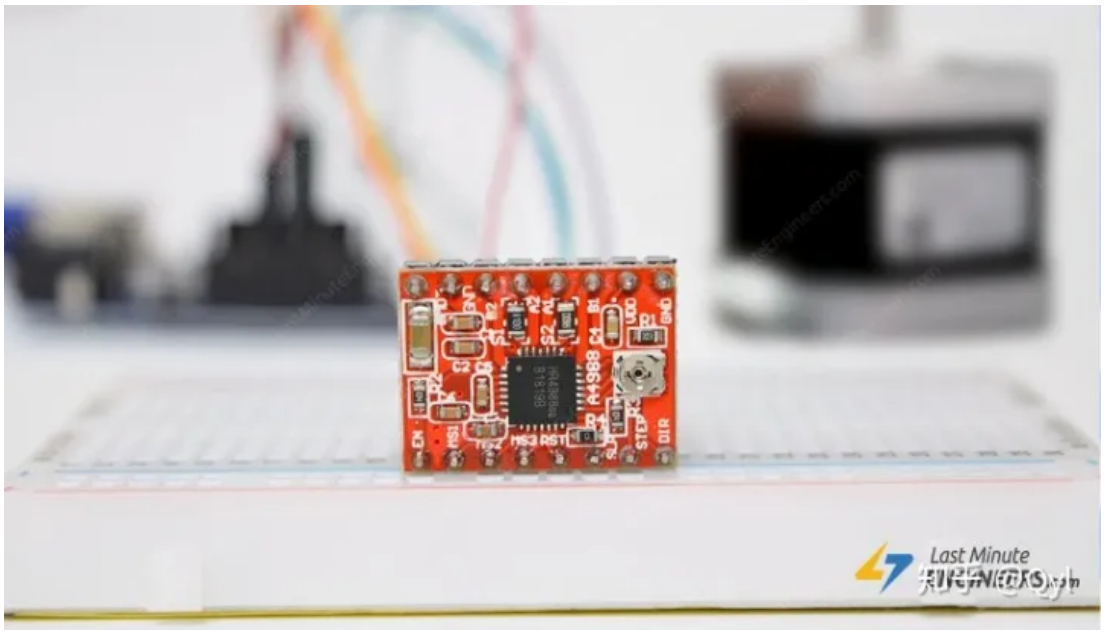
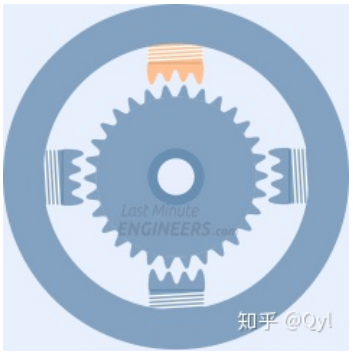
使用A4988驱动程序模块和Arduino控制步进电机



如果您打算建造自己的3D打印机或CNC机器，则需要控制一堆步进电机。而且，由一个人控制所有这些人可能会占用大量的处理过程，并且不会为您做任何其他事情留出很多空间。除非您使用独立的专用步进电机驱动器– **A4988**。

它只需两个引脚即可控制NEMA 17等双极步进电机的速度和旋转方向。多么酷啊！

您知道步进电机如何工作吗?
步进电机使用带齿的轮和电磁体来使轮一次“一步”旋转。
发送的每个HIGH脉冲都会使线圈通电，吸引齿轮的最近齿，并一步步驱动电动机。

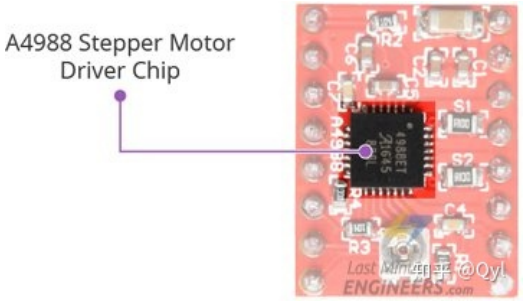


脉冲这些线圈的方式会极大地影响电动机的性能。

- 脉冲序列确定电动机的旋转方向。
- 脉冲的频率决定了电动机的速度。
- 脉冲数确定电动机将旋转多远。

A4988步进电机驱动器芯片

该模块的核心是Allegro的Microstepping驱动器– A4988。它的体积很小（只有0.8英寸×0.6英寸），但仍然有力气。



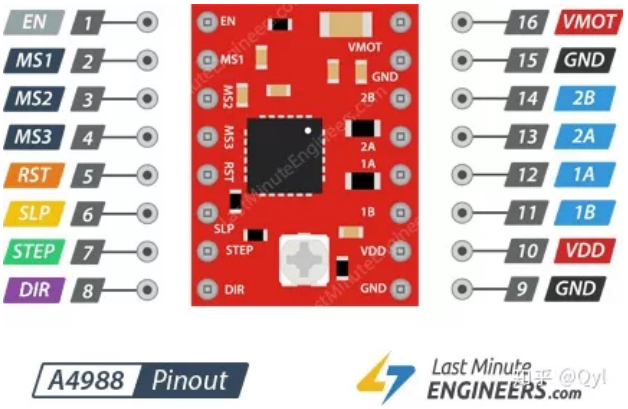
A4988步进电机驱动器具有高达35 V的输出驱动能力和±2A的输出能力，使您可以像NEMA 17一样，以每个线圈高达2A的输出电流控制一个双极步进电机。

该驱动程序具有内置转换器，易于操作。这样可以将控制针的数量减少到仅2个，一个用于控制步进，另一个用于控制旋转方向。

驱动程序提供5种不同的步进分辨率。全步，半步，四分之一步，八步和十六步。

A4988电机驱动器引脚排列

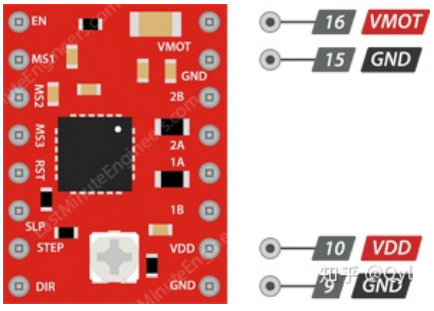
A4988驱动器共有16个引脚与外界连接。连接如下：



让我们一一熟悉一下所有的引脚。

电源连接引脚

A4988实际上需要两个电源连接。



VDD 和 GND 用于驱动3V至5.5V的内部逻辑电路。

VMOT&GND 为电动机提供电源，该电源可以为8V至35V。

根据数据表，电动机电源需要在靠近电路板的地方使用适当的去耦电容器，以维持4A的电流。

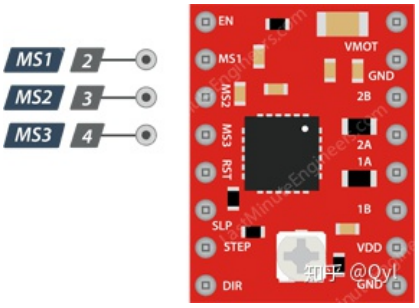
> **警告： 该驱动器板上装有低ESR陶瓷电容器，因此容易受到电压尖峰的影响。在某些情况下，这些峰值可能会超过35V（A4988的最大额定电压），可能会永久损坏电路板甚至电动机。 保护驱动器免受此类尖峰影响的一种方法是在电动机电源引脚之间放置一个较大的100µF**

> **（至少47μF）电解电容器。**

微步选择针

A4988驱动器通过允许中间步进位置来允许微步进。这是通过以中等电流水平向线圈通电来实现的。

例如，如果选择以四分之一步模式驱动每转1.8°或200步的NEMA 17，则电动机每转将给出800微步。



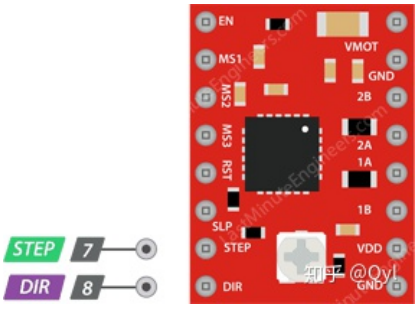
A4988驱动器具有三个步长（分辨率）选择器输入，即。 MS1，MS2和MS3。通过为这些引脚设置适当的逻辑电平，我们可以将电动机设置为五步分辨率之一。

MS1	MS2	MS3	微步分辨率
低	低	低	全步
高	低	低	半步
低	高	低	四分之一步
高	高	低	第八步
高	高	高	第十六步

这三个微步选择引脚被内部下拉电阻拉至低电平，因此，如果我们将其断开，电动机将以全步模式运行。

控制输入引脚

A4988有两个控制输入，即。STEP和DIR。



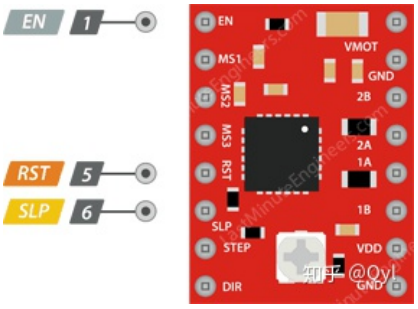
*STEP*输入控制电机的微步。发送到此引脚的每个HIGH脉冲都会通过微步选择引脚设置的微步数来步进电动机。脉冲越快，电动机旋转得越快。

*DIR*输入控制电机的旋转方向。将其拉高将驱动电动机顺时针旋转，将其拉低将驱动电动机逆时针旋转。

> **内部没有将STEP和DIR引脚拉至任何特定电压，因此您不应在应用中将其悬空。**

控制电源状态的引脚

A4988具有三个不同的输入，用于控制其电源状态。EN，RST和SLP。



EN引脚为低电平有效输入，当拉低至低电平（逻辑0）时，A4988驱动器使能。默认情况下，该引脚被拉低，因此驱动器始终处于使能状态，除非您将其拉高。

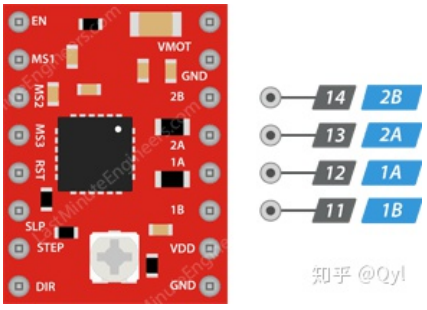
SLP引脚为低电平有效输入。这意味着将该引脚拉至低电平可将驱动器置于睡眠模式，从而将功耗降至最低。特别是在不使用电动机以节省功率时，可以调用此方法。

RST也是低电平有效输入。将其拉低时，将忽略所有STEP输入，直到将其拉高。它还通过将内部转换器设置为预定义的Home状态来重置驱动程序。原始状态基本上是电动机启动的初始位置，并且根据微步分辨率而不同。

- **提示**
  RST引脚悬空。如果不使用该引脚，则可以将其连接到相邻的SLP / SLEEP引脚以将其拉高并启用驱动器。

输出引脚

A4988电机驱动器的输出通道通过以下方式分解到模块的边缘： 1B，1A，2A和2B 针脚。



您可以将电压在8V至35V之间的任何双极步进电机连接到这些引脚。

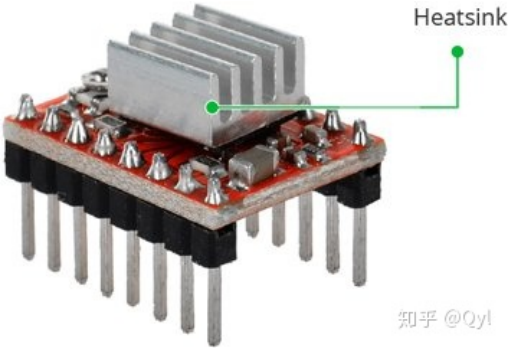
模块上的每个输出引脚都可以为电机提供高达2A的电流。但是，提供给电动机的电流量取决于系统的电源，冷却系统和电流限制设置。

冷却系统–散热器

A4988驱动器IC的过多功耗会导致温度上升，该温度上升可能超出IC的容量，可能会损坏自身。

即使A4988驱动器IC每个线圈的最大额定电流为2A，该芯片也只能为每个线圈提供大约1A的电流而不会过热。

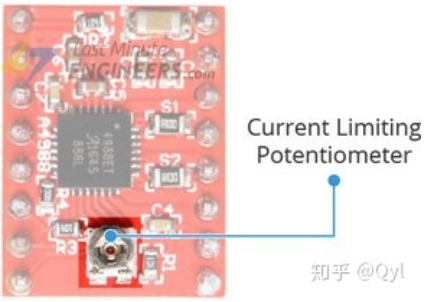
为了使每个线圈达到1A以上，需要使用散热器或其他冷却方法。



A4988驱动器通常带有散热器。建议您在使用驱动程序之前先安装它。

限流

在使用电动机之前，需要进行一些小的调整。我们需要限制流经步进线圈的最大电流，并防止其超过电机的额定电流。



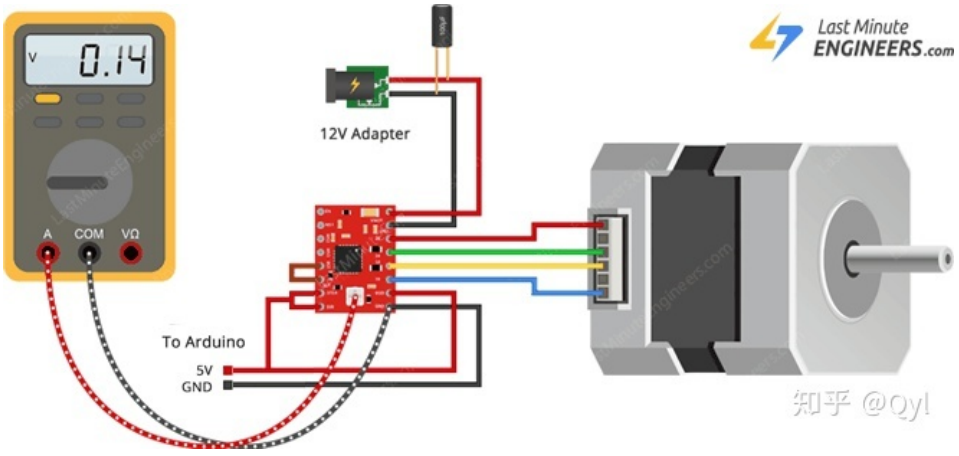
A4988驱动器上有一个微调电位器，可用于设置电流限制。您应将电流限制设置为等于或小于电动机的额定电流。

要进行此调整，有两种方法：

方法1：

在这种方法中，我们将通过测量“ ref”引脚上的电压（Vref）来设置电流限制。

1. 查看您的步进电机的数据表。记下它的额定电流。在我们的情况下，我们使用NEMA 17 200steps / rev，12V 350mA。
2. 断开三个微步选择引脚，使驱动器进入全步模式。
3. 不给STEP输入计时，将电动机固定在固定位置。
4. 调整时，请测量金属微调电位器本身的电压（Vref）。
5. 使用公式调整Vref电压电流限制= Vref x 2.5
   例如，如果电动机的额定电流为350mA，则可以将参考电压调整为0.14V。

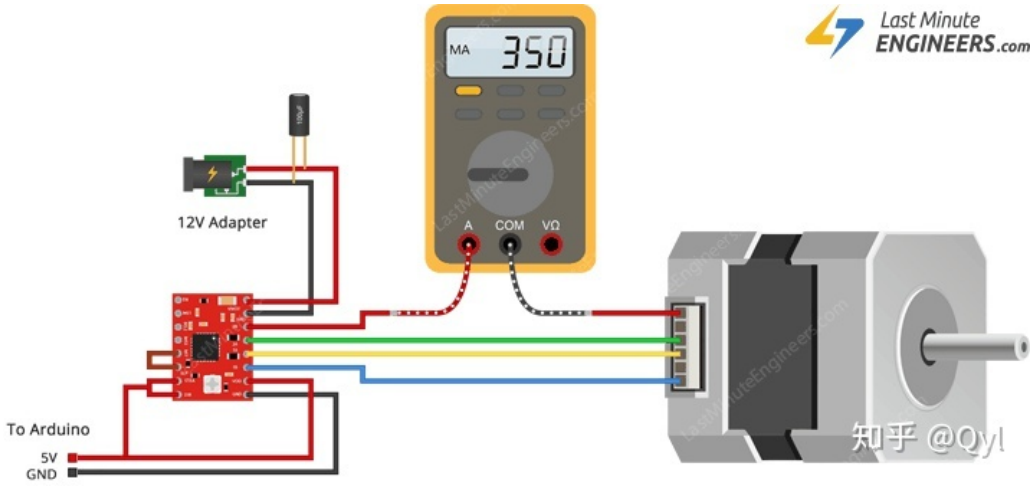


提示：
一种简单的调整方法是在金属螺丝刀的轴上使用一个鳄鱼夹，然后将其固定在万用表上，以便您可以同时使用螺丝刀测量和调整电压。

方法2：

在这种方法中，我们将通过测量流经线圈的电流来设置电流极限。

1. 查看您的步进电机的数据表。记下它的额定电流。在我们的情况下，我们使用NEMA 17 200steps / rev，12V 350mA。
2. 断开三个微步选择引脚，使驱动器进入全步模式。
3. 不给STEP输入计时，将电动机固定在固定位置。不要将STEP输入悬空，将其连接到逻辑电源（5V）
4. 将电流表与步进电机上的线圈之一串联，然后测量实际电流。
5. 拿起一把小螺丝刀，调节电流限制电位器，直到达到额定电流。



如果更改逻辑电压（VDD），则需要再次执行此调整

用Arduino UNO接线A4988步进电机驱动器

现在我们了解了有关驱动程序的所有知识，我们将其连接到我们的Arduino。

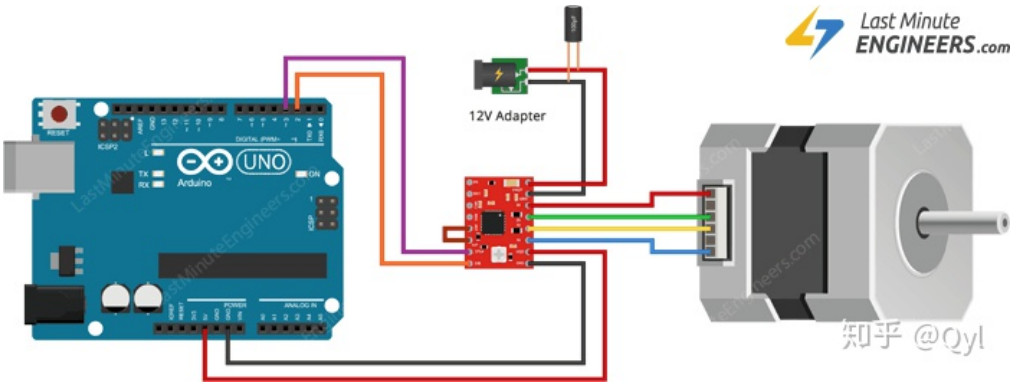
连接非常简单。首先将VDD和GND（VDD旁边）连接到Arduino的5V和接地引脚。DIR和STEP输入引脚分别连接到Arduino上的＃2和＃3数字输出引脚。

将步进电机连接到2B，2A，1A和1B引脚。实际上，A4988的布局很方便，可以与几台双极电机上的4针连接器匹配，因此这不是问题。

警告：
驱动器通电时连接或断开步进电机会损坏驱动器。

接下来，将RST引脚连接到相邻的SLP / SLEEP引脚以保持驱动器使能。还要保持微步选择引脚断开，以使电机以全步模式运行。

最后，将电动机电源连接到VMOT和GND引脚。请记住，在靠近电路板的电动机电源引脚之间放置一个较大的100pF去耦电解电容。



Arduino代码–基本示例

下图将使您对如何使用A4988步进电动机驱动器控制双极步进电动机的速度和旋转方向有完整的了解，并可作为更实际的实验和项目的基础。

```
// Define pin connections & motor's steps per revolution
const int dirPin = 2;
const int stepPin = 3;
const int stepsPerRevolution = 200;

void setup()
{
        // Declare pins as Outputs
        pinMode(stepPin, OUTPUT);
        pinMode(dirPin, OUTPUT);
}
void loop()
{
        // Set motor direction clockwise
        digitalWrite(dirPin, HIGH);

        // Spin motor slowly
        for(int x = 0; x < stepsPerRevolution; x++)
        {
                digitalWrite(stepPin, HIGH);
                delayMicroseconds(2000);
                digitalWrite(stepPin, LOW);
                delayMicroseconds(2000);
        }
        delay(1000); // Wait a second

        // Set motor direction counterclockwise
        digitalWrite(dirPin, LOW);

        // Spin motor quickly
        for(int x = 0; x < stepsPerRevolution; x++)
        {
                digitalWrite(stepPin, HIGH);
                delayMicroseconds(1000);
                digitalWrite(stepPin, LOW);
                delayMicroseconds(1000);
        }
        delay(1000); // Wait a second
}
```

代码说明：

草图从定义A4988的STEP＆DIR引脚连接到的Arduino引脚开始。我们也定义 `stepsPerRevolution` 。进行设置以符合您的步进电机规格。

```
const int dirPin = 2;
const int stepPin = 3;
const int stepsPerRevolution = 200;
```

在代码的设置部分，所有电机控制引脚均声明为数字输出。

```
pinMode(stepPin, OUTPUT);
pinMode(dirPin, OUTPUT);
```

在循环部分中，我们缓慢地顺时针旋转电动机，然后以一秒的间隔快速逆时针旋转电动机。

控制旋转方向：为了控制电动机的旋转方向，我们将DIR引脚设置为HIGH或LOW。高输入将使电动机顺时针旋转，而低输入将使电动机逆时针旋转。

```
digitalWrite(dirPin, HIGH);
```

控制速度：电动机的速度取决于我们发送到STEP引脚的脉冲的频率。脉冲越高，电动机运行得越快。脉冲不过是将输出拉为高电平，等待一会儿，然后将其拉低，然后再次等待。通过更改两个脉冲之间的延迟，可以更改这些脉冲的频率，从而更改电动机的速度。

```
for(int x = 0; x < stepsPerRevolution; x++) {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(1000);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(1000);
}
```

Arduino代码–使用AccelStepper库

在没有库的情况下控制步进器非常适合简单的单电机应用。但是，如果要控制多个步进器，则需要一个库。

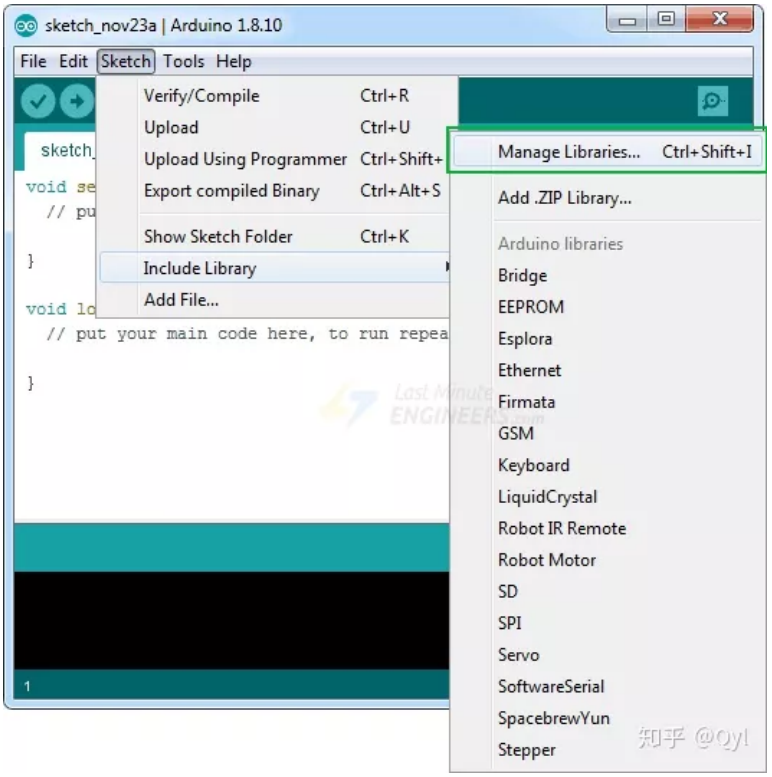
因此，在下一个实验中，我们将使用称为AccelStepper库的高级步进电机库。它支持：

- 加减速。
- 多个同时步进器，每个步进器具有独立的并发步进。

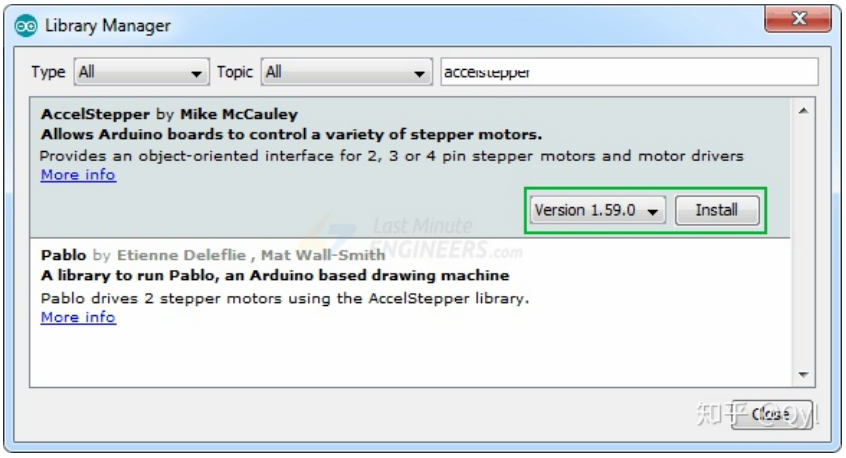
该库未包含在Arduino IDE中，因此您需要首先安装它。

库安装

要安装库，请导航至“草图”>“包含库”>“管理库...”，等待库管理器下载库索引并更新已安装库的列表。



输入“ accelstepper”以过滤搜索。单击第一个条目，然后选择“安装”。



Arduino代码

这是一个简单的草图，它使步进电机沿一个方向加速，然后减速以使其静止。电动机旋转一圈后，它将改变旋转方向。而且它会不断地重复这样做。

```
// Include the AccelStepper Library
#include <AccelStepper.h>

// Define pin connections
const int dirPin = 2;
const int stepPin = 3;

// Define motor interface type
#define motorInterfaceType 1

// Creates an instance
AccelStepper myStepper(motorInterfaceType, stepPin, dirPin);

void setup() {
        // set the maximum speed, acceleration factor,
        // initial speed and the target position
        myStepper.setMaxSpeed(1000);
        myStepper.setAcceleration(50);
        myStepper.setSpeed(200);
        myStepper.moveTo(200);
}

void loop() {
        // Change direction once the motor reaches target position
        if (myStepper.distanceToGo() == 0)
                myStepper.moveTo(-myStepper.currentPosition());

        // Move the motor one step
        myStepper.run();
}
```

代码说明：

我们首先包括新安装的AccelStepper库。

```
#include <AccelStepper.h>
```

我们定义了A4988的STEP和DIR引脚连接到的Arduino引脚。我们还将设置 `motorInterfaceType` 为1。（1表示带有Step和Direction引脚的外部步进驱动器）

```
// Define pin connections
const int dirPin = 2;
const int stepPin = 3;

// Define motor interface type
#define motorInterfaceType 1
```

接下来，我们创建一个名为myStepper的步进器库实例。

```
AccelStepper myStepper(motorInterfaceType, stepPin, dirPin);
```

在设置功能中，我们首先将电动机的最大速度设置为1000。然后，我们为电动机设置一个加速因子，以将加速度和减速度添加到步进电动机的运动中。

接下来，我们将常规速度设置为200，并将其移动到200，即步数（因为NEMA 17每转移动200步）。

```
void setup() {
        myStepper.setMaxSpeed(1000);
        myStepper.setAcceleration(50);
        myStepper.setSpeed(200);
        myStepper.moveTo(200);
}
```

在循环功能中，我们使用If语句来检查电动机 `distanceToGo` 到达目标位置（由设置）之前需要行驶多远（通过读取属性 `moveTo` ）。一旦 `distanceToGo` 达到零，我们将通过将 `moveTo` 位置更改为当前位置的负值来使电动机反向旋转。

现在，在循环的底部，您会注意到我们已经调用了一个 `run()` 函数。这是最重要的功能，因为直到执行此功能，步进器才会运行。

```
void loop() {
        if (myStepper.distanceToGo() == 0)
                myStepper.moveTo(-myStepper.currentPosition());

        myStepper.run();
}
```