

Lesson 3 - LK COKOINO NANO Shield

Need to prepare:

- ◆ A LK COKOINO Shield
- ◆ A LK COKOINO Nano Board
- ◆ A USB Cable
- ◆ A Battery Case and Two 18650 batteries(Batteries are not included in the kit)



Table of Content

1. Overview	2
2. Specification	2
3. Onboard buzzer of the nano board	3
3.11 Working principle	4
3.12 Drive circuit	5
3.13 Wiring diagram	5
3.15 Result	6
4. Onboard infrared receiver module	6
4.1 Receiving angle	7
4.3 NEC Infrared Transmission Protocol	8
4.4 Wiring diagram	10
4.5 upload code	10
4.6 Result	11

1. Overview

LK COKOINO NANO Shield is a high-current output board with integrated DC5V/5A. It solves the problem of the weak drive capability of the nano board. The nano board can be directly inserted into the shield to drive more peripheral devices for more functions, such as driving the servos, small fans, robots, etc.

2. Specification

Compatible with: arduino

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

Recommended voltage input range: DC7---12V

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

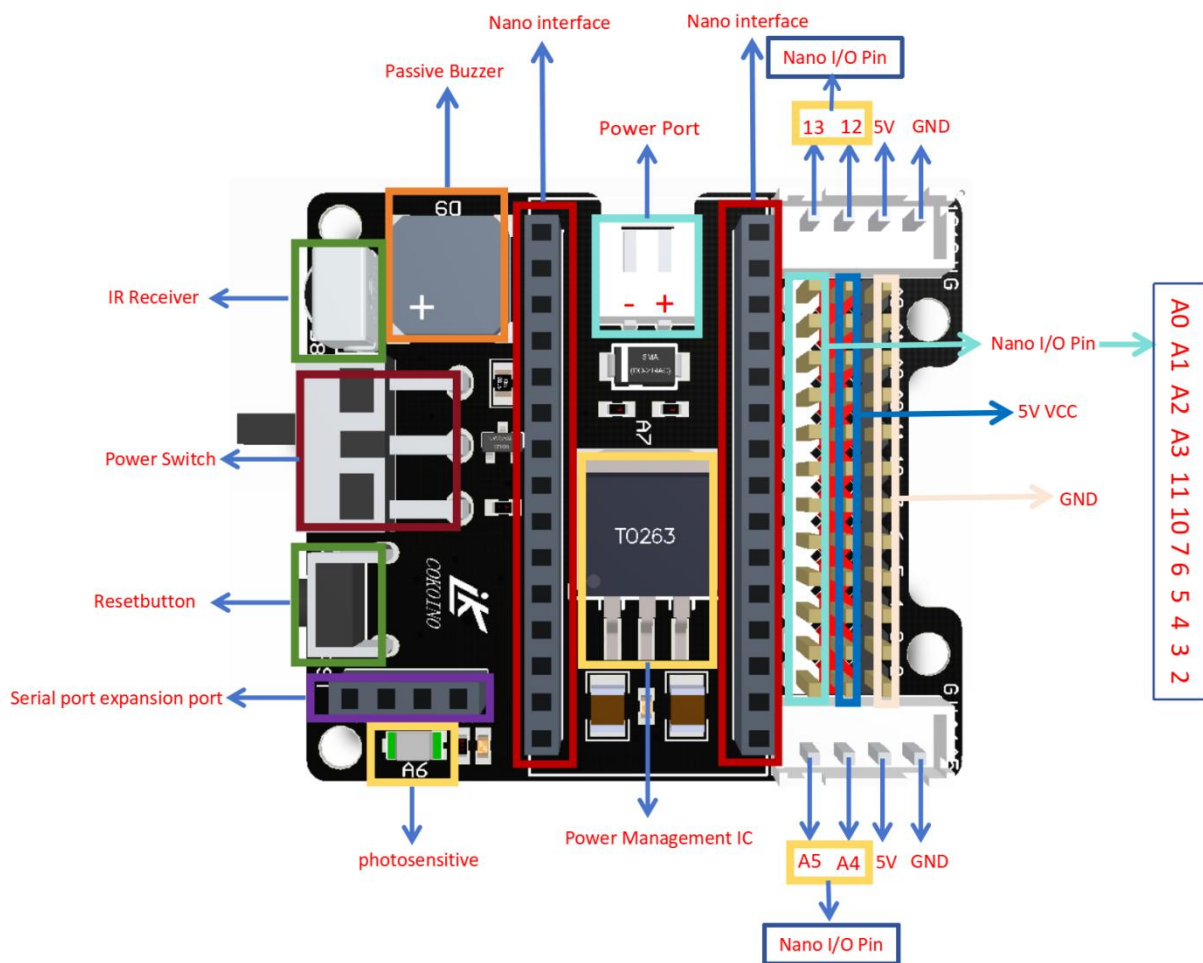
Pin header specification: 2.54mm pitch

Connector Specifications: XH2.54mm 4P

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

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

For the main components and functions of the Robotic Arm Driver Board, please refer to the following table.



IR Receiver	Infrared receiver, receiving infrared signals,Occupying Nano's D8 signal pin
Power Switch	Dial to ON to power on, dial to OFF to power off
Reset button	The reset button of the system, press it to trigger the reset of the control board (reset the main control IC).
Serial port expansion port	Serial port: 2.54mm pitch, can be directly inserted into hc-06 Bluetooth
photosensitive	Photodiode, sensing the strength of light signals, occupies the A6 signal pin of Nano
Power Management IC	DC to DC 5V voltage regulator system to convert external power supply to DC5V
A5	SCL : IIC clock port (multifunction IO port, common pin with A5)
A4	SDA : IIC data port (multifunction IO port, common pin with A4)
2	D2,The digital IO port of the Nano
3	D3,The digital IO port of the Nano
4	D4,The digital IO port of the Nano
5	D5,The digital IO port of the Nano
6	D6,The digital IO port of the Nano
7	D7,The digital IO port of the Nano
10	D10,The digital IO port of the Nano
11	D11,The digital IO port of the Nano
A2	A2:Analog input port of the Nano, it is also used for digital IO port
A1	A1:Analog input port of the Nano, it is also used for digital IO port
A0	A0:Analog input port of the Nano, it is also used for digital IO port
12	D12,The digital IO port of the Nano
13	D13,The digital IO port of the Nano
Nano interface	Insert the Nano board and match the pins of the Nano board one by one
Power Port	DC Power Port,power supply range: 7-12V DC.
Passive buzzer	Passive buzzer,Can directly drive high-level operation and sound,Occupying Nano's D9 signal pin

3. Onboard buzzer of the nano board

Model: MLT-8530 (voltage type)
Operating voltage: 4-6.5V (rated: 5V)
Current: 90MA (max)
Sound (10CM): 86dB

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

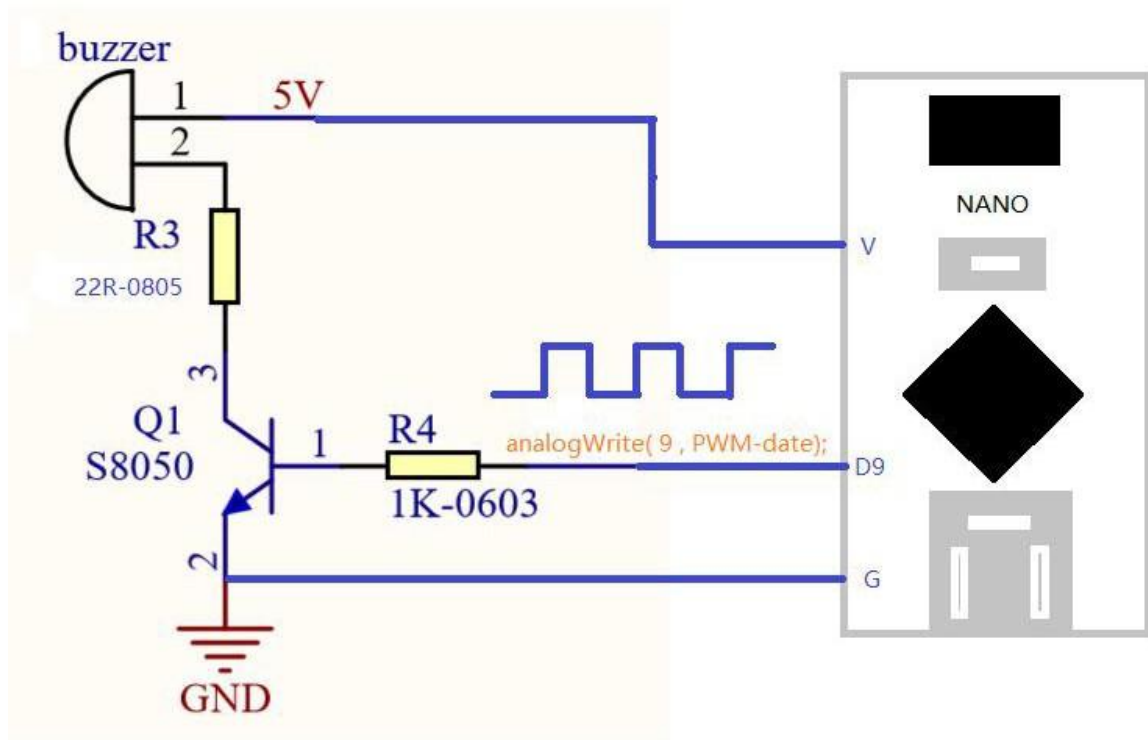
Operating ambient temperature: -20 to +60 degrees Celsius

3.11 Working principle

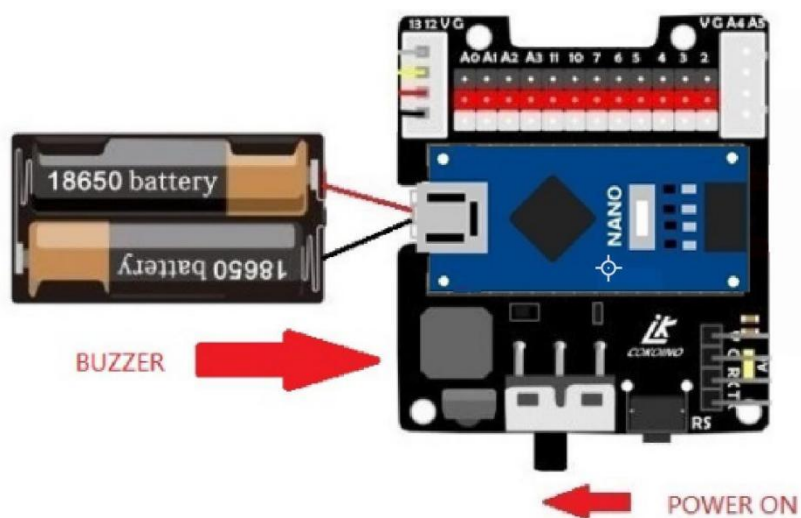
The piezoelectric buzzer is made of a piezoelectric ceramic piece that is pressed by high pressure and adhered to a vibrating metal piece.

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.

3.12 Drive circuit

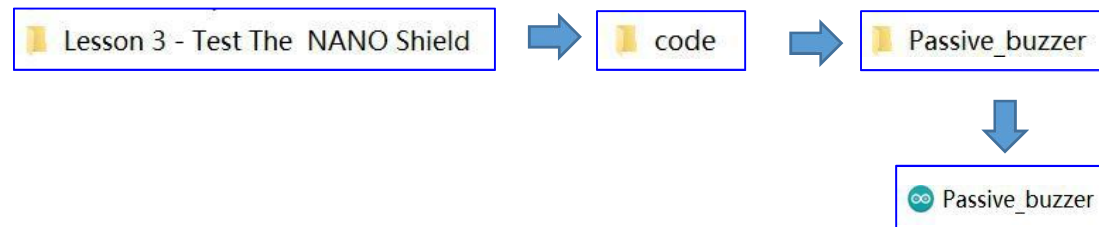


3.13 Wiring diagram



3.13 Upload Sample code

Find the "[Passive_buzzer](#)" code from the following path, and open it with the Arduino IDE and upload it.



3.15 Result

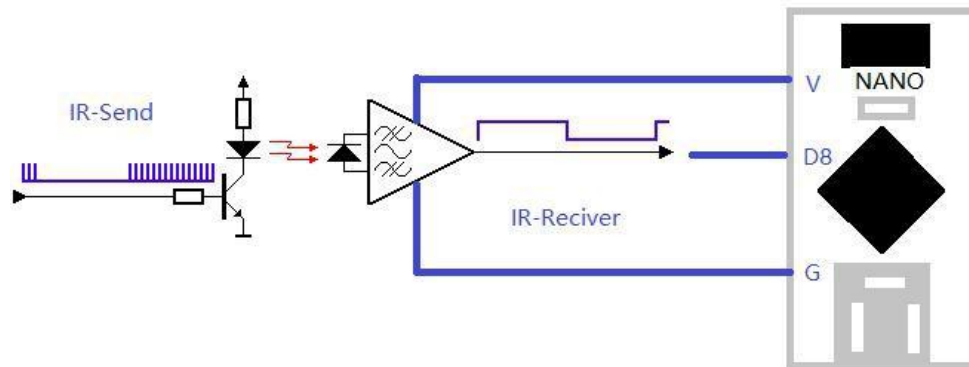
Turn on the power switch and the buzzer will sound.

4. Onboard infrared receiver module

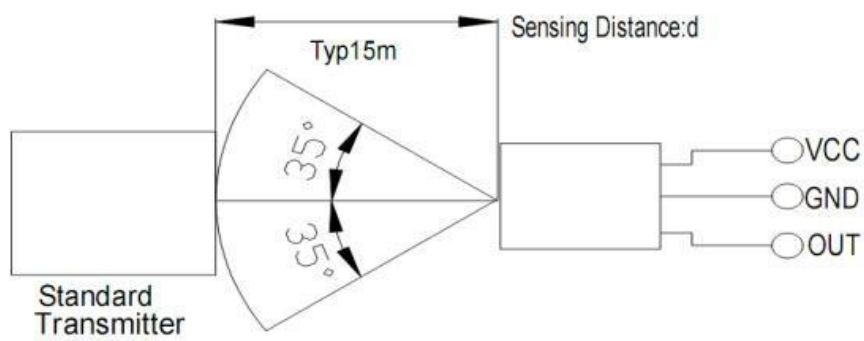
The visible light that can be seen by the human eye is arranged in wavelength from long to short, followed by red, orange, yellow, green, cyan, blue, and purple. The wavelength range of red light is 0.62 to 0.76 μm ; the wavelength range of violet light is 0.38 to 0.46 μm . Light that is shorter than the wavelength of violet light is called ultraviolet light, and light that is longer than the wavelength of red light is called infrared light. Infrared remote control uses a near infrared ray with a wavelength between 0.76 and 1.5 μm to transmit control signals.

The infrared remote control is a control method for transmitting information by using infrared rays. The infrared remote control has the advantages of anti-interference, simple circuit, easy encoding and decoding, low power consumption and low cost. Infrared remote control is suitable for almost all home appliances.

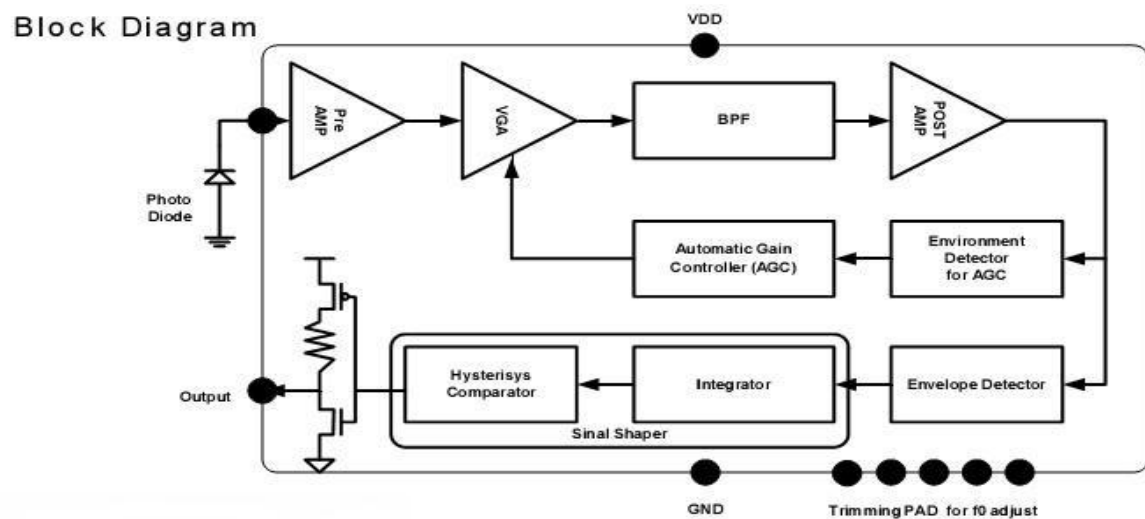
The main part of the infrared remote control system is modulation, transmission and reception, as shown in the figure :



4.1 Receiving angle



4.2 Internal principle



4.3 NEC Infrared Transmission Protocol

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

- ① Logical '0' – a 562.5 μ s pulse burst followed by a 562.5 μ s space, with a total transmit time of 1.125ms
- ② Logical '1' – a 562.5 μ s 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 562.5 μ s 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).

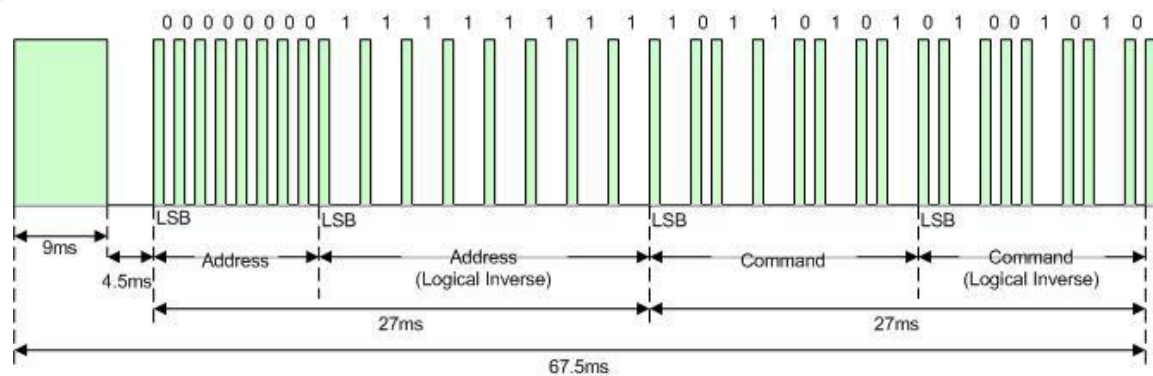


Figure 1. Example message frame using the NEC IR transmission protocol.

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).

Figure 2 illustrates the transmission of two repeat codes after an initial message frame is sent.

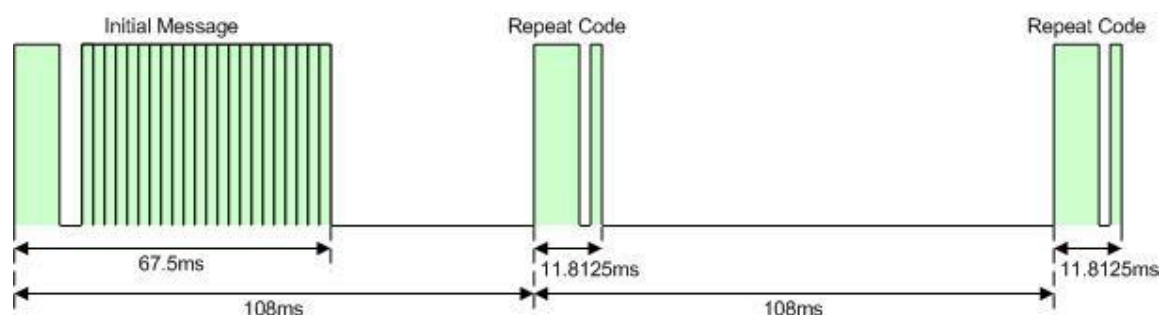
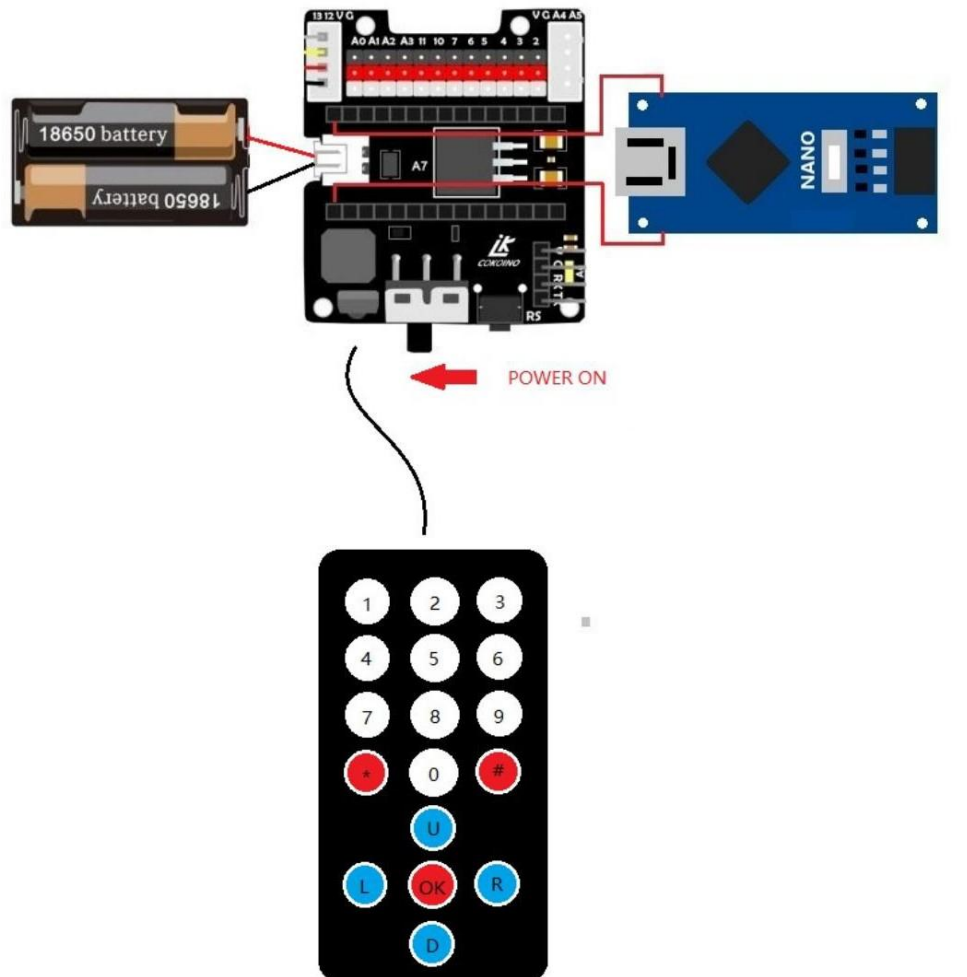


Figure 2. Example repeat codes sent for a key held down on the transmitting remote controller.

<https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>

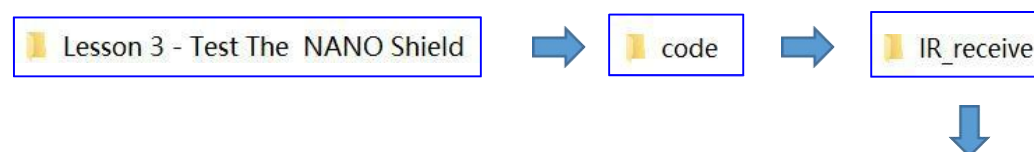
4.4 Wiring diagram

Note: The infrared remote control is not included in the kit, we only provide a test solution.



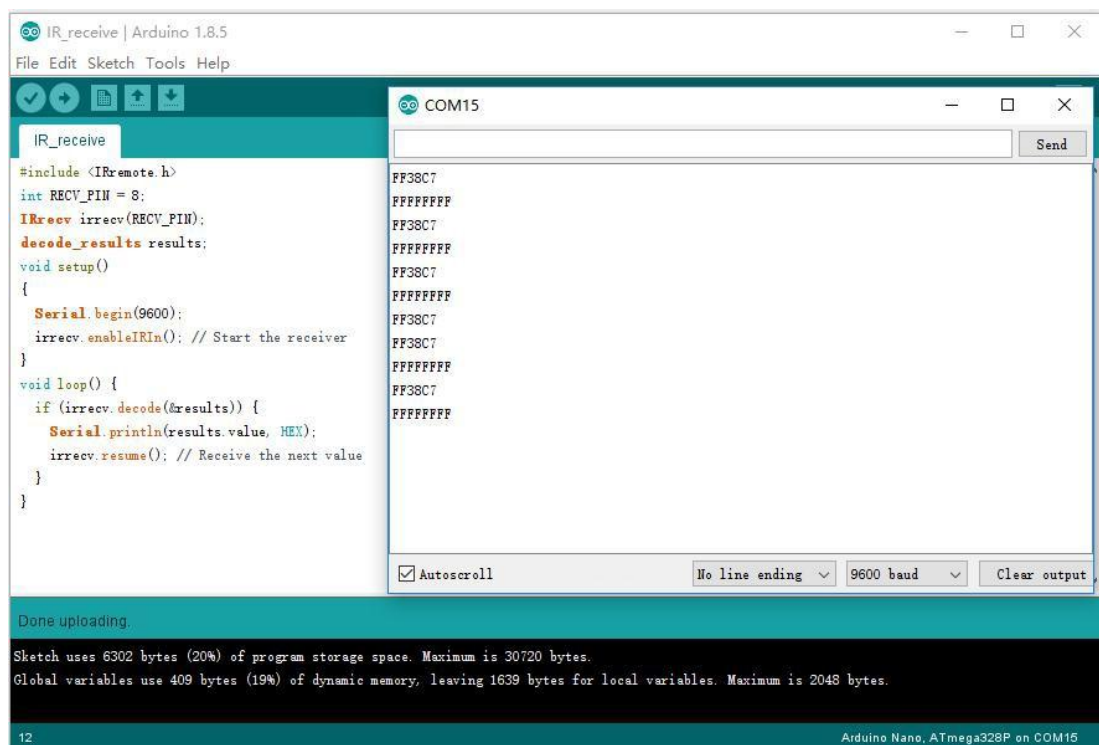
4.5 upload code

Find the "[IR_receive](#)" code from the following path, and open it with the Arduino IDE and upload it.



4.6 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:



The screenshot displays the Arduino IDE interface. The left pane shows the sketch 'IR_receive' with the following 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
  }
}
```

The right pane shows the serial monitor for COM15, displaying the received hexadecimal values:

```
FF38C7
FFFFFFFF
FF38C7
FFFFFFFF
FF38C7
FFFFFFFF
FF38C7
FFFFFFFF
FF38C7
FFFFFFFF
```

At the bottom, a status bar indicates 'Done uploading.' and provides memory usage details: 'Sketch uses 6302 bytes (20%) of program storage space. Maximum is 30720 bytes. Global variables use 409 bytes (19%) of dynamic memory, leaving 1639 bytes for local variables. Maximum is 2048 bytes.'