工作原理

- **信号类型：** I2C总线在传送数据过程中共有三种类型信号，它们分别是：
  - 开始信号：SCL为高电平时，SDA由高电平向低电平跳变，开始传送数据。
  - 结束信号：SCL为高电平时，SDA由低电平向高电平跳变，结束传送数据。
  - 应答信号：接收数据的IC在接收到8bit数据后，向发送数据的IC发出特定的低电平脉冲，表示已收到数据。CPU向受控单元发出一个信号后，等待受控单元发出一个应答信号，CPU接收到应答信号后，根据实际情况作出是否继续传递信号的判断。若未收到应答信号，由判断为受控单元出现故障。

➢I2C总线工作原理

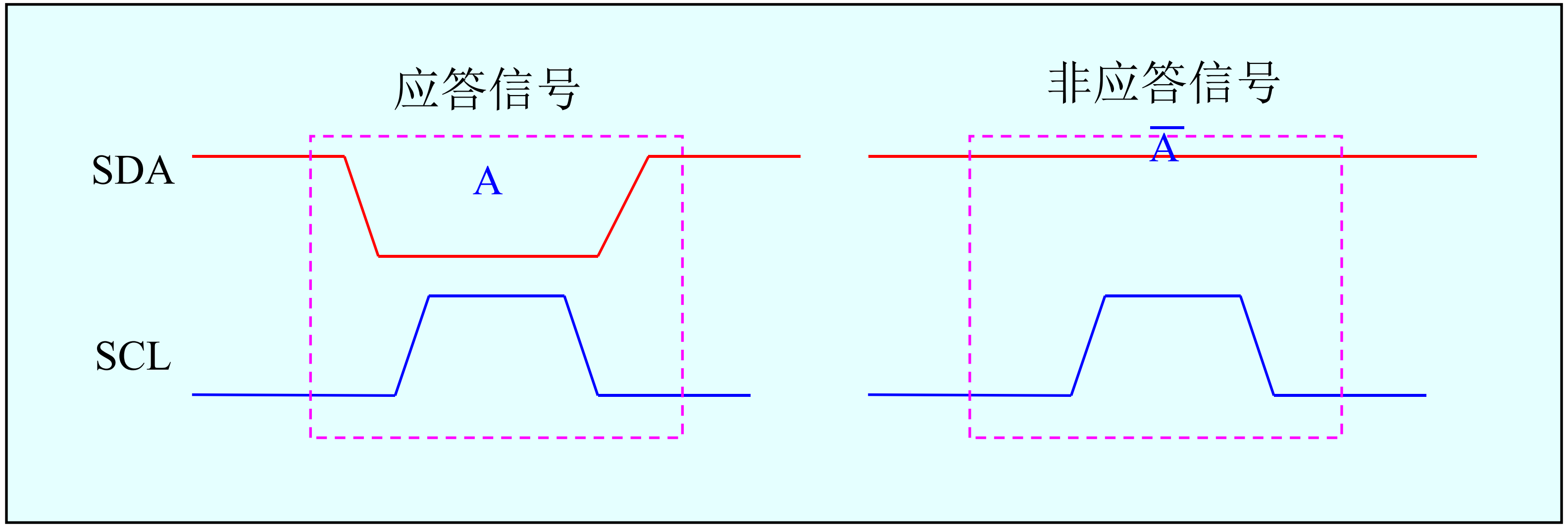
- **信号类型：**I2C总线在传送数据过程中共有三种类型信号，它们分别是：开始信号、结束信号、应答信号。



补充：I²C总线原理及应用（5）

➢I2C总线工作原理

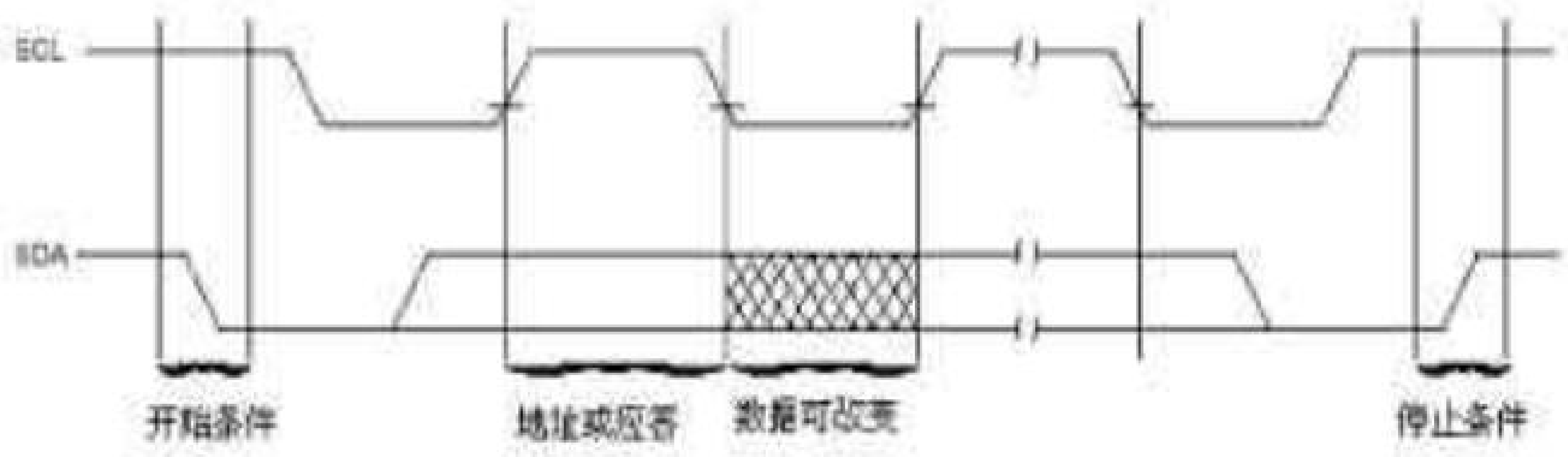
- **信号类型：**I2C总线在传送数据过程中共有三种类型信号，它们分别是：开始信号、结束信号、应答信号。



补充：I²C总线原理及应用（6）

➢I2C总线工作原理

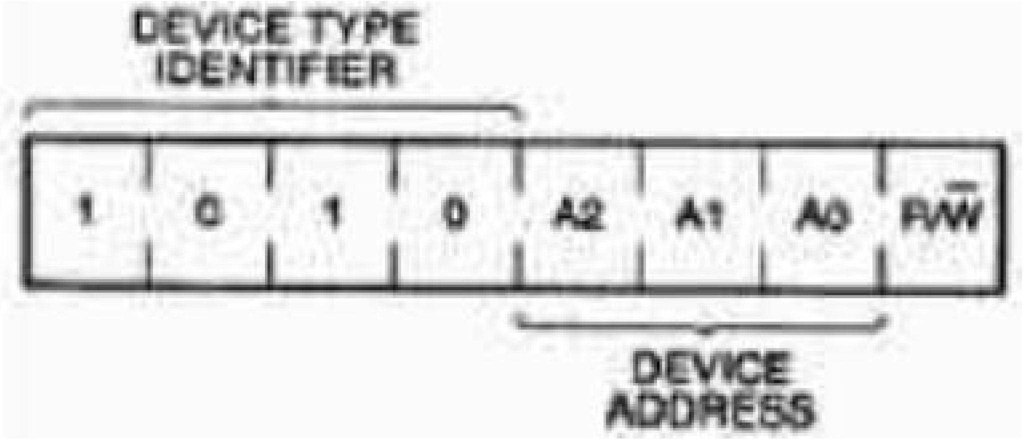
- **总线基本操作：**总线由主器件控制，主器件产生串行时钟（SCL）控制总线的传输方向，并产生起始和停止条件。SDA线上的数据状态仅在SCL为低电平的期间才能改变，SCL为高电平的期间，SDA状态的改变被用来表示起始和停止条件。



补充：I^2C总线原理及应用（7）

➢I2C总线工作原理

- **数据格式：**主控产生起始信号以后，开始传送数据，数据的第一个字节必须是控制字节，其中高四位为器件类型识别符（不同的芯片类型有不同的定义，EEPROM一般应为1010），接着三位为片选（即受控设备的地址），最后一位为读写位，当为1时为读操作，为0时为写操作。



补充：I²C总线原理及应用（8）

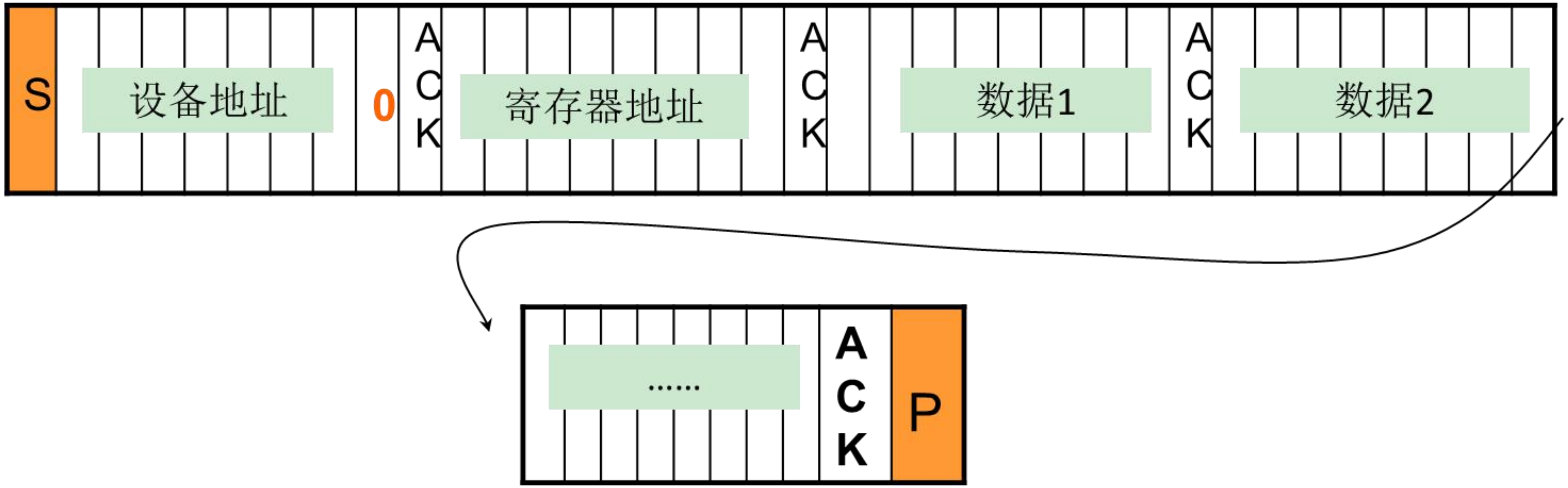
➢I2C总线工作原理

- **数据传送过程：** 在I2C总线上挂接的所有被控IC都有一个自己的地址，CPU在发送数据时，I2C总线上的所有被控IC都会将CPU发出的位于起始信号后面的地址与自己的地址相比较，如果两者相同，则该被控IC认为自己被CPU选中，然后按照读/写位规定的工作方式接收或发送数据。

➢I2C总线工作原理

◻ 写操作

在写入时，I2C主控设备先发送起始位（S），抢占总线；然后，发送7位设备地址和1位0，表示对设备的写入；接着就是向设备传送数据。

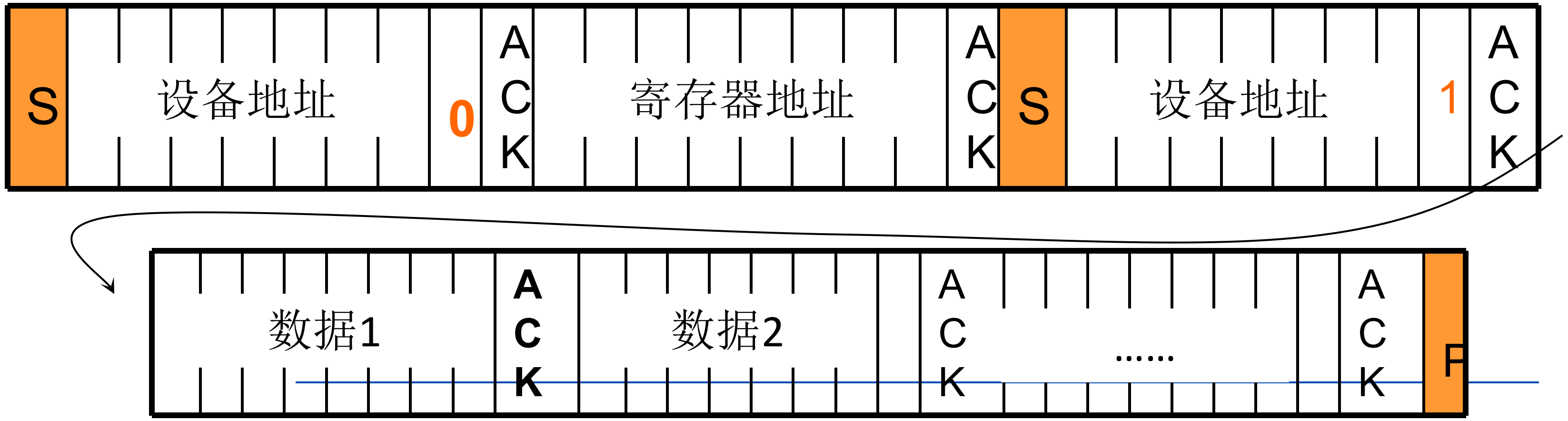


补充：I²C总线原理及应用（10）

➢I2C总线工作原理

◻ 读操作（稍微复杂一些，因为总线的数据传输方向要改变）

- I2C主控设备发送起始位（S），抢占总线；
- 发送7位的设备地址和1位0，表示对设备的写入；
- 发送要读取数据的寄存器地址；
- 再次发送起始位（S）
- 发送7位的设备地址和1位1，表示对设备的读取；
- 从设备读取数据



Example 8-3. mpu_6050.ino

```
// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>       //Arduino的I2C协议库，已经整合在IDE中，所以不需要独立安装，include即可

const char i2c_address =  0x68;         // MPU6050传感器的I2C地址

const unsigned char sleep_mgmt = 0x6B;  // 命令寄存器地址
const unsigned char accel_x_out = 0x3B;

struct data_pdu {  //结构体用于解码从传感器获取的数据
        int16_t x_accel;          //加速度变量
        int16_t y_accel;
        int16_t z_accel;
        int16_t temperature;      //温度变量
        int16_t x_gyro;           //角速度变量
        int16_t y_gyro;
        int16_t z_gyro;
};
```

Example 8-3. mpu_6050.ino

// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
void setup() {
	Serial.begin(115200);
	Wire.begin();	//初始化I2C通信

	//向电源管理寄存器写入命令0，从而唤醒传感器（MPU6050初始状态处于睡眠模式）
	write_i2c(sleep_mgmt,0x00);
}
```

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

```
//交换参数的两个字节。MPU6050是大端字节序，而 Arduino和大多处理器一样都是小端字节序。
int16_t swap_int16_t(int16_t value) {
	int16_t left = value << 8;
	int16_t right = value >> 8;
	right = right & 0xFF;
	return left | right;
}
```

Example 8-3. mpu_6050.ino

*// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)*
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
void loop() {
	data_pdu pdu;		// 创建类型为data_pdu的变量pdu，data_pdu是之前定义的结构体类型
	read_i2c(accel_x_out, (uint8_t *)&pdu, sizeof(data_pdu));	//读传感器数据填充PDU结构体
	//第一个参数是要读取的寄存器0x3B，第二个参数是结构的地址指针，第三个参数是要读取的字节量

	//将传感器返回的大端字节序数据，转换成Arduino使用的小端字节序
	pdu.x_accel = swap_int16_t(pdu.x_accel);
	pdu.y_accel = swap_int16_t(pdu.y_accel);
	pdu.z_accel = swap_int16_t(pdu.z_accel);
	pdu.temperature = swap_int16_t(pdu.temperature);
	pdu.x_gyro = swap_int16_t(pdu.x_gyro);
	pdu.y_gyro = swap_int16_t(pdu.y_gyro);
	pdu.z_gyro = swap_int16_t(pdu.z_gyro);
```

```
struct data_pdu // {
		int16_t x_accel;
		int16_t y_accel;
		int16_t z_accel;
		int16_t temperature;
		int16_t x_gyro;
		int16_t y_gyro;
		int16_t z_gyro;
};
```

Example 8-3. mpu_6050.ino

```
// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
float acc_x = pdu.x_accel / 16384.0f;      //将原始数据转换成真实世界中单位为g的数据
float acc_y = pdu.y_accel / 16384.0f;      //转换因子查手册获得的
float acc_z = pdu.z_accel / 16384.0f;      //为了获得浮点数结果，除数必须是浮点数类型的
```

Each 16-bit accelerometer measurement has a full scale defined in *ACCEL_FS* (Register 28). For each full scale setting, the accelerometers' sensitivity per LSB in *ACCEL_xOUT* is shown in the table below.

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*

Example 8-3. mpu_6050.ino

*// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)*
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
Serial.print("Accelerometer: x,y,z (");
Serial.print(acc_x,3); Serial.print("g, ");        //向串口监视器输出加速度值，单位是重力加速度g
Serial.print(acc_y,3); Serial.print("g, ");
Serial.print(acc_z,3); Serial.println("g)");

int zero_point = -512 - (340 * 35);   //  根据手册，把原始数据转换为摄氏度，先计算0℃对应的原始数据
double temperature = (pdu.temperature - zero_point) / 340.0;  // 计算温度的摄氏度表示
Serial.print("Temperature (C): ");
Serial.println(temperature,2);
```

The temperature in degrees C for a given register value may be computed as:

Temperature in degrees C = (TEMP_OUT Register Value as a signed quantity)/340 + 36.53

Example 8-3. mpu_6050.ino

// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
    Serial.print("Gyro: x,y,z (");
    Serial.print(pdu.x_gyro / 131.0f);         //  根据手册，将原始数据转换成角速度需要除以因子131.0
    Serial.print(“ deg/s, ”);
    Serial.print(pdu.y_gyro / 131.0f);         Serial.print(" deg/s, ");
    Serial.print(pdu.z_gyro / 131.0f);         Serial.println(" deg/s)");
    delay(1000);
}
```

Each 16-bit gyroscope measurement has a full scale defined in *FS_SEL* (Register 27). For each full scale setting, the gyroscopes' sensitivity per LSB in *GYRO_xOUT* is shown in the table below:

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 8-3. mpu_6050.ino

```
// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

// 定义函数，从寄存器reg中读取size字节的数据存入结构体中
void read_i2c(unsigned char reg, uint8_t *buffer, int size) {
	Wire.beginTransmission(i2c_address);	// 向设备发送I2C命令（MPU6050传感器的地址位于0x69)
	Wire.write(reg);	// 指定要读取的寄存器地址
	Wire.endTransmission(false);	// 保持连接处于打开状态，这样才能从下一行代码中读取数据

	// 向传感器请求size字节的数据，参数true表示读取结束后关闭连接并释放I2C总线资源
	Wire.requestFrom(i2c_address, size, true);
	int i = 0;	// 定义while循环迭代次数
	while(Wire.available() && (i < size)) {	// 当有字节可读，且还未读完所有的字节请求时，进入循环
		buffer[i] = Wire.read();	// 读取一个字节，并存入buffer中
		i++;
	}
	if(i != size) {	//如果可供读取的字节不足，说明有错误
		Serial.println("Error reading from i2c");
	}
}
```

Example 8-3. mpu_6050.ino

```
// mpu_6050.ino - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void write_i2c(unsigned char reg, const uint8_t data) {      // 向传感器的寄存器reg写入一个字节data
    Wire.beginTransmission(i2c_address);                     // 传感器的地址就是全局变量i2c_address的值
    Wire.write(reg);
    Wire.write(data);
    Wire.endTransmission(true);
}
```

MPU 6050 Code and Connection for Raspberry Pi

- Figure 8-6 shows the wiring diagram for Raspberry Pi.
- Hook it up as shown, and then run the code from Example 8-4.

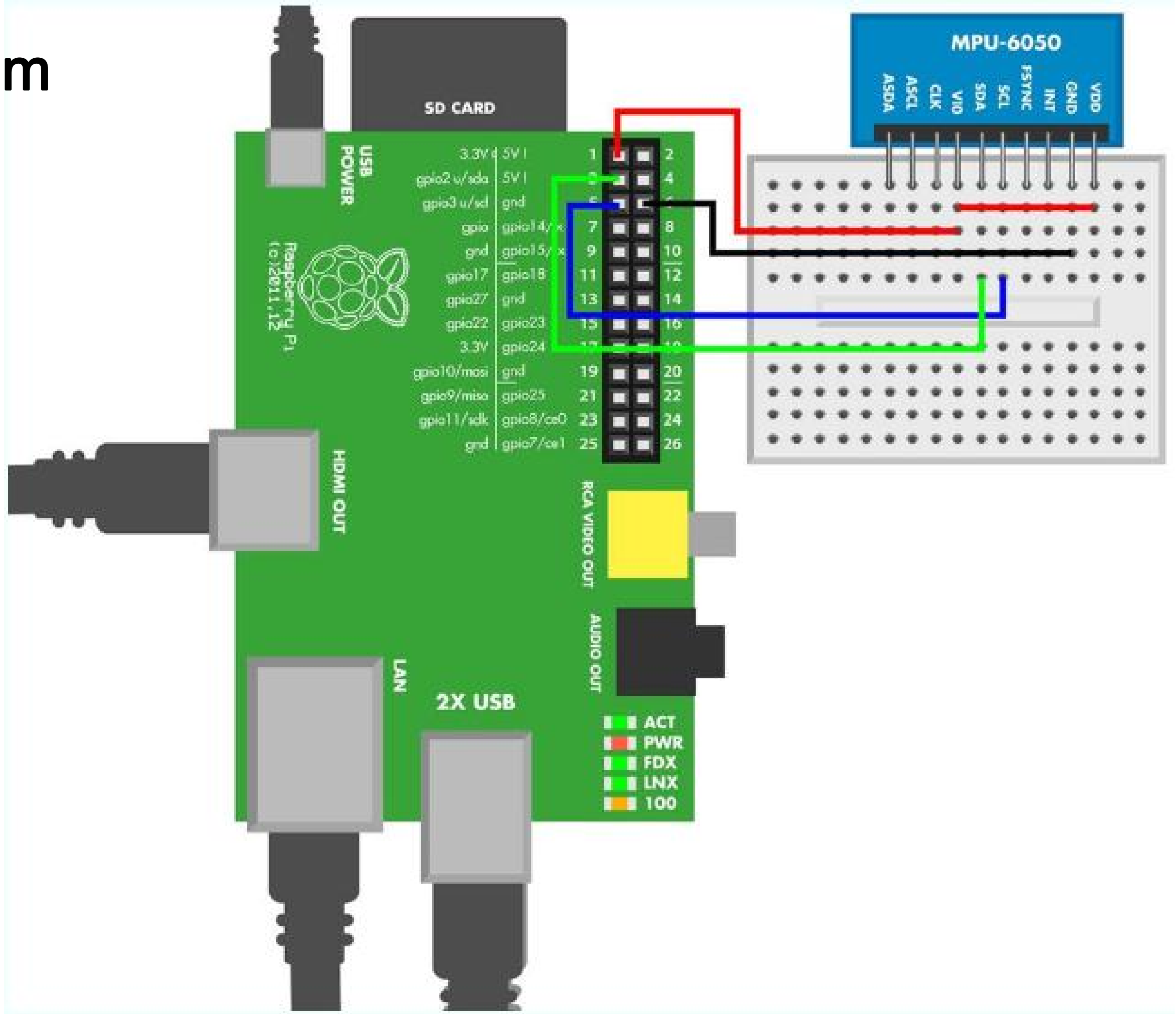


Figure 8-6. MPU 6050 six-axis accelerometer circuit for Raspberry Pi

Example 8-4. mpu_6050.py

```
# mpu_6050.py - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import smbus                    # SMBus库实现了I2C通信协议，也使得树莓派程序比Arduino简单很多
import struct

i2c_address =  0x68             #MPU 6050传感器的I2C地址
sleep_mgmt = 0x6B               #命令的寄存器地址
accel_x_out = 0x3B              #以x轴加速度的寄存器为起点，从0x3B~0x48是加速度、温度、角速度寄存器地址

bus = None                      # 变量BUS是给所有函数使用的全局变量
acc_x = 0
acc_y = 0
acc_z = 0
temp = 0
gyro_x = 0
gyro_y = 0
gyro_z = 0
```

Example 8-4. mpu_6050.py

```
# mpu_6050.py - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def initmpu():
        global bus                  #为了在函数中修改全局变量的值，必须在函数一开始指明它是全局的
        bus = smbus.SMBus(1)  #初始化SMBus(I2C)，将新建的SMBus类的对象存储到变量bus中
        bus.write_byte_data(i2c_address, sleep_mgmt, 0x00)      # 将传感器从睡眠模式中唤醒
```

Example 8-4. mpu_6050.py

```
# mpu_6050.py - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def get_data():
        global acc_x,acc_y,acc_z,temp,gyro_x,gyro_y,gyro_z
        bus.write_byte(i2c_address, accel_x_out)          # 从x轴加速度地址开始读数据
        rawData = ""
        for i in range(14):   # 循环14次
            rawData += chr(bus.read_byte_data(i2c_address,accel_x_out+i))   #读字节并转换成ASCII码
        data = struct.unpack('>hhhhhhh', rawData)     # 将字符串rawData转换成python元组
        #  '>hhhhhhh' 是格式化字符串， >小端模式， h表示有符号的两字节短整型

        acc_x = data[0] / 16384.0       # 把传感器原始数据转换为实际生活中的重力加速度g的倍数
        acc_y = data[1] / 16384.0
        acc_z = data[2] / 16384.0
```

Example 8-4. mpu_6050.py

```
# mpu_6050.py - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

zero_point = -512 - (340 * 35)     #把原始传感器温度值转换为摄氏度，首先计算0℃对应值
temp = (data[3] - zero_point) / 340.0     #

gyro_x = data[4] / 131.0     # 把原始传感器角速度值转换为实际生活中以“角度/秒”为单位的角速度
gyro_y = data[5] / 131.0
gyro_z = data[6] / 131.0
```

Example 8-4. mpu_6050.py

```
# mpu_6050.py - print acceleration (m/s**2) and angular velocity (gyro, deg/s)
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def main():
	initmpu()
	while True:   #
		get_data() #调用之前的函数，更新全局变量，所以不需要返回值
		print("DATA:")
		print("Acc (%.3f,%.3f,%.3f) g, " % (acc_x, acc_y, acc_z))     #使用格式化输出加速度
		print("temp %.1f C, " % temp)     #使用格式化输出温度
		print("gyro (%.3f,%.3f,%.3f) deg/s" % (gyro_x, gyro_y, gyro_z)) #使用格式化输出角速度
		time.sleep(0.5) # s #

if __name__ == "__main__":
	main()
```

Contents

Experiment: Hacking Wii Nunchuk (with I2C)

- Would you like to have an accelerometer, a joystick, and a button—all in a cheap package?
- Look no further, because the Nunchuk controller for the Wii gaming console is all that.
- If you want to go even cheaper, there are cheap compatible copies available, too.

Experiment: Hacking Wii Nunchuk (with I2C)

- The Wii Nunchuk also teaches an important hacking lesson: engineers are human.
- And just like the rest of us humans, engineers want to work with tried and true protocols, where libraries and tools are available.
- The Wii has its own proprietary connector, and originally, very little documentation. But under the surface, it's the standard I2C protocol.
- As you saw earlier, I2C is our favorite industry standard short-range communication protocol.

Experiment: Hacking Wii Nunchuk (with I2C)

- Nintendo produces the Wii Nunchuk in large batches, which keeps quality up and prices down. The wide availability of Nunchuk also means there is a lot of example code and documentation available. You don’t even need to cut the cord, as there is even a WiiChuck adapter you can push into Nunchuk’s connector to connect it to Arduino or a breadboard ([Figure 8-7](#)).

Figure 8-7. Nunchuk connected to Arduino with WiiChuck adapter

Experiment: Hacking Wii Nunchuk (with I2C)

- Nintendo produces the Wii Nunchuk in large batches, which keeps quality up and prices down. The wide availability of Nunchuk also means there is a lot of example code and documentation available.
- You don’t even need to cut the cord, as there is even a WiiChuck adapter you can push into Nunchuk’s connector to connect it to Arduino or a breadboard (Figure 8-7).

Figure 8-7. Nunchuk connected to Arduino with WiiChuck adapter

Nunchuk Code and Connection for Arduino

- Figure 8-8 shows the connection diagram for Arduino. Wire things up as shown, and run the sketch shown in Example 8-6.

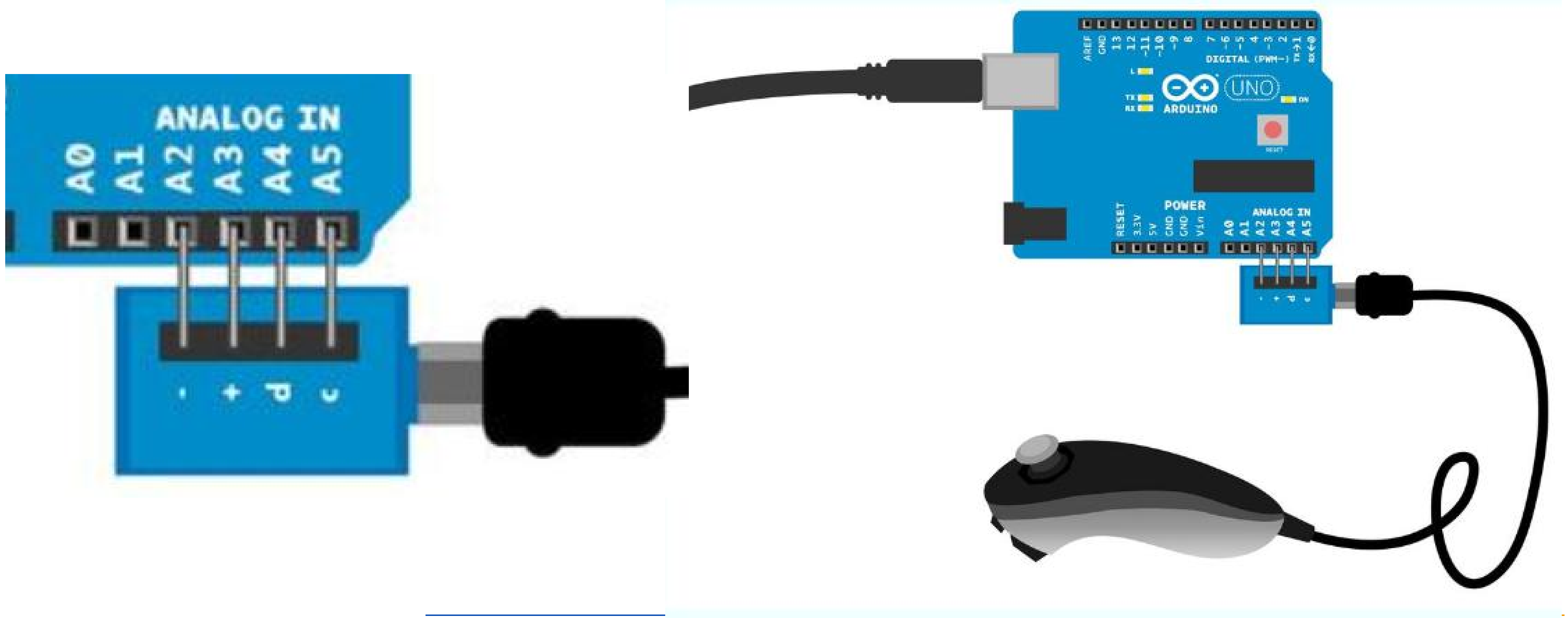
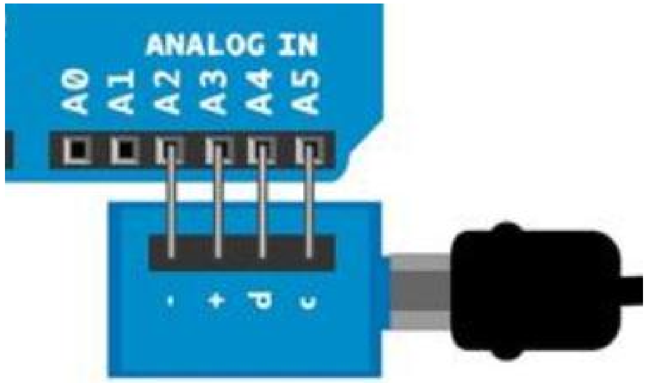


Figure 8-8. WiiChuck circuit for Arduino

Example 8-6. wiichuck_adapter.ino




```
// wiichuck_adapter.ino - print joystick, accelerometer and button data to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>

const char i2c_address = 0x52;
unsigned long lastGet=0; // ms
int jx = 0, jy = 0, accX = 0, accY = 0, accZ = 0, buttonZ = 0, buttonC = 0;
// 定义全局变量，因Arduino的函数无法返回多值，所以使用全局变量在函数之间传递数据

void setup() {
		Serial.begin(115200);
		Wire.begin();
		pinMode(A2, OUTPUT);
		pinMode(A3, OUTPUT);
		digitalWrite(A2, LOW);		// 接GND，目的是为了让WiiChunk无需其他接地跳线
		digitalWrite(A3, HIGH);		// 接VCC，理由同上
		delay(100);
		initNunchuck();			//初始化WiiChunk，函数通过I2C发送0x40和0x00
}
```

Example 8-6. wiichuck_adapter.ino

```
// wiichuck_adapter.ino - print joystick, accelerometer and button data to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
    if(millis() - lastGet > 100) {  // 每100ms获取一次数据，这是常见的每xms执行一段代码的模式
        get_data();                 // 更新全局变量，所以无返回值
        lastGet = millis();         // 更新最后一次调用get_data的时间
    }
    Serial.print("Button Z: ");
    Serial.print(buttonZ);          //数字0表示按钮按下，1表示松开
    Serial.print(" Button C: ");
    Serial.print(buttonC);
    Serial.print(" Joystick: (x,y) (");
    Serial.print(jx);               //显示摇杆坐标（jx，jy），原始数据在30~220之间
    Serial.print(",");
    Serial.print(jy);
//未完待续……
```

Example 8-6. wiichuck_adapter.ino

```
// wiichuck_adapter.ino - print joystick, accelerometer and button data to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

//接续......
    Serial.print(“) Acceleration (x,y,z) (”);
    Serial.print(accX);            //显示加速度传感器数值（accX，accY，accZ），原始数据在80~190之间
    Serial.print(",");
    Serial.print(accY);
    Serial.print(",");
    Serial.print(accZ);
    Serial.println(“)”);
    delay(10);                     // ms
}
```

Example 8-6. wiichuck_adapter.ino

// wiichuck_adapter.ino - print joystick, accelerometer and button data to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
void get_data() {
	int buffer[6];                         // 定义一个新的含有六个整数的数组
	Wire.requestFrom(i2c_address, 6);      //向Nunchuk请求6个字节的数据，数据含义见下页ppt
	int i = 0;                             //循环变量 i 指明当前正在处理的字节数
	while(Wire.available()) {              //如果存在可读取数据，则进入循环
		buffer[i] = Wire.read();           // 读取一个字节数据
		buffer[i] ^= 0x17;                 // 对读取的数据进行解码（对0x17做XOR操作）
		buffer[i] += 0x17;                 // 将当前值递增0x17
		i++;
	}
```

//未完待续……

Nunchuk Code and Connection for Arduino

Table 8-5. Nunchuk的六个字节的数据块

字节	含义
0	摇杆X
1	摇杆Y
2	加速度传感器X
3	加速度传感器Y
4	加速度传感器Z
5	ZCxxyyzz：按钮“Z” “C”，加速度传感器的精度

Example 8-6. wiichuck_adapter.ino

// wiichuck_adapter.ino - print joystick, accelerometer and button data to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

//接续……

```
    if(i != 6) {          // 如果读取的字节数不等于六个字节，输出警告
        Serial.println("Error reading from i2c");
    }
    write_i2c_zero();                       //通过12C发送0x00
    buttonZ = buffer[5] & 0x01;             //最低位是按钮Z的状态
    buttonC = (buffer[5] >> 1) & 0x01;      //倒数第二位是按钮C的状态
    jx = buffer[0];                         //变量jx代表摇杆的x轴，它的值是一个完整的字节
    jy = buffer[1];
    accX = buffer[2];
    accY = buffer[3];
    accZ = buffer[4];
}
```

Example 8-6. wiichuck_adapter.ino

```
// wiichuck_adapter.ino - print joystick, accelerometer and button data to serial
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void write_i2c_zero() {
	Wire.beginTransmission(i2c_address);
	Wire.write(0x00);
	Wire.endTransmission();
}

void initNunchuck(){
	Wire.beginTransmission(i2c_address);
	Wire.write(0x40);
	Wire.write(0x00);
	Wire.endTransmission();
}
```

Nunchuk Code and Connection for Raspberry Pi

- Figure 8-9 shows the connectior
- Hook it up, and run the program

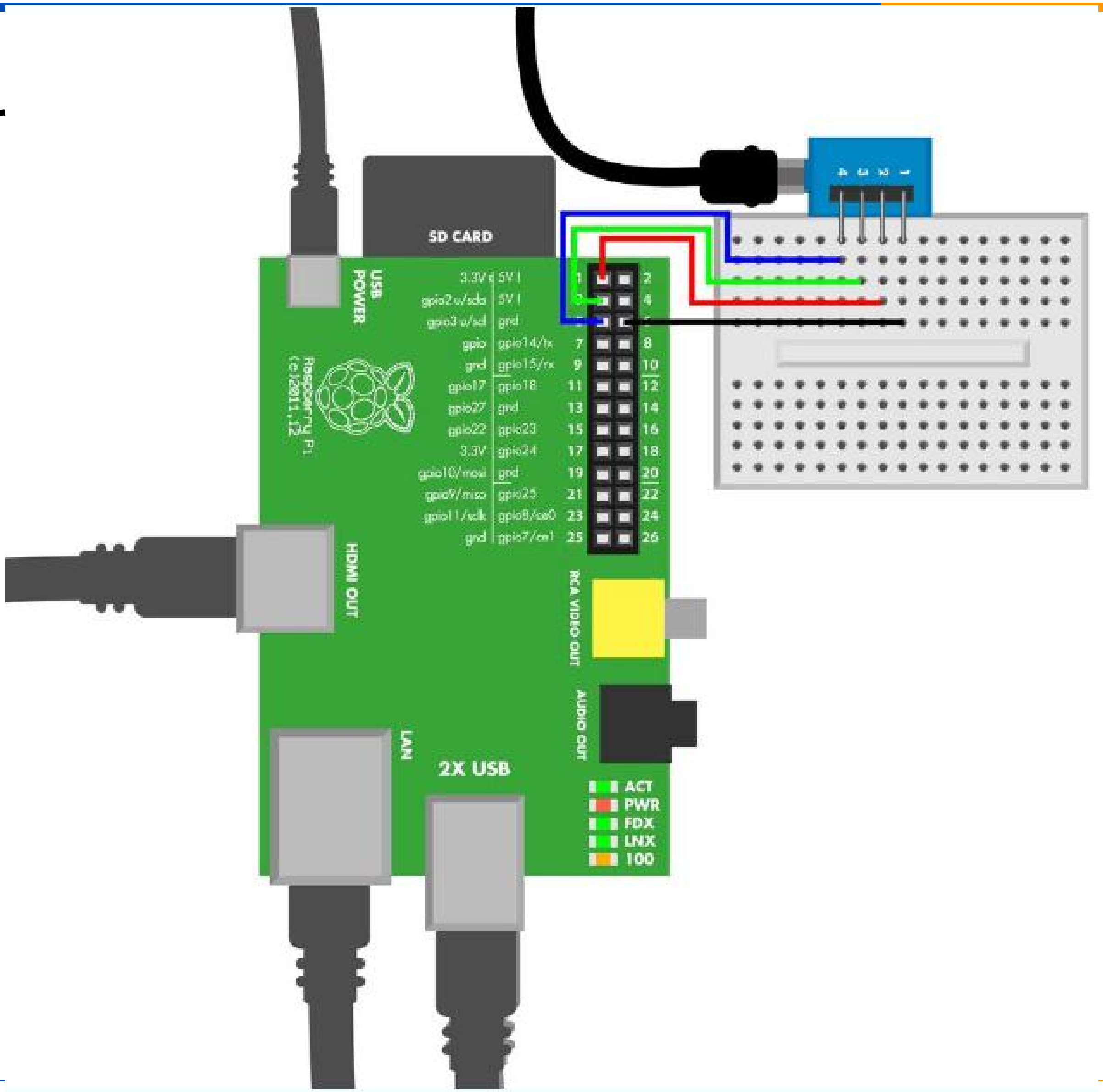


Figure 8-9. WiiChuck circuit for Raspberry Pi

Example 8-7. wiichuck_adapter.py

```
# wiichuck_adapter.py - print Wii Nunchuck acceleration and joystick
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import smbus            # sudo apt-get -y install python-smbus    # 先安装python-smbus库

bus = None
address = 0x52          #Wii Nunchuk的地址
z = 0                   #Z按钮状态的全局变量
c = 0                   #C按钮状态的全局变量
joystick_x = 0          #摇杆x状态的全局变量
joystick_y = 0
ax_x = 0                #加速度x的全局变量
ax_y = 0
ax_z = 0
```

Example 8-7. wiichuck_adapter.py

```
# wiichuck_adapter.py - print Wii Nunchuck acceleration and joystick
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def initNunchuck():
	global bus
	bus = smbus.SMBus(1)		#创建一个SMBus类的对象，存储到变量bus中，参数1是设备号

	#使用I2C命令初始化Wii，设备Wii地址address是0x52，命令是0x40，命令的参数是0x00
	bus.write_byte_data(address, 0x40, 0x00)

def send_request():
	bus.write_byte(address, 0x00)	#使用空字符0x00向Wii询问下一组数据
```

Example 8-7. wiichuck_adapter.py

```
# wiichuck_adapter.py - print Wii Nunchuck acceleration and joystick
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def get_data():
        global bus, z, c, joystick_x, joystick_y, ax_x, ax_y, ax_z
        data = [0]*6
        for i in range(len(data)):          # 依次读取整组数据的六个字节
                data[i] = bus.read_byte(address)
                data[i] ^= 0x17             #解码数据值
                data[i] += 0x17

        z = data[5] & 0x01                  # 获取按钮“Z”的状态，1代表按下，0代表松开
        c = (data[5] >> 1) & 0x01           # 获取按钮“C”的状态，1代表按下，0代表松开

        joystick_x = data[0]
        joystick_y = data[1]
        ax_x = data[2]
        ax_y = data[3]
        ax_z = data[4]
        send_request()
```

Example 8-7. wiichuck_adapter.py

```
# wiichuck_adapter.py - print Wii Nunchuck acceleration and joystick
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def main():
	initNunchuck()
	while True:
		get_data()
		print("Button Z: %d Button C: %d joy (x,y) (%d,%d) \
			acceleration (x,y,z) (%d,%d,%d)" \
			% (z, c, joystick_x, joystick_y, ax_x, ax_y, ax_z))
		time.sleep(0.1)

if __name__ == "__main__":
	main()
```

Contents

Test Project: Robot Hand Controlled by Wii Nunchuk

- Control a robot hand with the Nunchuk.
- As you can already read acceleration and joystick position, you can simply turn servos according to these numbers.
- Add mechanics, and you've got a Nunchuk-controlled robot hand.

Figure 8-10. Nunchuk-controlled robot hand

Test Project: Robot Hand Controlled by Wii Nunchuk

- In the Robot Hand project, you'll learn how to:
  - Use the accelerometer and a mechanical joystick with outputs.
  - Combine servos for complex movement.

- You'll also
  - refresh your skills on servo control
  - filtering noise with running averages (see Servo Motors and Moving Average).

Test Project: Robot Hand Controlled by Wii Nunchuk

- Start with just the servos (Figure 8-11).
- Once you get some movement, you can continue with hand mechanics.

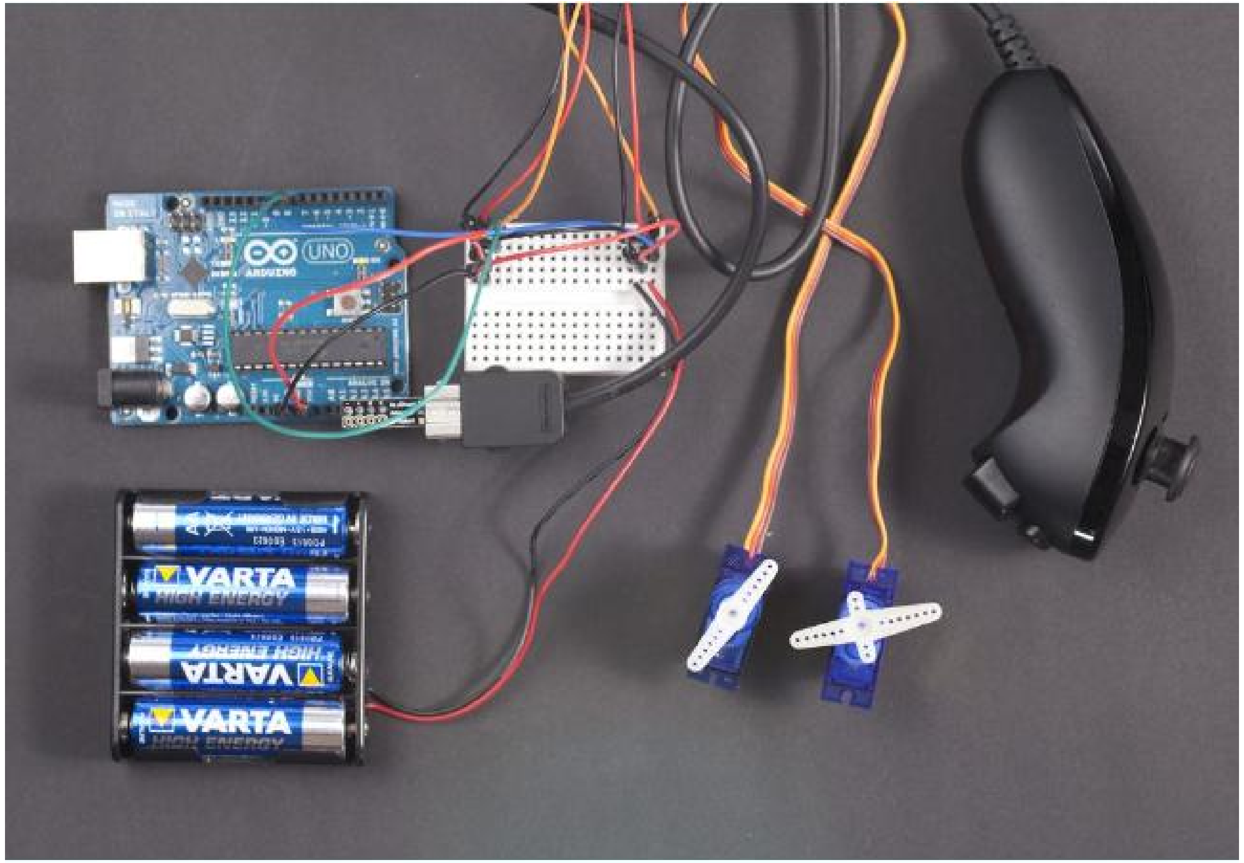


Figure 8-11. Nunchuk controls two servos with Arduino

Test Project: Robot Hand Controlled by Wii Nunchuk

- Figure 8-12 shows the wiring diagram. Hook everything up as shown, and then run the code shown in Example 8-8.

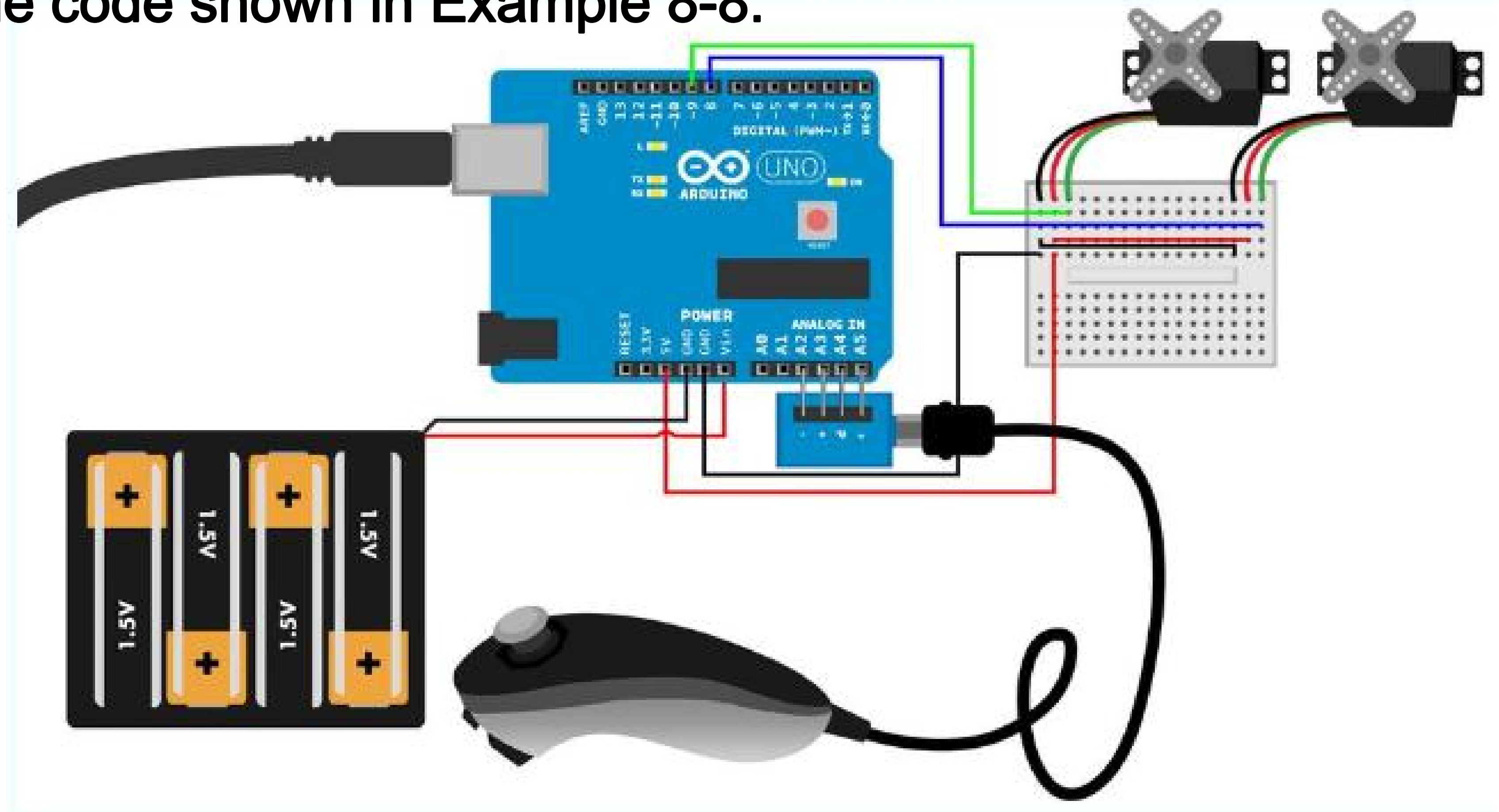


Figure 8-12. Claw circuit for Arduino

Example 8-8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

#include <Wire.h>

const int clawPin = 8;          // 一个舵机模仿手，一个舵机模仿臂
const int armPin = 9;
int armPos=0, clawPos=0;
float wiiP = 0.0;               // 保存的是Wii的倾斜程度的百分比：向后是0.0，向前是1.0
float wiiPAvg = 0.0;            // WiiP的移动平均值
int lastarmPos = 350;

const char i2c_address = 0x52;
int jx = 0, jy = 0, accX = 0, accY = 0, accZ = 0, buttonZ = 0, buttonC = 0;
```

Example 8-8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void setup() {
        Serial.begin(115200);
        Wire.begin();               // Wii Nunchuck作为I2C设备，进行初始化
        pinMode(A2, OUTPUT);
        pinMode(A3, OUTPUT);
        digitalWrite(A2, LOW);
        digitalWrite(A3, HIGH);
        delay(100);
        initNunchuck();

        pinMode(clawPin, OUTPUT);  // Servos   模仿手臂的舵机控制引脚初始化
        pinMode(armPin, OUTPUT);
}
```

Example 8-8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void loop() {
        get_data();                                  //

        // 根据从Wi中读取的原始数据计算百分数。这个公式与 Arduino内置的map（）函数异曲同工
        wiiP = (accZ-70.0)/(178.0-70.0);
        if (accY>120 && accZ>100) wiiP=1;
        if (accY>120 && accZ<100) wiiP=0;
        if (wiiP>1) wiiP=1;
        if (wiiP<0) wiiP=0;

        // 为了过滤随机尖峰，对加速度传感器的z轴应用移动平均法。
        wiiPAvg = runningAvg(wiiP, wiiPAvg);
//未完待续......
```

Example 8-8. wiichuck_adapter_claw.ino

// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

接续……

```
    armPos = map(wiiPAvg*10*1000, 0, 10*1000, 2200, 350);
    //将超杆的原始数据（30 - 220）映射到舵机脉冲（1500 - 2400）中
    //例如，摇杆的30映射到1500ps的舵机脉冲长度
    clawPos = map(jy, 30, 220, 1600, 2250);

    //向舵机发送脉冲控制机械手臂。为了让舵机保持在该位置，你必须向舵机持续发送脉冲
    pulseServo(armPin, armPos);
    pulseServo(clawPin, clawPos);
    printDebug();
}
```

Example 8-8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

float runningAvg(float current, float old) {
	float newWeight=0.3;
	//为了从多个数据中获取移动平均值，使用加权平均值
	//这样仅需要存储上一次的结果，同时之前的结果依然会影响平均值。
	return newWeight*current + (1-newWeight)*old;
}

//servo 向舵机发送一个脉冲。为了控制的可靠性，该函数每秒需要调用约50次。
void pulseServo(int servoPin, int pulseLenUs) {
	digitalWrite(servoPin, HIGH);
	delayMicroseconds(pulseLenUs);
	digitalWrite(servoPin, LOW);
	delay(15);
}
```

Example 8-8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void get_data() {
        int buffer[6];
        Wire.requestFrom(i2c_address,6);
        int i = 0;
        while(Wire.available()) {
                buffer[i] = Wire.read();
                buffer[i] ^= 0x17;
                buffer[i] += 0x17;
                i++;
        }

//未完待续……
```

Example 8-8. wiichuck_adapter_claw.ino

// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

接续……

```
        if(i != 6) {
                Serial.println("Error reading from i2c");
        }
        write_i2c_zero();
        buttonZ = buffer[5] & 0x01;
        buttonC = (buffer[5] >> 1) & 0x01;
        jx = buffer[0];
        jy = buffer[1];
        accX = buffer[2];
        accY = buffer[3];
        accZ = buffer[4];
}
```

Example 8-8. wiichuck_adapter_claw.ino

```
// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

void write_i2c_zero() {
	Wire.beginTransmission(i2c_address);
	Wire.write((byte)0x00);
	Wire.endTransmission();
}

void initNunchuck(){
	Wire.beginTransmission(i2c_address);
	Wire.write((byte)0x40);
	Wire.write((byte)0x00);
	Wire.endTransmission();
}
```

Example 8-8. wiichuck_adapter_claw.ino

// wiichuck_adapter_claw.ino - control robot hand with Nunchuck
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
// debug
void printDebug(){
        Serial.print("accZ:");
        Serial.print(accZ);
        Serial.print("      wiiP:");
        Serial.print(wiiP);
        Serial.print("      wiiPAvg:");
        Serial.print(wiiPAvg);  Serial.print("      jy:");
        Serial.print(jy);
        Serial.print("      clawPos:");
        Serial.println(clawPos);
}
```

Adding Hand Mechanics

- You already know how to control servo motors with Wii Nunchuk, and it's easy to adapt this to commercial robot arms and hands.
- There are plenty of choices available, and if you have strong mechanical skills you can even build your own.

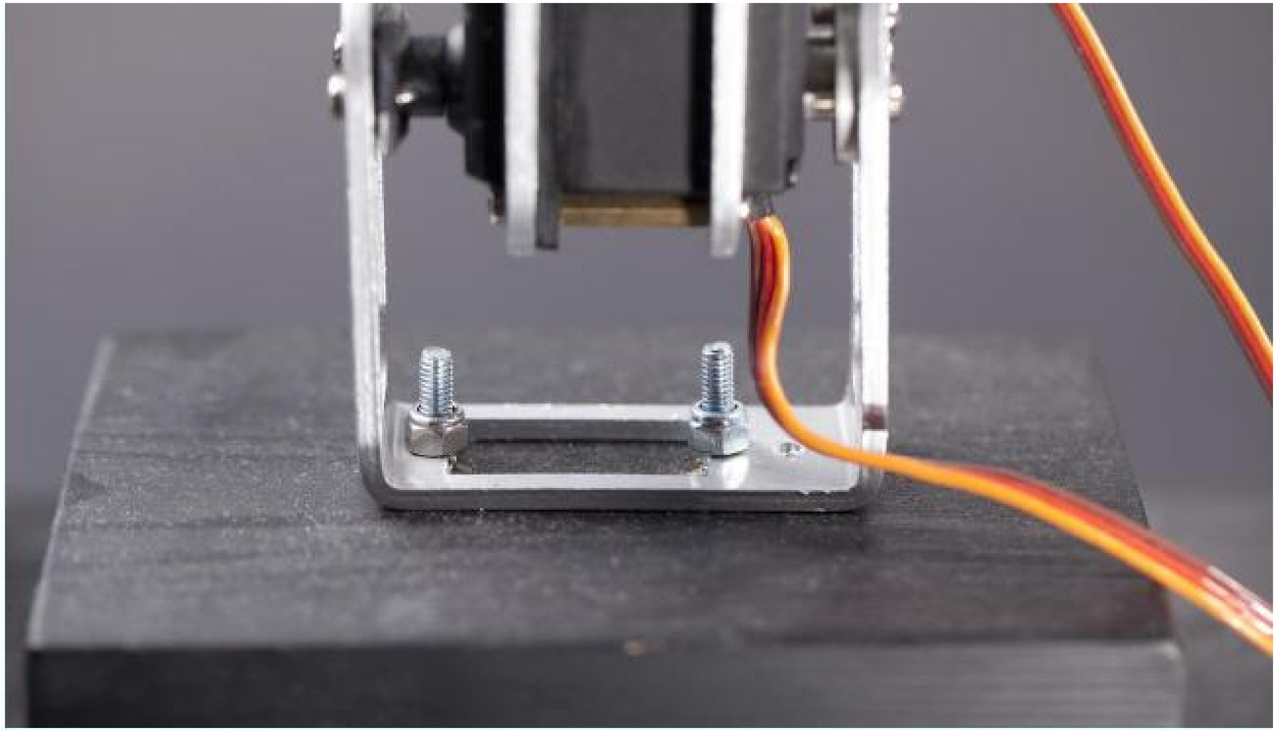


Figure 8-13. Robot arm screwed to base

Adding Hand Mechanics

- Whichever arm or hand you choose, it's a good idea to mount it firmly on a solid base to keep it upright and prevent it from tipping over. We attached our arm to a thick wood base with screws (see Figure 8-13).

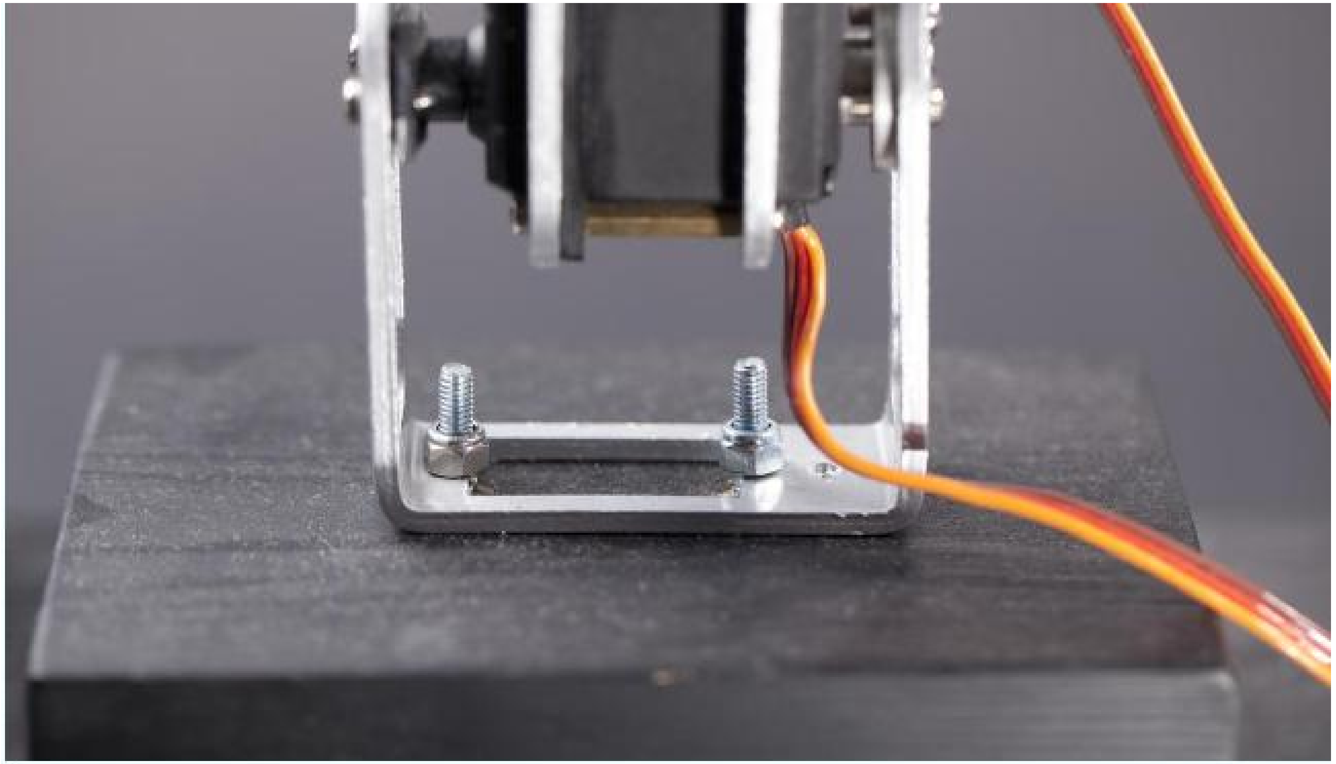


Figure 8-13. Robot arm screwed to base

Adding Hand Mechanics

- All the code is finished already (see Example 8-8). Connect the servos so that you control the gripper with the Nunchuk thumbstick, and arm movement with the accelerometer. See Figure 8-14 for the final result.

Figure 8-14. Robot arm controlled by Wii