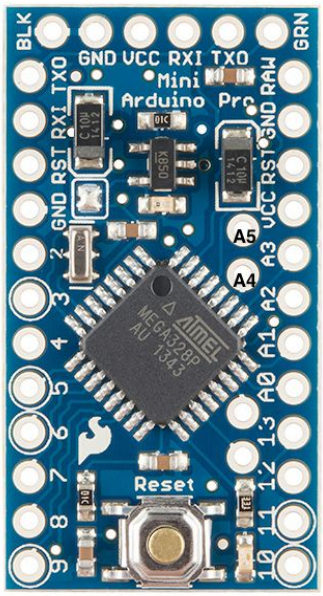
[All files are kept at https://goo.gl/XCJGKx]

Target Processor Boards

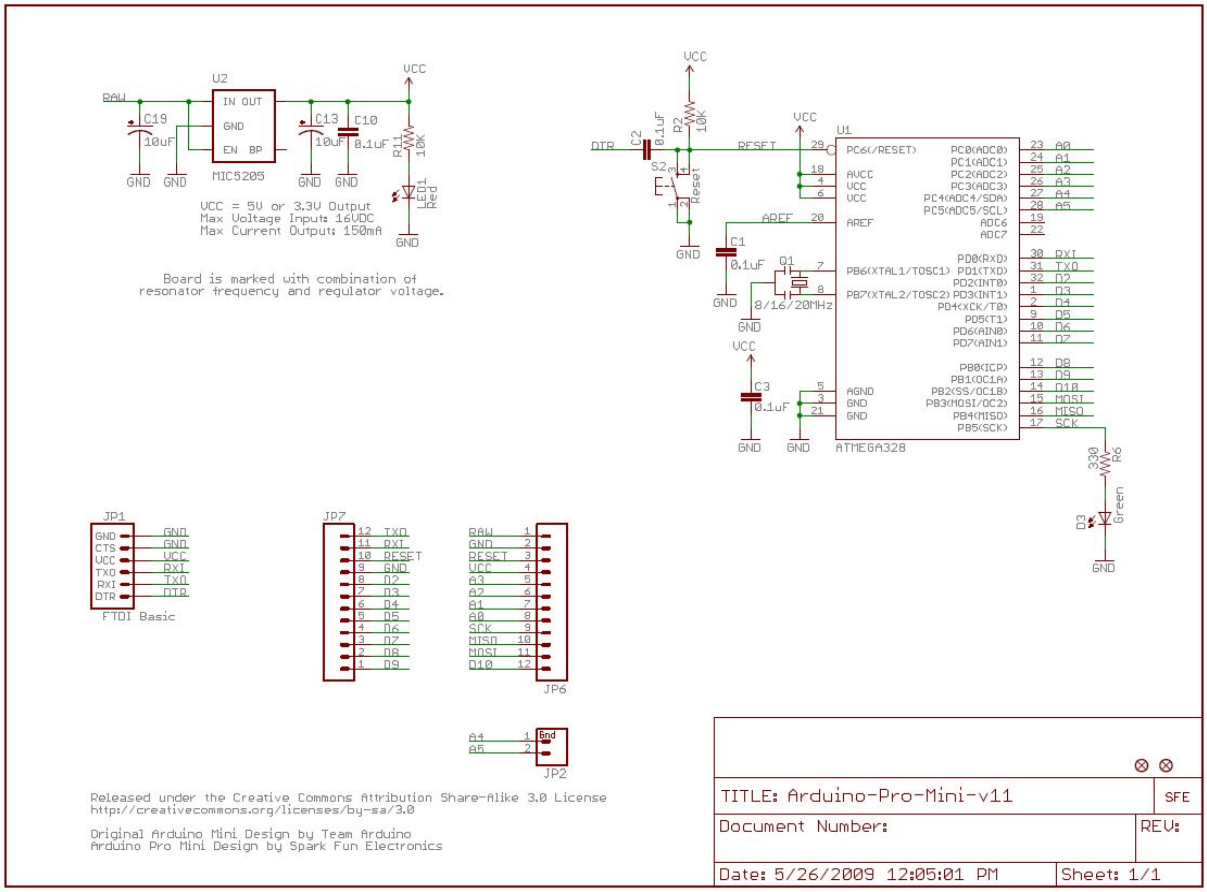
1 Arduino Pro Mini

We have two versions of the Arduino Pro Mini that only differ according to how many extra analogue-compatible input lines are brought out to headers. The Pro Mini board provides the basic microcontroller with reset circuit and host serial port connector but none of the development circuitry such as a USB interface. This means that any product is left with only the essential hardware after development.

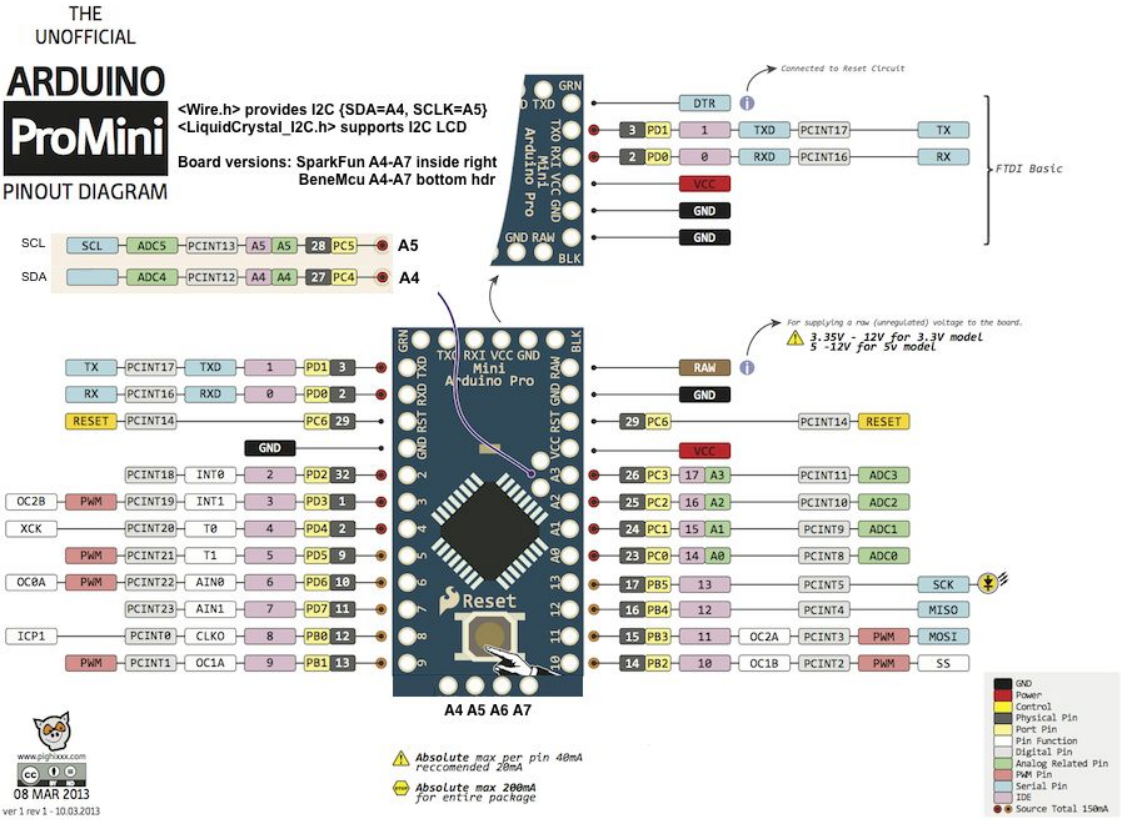


SparkFun Pro-Mini (A4,A5 next to A3) BeteMcu Pro-Mini (A4,A5,A6,A7 across bottom)

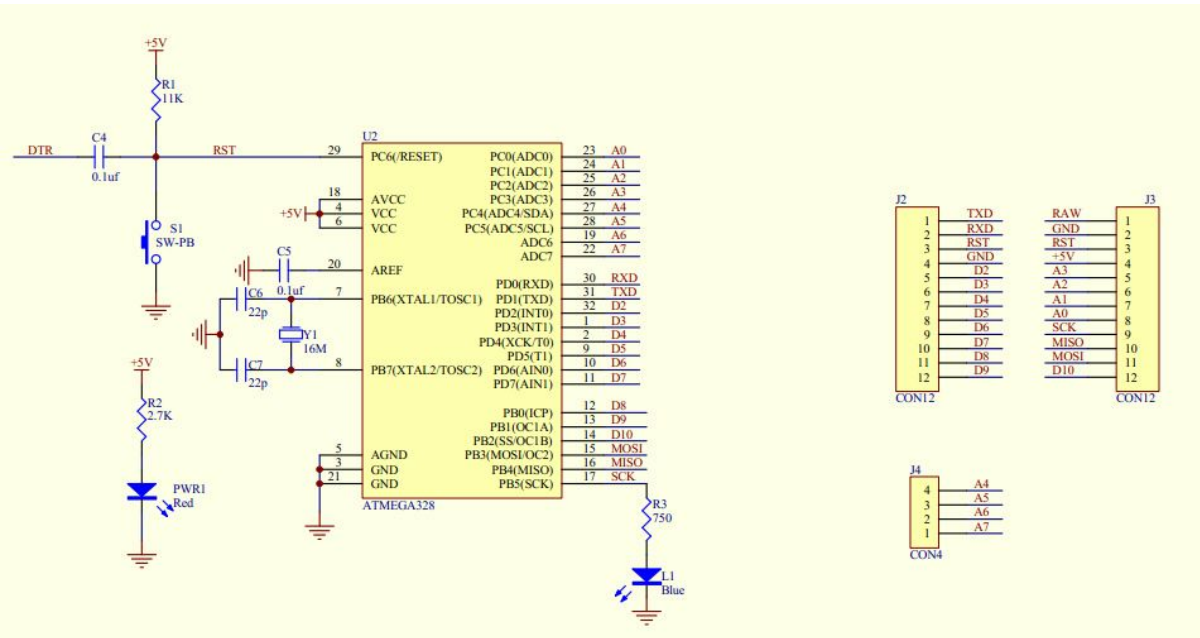
The SparkFun version provides {A4,A5} on a header located mid-board and has the circuit (full size is file ArduinoProMini-5V-16MHz SparkFun.png):



The voltage regulator to convert raw DC to Vcc is a low dropout version but for this workshop, we power the Vcc directly from the USB power coming from the host. For an overview of connections, see the next diagram and note that the A4 and A5 positions have been corrected (full size is file Arduino-Pro-Mini-v14-v2a.png).



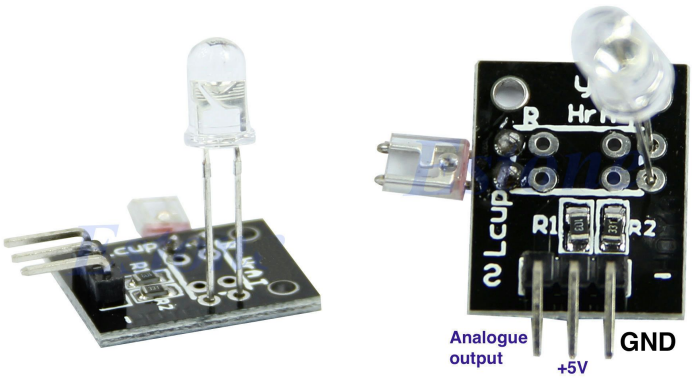
The BeteMcu version provides {A4,A5,A6,A7} on a header at the opposite end of the host connector (i.e. the extra connector at the bottom of the image above) but the circuit is essentially the same (full size is file Arduino-Pro-Mini image BM.png):



Peripherals

1 Heartbeat sensor

Information comes from the ebay vendow listing http://www.ebay.com.au/itm/181256874174 (software) and https://tkkrlab.nl/wiki/Arduino_KY-039_Detect_the_heartbeat_module (pulse sensor overview).



The circuit provides a bright infrared (IR) LED and a phototransistor to detect the pulse of the finger. The system is set up with the LED on one side of the finger and the phototransistor on the other side. The phototransistor senses the light flux transmitted and pulse detection is based on the fact that this changes according to the blood pressure. The software uses the ADC to obtain a number representing the received light detected by the phototransistor, and then this value is smoothed using the recursive relationship

```
double value = alpha * oldValue + (1 - alpha) * rawValue;
```

where alpha is in the range 0 to 1. The sample code uses alpha=0.75. Note that choosing a high alpha smooths “value” as the contribution of the ADC value “rawValue” is lowered (due to term 1-alpha). In contrast, choosing a low alpha makes “value” depend more on the latest rawValue. Next, the expression

```
(change<0.0 && oldChange>0.0)
```

is used as a turning point detector i.e. it is true when the current “value” is decreasing *and* the immediate previous value was increasing. Each turning point in the smoothed “value” indicates a pulse.

Code sample found on-line

```
// Sample code flashes a LED on each detected pulse

int ledPin=13;
int sensorPin=0;

double alpha=0.75;
int period=20;
double change=0.0;

void setup()
{
  pinMode(ledPin,OUTPUT);
}

void loop()
{
  static double oldValue=0;
  static double oldChange=0;

  int rawValue=analogRead(sensorPin);
  double value=alpha*oldValue+(1-alpha)*rawValue;

  change=value-oldValue;
  Digitalwrite(ledPin,(change<0.0&&oldChange>0.0));

  oldValue=value;
  oldChange=change;

  delay(period);
}
```

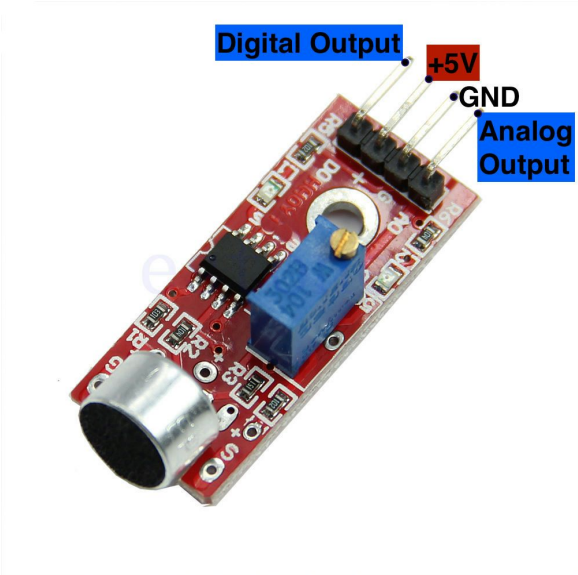
Local experimental code

```
/*
 * detect_pulse_adc
 *   reads the light transmission sensor attached to pin SENSOR_pin_adc and
 *   smooths it via an exponential decay filter
 *                sensorValue = alpha * oldSensorValue + (1-alpha) * rawSensorValue;
 *   A pulse is detected when previous change and current change indicate a
 *   turning point.
 *
 * To do:
 * . experiment with values for alpha
 * . add a downcount debouncer so detection of new pulses is prevented
 *   immediately after a new pulse
 * . replace the use of the relatively slow serial port for checking the
 *   sensorValue with a call to lcd.print (see the definitions and setup
 *   in directory examples/LCD)
 *
 * The original code was found online at
 *   https://tkkrlab.nl/wiki/Arduino_KY039_Detect_the_heartbeat_module
 */

//////////////////////////////////////////////////////////////////////////

// Define signal names in terms of standard Arduino board pin numbers
#define LED_pin         13       // pin 13
#define SENSOR_pin_adc  A0         // adc0

const double alpha_forgettingFactor = 0.75;
const int sampling_period = 20;         // ms delay between sampling

// initialise IO ports
void setup()
{
  // initialize digital LED_pin as an output
  pinMode(LED_pin, OUTPUT);
  // set UART to 57600 baud
  Serial.begin(57600);
}

// main loop
// - filter sensor data looking for a turning point that indicates a pulse
// - display sensor data on console
char status_msg[20];
void loop()
{
```

```
  static double oldSensorValue = 0.0;
  static double oldChange      = 0.0;
  static double change = 0.0;
  static int debounce = 0;

  // read sensor and smooth sensorValue
  int rawSensorValue = analogRead(SENSOR_pin_adc);
  double sensorValue = alpha_forgettingFactor * oldSensorValue +
(1-alpha_forgettingFactor) * rawSensorValue;

  // calculate change and detect turning point
  // (previous change < 0 and this change > 0 means minimum turning point found)
  change = sensorValue-oldSensorValue;
  int is_pulse = change<0.0&&oldChange>0.0;
  digitalWrite(LED_pin,is_pulse);

  // maintain history for next update, and wait before next sampling
  oldSensorValue = sensorValue;
  oldChange      = change;
  delay(sampling_period);

  sprintf(status_msg,"%d\n", (int) sensorValue);
  Serial.print( status_msg );
  //  lcd.setCursor( 0,0 ); lcd.print( rawSensorValue );
  //  lcd.print( ", ");     lcd.print( (int) sensorValue );
}
```

2 Microphone board

The ebay vendor page is http://www.ebay.com.au/itm/181261402839



Sample analogue input program found online

```
// Continuously input mic value
int sensorPin = A5;     // select the input pin for the potentiometer
int ledPin = 13;        // select the pin for the LED
int sensorValue = 0;    // variable to store the value coming from the sensor

void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  sensorValue = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);
  delay(sensorValue);
  digitalWrite(ledPin, LOW);
  delay(sensorValue);
  Serial.println(sensorValue, DEC);
}
```

A local version with LCD display

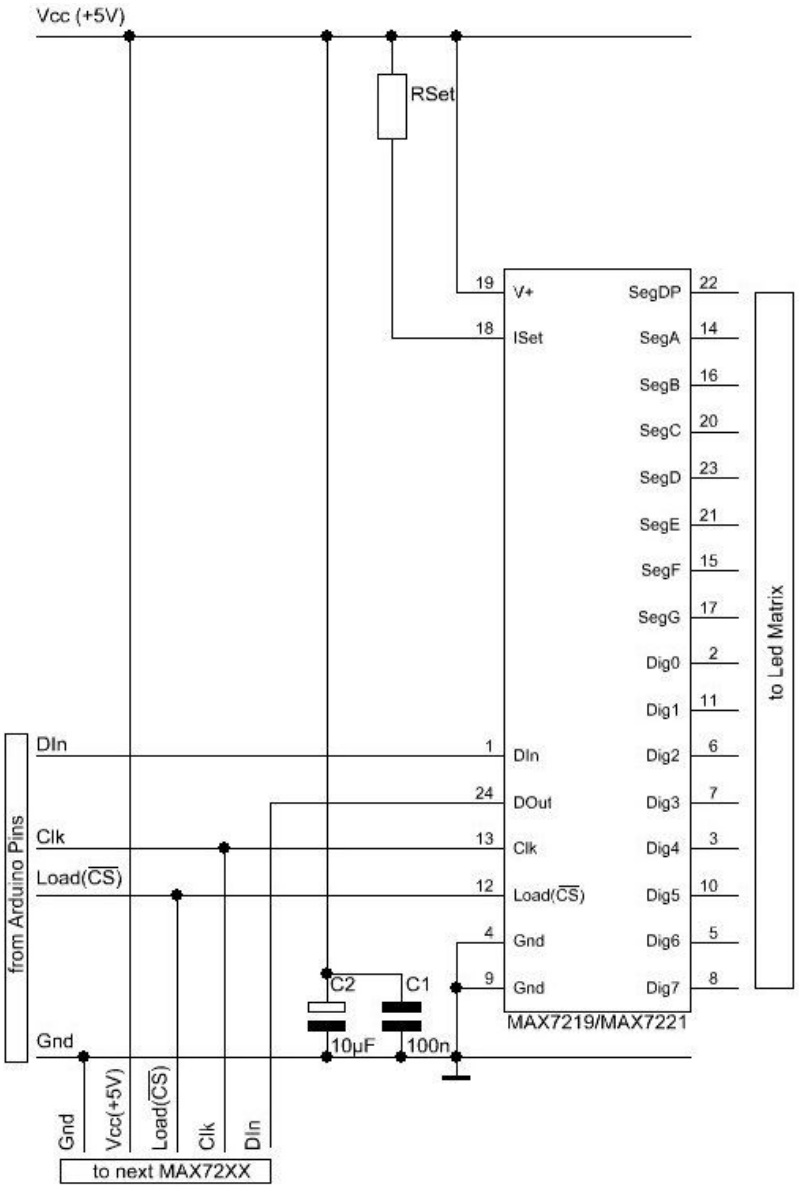
```
/*
 * echo_audio
 *   reads in microphone signal via pin SENSOR_pin_adc and averages adjacent values
 *   input data is 12 bit i.e. 0..1023
 *   output data is 8 bit i.e. 0..255 so sum(new+old)/2 must be divided by 16 for scaling
 *
 * See also analogue input examples
 */

////////////////////////////////////////////////////////////////////////
#include <Wire.h>      // access I2C
#include <LCD.h>       // and LCD
#include <LiquidCrystal_I2C.h>

// Specify LCD geometry and optionally choose "fat" digits
#define nCOLS          20
#define nROWS           4

// Instantiate I2C connected LCD at backpack address 0x27
// - this is like 4-bit mode for standard LiquidCrystal described at
//     https://www.arduino.cc/en/Reference/LiquidCrystalConstructor
//   but with extra arguments defining I2C port address and E pin number
// - the address = base address 0x20 + least significant 3 bits set by user
#define BACKLIGHT_PIN  3
#define En_pin  2
#define Rw_pin  1               // Note that these pin numbers refer
#define Rs_pin  0               // to connections between the LCD
#define D4_pin  4               // backpack PCF8574 chip and the LCD
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7

LiquidCrystal_I2C  lcd(0x27,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
////////////////////////////////////////////////////////////////////////

// Define signal names in terms of standard Arduino board pin numbers
#define LED_pin         13      // pin 13
#define SENSOR_pin_adc  A0          // adc0

// Define array for sensor values using smallest possible int types
#define nHistory 16             // number of samples to keep for processing
static uint16_t sensorValues[nHistory];
static uint8_t  nexts = 0;          // note where next sample is stored
static uint8_t  lasts = nHistory-1;
```

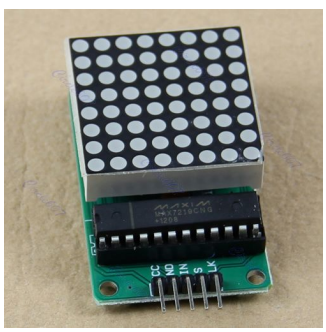
```
// initialise IO ports
void setup()
{
  // initialize digital LED_pin as an output
  pinMode(LED_pin, OUTPUT);

  lcd.begin(nCOLS,nROWS);                    // activate LCD module
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);
  delay(100);

  // clear sample buffer
  for ( int i=0; i<nHistory; i++ )
    sensorValues[i] = 0;
}

// main loop
// - average input data
// - display sensor data on LCD
void loop()
{
  // read sensor and smooth sensorValue
  sensorValues[nexts] = analogRead(SENSOR_pin_adc);
  int average_sv = (sensorValues[nexts] + sensorValues[lasts])/2;

  lasts = nexts;                             // update cyclic indices
  if ( ++nexts >= nHistory ) nexts = 0;

  lcd.setCursor( 0,0 ); lcd.print( average_sv );
}
```

3 LED Array

The ebay vendor page is http://www.ebay.com.au/itm/190867212840
Features:

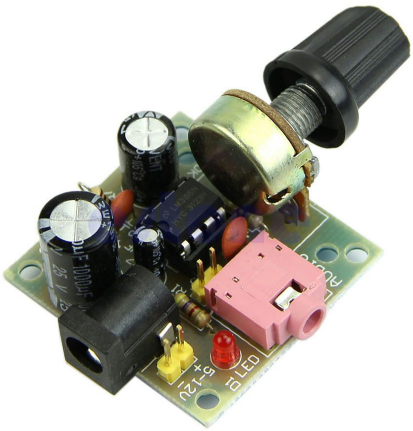
1. Driver chip: MAX7219 (http://playground.arduino.cc/Main/MAX72XXHardware)
2. Module has SPI ports that support cascading. Multiple modules cascade when Dout from one unit feeds into the Din of the next.

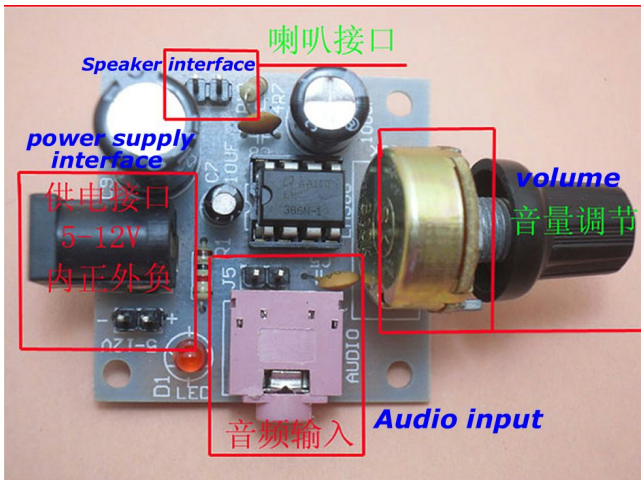




4 Audio amplifier

The ebay vendor page is http://www.ebay.com.au/itm/400814032036





(basic LM386 circuit, not yet tested to see if 5Vdc operation is realistic)