QUICK REFERENCE · TK16

IR Sender

SIGNAL HIGH lights the infrared LED.

VCC: 5 V at most Nothing but a 150 Ω resistor limits the LED's current. **Feed VCC from 5V or 3V3, never a 9 V or 12 V rail**: near 9 V the LED reaches its 50 mA limit.

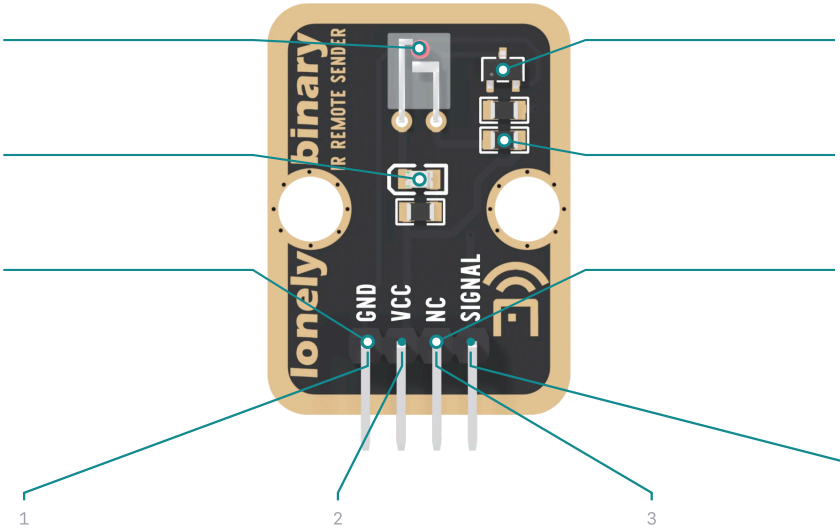
Pins, and which to wire

Parts up, header at the bottom. Wire GND, VCC and SIGNAL; NC is connected to nothing on the board.

IR908-7C infrared LED
940 nm, lens facing up. Beam ±20°: aim it.

Red LED
Lights whenever the infrared LED does.

GND is the square pad
Count from it. The back prints no pin names.



S8050 transistor
SIGNAL only works the switch; VCC feeds the LED.

150 Ω sets the current
About 24 mA from 5 V, 13 mA from 3.3 V.

NC
Connected to nothing on the board.

Front	GND	VCC	NC	SIGNAL
Send	GND	VCC	●	SIGNAL

HIGH = LED on The sender is active high, the opposite of the receiver. Your board makes the 38 kHz carrier; the TK16 is an LED and a switch.

What VCC buys

VCC on 5V

24 mA in the LED

The full range. VCC never reaches SIGNAL, so 5V is safe beside a 3.3 V board.

VCC on 3V3

13 mA in the LED

About 73 per cent of the 5 V reach. Fine across a desk.

From your pin

2.6 mA from 3.3 V

Into the transistor's base through 1 kΩ: many times what it needs.

The 150 Ω resistor is the only limit, so VCC decides the current, and reach follows its square root: 5 V goes about a third further than 3.3 V, for moving one wire.

From Everlight's datasheet DIR-0000453: 940 nm, 1.25 V at 20 mA, 50 mA continuous, 40° full angle. Currents are worked out with a 0.15 V transistor drop, not measured. The carrier runs at 30 per cent duty, so the average is lower.

Send a code

Arduino IDE · Board: your board · ESP32-S3: USB CDC On Boot: Enabled

```
#include <IRremote.hpp>   // IRremote by shirriff, z3t0, ArminJo

// SIGNAL's pin. Uno: 3. ESP32: 22. ESP32-S3: 6. Pico: 17.
const int IR_TX_PIN = 3;

// REPLACE with what your own remote sent to the receiver.
const uint16_t ADDRESS = 0x04;
const uint8_t COMMAND = 0x16;

void setup() {
  Serial.begin(115200);
  IrSender.begin(IR_TX_PIN);
}

void loop() {
  Serial.flush();        // on an Uno: no interrupt mid-frame
  IrSender.sendNEC(ADDRESS, COMMAND, 0);   // one press
  Serial.println("sent");
  delay(2000);
}
```

```
sent
sent
```

Serial Monitor at 115200. A TK15 running the key-code sketch 30 cm away prints the same address and command.

Wiring

	FRONT	UNO	ESP32	S3	PICO
1	GND	GND	GND	GND	GND
2	VCC	5V	5V	5V	VBUS
3	NC				
4	SIGNAL	D3	GPIO 22	GPIO 6	GP17

Hole 1 is the square pad. VCC to 5V (VBUS on a Pico) for range; 3V3 works, shorter.

Where the carrier comes from

On an Uno IRremote makes the 38 kHz carrier in software on any pin, so let Serial finish first: an interrupt mid-frame puts a gap in the tone. On an ESP32, ESP32-S3 or Pico it uses a hardware PWM channel by default, also on any pin. Either way sendNEC blocks for about 68 ms.

When it does not work

The LED looks dead

940 nm is invisible. Point a phone camera at it, or watch the red LED: it lights with the infrared one.

The other end prints UNKNOWN

The boards are too close: under about 20 cm the receiver is swamped. Try 30 cm.

It only reaches a metre or two

Move VCC from 3V3 to 5V and aim: the beam is only ±20°. Shade the receiver.

The receiver hears nothing

Red LED dark: check SIGNAL's wire and pin, and GND. Flickering: check aim and distance.

A television ignores it

Wrong protocol. Send the one the receiver decoded: sendSamsung for Samsung, sendSony for Sony.

The sketch stalls while sending

sendNEC blocks for the frame, about 68 ms, plus 110 ms per repeat.

Specifications

LED	Everlight IR908-7C, 940 nm	**Beam**	40° in all, ±20°
Driver	S8050 NPN, 1 kΩ to the base	**LED current**	24 mA from 5 V, 13 mA from 3.3 V
Supply	3.3 or 5 V; 5 V for range	**Signal**	Active HIGH
Header	4 pins, right-angle, 2.54 mm	**Size**	22.4 × 30.4 mm