QUICK REFERENCE · TK15 + TK16

IR Remote Kit

A receiver, a sender and a remote: 3 of each.

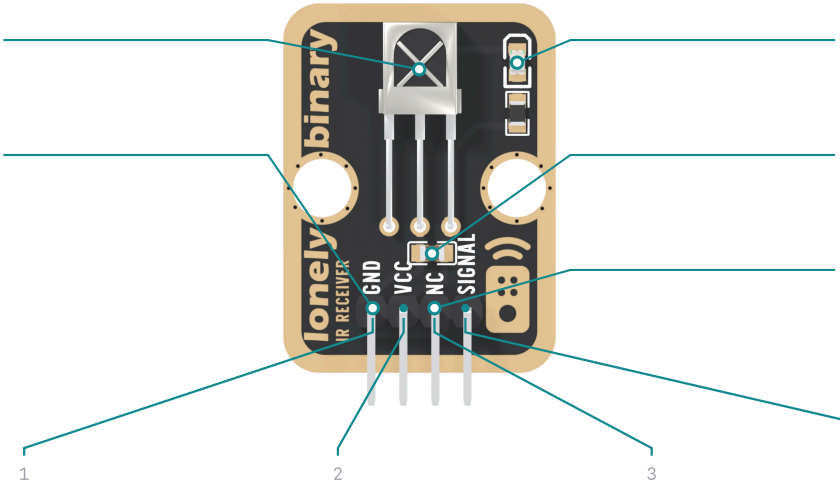
VCC sets the signal SIGNAL idles at VCC through 30 kΩ inside the receiver. **On an ESP32, ESP32-S3 or Pico, feed VCC from 3V3, never 5V**: 5 V would sit on a 3.3 V input all the time.

The receiver: TK15

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

TSOP18138 receiver
Tuned to 38 kHz, in a steel shield. Face it to the remote.

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



Red LED, 1 kΩ
Flickers while a remote sends, before any code.

100 nF across VCC
The only supply filter: keep motors off its rail.

NC
Connected to nothing on the board.

Front	GND	VCC	NC	SIGNAL
Receive	GND	VCC	•	SIGNAL

LOW = a remote is sending The receiver is active low: HIGH at rest, LOW while a 38 kHz burst arrives. No pull-up needed; it is inside the chip.

Two boards, opposite jobs

TK15 receiver

38 kHz, tuned

Sees ±45° off axis. SIGNAL active LOW. VCC 2.0 to 5.5 V, as your board.

TK16 sender

24 mA from 5 V

Fires ±20° off axis. SIGNAL active HIGH. 13 mA from 3.3 V, about 73 per cent of the reach.

The remote

17 buttons, NEC

Two AAA cells each, not included. A held button repeats every 110 ms.

The sender is the narrow one: aim the TK16, not the TK15. Put the sender on 5V, keep the two at least 20 cm apart, and shade the receiver from the sun.

Vishay 82802 and Everlight DIR-0000453. The receiver's 26 m was measured in the dark with an LED at 50 mA; indoors, plan on two to five metres. Currents are worked out, not measured.

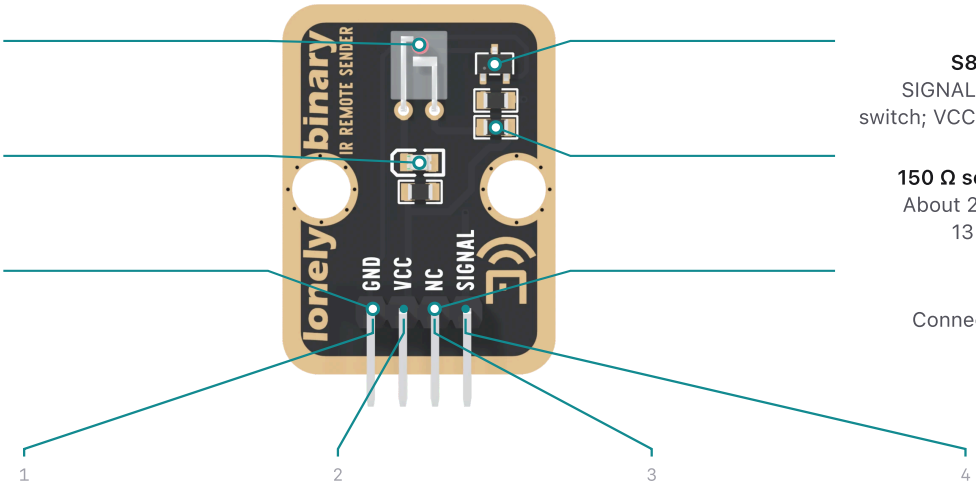
The sender: TK16

Same header, same order. SIGNAL does the opposite job. 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.

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.

Which board is which

	TK15 IR RECEIVER	TK16 IR SENDER
SIGNAL is	an output your board reads	an input your board drives
Active	LOW, while a burst arrives	HIGH, to light the LED
VCC	your board's logic: 5V or 3V3	5V for range; 3V3 works
Beam	±45°	±20°
Red LED	lights while a burst arrives	lights while SIGNAL is HIGH
Library call	IrReceiver.decode()	IrSender.sendNEC()

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.

Test both at once

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

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

// Uno 2, 3. ESP32 23, 22. ESP32-S3 9, 6. Pico 16, 17.
const int IR_RX_PIN = 2;      // TK15 SIGNAL
const int IR_TX_PIN = 3;      // TK16 SIGNAL

void setup() {
  Serial.begin(115200);
  IrSender.begin(IR_TX_PIN);
  IrReceiver.begin(IR_RX_PIN, DISABLE_LED_FEEDBACK);
}

void loop() {
  Serial.flush();
  IrSender.sendNEC(0x12, 0x34, 0);      // a test code
  IrReceiver.restartAfterSend();
  delay(RECORD_GAP_MICROS / 1000 + 5);  // let the frame end
  if (IrReceiver.decode()) {
    IRData &d = IrReceiver.decodedIRData;
    Serial.println(d.command == 0x34 ? "ok" : "other frame");
    IrReceiver.resume();
  } else {
    Serial.println("nothing came back");
  }
  delay(1000);
}
```

```
ok
ok
```

Face the two boards, parts side to parts side, about 30 cm apart.

Wiring

	FRONT	UNO	ESP32	S3	PICO
1	GND	GND	GND	GND	GND
2	VCC · TK15	5V	3V3	3V3	3V3
2	VCC · TK16	5V	5V	5V	VBUS
3	NC				
4	SIGNAL · TK15	D2	GPIO 23	GPIO 9	GP16
4	SIGNAL · TK16	D3	GPIO 22	GPIO 6	GP17

Hole 1 is the square pad. The receiver's VCC is your board's logic voltage, because its SIGNAL idles at VCC; the sender's is 5V (VBUS on a Pico) for range.

Read key codes first

Run the receiver on its own, press every button on the remote, and write down the address and command of each. The sender can only send codes you have read: there is no table to look them up in. A held button repeats every 110 ms; skip repeats for anything that toggles.

When it does not work

Nothing prints at all

Watch the red LED as you press a button. It flickers: check SIGNAL's wire and pin. It does not: check GND, VCC and the remote's batteries.

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.

It only reaches a metre or two

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

It decodes once, then goes silent

IrReceiver.resume() is missing. The library looks at nothing else until it runs.

The other end prints UNKNOWN

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

analogWrite stopped on pins 3 and 11

On an Uno, receiving takes Timer2. Move that output to 5, 6, 9 or 10.

Specifications

TK15 chip	Vishay TSOP18138, 38 kHz	**TK15 VCC**	2.0 to 5.5 V, as your board
TK15 out	Active LOW, 30 kΩ pull-up inside	**TK16 LED**	Everlight IR908-7C, 940 nm, ±20°
TK16 VCC	3.3 or 5 V; 24 mA from 5 V	**TK16 in**	Active HIGH, 1 kΩ to the base
Remote	17 buttons, NEC, 2 × AAA	**Boards**	22.4 × 30.4 mm, 4-pin header