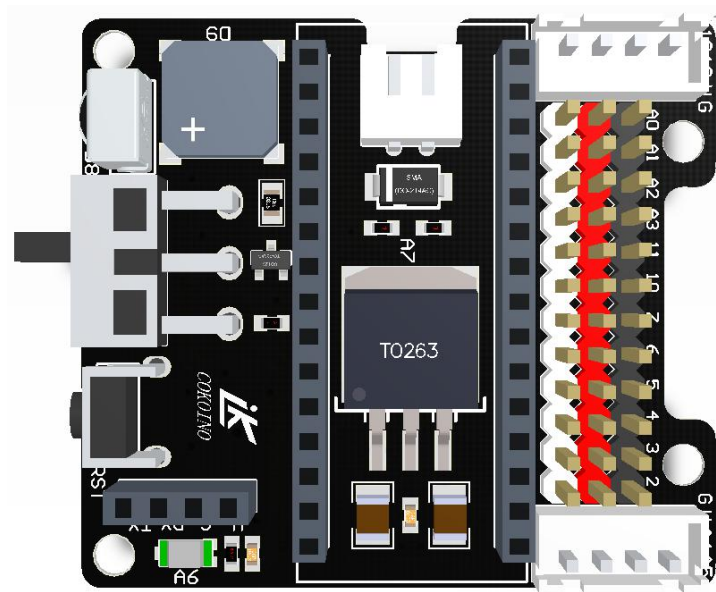


LK COKOINO NANO Shield Tutorial



I Overview

LK COKOINO NANO Shield is a high-current output board with integrated DC5V/5A. It solves the problem of weak drive capability of nano motherboard. The nano motherboard is directly inserted on the expansion board to drive more peripheral modules for more functions. Such as driving the steering gear to DIY robots, robots, etc.

II Specification

Compatible with: arduino

Maximum voltage input range: DC6.5---15V

Recommended voltage input range: DC7---12V

Maximum output current: $6.5V \leq V_{IN} \leq 10V/5A$

Pin header specification: 2.54mm pitch

Connector Specifications: XH2.54mm 4P

Infrared receiving specification: IRM-3638T f=38KHz $\lambda=940nm$ d=5.5m

Serial port mother: 2.54mm pitch, can be directly inserted into hc-06 Bluetooth

III Onboard buzzer specifications

Model: MLT-8530 (voltage type)

Operating voltage: 4-6.5V (rated: 5V)

Current: 90MA (max)

Sound (10CM): 86dB

Frequency: 2700Hz (31+/-3R)

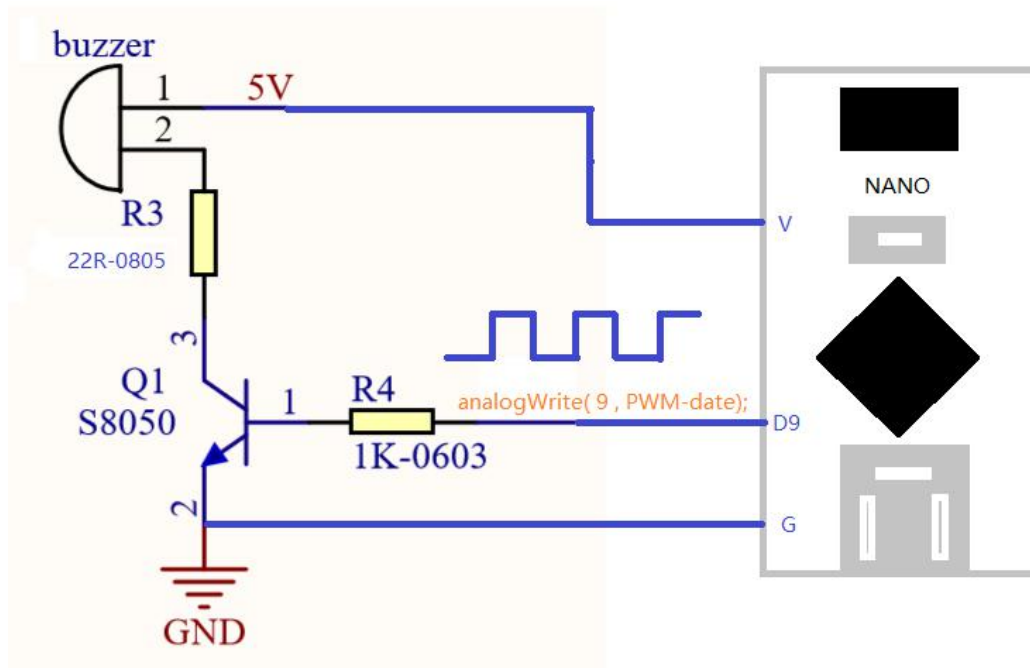
Operating ambient temperature: -20 to +60 degrees Celsius

1. Working principle

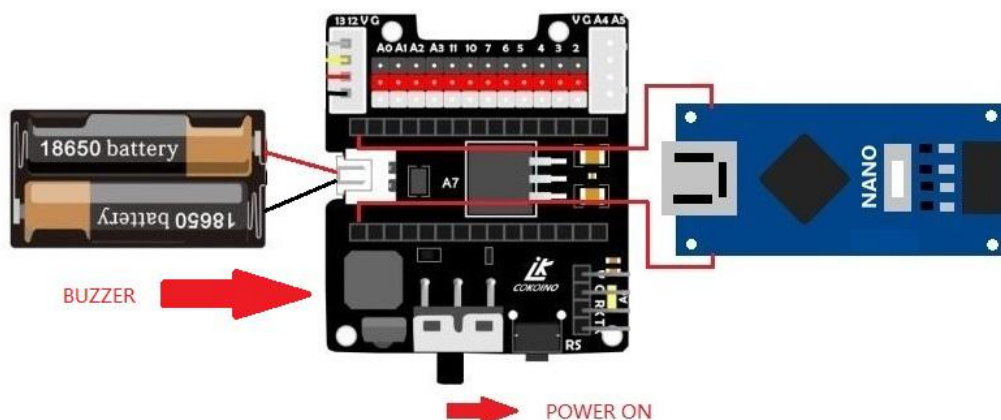
The piezoelectric buzzer is a piezoelectric ceramic piece that is pressed by a high pressure and adhered to a vibrating metal piece.

When an alternating voltage is applied across the ceramic piece and the metal piece, they generate mechanical deformation-stretching and contraction due to the piezoelectric effect, causing the metal piece to vibrate and make a sound.

2. Drive circuit



3. Wiring diagram



4. Upload Sample code

```
int Buzzer_pin = 9;
void setup() {
  pinMode(Buzzer_pin, OUTPUT);
}
void loop() {
  analogWrite(Buzzer_pin,150);
  delay(1000);
  analogWrite(Buzzer_pin,200);
  delay(1000);
  analogWrite(Buzzer_pin,250);
  delay(1000);
}
```

5. Result

Turn on the power switch and the buzzer will sound.

IV Onboard TEMT6000 light sensor

Model: TEMT600

Photosensitive angle: +/- 60 degrees

Operating temperature: -40 to +85 degrees Celsius

Vceo: 6V

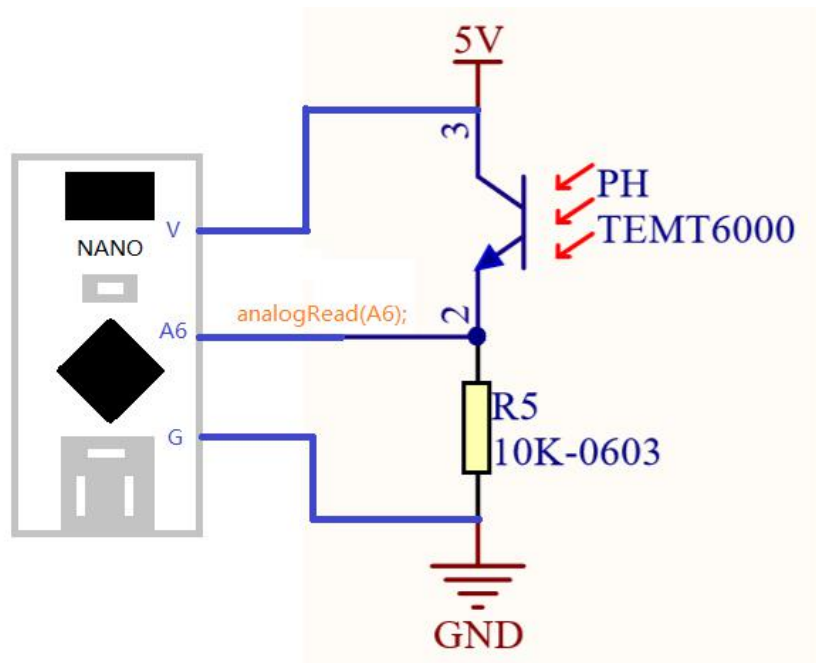
Vcco: 1.5V

Ic: 20mA (max)

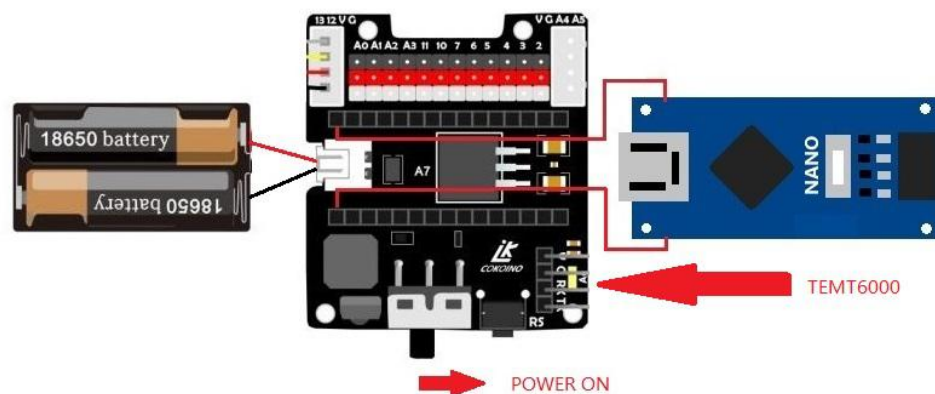
1. Description

TEMT6000 is a silicon NPN epitaxial planar photo-transistor in a miniature transparent mold for surface mounting onto a printed circuit board. The device is sensitive to the visible spectrum.

2. Working circuit



3. Wiring diagram

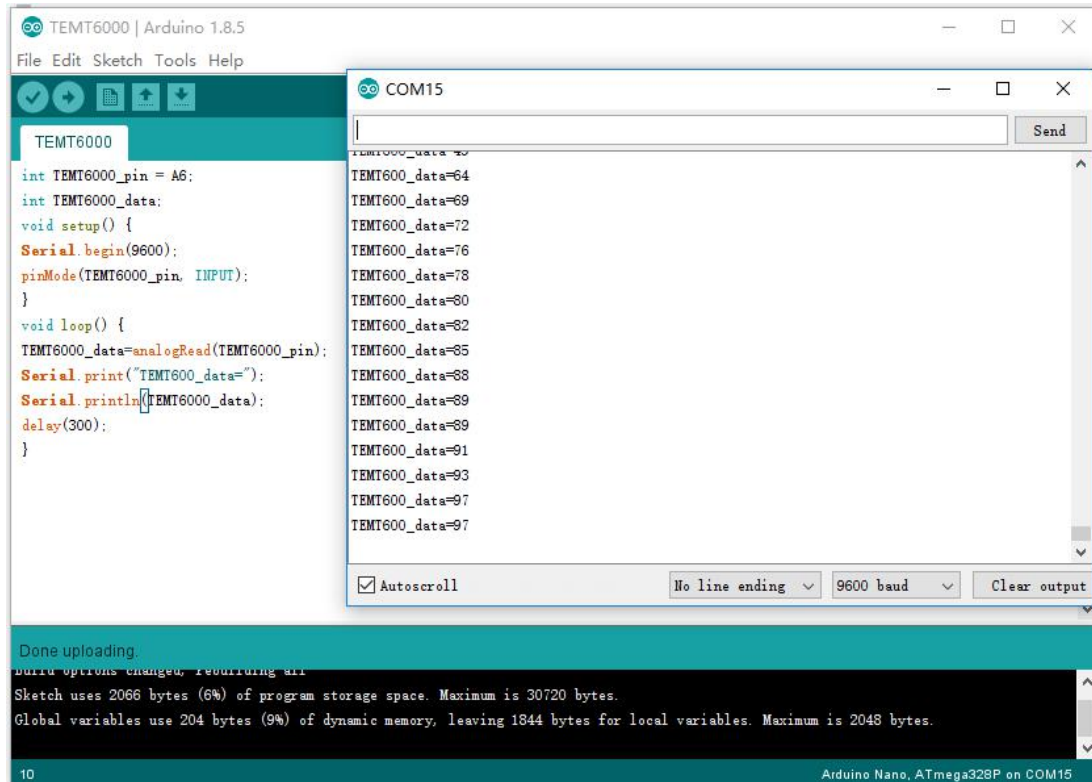


4. Upload sample code

```
int TEMT6000_pin = A6;
int TEMT6000_data;
void setup() {
  Serial.begin(9600);
  pinMode(TEMT6000_pin, INPUT);
}
void loop() {
  TEMT6000_data=analogRead(TEMT6000_pin);
  Serial.print("TEMT600_data=");
  Serial.println(TEMT6000_data);
  delay(300);
}
```

5. Result

Turn on the power switch. When the TEMT6000 sensor is blocked by hand, the value printed by the serial monitor will become smaller. If the light is applied to the TEMT6000 sensor, the value printed by the serial monitor will become larger, as shown below:



The screenshot displays the Arduino IDE interface. The main window shows the sketch for the TEMT6000 sensor. The code defines a pin (A6) and a variable (data) to read the sensor's output. The setup function initializes the serial port at 9600 baud and sets the pin mode. The loop function reads the sensor data and prints it to the serial monitor every 300 milliseconds.

```
int TEMT6000_pin = A6;
int TEMT6000_data;
void setup() {
  Serial.begin(9600);
  pinMode(TEMT6000_pin, INPUT);
}
void loop() {
  TEMT6000_data = analogRead(TEMT6000_pin);
  Serial.print("TEMT6000_data=");
  Serial.println(TEMT6000_data);
  delay(300);
}
```

The serial monitor (COM15) shows the output of the sketch, displaying the sensor data values. The values range from 64 to 97, with a slight increase in the final two readings.

Done uploading.
Arduino options changed, reuploading all.
Sketch uses 2066 bytes (6%) of program storage space. Maximum is 30720 bytes.
Global variables use 204 bytes (9%) of dynamic memory, leaving 1844 bytes for local variables. Maximum is 2048 bytes.

10 Arduino Nano, ATmega328P on COM15

V Infrared remote

1. Onboard infrared receiver module

The infrared receiver is mainly composed of the infrared receiving head, which is a device integrating reception, amplification and demodulation. Its internal IC has been completed to understand the modulation, and the output is a digital signal. Can be used in audiovisual equipment, home appliances, etc.



specification parameter

Working voltage: 2.7-5.5V (DC)

Output signal: digital signal

Working current: 0.8mA

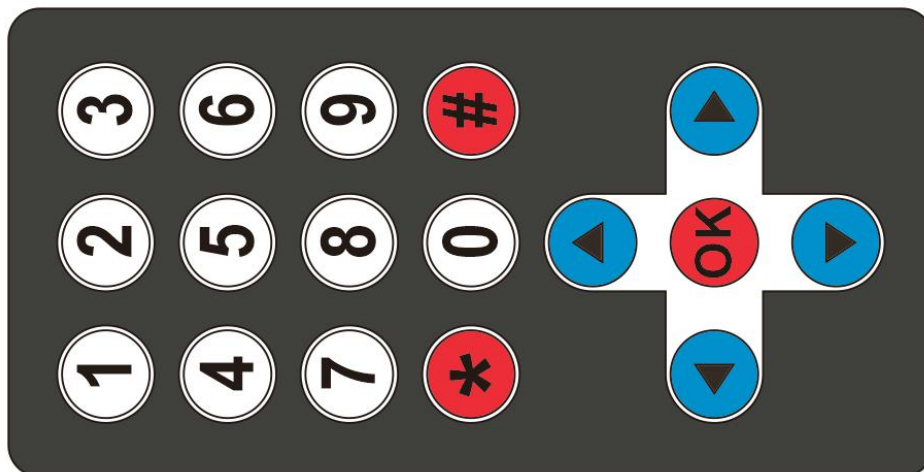
Carrier frequency: 38KHz

Receiving angle: $\pm 35^\circ$

Receiving distance: 5-10M

2. Infrared transmitter

send the key value to the infrared receiver via NEC infrared wireless transmission protocol.



specification parameter

Working voltage: 3V

Protocol: NEC protocol

Transmitter distance: 5-10M

Key values for the Arduino code

0	1	2	3	4	5	6	7	8
FF9867	FFA25D	FF629D	FFE21D	FF22DD	FF02FD	FFC23D	FFE01F	FFA857
9	*	#	▲	▼	◀	▶	OK	
FF906F	FF6897	FFB04F	FF18E7	FF4AB5	FF10EF	FF5AA5	FF38C7	

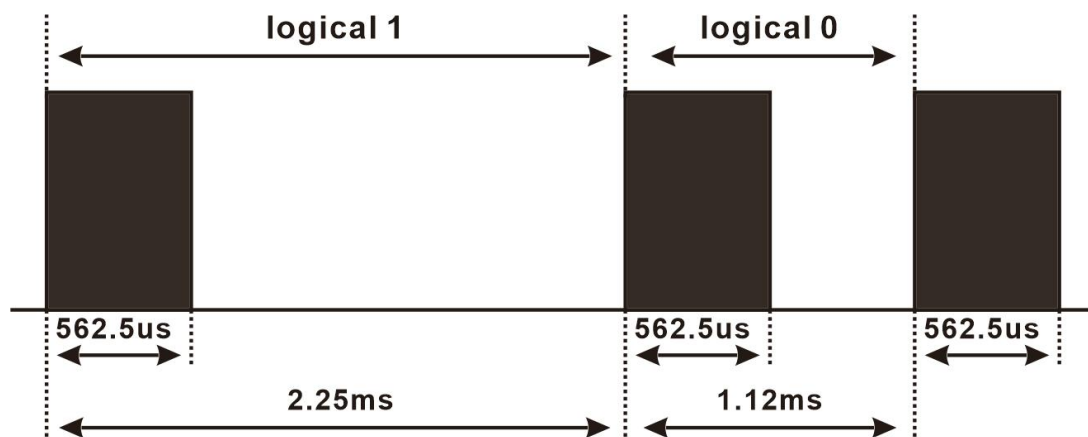
Note: Use the arduino IIRemote library to get the above values.

3. NEC Protocol

The NEC IR transmission protocol uses pulse distance encoding of the message bits. Each pulse burst (mark – RC transmitter ON) is 562.5µs in length, at a carrier frequency of 38kHz (26.3µs). Logical bits are transmitted as follows:

Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms

Logical '1' – a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms



When transmitting or receiving remote control codes using the NEC IR transmission protocol, the WB_IRRC performs optimally when the carrier frequency (used for modulation/demodulation) is set to 38.222kHz.

When a key is pressed on the remote controller, the message transmitted consists of the following, in order:

- a 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
- a 4.5ms space
- the 8-bit address for the receiving device
- the 8-bit logical inverse of the address
- the 8-bit command
- the 8-bit logical inverse of the command
- a final 562.5µs pulse burst to signify the end of message transmission.

The four bytes of data bits are each sent least significant bit first. Figure 1 illustrates the format of an NEC IR transmission frame, for an address of 00h (00000000b) and a command of ADh (10101101b).

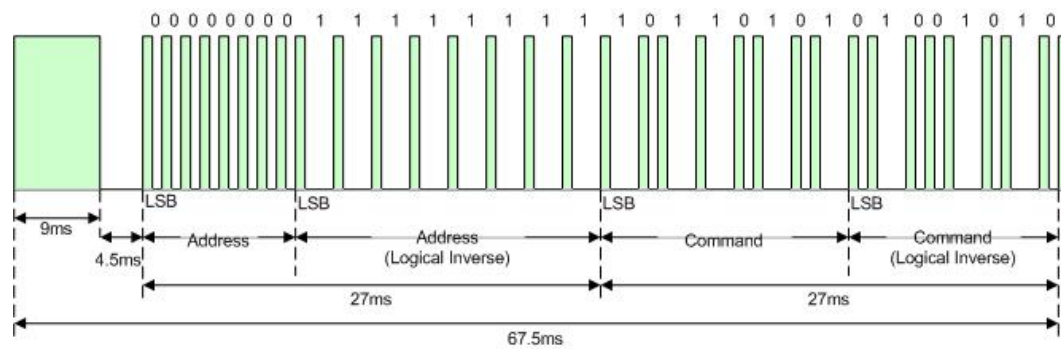


Figure 1. Example message frame using the NEC IR transmission protocol.

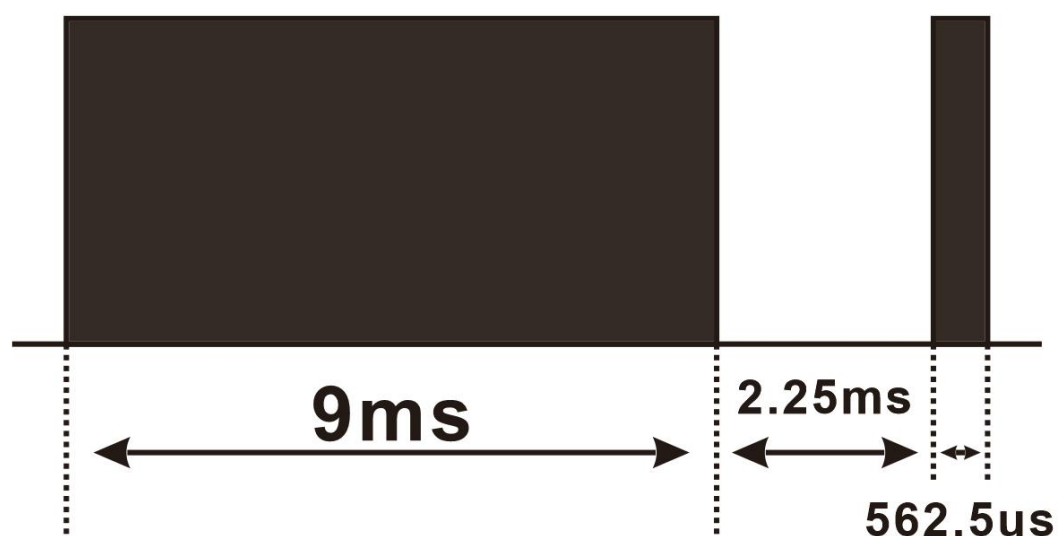
Notice from Figure 1 that it takes:

- 27ms to transmit both the 16 bits for the address (address + inverse) and the 16 bits for the command (command + inverse). This comes from each of the 16 bit blocks ultimately containing eight '0's and eight '1's - giving $(8 * 1.125\text{ms}) + (8 * 2.25\text{ms})$.
- 67.5ms to fully transmit the message frame (discounting the final 562.5μs pulse burst that signifies the end of message).

REPEAT CODES

If the key on the remote controller is kept depressed, a repeat code will be issued, typically around 40ms after the pulse burst that signified the end of the message. A repeat code will continue to be sent out at 108ms intervals, until the key is finally released. The repeat code consists of the following, in order:

- a 9ms leading pulse burst
- a 2.25ms space
- a 562.5μs pulse burst to mark the end of the space (and hence end of the transmitted repeat code).



Initial Message

Repeat Code

Repeat Code

67.5ms

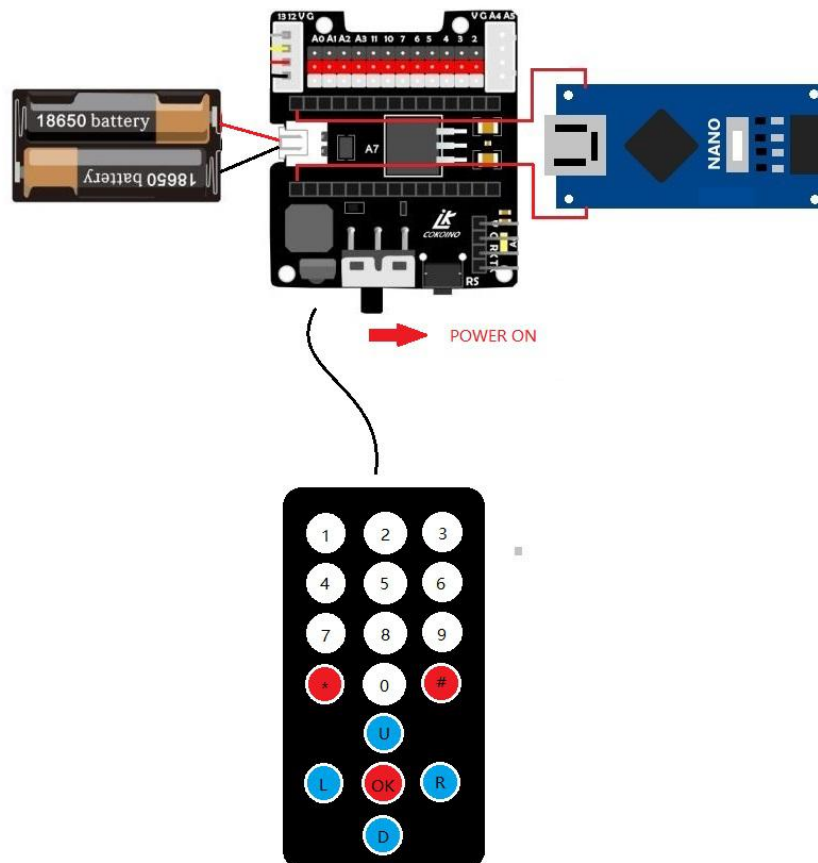
11.8125ms

11.8125ms

108ms

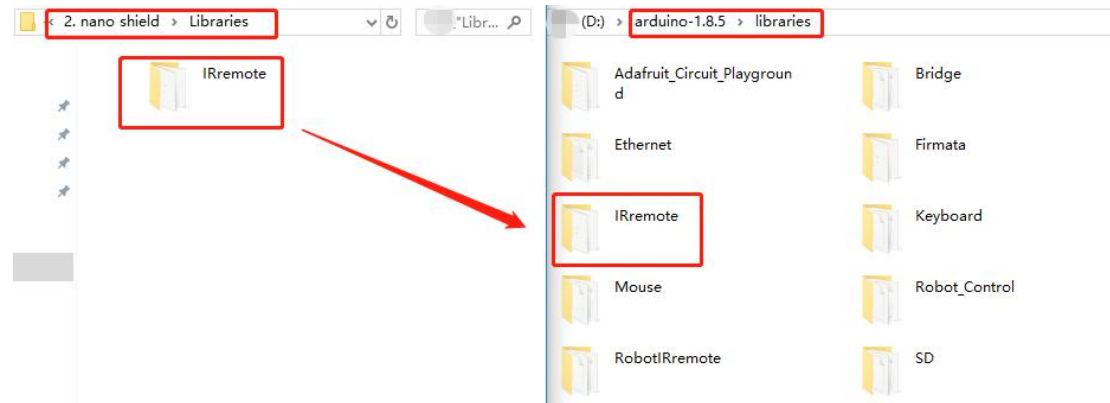
108ms

4. How to use infrared transmitter and receiver



6. Install the infrared library files

Our code uses the arduino library file "IRremote". Before running the code, the "IRremote" folder under the Libraries folder provided by us must be copied to the arduino installation directory. In the "libraries" file, as shown below:



7. Upload code

```
#include <IRremote.h>
int RECV_PIN = 8;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}
void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

8. Result

Open the arduino IDE serial monitor and send the key value with an infrared remote control that can match the infrared receiver. If the key value is received, the serial monitor will print it, as shown below:

