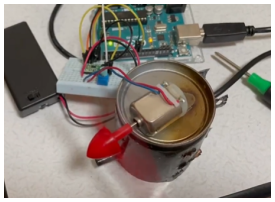
Arduino version pseudo VVVF fan

We released an Arduino program that produces 5 types of train-like sounds with the familiar "FA-130 motor" in Plarail and Mini 4WD. Here, we will briefly explain how to wire the Arduino, how to write a program, how to play, how it works, how to modify it, and so on. Please take advantage of it.

photograph

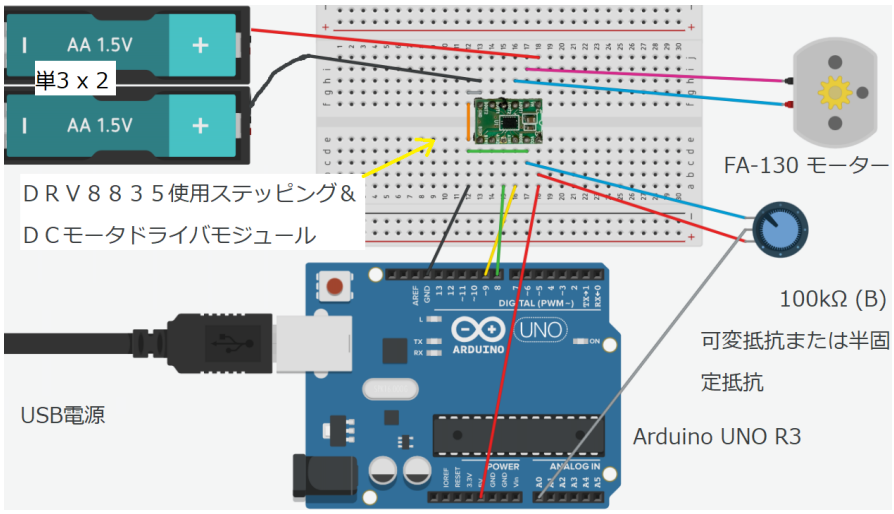


YouTube video (demonstrating the operation of the AVR version and introducing this site)

---

content

Wiring diagram



*Tinkercad (https://www.tinkercad.com/) is used to create wiring diagrams .

Things necessary

- Arduino UNO R3
- USB cable A male - B male (If your computer's USB port is Type A. If not, change the A male side to a type that can be inserted into your computer's USB port.)
- bread board
- Jumper
- Motor FA-130 (with cord, especially MABUCHI MOTOR FA-130RA is good)
- Stepping & DC motor drive module using Akizuki Denshi Tsusho DRV8835
  https://akizukidenshi.com/catalog/g/gK-09848/
  *Soldering is required
- Variable resistor or semi-fixed resistor 100kΩ B curve (Actually, anything from 1kΩ to 100kΩ can be used.)
- Battery box for AA x 2 (convenient with a switch)
- 2 AA batteries
- Propellers (red and blue sets for about 300 yen are available at home centers, model stores, and electronic parts stores)
- Motor reverberation material (Since it is difficult to hear the sound of the motor alone, using double-sided tape to attach it to an appropriate reverberation material such as a paper cup, steel can, tin can, Tupperware, or desk will make the sound overwhelmingly easier to hear. We recommend the bottom of a tin can or steel can for .The texture of the sound changes considerably depending on the material, so it is interesting to try various things.)

If you want to know more about how to use Arduino, please refer to this page (an explanation page I wrote a long time ago). → Learn C

language with Arduino

Note

● We will not be held responsible for any injury or damage caused by using this page as a reference . Please understand that the work is safe and that you are responsible for anything that happens.

● In the above circuit, the Arduino is powered by USB, and the motor is powered by two AA batteries. This battery, that is, the power supply for the motor, is absolutely necessary. Do not connect the motor driver's power supply (VM) to the Arduino's 5V pin, etc., instead of the battery, or take the motor power supply from the computer's USB power supply . The Arduino power supply has an overcurrent protection circuit and should not allow a large current to flow through the motor (the Arduino power supply will be reset), but if you force it to run using the computer's USB power supply, the motor's noise and large current will affect it. In the worst case, your PC will be broken.

● You may freely use or modify the source code, but please refrain from redistributing it under another name. Please remodel at your own risk.

● If you have any other questions, please contact us via Twitter DM or reply below.

---

how to make

1. Install the Arduino IDE from the official website . Install the one that matches the OS of your computer, and follow the on-screen instructions to make the initial settings.

2. Right click here → "Save link as" to download pseudo_vvvf.ino.

3. Create **a new folder called** pseudo_vvvf and move the downloaded pseudo_vvvf.ino into it. (If you don't do this, an error will be displayed in 4.)

4. Double-click pseudo_vvvf.ino to open it in the Arduino IDE.

5. Check if the item "MsTimer2" is displayed at the bottom of the tab "Sketch" → "Include Library". If there is, click an appropriate place outside the list (such as on the source code) to close the list and proceed to 6.

If you don't have it, click "Manage Library" at the top of the list and enter "MsTimer2" in the search field at the top right of the window that appears to search. Move the cursor to MsTimer2 that appears in the list at the bottom, and the "Install" button will be displayed. Press it to

install the MsTimer2 library. Close the window when the installation is complete.

6. Assemble the circuit according to the wiring diagram, connect the Arduino to the computer, and press the "→" (write) button to write the program. If a message like "Would you like to change the COM port to COM* and write?" is displayed, click "Yes". When the writing is finished, if the circuit is correct, the motor should turn with some train sound selected with the volume.

Usage

You can select the sound of the train by turning the variable resistor (volume). In the case of a normal variable resistor (B curve), each section obtained by dividing the volume rotation range into 5 equal parts corresponds to the following trains.

Keikyu New 1000 type image courtesy of [Toyoko @PDSto_yoko]

When you turn on the power of the Arduino (insert the USB) with two AA batteries connected, the fan automatically accelerates with the sound of the train according to the volume at that time. When the acceleration is over, it will automatically decelerate. If you want to change the sound in the middle, adjust the volume first and then press the reset button on the Arduino.

When not in use, remove the battery box and USB to reduce unnecessary consumption.

Mechanism of producing sound

In the first place, Plarail and mini 4WD FA-130 are " DC motors " that rotate just by connecting batteries, and the structure is completely different from " **cage-type three-phase induction motors** " of trains equipped with VVVF inverters . In other words, it is necessary to use a method different from the VVVF device of the actual train to make the sound from the DC motor.

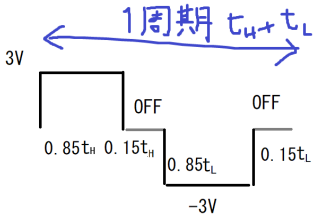
The key to producing sound in this craft is " **PWM control** ". If you've played with Arduino, you'll probably be familiar with this term (analogWrite function). It is a method of slowly rotating a motor by 'smoothing the state in time'.

At this time, you can imagine that the longer the ON time is relative to the OFF time (the higher the duty ratio), the faster the motor will rotate. The important point is that the number of ON/OFF cycles per second (frequency) has little effect on the speed of rotation, so it can be freely determined. In addition, when the motor is turned on and off with PWM, it is thought that small vibrations are generated in the coil and iron core inside the motor, and this vibration is audible.

What I'm trying to say is that if you gradually change the PWM ON and OFF time ratio to accelerate the motor, and at the same time change the ON/OFF speed (frequency) to the pitch that you can hear from the train, you can create something like a VVVF train. You can accelerate the motor while making a sound. By the way, PWM control itself is used as the basis of VVVF inverter control.

The above is a general description of PWM, but this time we made the PWM a special waveform to make a loud sound on purpose. Specifically, instead of turning the motor on and off, the direction of the power supply applied to the motor is changed to the + and - polarity. Force it to vibrate.

However, if you suddenly switch between forward and reverse rotation, the power consumption will be uselessly increased due to the shock (about 600 mA according to the actual measurement), so by adding a "rest" in which the motor is turned off for a short time between switching, the volume can be reduced. Power consumption is reduced with almost no change. The actual waveform is shown below.



$$Duty = \frac{t_H}{t_H + t_L} \times 100\ [\%]$$（と定義する）

It should be noted here that the motor stops when the duty ratio is 50%, not 0% (= when the forward and reverse rotation times are the same). The rotation direction of the motor is switched. In this source code, only the area of 50% or more is used for motor rotation, and the motor is accelerated by changing the duty ratio defined in the figure from 56% to 80%. (56% instead of 50% is because a certain amount of duty ratio is required before the motor starts rotating. The upper limit is 80% to reduce speed.)

Customization/modification points

1. I want to reproduce my favorite train sounds

This source code already contains 5 types of train sounds, but you can reproduce various train sounds by rewriting the program. Here, we will explain how to analyze the sound of the train and how to input the program, using the JR East 209 series train as an example.

In addition, I refer to this page for the method of analyzing the pitch, and the sound of the 209 series (recorded by Mohara Radio) used for analysis is also borrowed from this page.

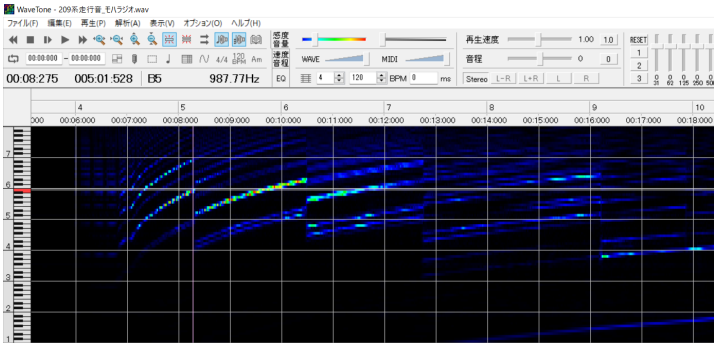
VVVF Factory (CAD Railway Studio)
http://vvvf21851.jp/

First, download the free software called WaveTone. This is excellent software that analyzes the frequency of the voice and displays the time transition. You can download it from this page .

Next, get the sound source as a wav file. If you can hear the motor sound clearly, you can use the actual sound or the sound picked up by Mohara Radio. Since the sound source itself is only used for analysis, you can download the YouTube video as wav from the download site.

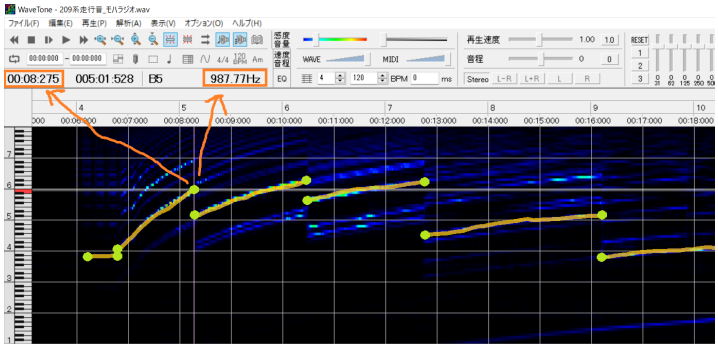
When you drag and drop a wav file containing train sounds into WaveTone and open it, a small window like the one shown in the figure will appear. Press the "Analyze" button.

After that, the main screen that appears looks like the one below, showing the various frequencies involved, with time on the horizontal axis and frequency on the vertical axis.



Looking at this, you can see that there are about 5 lines in the pink vertical line alone, and that it contains complex frequency components. However, you don't have to worry about all of these, just focus on the frequency component that sounds the strongest. In the

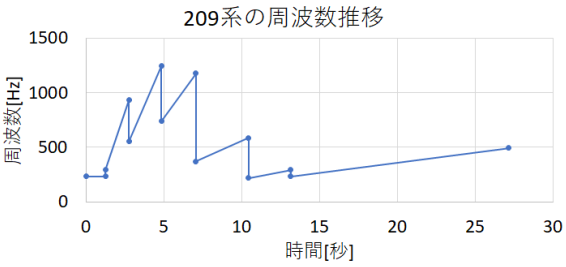
case of 209 series, only the frequency highlighted in orange in the figure below is seen.



As you can see, the strongest frequencies (picked from the figure at a glance) are cut off at several points. The next thing to do is place the cursor over the broken end of the line and note the frequency and time at that point. In the figure, the cursor is placed at 8.275 seconds from the wav start point, frequency 987.77 Hz.

After writing down the times and frequencies of all the points indicated by the yellow-green circles in the figure (actually, there are three more points outside the screen on the right...), type them into the program.

As an aside before that, on the WaveTone screen, the frequency seems to change in a strange curve with respect to time. This is because the value (frequency) increases by the same amount every time), and the normal scale (the most common display method, linear scale, in which the value (frequency) increases by the same amount for the same distance along the vertical axis). Straighten it out. In the case of 209 series, it looks like this (the time on the horizontal axis is slightly corrected, so it does not necessarily match the above figure).



Now let me explain how to enter the program. The program that sounds the 209 series is a function called "void vvvf209()", and the sound is generated by calling this from the setup function.

```
void  vvvf209 ( ) {
  speedUp ( ) ;
  changeTone ( 233 ,  233 ,  1250 ) ;

  changeTone ( 293 ,  932 ,  1500 ) ;
```

```
  changeTone ( 554 ,  1245 ,  2100 ) ;
  _ _ _ _  _ _  _ _ _
  _ _ _ _  _ ,  3400 ) ;
  changeTone ( 220 ,  294 ,  2700 ) ;
  changeTone ( 232 ,  492 ,  14000 ) ;

  freeRun ( 492 ) ;
  changeTone ( 246 ,  138 ,  9600 ) ;
  _ _ _ _
  _ _ _ _  _ _  _ _ _
  _ _ _ _  _ ,  4700 ) ;
  changeTone ( 987 ,  659 ,  3200 ) ;
  changeTone ( 1244 ,  784 ,  2500 ) ;
  changeTone ( 1244 ,  440 ,  2500 ) ;


_
```

First, execute speedUp(); before starting rotation. This initializes the duty ratio, and the motor is automatically accelerated over the next 18 seconds. Acceleration processing is executed in parallel with changeTone next to this, and after 18 seconds, it switches to constant speed operation with a constant duty ratio.

changeTone is a function that changes the sound linearly with time,

```
changeTone(start frequency[Hz], end frequency[Hz], change time[ms])
```

format. For example, in WavTone analysis,

300 Hz at time 2.55 seconds became 500 Hz at time 4.75 seconds

then the change time is 4.75 - 2.55 = 2.20 seconds, so

```
changeTone(300, 500, 2200)
```

It is OK if you write

After reproducing the acceleration for the frequency obtained by the analysis,

```
freeRun(Frequency[Hz])
```

is running. The frequency will be the frequency set just before with changeTone.

The changeTone that follows is the reproduction part of the deceleration sound, and the reproduction method is exactly the same as during acceleration, but it is necessary to pay attention to the timing of executing speedDown();.

When speedDown(); is executed, PWM duty ratio subtraction processing (due to timer interrupt) starts, and the motor decelerates and stops over 18 seconds. In other words, in order to match "end of reproduced deceleration sound" and "motor stop", it is necessary to call speedDown(); about 18 seconds before the end of motor deceleration sound reproduction. There is no problem even if it is off by a few seconds, but if the deceleration sound and the actual stop of the motor are far apart, it will feel strange, so speedDown(); is executed at an appropriate position. In the 209 series example, adding all the times of changeTone() called under speedDown(); gives 15400 milliseconds, or 15.4 seconds, which is 2.6 seconds off, but set to a value relatively close to 18 seconds. There is (conversely, it is OK with such an appropriate setting).

If you want to incorporate a train sound that you really like into a program, the quickest way is to write the contents of a function that corresponds to one of the train sounds (the function name is the name of the train, such as vvvf209, seibu6000).) and replace it with the train sound program you want to play.

Another method is to create a new function that emits train sounds, and if you slightly modify the process in the setup function that selects the train sounds to be called by the volume value, you can incorporate the sounds in a way that increases the variation. can also do. The number of registered trains (5 by default) is assigned to the variable pattern, so change this to 6, and add a new train with case 5: to the switch case statement (because it starts with 0, it ends with 4). increase). For example:

```
  int  pattern  =  6 ;
//     Additions/changes in red
  ... switch ( r ) {
    ... case 4 : seibu6000 ( ) ; break ; case 5:
    meijo2000
       ();
    break
       ; Create a new function by arranging it with the train function (in any order)
 void meijo2000(){
   speedUp();
   …
 }
```

However, if you add too many types of sounds, it will be difficult to select by volume, so please try to devise a suitable way, such as choosing between them or changing the method of selecting sounds. Once the program is done, let's write it to Arduino and actually listen to the sound. If you hear something that sounds strange, try changing the frequency you are interested in on WaveTone or changing the length of the sound to make adjustments that are more comfortable. Also, I think that there are various types of train sounds that are easy to reproduce and difficult to reproduce, so let's try to reproduce various train sounds.

2. I want to change the pin used

You can change the pin to be used by changing this number at the beginning of the source code and writing it.

```
#define AIN1 9 //Pin to connect to AIN1 of motor driver
 #define AIN2 8 //Pin to connect to AIN2 of motor driver
 #define VR A0 //Pin to connect volume analog input
```

3. I want to use it with other Arduino (Nano, Mega, etc.)

I haven't tried it, but I think it will probably work. This is because it does not use the notation unique to Arduino UNO (such as port manipulation) and special functions. In the case of a compatible board with special functions, it is impossible to know until it is moved.

4. I want to run with only two batteries

By using the boost module, you can run the Arduino and the motor together with two batteries. Here are some recommended examples.

5V output step-up DCDC converter
https://akizukidenshi.com/catalog/g/gK-13065/

With the wiring diagram at the beginning, the USB is not connected to the Arduino, and the boost module

- From the positive pole of two batteries to the IN pin
- 2 batteries - pole to GND pin

- From the Vcc pin of the motor driver to the OUT pin

just connect the wires. This method can be used with Arduino UNO and Arduino Nano, but it cannot work with Arduino Mega, which consumes a large amount of current and exceeds the limit of the booster circuit. Also, using a booster circuit will increase current consumption, so it is better to install a switch in the battery box or the wiring from the battery box to turn off the main power supply when not in use.

5. I want to make it compact

Since the Arduino UNO is large, there may be cases where you want to pack it more compactly and store it in a box or something. Arduino Nano Every is convenient because it is compact and has a large number of pins. I would like to say that this source code is compatible with Nano Every, so I would recommend it. If anyone has tried it, I would appreciate it if you could let me know if you succeeded or if it didn't go well.

Combining this with 4. should be possible to fit in a compact box like the YouTube version.

6. I want to increase/decrease the maximum speed of the motor/I want to change the ease of starting

about the 3rd line from the top

```
volatile  int  V_START  =  56 ,  V_END  =  80 ,  V_OFF  =  50 ,  T  =  180 ;
```

The first three "V_○○" define the PWM duty ratio range during acceleration/deceleration. 50 is stop, 100 is full forward rotation, 0 is full reverse rotation. For the meaning of the value, please refer to "Mechanism for producing sound". in particular

Acceleration: Linear change from V_START to V_END
Deceleration: Linear change from V_END to V_OFF

It is a program called. In other words, the magnitude of V_END determines the maximum speed of the motor, and V_START is the PWM value at which it begins to rotate. If V_START is too large than 50, it will turn quickly as soon as the switch is turned on, and if it is too close to 50, it will take time to start rotating. The remaining V_OFF is normally 50, but if the motor resistance is high or the load is applied and the vehicle stops too quickly (e.g. when loaded on a Plarail), set this to a value higher than 50. You can respond by doing If you overwrite the program that changed these values to the Arduino, you can correct the rotation speed of the motor. It may be better to adjust V_START, V_END, and V_OFF because the ease of rotation changes when a load is applied to the motor.

T specifies the time to accelerate and decelerate the motor with 100 milliseconds as 1. So 180 means 18 seconds. It is possible to change here as well, but since the execution position of speedDown(); explained in 1. is made with the assumption that the deceleration time is 18 seconds, if only T is changed greatly, the deceleration will not be clean. If you want to change T significantly (more than 30 as a guideline?), you need to change the speedDown(); execution position of all recorded train sounds as well.

URLs

Twitter of the author (Iwango)
https://twitter.com/iwango_system

AVR version circuit diagram and program

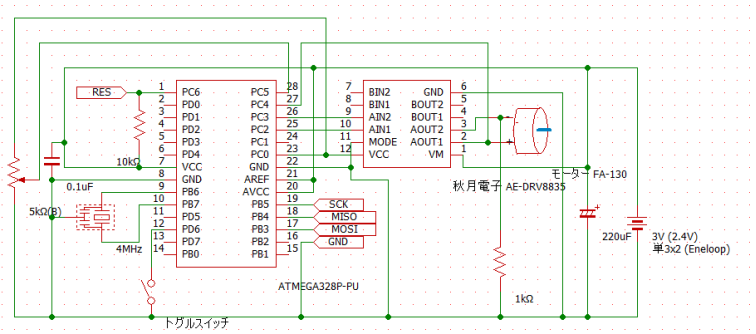
Video of VVVF fan (AVR version) (Twitter)
https://twitter.com/iwango_system/status/1472130548579844096?s=20

The "AVR version" published on YouTube and Twitter is developed with a raw AVR microcomputer (ATmega328p) that does not go through Arduino functions in order to fit compactly in a tin can of Seria without driving with two batteries or boosting. . The "Arduino version" that we have introduced so far is a slightly improved version of the Arduino prototype program that was used when developing the battery-powered version. The circuit diagram and source code are posted for reference.

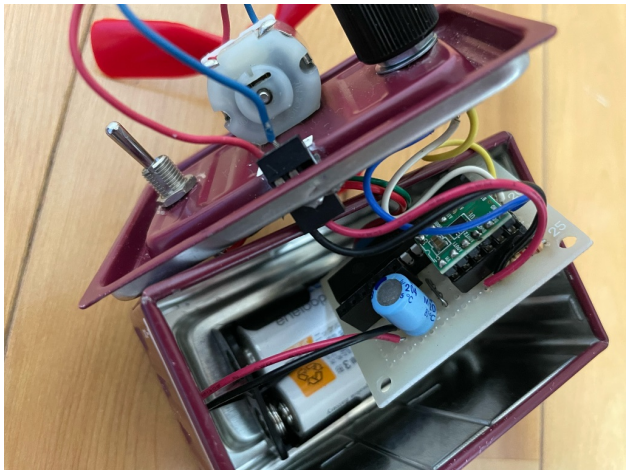
It's a little more difficult to make than the Arduino version because it requires board mounting/wiring and AVR writing, but it's compact and cheap, so please give it a try.

source code
pseudo_vvvf.c

AVR version schematic



inside



Top page