How to use arduino control board LY-F2 and STONE serial display to implement piano key project

This project uses a serial touch screen STVI056WT-01 with arduino control board LY-F2 design, the main accessories are as follows.

1, STVI056WT-01 serial touch screen and STONE adapter board V1.2.

2, gift box speaker.

3, LY-F2 development board.

Design idea

Serial screen carries piano keys, returns different key values, serial port notifies arduino decoding, arduino development board uses instruction tone (), outputs the corresponding key tone.

Working steps

1. import the piano key images into the STONE screen development

platform Toolbox.

a. add return value touch keys to the piano keys.

b. Plan the data address and return value of the keys.

c. Set the power-on screen to piano keys, check the return value to upload the serial port immediately.

2. debugging speaker sounding using arduino programming.

a) connect the speaker and microcontroller output IO.

b) programming different frequencies, debugging sound compared with the actual piano.

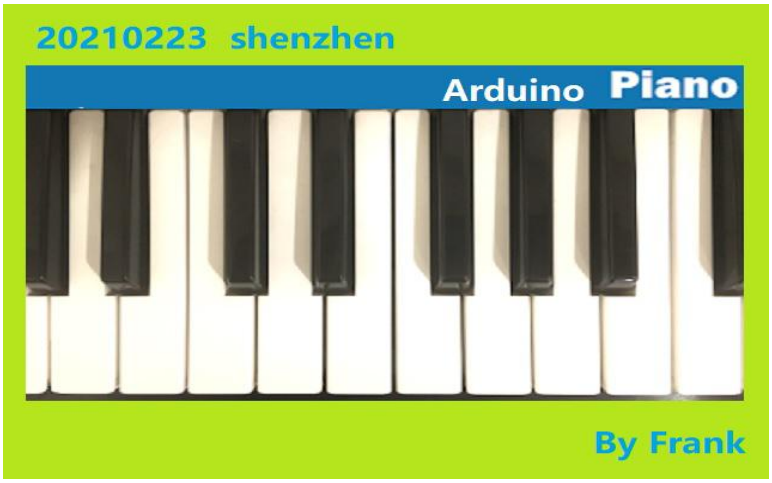
c) create a table of key output frequencies.

3. Connect the touch screen and arduino development board through the serial port, decoding and coupling to achieve real-time analog sounding function.

Next, record the specific development process.

First, create the piano keys image.

The 640*480 piano keys picture made according to the resolution of the STONE screen used in this project is as follows.



Next, import the image into Tool4.3 and create a return value button.

Check "Return pressed key value" under the menu Touch Cinfiguration (M) in Tool4.3 tool software, or click on the button tool in the picture above to create a button in the desired location, and fill in the variable address and key value in the marker as above. Check the box to automatically upload the return value when pressed. The 10 piano key variable addresses here are 0x0800, 0X0802, 0X0804 ..... .0X0812, and the key return values are 0x0060, 0x0061, 0x0062..... , 0x0069; as follows.

As monitored by the serial assistant, the values returned by the serial assistant for each key are as follows

a5 5a 06 83 08 00 01 00 60

A5 5A 06 83 08 02 01 00 61

A5 5A 06 83 08 04 01 00 62

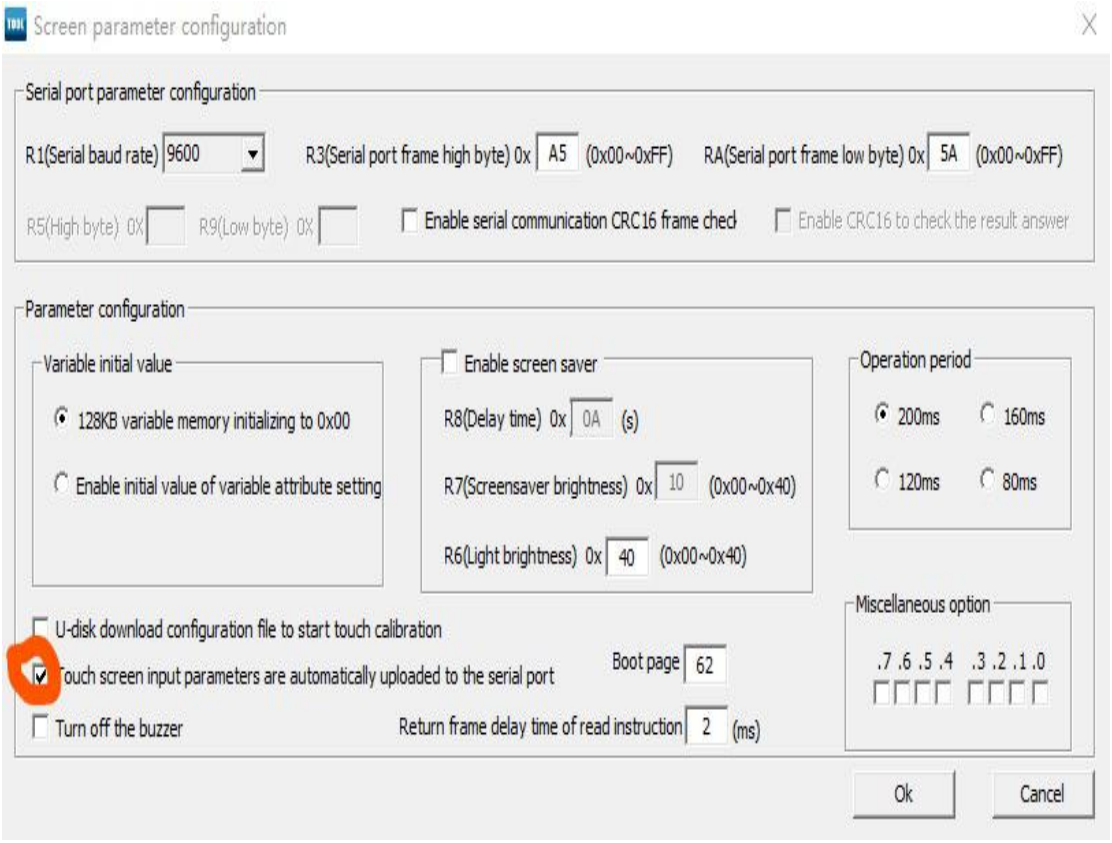
A5 5A 06 83 08 06 01 00 63

.....

a5 5a 06 83 08 12 01 00 69

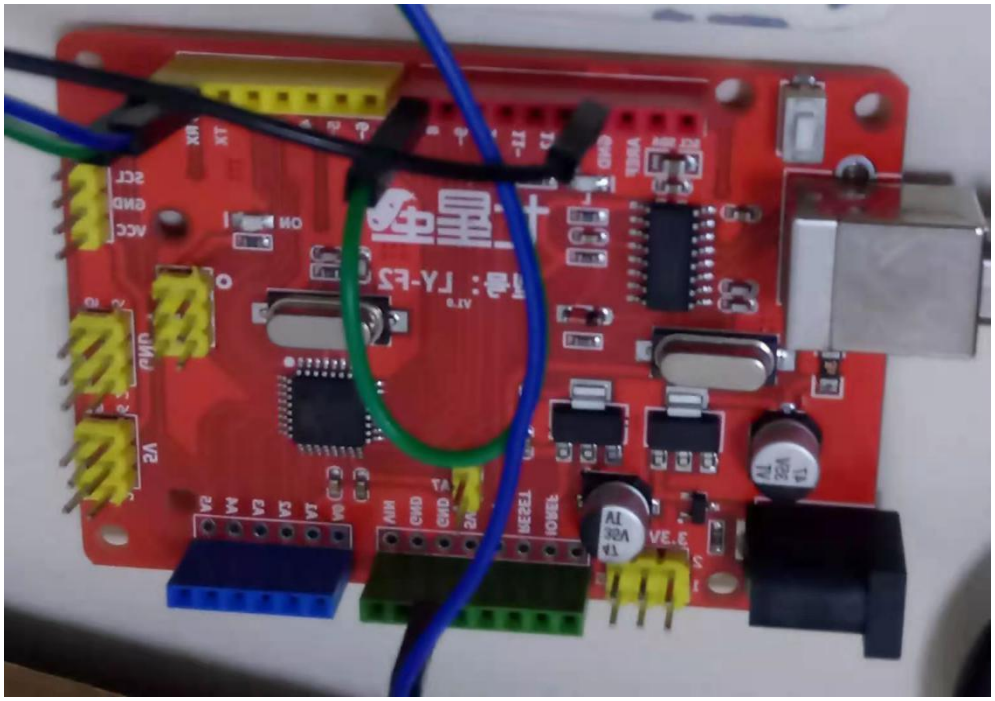
If there is no key return value, remember to check and tick the red marker in the picture below.

Location: Main Menu --》 Tool (T) --》 Screen Configuration.



Here, we introduce the arduino development platform before we do the programming of the sound control function.

The following figure is the arduino development board used in this project, the model is LY-F2.



Arduino platform is very good, no need to layout and play the board, also do not need to find parts welding, simple development environment, also comes with a variety of routines.

To the "arduino language instructions" focus on the application of part of the serial port transceiver function, instructions are extracted as follows.

Serial port send/receive function

Serial.begin(speed) Serial port definition baud rate function, speed means baud rate, such as 9600, 19200, etc.

int Serial.available() Determine the buffer status.

int Serial.read() Read the serial port and return the received parameters.

Serial.flush() flushes the buffer.

Serial.print(data) Output data from the serial port.

Serial.println(data) Serial port outputs data with a carriage return character.

There is a more detailed description of the instructions in the 232 page "Getting.Started.with.Arduino" pdf full English electronic document, screenshot saved below.

Serial.begin(speed)

Prepares Arduino to begin sending and receiving serial data. You'll generally use 9600 bits per second (bps) with the Arduino IDE serial monitor, but other speeds are available, usually no more than 115,200 bps.

Example:

```
Serial.begin(9600);
```

(arduino serial port maximum baud rate setting)

As you can see, arduino can support up to 115200 baud rate of serial screen, this time the selected one is 9600.

Serial.print(data)
Serial.print(data, encoding)

Sends some data to the serial port. The encoding is optional; if not supplied, the data is treated as much like plain text as possible.

Examples:

```
Serial.print(75);       // Prints "75"
Serial.print(75, DEC);  // The same as above.
Serial.print(75, HEX);  // "4B" (75 in hexadecimal)
Serial.print(75, OCT);  // "113" (75 in octal)
Serial.print(75, BIN);  // "1001011" (75 in binary)
Serial.print(75, BYTE); // "K" (the raw byte happens to
                        // be 75 in the ASCII set)
```

Serial.println(data)
Serial.println(data, encoding)

Same as *Serial.print()*, except that it adds a carriage return and linefeed (*\r\n*) as if you had typed the data and then pressed Return or Enter.

Examples:

```
Serial.println(75);       // Prints "75\r\n"
Serial.println(75, DEC);  // The same as above.
Serial.println(75, HEX);  // "4B\r\n"
Serial.println(75, OCT);  // "113\r\n"
Serial.println(75, BIN);  // "1001011\r\n"
Serial.println(75, BYTE); // "K\r\n"
```

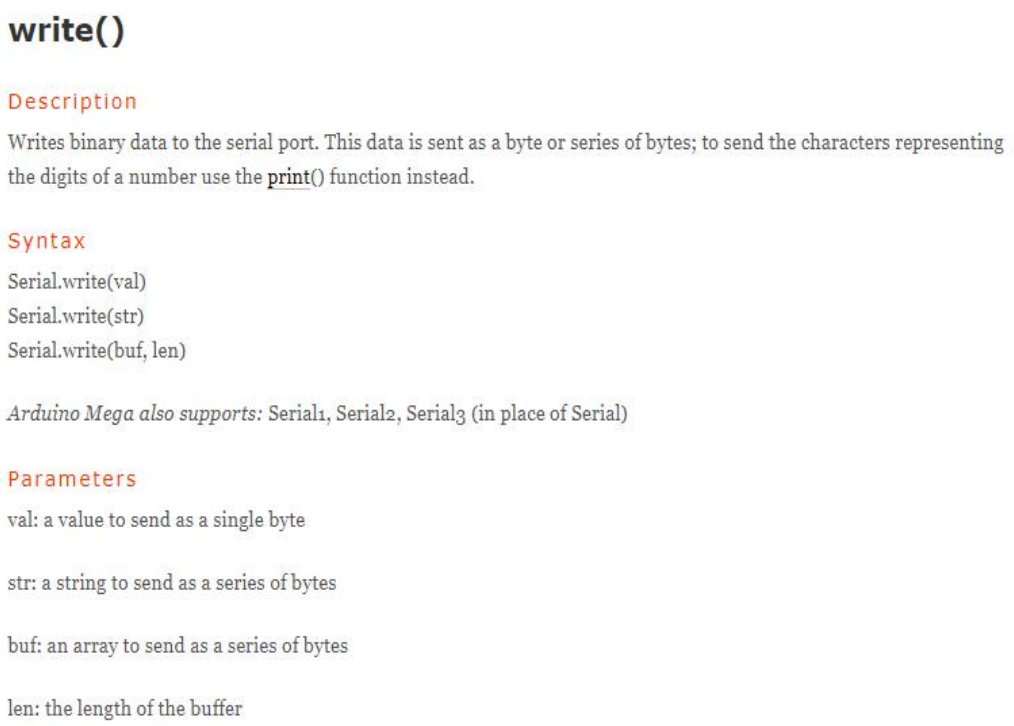
After testing with the serial assistant, the two print functions above transmit in character form.

For example：

Serial.print(75,HEX);

The string transmitted is "4B", which is the ASCII value of the characters "4" and "B" (0x34, 0x42); therefore, serial. print() function, whether string, or data, are sent ASCII, not suitable for command transmission!

In arduino 1.0.5, under the menu help, click on "Reference", then you will see the function descriptions of each function, among which the latest serial function is serial.write(Val), the screenshot is as follows.



write()

Description

Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead.

Syntax

Serial.write(val)
Serial.write(str)
Serial.write(buf, len)

Arduino Mega also supports: Serial1, Serial2, Serial3 (in place of Serial)

Parameters

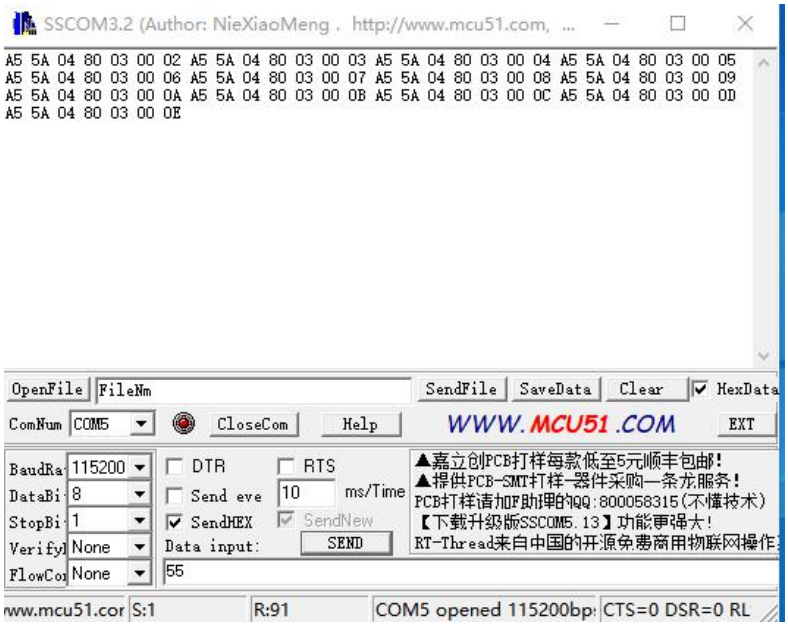
val: a value to send as a single byte

str: a string to send as a series of bytes

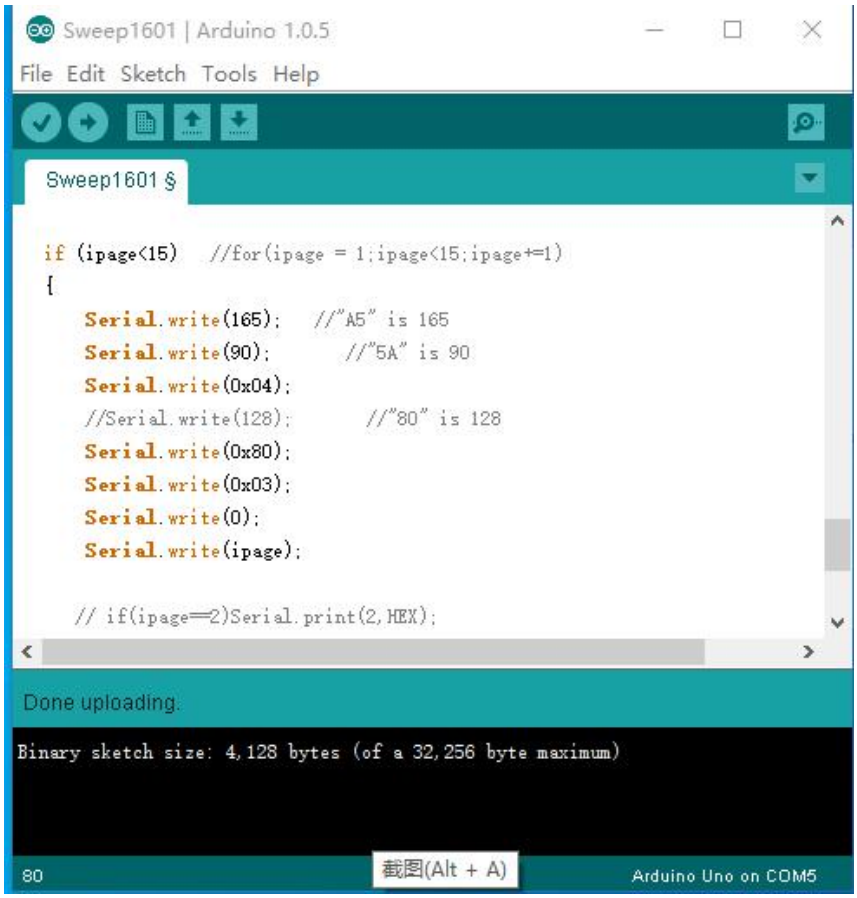
buf: an array to send as a series of bytes

len: the length of the buffer

The function transmits the binary code, and the test transmission is correct and can jump pages correctly; as follows.



The serial.write(Val) function supports writing hexadecimal directly, such as serial.write(0x80), which compiles and uploads successfully, as shown below. Therefore, the serialwrite() function sends the value VAL in binary and string in ASCII.



int Serial.available()

Returns how many unread bytes are available on the Serial port for reading via the *read()* function. After you have *read()* everything available, *Serial.available()* returns 0 until new data arrives on the serial port.

Example:

```
int count = Serial.available();
```

int Serial.read()

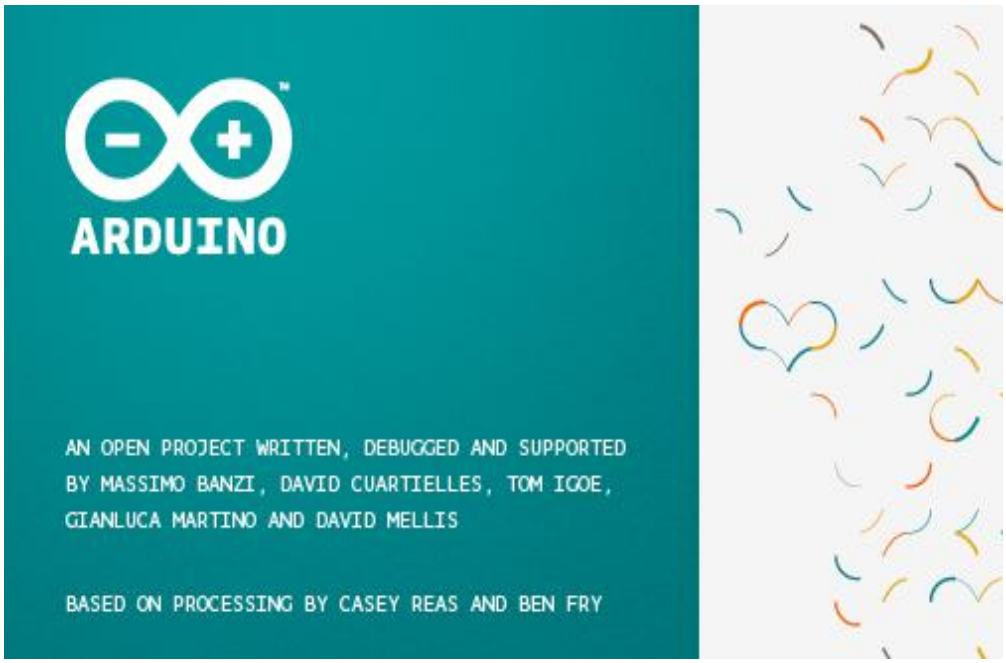
Fetches one byte of incoming serial data.

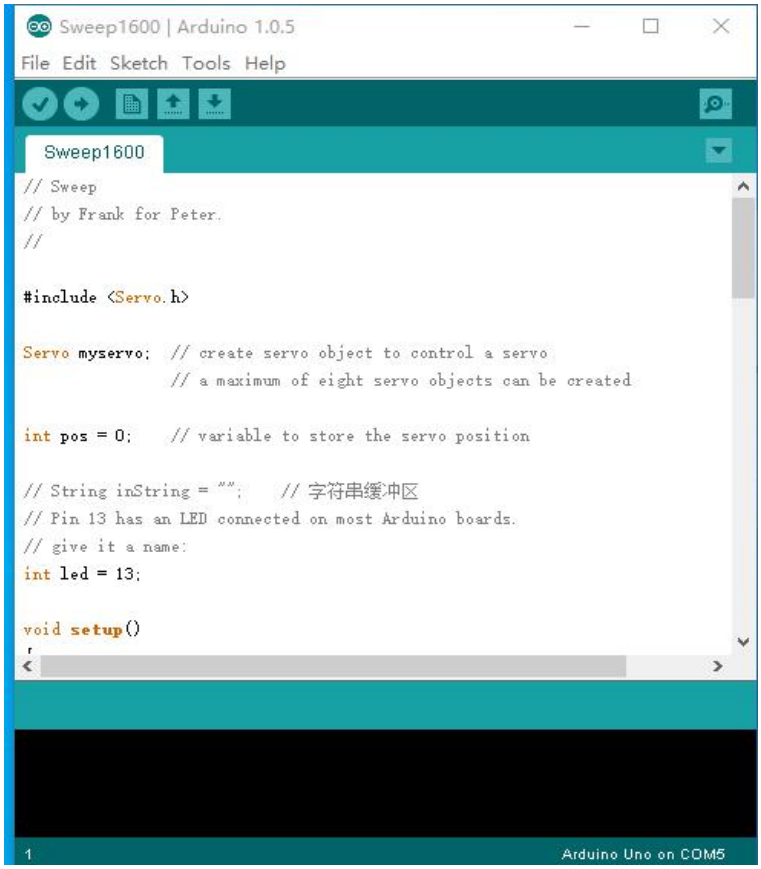
Example:

```
int data = Serial.read();
```

For the arduino development environment, just run the arduino.exe directly under the arduino-1.0.5 package directory.

After double-clicking, wait for a while, the following startup screen will appear.





The extension .ino is the arduino file and must be placed in the folder with the same name; to download the program written in arduino to the development board is to use "upload" under the menu "file". In other places (such as Tool4.3), it may be "download".

Arduino will first compile the .ino file during the upload operation, and the status bar below will show "uploading" after compiling and "Done uploading" after finishing. ". If there is an error in the compilation process, it will be displayed (e.g., variable not defined, etc.) and the upload operation will be terminated. When finished, if there is a red warning under the status bar, it means that there is an error and you need to check and re-upload.

How about making a sound?

In the main menu of arduino 1.0.5 software, help--->Reference, a browser will pop up, which has the function of tone().

The speaker of this project is connected to 8 pins, and the frequency table is in notes[]. Each keystroke sounds last 0.36 seconds. It can be used like this: tone(8, notes[], 360). The notes[] array holds the frequency of the 10 keys. With the key value returned by the serial port, you can use this tone() function to drive the speaker to emit the corresponding tone!

The frequencies determined by debugging this project are as follows.

int notes[] = {262, 294, 330, 349, 389, 430, 449, 469, 489, 509};

These 10 frequencies emit a detailed breakdown of the variation of each tone, which sounds great with the chosen speakers!

tone()

Description

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

NOTE: if you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

Syntax

tone(pin, frequency)
tone(pin, frequency, duration)

Parameters

pin: the pin on which to generate the tone

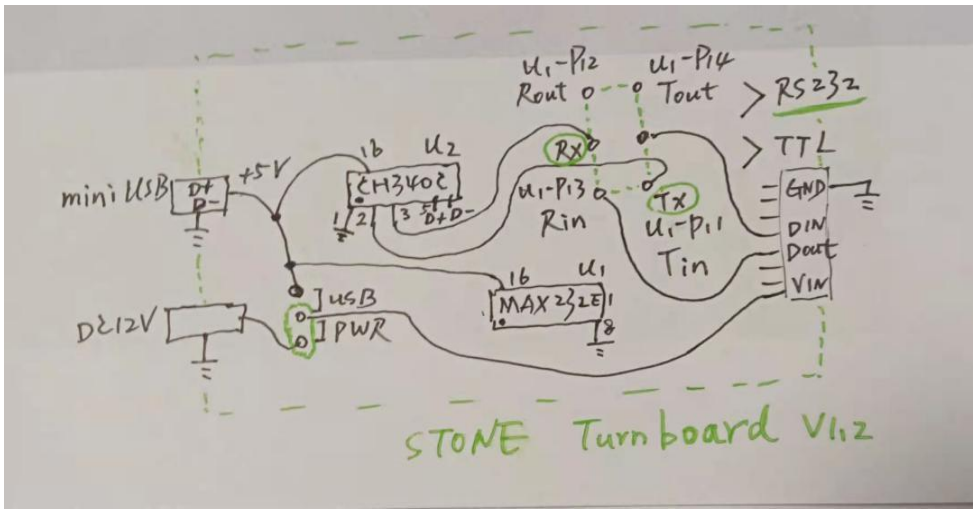
frequency: the frequency of the tone in hertz - *unsigned int*

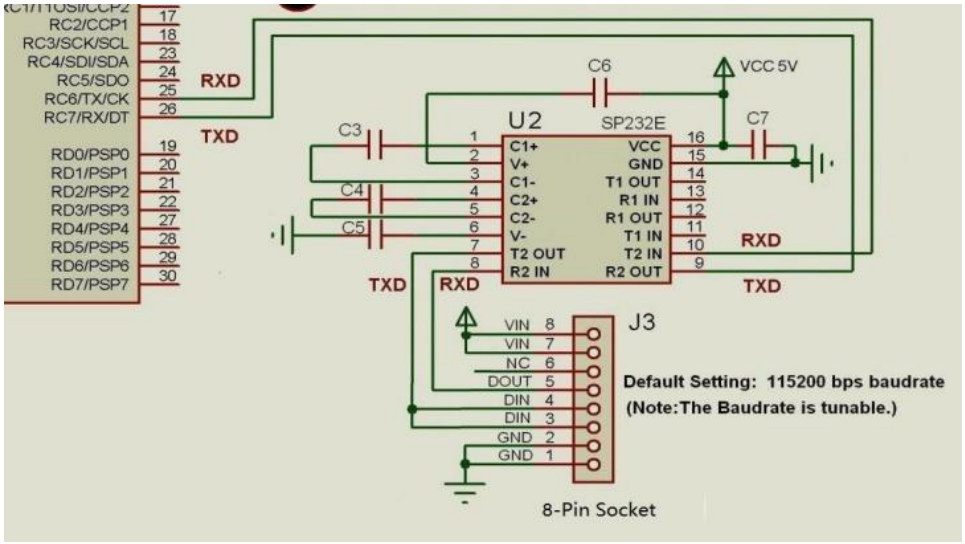
duration: the duration of the tone in milliseconds (optional) - *unsigned long*

Next, connect the serial screen to the MCU.

Check the "Datasheet STVI056WT-01" pdf, find the serial screen circuit diagram on page 31, and find that the DIN and DOUT signals of the screen interface are RS232. STONE included an adapter board with the screen, but I didn't see any circuit diagram of the adapter board, so I

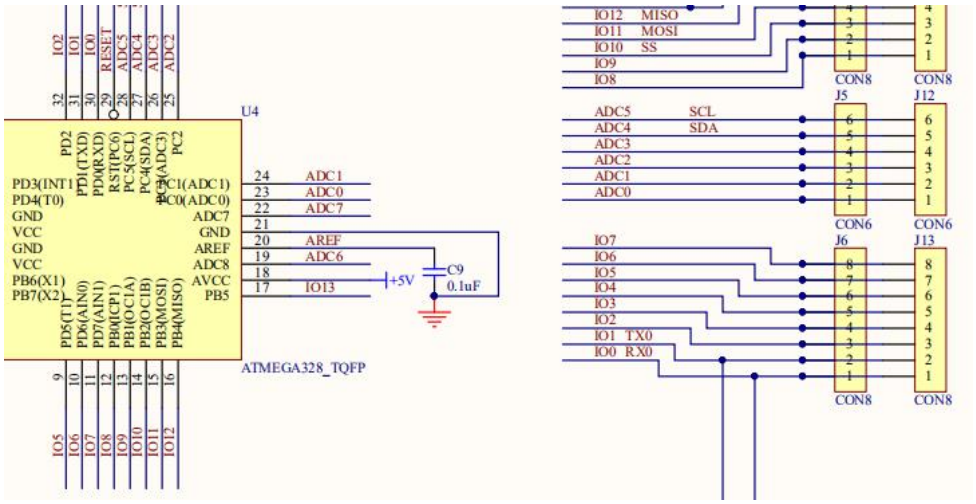
drew one myself. From the drawing, we found that there is MAX232E1 chip on the adapter board, and the TX and RX signals are led out through the jumper, it plays a conversion role, the serial TTL of the arduino development board can match the RS232 signal of the STVI056WT-01 serial screen! From the drawing also know that the STONE adapter board DC12V power supply is supplied to the serial screen, the internal MAX232E1, CH340C power supply +5V are provided by the USB, so when using RS232 to serial TTL level conversion, the USB needs to be powered. For the adapter board MAX232 chip, Tin, Rout are 0-5V, while Tout, Rin pin is + - 15V logic. Detailed connection see the hand drawing, STONE adapter board USB and 12V power supply, LY-F2 development board USB power supply should be connected, LY-F2 development board TX ----- adapter board TX, LY-F2 development board RX ------ adapter board Rout (if you connect RX you need to short the Rout and RX).

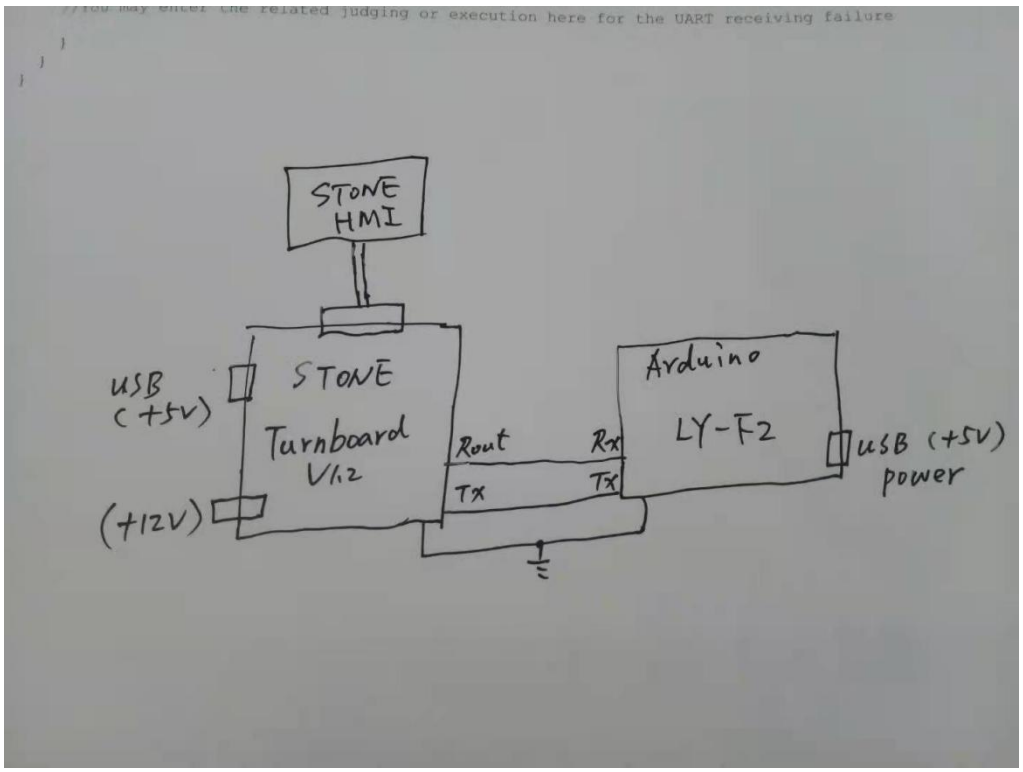




As you can see in the picture, the serial screen comes out at RS232 level, and since RS232 is plus or minus 15V logic, it is able to transmit farther!

Check the "LY-F2 development board circuit diagram", screenshot as follows.





Then secondly, the serial communication and control functions are programmed.

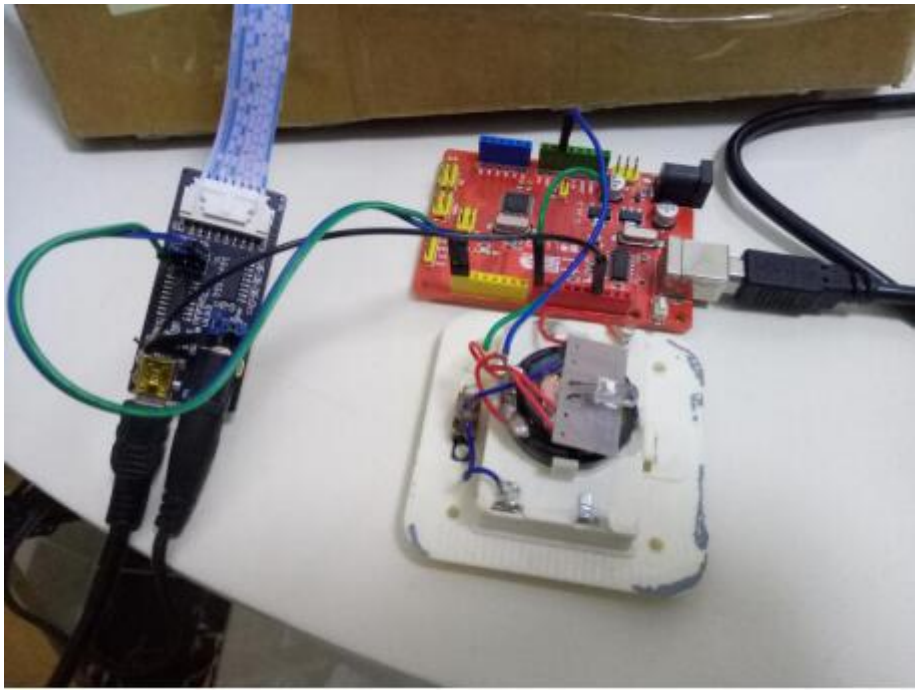
With the above introduction, it is easy to program and the demo code is as follows.

```
/Piano
// by Frank 20210223.
//--------------------------------------------------------------------------------
int inDelay = 0;
String inString = ""; // String buffer
int ipage = 2;
int notes[] = {262, 294, 330, 349, 389, 430, 449, 469, 489, 509}; // Pronunciation frequency array table!

void setup()
{
    Serial.begin(9600); // Open serial communication function wait for serial port to open, baud rate preset.
   while (!Serial) {
      Needed for Leonardo only
   Needed for Leonardo only } }
```

```
void loop()
{
   int inChar;

   // Read the information sent by the serial port:
   if (Serial.available() > 0) { inChar = Serial.read(); }

/* Piano key out music!

if (inChar >= 0x60) { // 0x0060--0x0069 is the Piano key value!
   if (inChar <= 0x69) {
       tone(8, notes[inChar - 96]); // 0x60 is 96!
       delay(360); // every key is pressed, the music output 0.36s!
noTone(8); // close output!
/* Too, Can write sampe:
      Tone(8,notes[inChar - 96],360);
*/
   }
}}
```

Finally, online debugging.

Tool4.3 tool software to edit a good screen file download, arduino code file upload, connect the power supply, communication, arduino development board IO8 and GND connected to the speaker two lines, operation touch screen piano keys, pronunciation function are demonstrated normal!




Arduino Piano