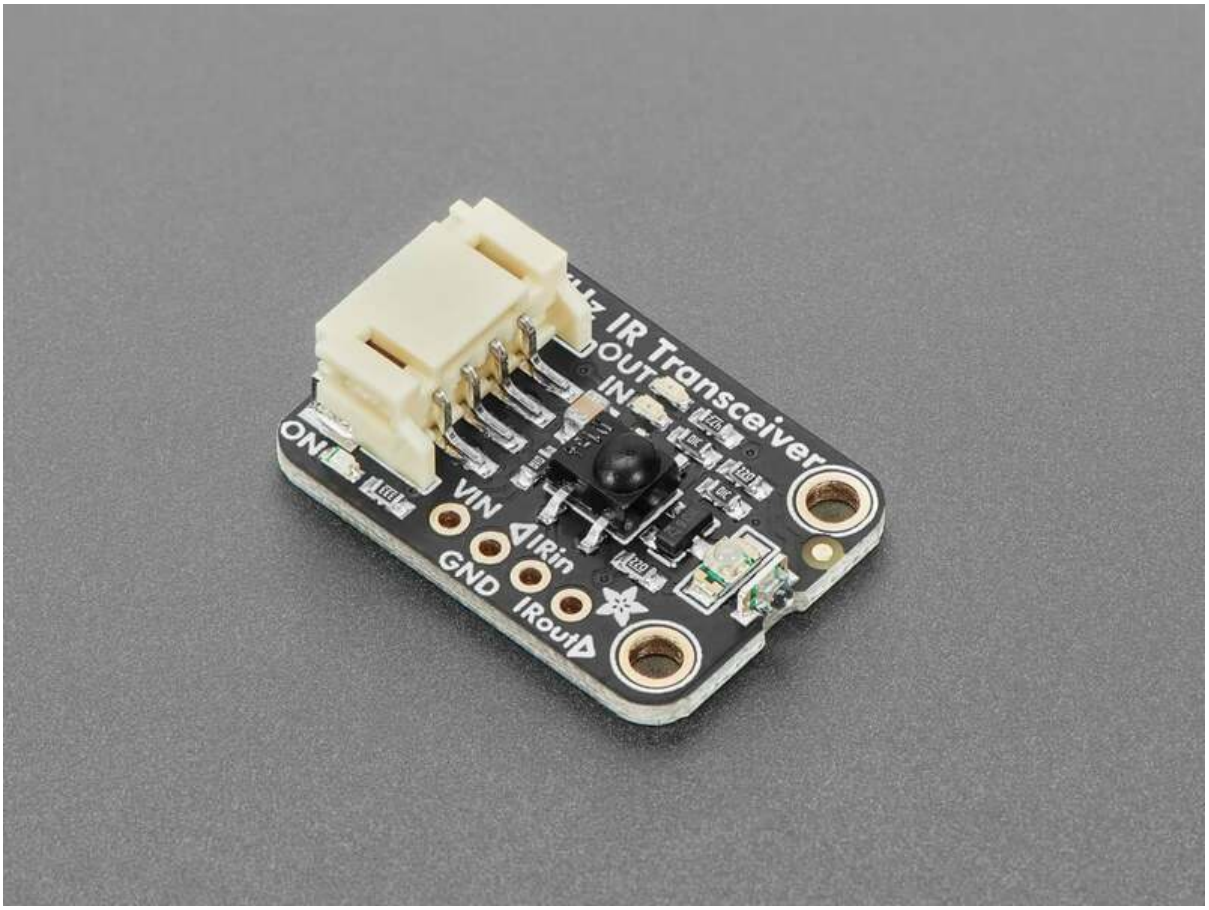
Adafruit Infrared IR Remote Transceiver

Created by Liz Clark

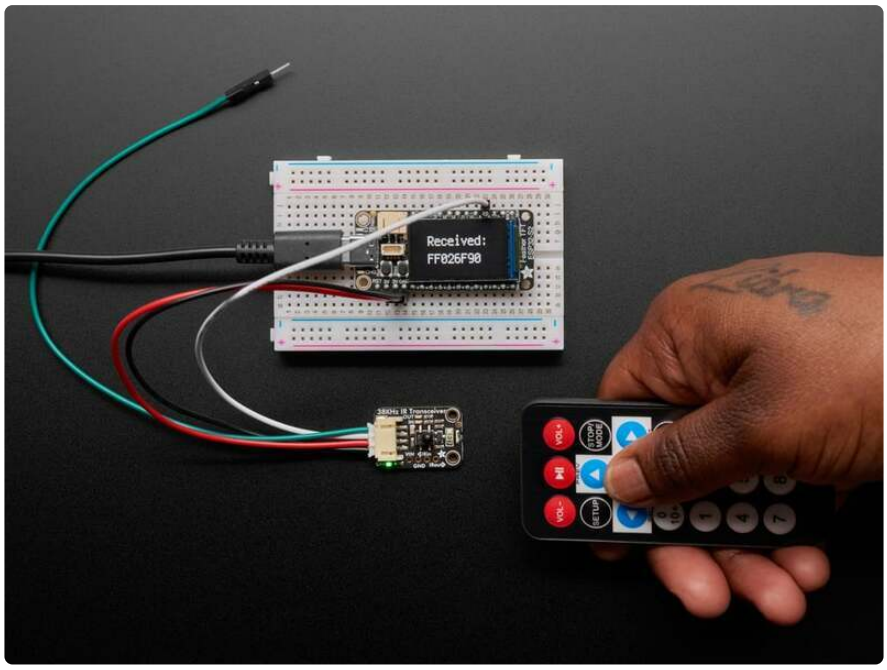


https://learn.adafruit.com/adafruit-infrared-ir-remote-transceiver

Last updated on 2024-07-09 05:26:25 PM EDT

Table of Contents

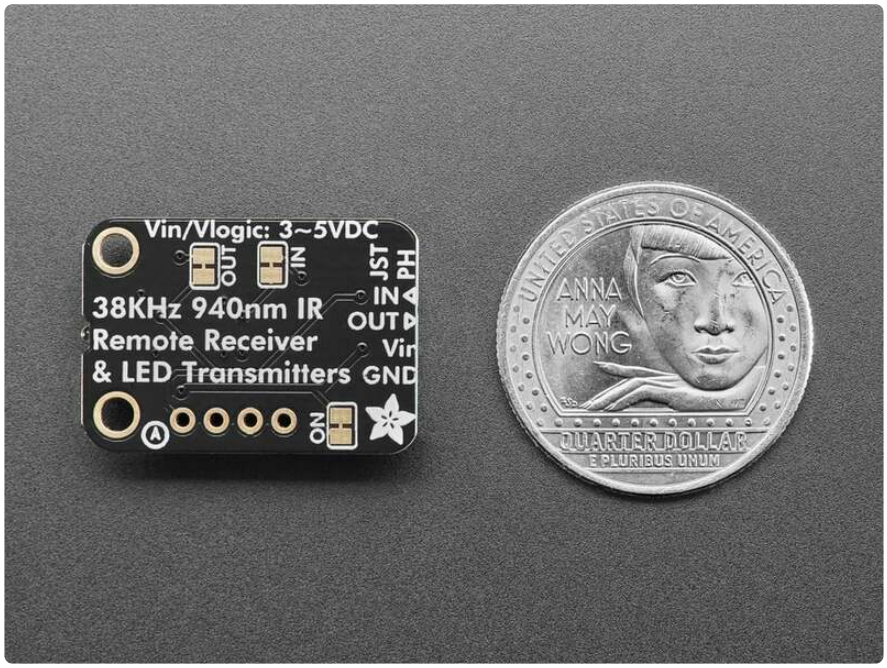
Overview



We've got a high-powered infrared emitter board (http://adafru.it/5639), and a super-sensitive infrared remote receiver board (http://adafru.it/5939) - but why buy two boards when you can pick up one that does both? This board is the **Adafruit Infrared Tranceiver breakout**, which can transmit and receiver 940nm remote control data in one handy solder-free package.

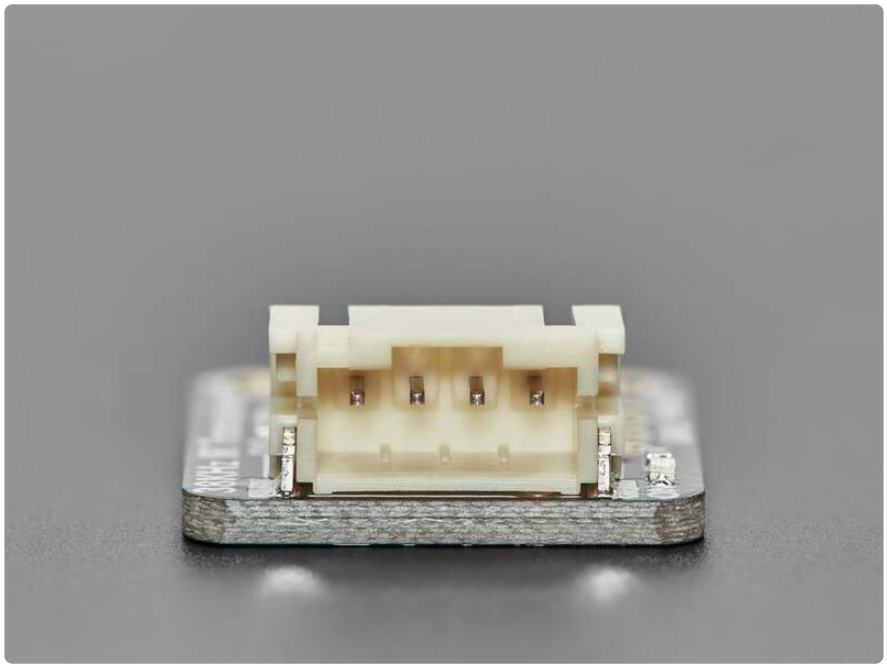
Please Note: to make this board easy to use, we have both solder-able breakout pads and also a JST PH 4-pin cable. We often use JST PH cables for I2C but this board **does not use I2C** - the green wire is data from a microcontroller to be emitted over

the IR LED, the white wire is demodulated IR remote signal data to a microcontroller. The black wire is ground, red wire is 3V~5V power and logic level.

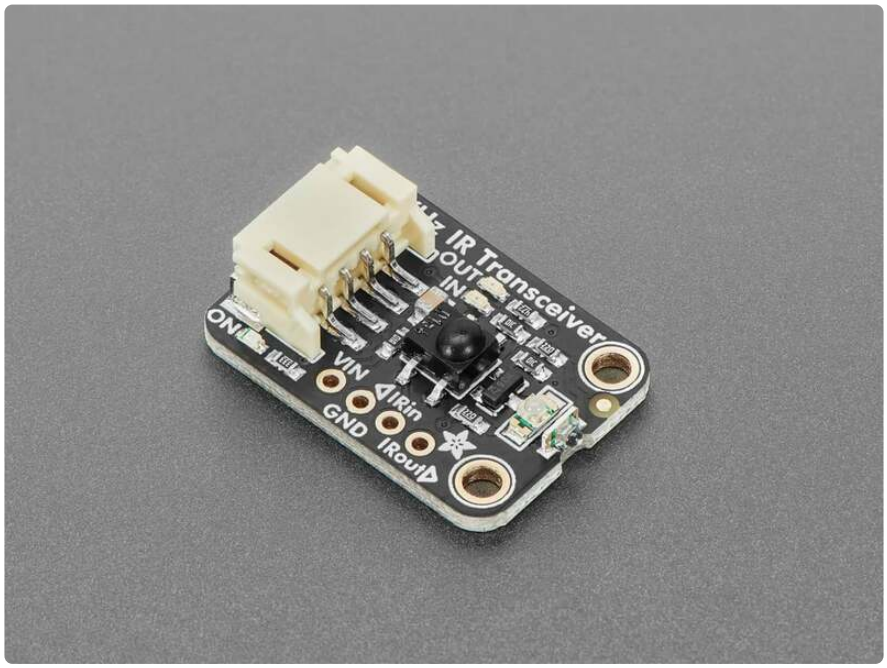


For the emitter half, we use the same schematic and setup as our "High Power IR LED Emitter" breakout (http://adafru.it/5639), with an onboard N-Channel FET driver, to blast 100mA-200mA of current pulsing through each LED for 10+ meters of range!

One LED is vertical and one is horizontal so you get tons of coverage. If powering with 5V, the board will draw about 200mA per LED (400mA total) when pulsing on. If powering with 3V the board will draw about 100mA per LED (200mA total). Since you can't see IR with a human eyeball, we have a small yellow LED labeled 'IN' that will blink when the IR LEDs are on.



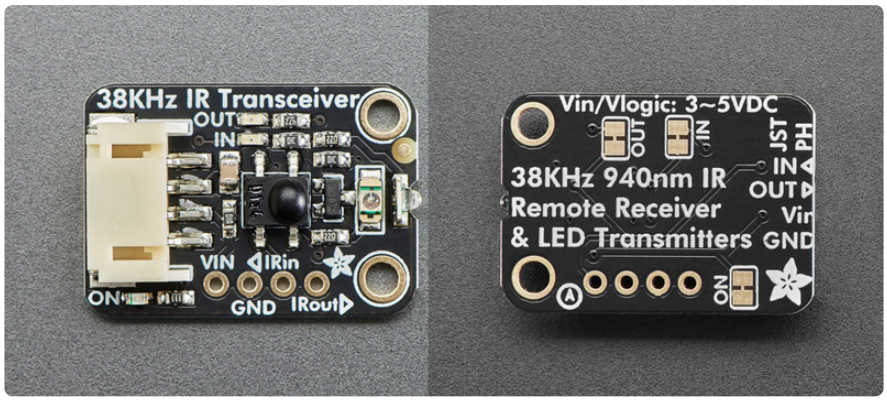
For the receiver half, we use the same 'vertical' IR sensor in our "IR Receiver breakout" (http://adafru.it/5939) The sensor is designed to recognize remote control style modulated signal at 38KHz and 940nm wavelength. The demodulated IR envelope is piped out the OUT labeled pin into your microcontroller which will then need to decode it. To make debugging easy, there's a second yellow LED labelled "OUT" that will blink when data is received.



Each STEMMA board is a fully assembled and tested PCB, but no cable. No soldering is required to use it, but you will need to pick up a 2mm pitch, 4-pin STEMMA JST PH cable (https://adafru.it/1a3Y). Alternatively, if you do want to solder, there's a 0.1" spaced header for power/ground/signals.

Note that this board is specifically for receiving 940nm 38KHz IR remote control signals - it isn't going to work for proximity/distance sensing or other frequency signals. The signal must be read by a microcontroller that has pulse-input reading capabilities - basically just check that it supports common IR Receiver connectivity and decoding. Sometimes you need to use special code or pins.

Pinouts



Power Pins

- **VIN** - this is the power pin. We have included an N-Channel FET driver on board for the IR LEDs that will take 3-5VDC. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V microcontroller like Arduino, use 5V.
    - If powering with **5V**, the board will draw about **200mA per LED** (400mA total) when pulsing on.
    - If powering with **3V** the board will draw about **100mA per LED** (200mA total).
- **GND** - common ground for power and logic.

Signal Pins

- **IRin** - this is the input signal pin from the infrared receiver in the center of the board. When a 38KHz signal is received, the demodulated IR envelope is piped out the **IRin** pin into your microcontroller which will then need to decode it.
- **IRout** - this is the infrared signal output pin. When the pin is high, the IR LEDs are on, when the signal pin is low, the IR LEDs are off.

The infrared receiver on this board is specifically for receiving 38KHz IR remote control signals - it isn't going to work for proximity/distance sensing or other frequency signals.

STEMMA JST PH

- **STEMMA JST PH** (https://adafru.it/Ft4) - 2mm pitch STEMMA JST port for use with 4-pin STEMMA JST PH cables (https://adafru.it/1a3Y). It has connections for:
    - **GND** - common ground for power and data. It is the black wire on the JST PH cable.
    - **VIN** - power input for the infrared receiver and infrared LEDs. It is the red wire on the JST PH cable.
    - **IRin** - infrared signal input. It is the white wire on the JST PH cable.
    - **IRout** - infrared signal output. It is the green wire on the JST PH cable.

Signal LEDs and LED Jumpers

- **IN LED -** In the center of the board, above the IR receiver, is the signal LED, labeled **IN**. It a yellow LED. It will light up when an IR signal is read by the IR receiver.
- **IN LED jumper** - This jumper is located on the back of the board and is labeled **IN**. Cut the trace on this jumper to cut power to the "IN" LED.
- **OUT LED -** In the center of the board, directly above the **IN** LED, is the signal LED, labeled **OUT**. It a yellow LED. Since humans don't see in IR, the **OUT** LED lights up to let you know when the IR LEDs are lit.
- **OUT LED jumper** - This jumper is located on the back of the board and is labeled **OUT**. Cut the trace on this jumper to cut power to the "OUT" LED.

Power LED and LED Jumper

- **Power LED -** In the bottom left corner, below the STEMMA JST PH connector, on the front of the board, is the power LED, labeled **ON**. It is the green LED.
- **Power LED jumper** - This jumper is located on the back of the board and is labeled **ON**. Cut the trace on this jumper to cut power to the "ON" LED.

CircuitPython

It's easy to use the **Infrared IR Remote Transceiver** with CircuitPython and the Adafruit_CircuitPython_IRRemote (https://adafru.it/BBm) module. This module allows you to easily write Python code that allows you to send and receive IR remote pulses using the pulseio (https://adafru.it/18cT) core module.

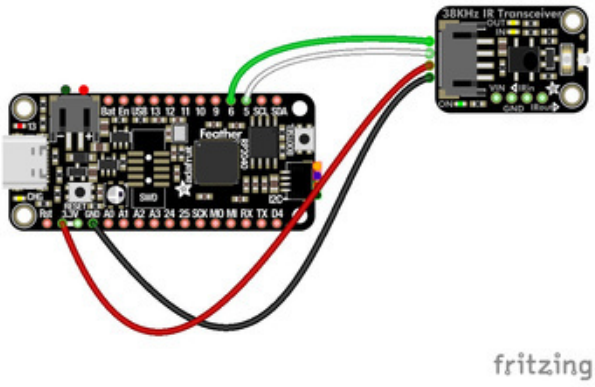
You'll need an IR remote controller to use this example:

Mini Remote Control
This little remote control would be handy for controlling a robot or other project from across the room. It has 21 buttons and a layout we thought was handy: directional buttons and...
https://www.adafruit.com/product/389

This receiver on this board is specifically for receiving 38KHz IR remote control signals - it isn't going to work for proximity/distance sensing or other frequency signals.

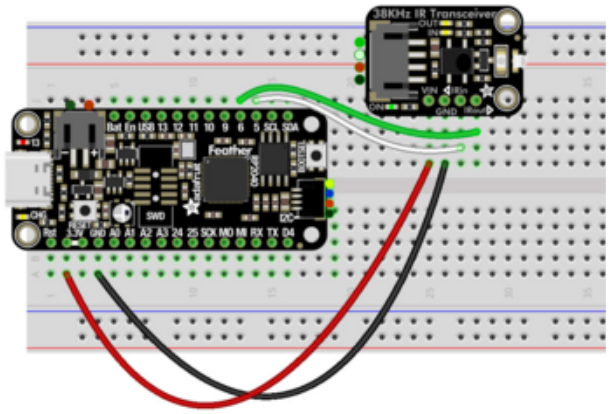
CircuitPython Microcontroller Wiring

First, wire up the transceiver to your board exactly as shown below. Here's an example of wiring a Feather RP2040 to the transceiver using one of the handy STEMMA JST PH (https://adafru.it/1a3Y)cables (https://adafru.it/1a3Y):



Board 3V to **VIN (red wire)**
Board GND to **GND (black wire)**
Board pin 5 to **IRin (white wire)**
Board pin 6 to **IRout (green wire)**

You can also use standard **0.100" pitch** headers to wire it up on a breadboard:



Board 3V to **VIN (red wire)**
Board GND to **GND (black wire)**
Board pin 5 to **IRin (white wire)**
Board pin 6 to **IRout (green wire)**

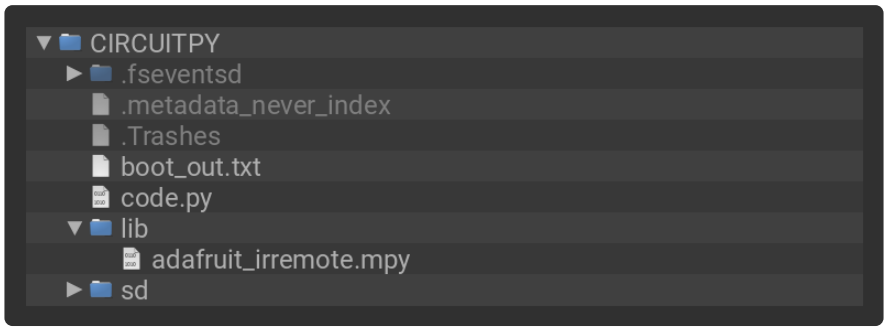
CircuitPython Usage

To use with CircuitPython, you need to first install the IRRemote library into the **lib** folder on your **CIRCUITPY** drive. Then you need to update **code.py** with the example script.

Thankfully, we can do this in one go. In the example below, click the **Download Project Bundle** button below to download the necessary library and the **code.py** file in a zip file. Extract the contents of the zip file, and copy the **entire lib folder** and the **code.py** file to your **CIRCUITPY** drive.

Your **CIRCUITPY/lib** folder should contain the following file:

- **adafruit_irremote.mpy**



Example Code

Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

```
# SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
# SPDX-License-Identifier: MIT
```

```
import time
import array
import pulseio
import board
import adafruit_irremote

# Create a 'PulseOut' to send infrared signals on the IR transmitter @ 38KHz
pulseout = pulseio.PulseOut(board.D6, frequency=38000, duty_cycle=2**15)
# Create an encoder that will take numbers and turn them into NEC IR pulses
encoder = adafruit_irremote.GenericTransmit(header=[9000, 4500],
                                            one=[560, 1700],
                                            zero=[560, 560],
                                            trail=0)
# IR receiver setup
ir_receiver = pulseio.PulseIn(board.D5, maxlen=120, idle_state=True)
decoder = adafruit_irremote.GenericDecode()

while True:
    pulses = decoder.read_pulses(ir_receiver)
    try:
        # Attempt to decode the received pulses
        received_code = decoder.decode_bits(pulses)
        if received_code:
            hex_code = ''.join(["%02X" % x for x in received_code])
            print(f"Received: {hex_code}")
            # Convert pulses to an array of type 'H' (unsigned short)
            pulse_array = array.array('H', pulses)
            # send code back using original pulses
            pulseout.send(pulse_array)
            print(f"Sent: {pulse_array}")
    except adafruit_irremote.IRNECRepeatException:  # Signal was repeated, ignore
        pass
    except adafruit_irremote.IRDecodeException:  # Failed to decode signal
        print("Error decoding")
    ir_receiver.clear()  # Clear the receiver buffer
    time.sleep(1)  # Delay to allow the receiver to settle
    print()
```

The code begins by creating `pulseio` objects on pin `D5` and `D6`. These will send and receive infrared signals at 38KHz. Then, `emitter` and `decoder` objects are created with the `adafruit_irremote` library.

In the loop, if an IR remote command is received, its HEX (hexadecimal) code is printed to the serial monitor. Then, that same command is sent via the IR LEDs as an array. `ir_receiver.clear()` is called to clear the buffer so that the receiver does not go into an infinite loop of receiving the code sent by the IR LEDs.

```
CircuitPython REPL
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Received: 00FD08F7
Sent: array('H', [9133, 4487, 602, 562, 601, 562, 578, 586, 600, 564, 576, 587, 601, 562,
601, 563, 600, 563, 577, 1667, 600, 1643, 601, 1667, 577, 1643, 601, 1666, 577, 1667,
577, 563, 600, 1643, 601, 563, 600, 563, 600, 564, 600, 563, 600, 1667, 577, 564, 575,
588, 599, 564, 600, 1667, 577, 1667, 576, 1667, 576, 1668, 576, 565, 599, 1667, 577,
1667, 577, 1666, 577])

Received: 00FD8877
Sent: array('H', [9127, 4490, 602, 563, 601, 562, 603, 562, 603, 564, 602, 563, 602, 562,
603, 562, 603, 563, 601, 1643, 602, 1644, 602, 1644, 601, 1643, 603, 1642, 603, 1642,
603, 562, 578, 1667, 603, 1642, 579, 587, 601, 564, 577, 588, 601, 1643, 602, 563, 601,
564, 601, 563, 602, 564, 576, 1668, 602, 1643, 602, 1644, 601, 587, 578, 1667, 577, 1668,
577, 1667, 577])

Received: 00FD6897
Sent: array('H', [9178, 4487, 605, 562, 605, 562, 605, 563, 605, 562, 604, 563, 604, 562,
604, 562, 604, 562, 604, 1642, 604, 1642, 604, 1642, 604, 1642, 579, 1667, 603, 1643,
603, 563, 603, 1642, 603, 563, 602, 1643, 602, 1668, 554, 588, 577, 1668, 602, 563, 602,
564, 601, 564, 602, 1667, 578, 564, 577, 588, 601, 1668, 578, 564, 601, 1668, 554, 1691,
578, 1667, 578])
```

Python Docs

Python Docs (https://adafru.it/18aP)

Arduino

Using the Infrared IR Remote Transceiver with Arduino involves wiring up the transceiver to your Arduino-compatible microcontroller, installing the IRremote (https://adafru.it/18bs) library and running the provided example code.

You'll need an IR remote controller to use this example:

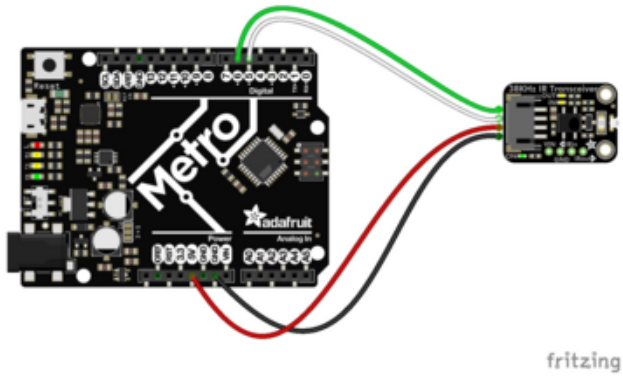
Mini Remote Control
This little remote control would be handy for controlling a robot or other project from across the room. It has 21 buttons and a layout we thought was handy: directional buttons and...
https://www.adafruit.com/product/389

This receiver on this board is specifically for receiving 38KHz IR remote control signals - it isn't going to work for proximity/distance sensing or other frequency signals.

Wiring

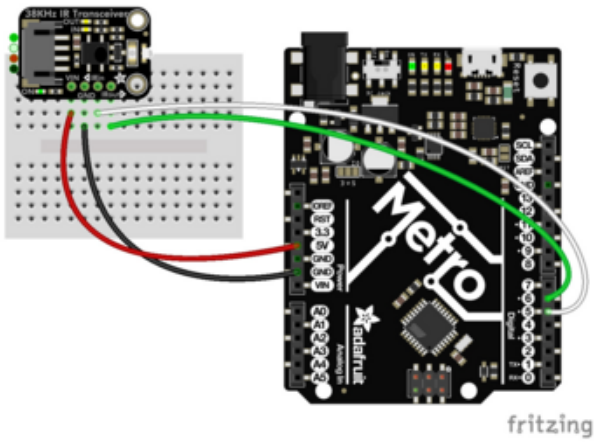
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the transceiver VIN.

Here is an Adafruit Metro wired up to the transceiver using the STEMMA JST PH cable:



Board 5V to **VIN (red wire)**
Board GND to **GND (black wire)**
Board pin 5 to **IRin (white wire)**
Board pin 6 to **IROut (green wire)**

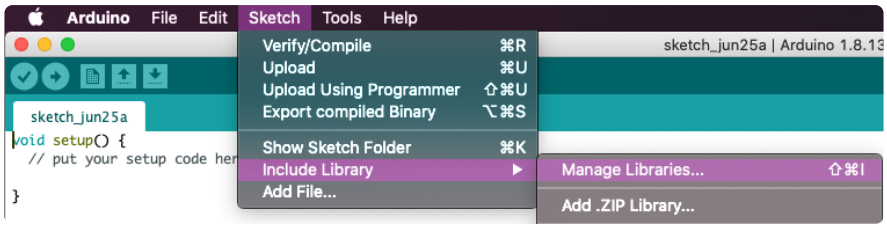
Here is an Adafruit Metro wired up using a solderless breadboard:



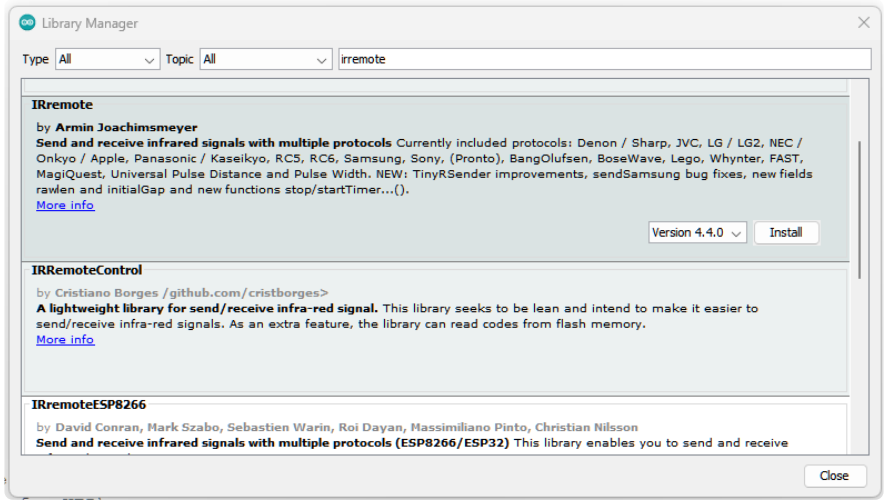
Board 5V to **VIN (red wire)**
Board GND to **GND (black wire)**
Board pin 5 to **IRin (white wire)**
Board pin 6 to **IROut (green wire)**

Library Installation

You can install the **IRremote** library for Arduino using the Library Manager in the Arduino IDE.



Click the **Manage Libraries ...** menu item, search for **IRremote,** and select the **IRremote** library:



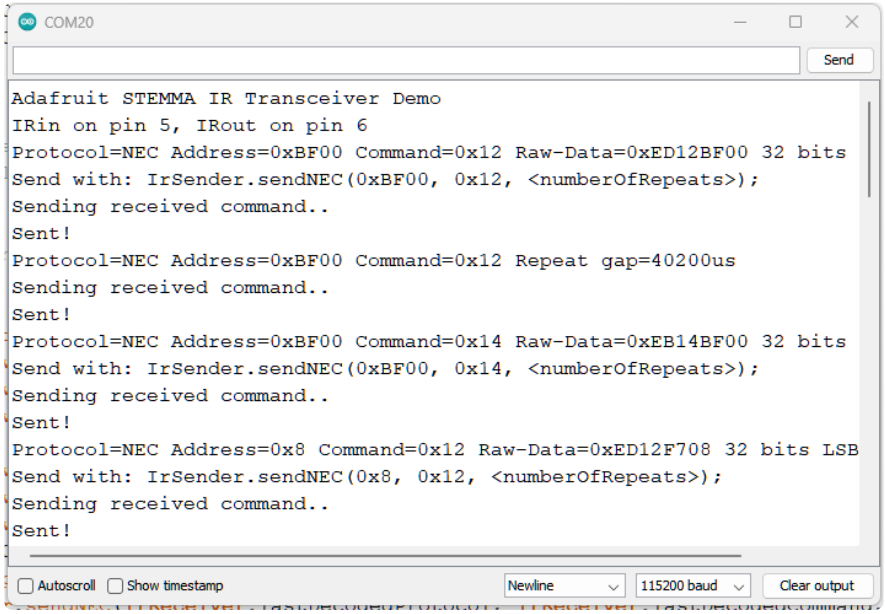
There are no additional dependencies needed for this library.

Example Code

```
// SPDX-FileCopyrightText: 2024 Liz Clark for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
 * Based on the SimpleReceiver.cpp and SimpleSender.cpp from the
 * Arduino-IRremote https://github.com/Arduino-IRremote/Arduino-IRremote.
 * by Armin Joachimsmeyer

************************************************************************************
 * MIT License
 *
 * Copyright (c) 2020-2023 Armin Joachimsmeyer
 *
 */

#include <Arduino.h>

#include <IRremote.hpp> // include the library
#define IR_RECEIVE_PIN   5
#define IR_SEND_PIN      6

void setup() {
```

```
    Serial.begin(115200);
    while (!Serial)
        ;
    Serial.println("Adafruit STEMMA IR Transceiver Demo");
    IrReceiver.begin(IR_RECEIVE_PIN);
    IrSender.begin(IR_SEND_PIN); // Start with IR_SEND_PIN -which is defined in
PinDefinitionsAndMore.h- as send pin and enable feedback LED at default feedback
LED pin
    Serial.print("IRin on pin ");
    Serial.print(IR_RECEIVE_PIN);
    Serial.print(", IRout on pin ");
    Serial.println(IR_SEND_PIN);
}

uint8_t sCommand = 0x34;
uint8_t sRepeats = 0;

void loop() {
    /*
     * Check if received data is available and if yes, try to decode it.
     * Decoded result is in the IrReceiver.decodedIRData structure.
     *
     * E.g. command is in IrReceiver.decodedIRData.command
     * address is in command is in IrReceiver.decodedIRData.address
     * and up to 32 bit raw data in IrReceiver.decodedIRData.decodedRawData
     */
    if (IrReceiver.decode()) {
        if (IrReceiver.decodedIRData.protocol == UNKNOWN) {
            IrReceiver.printIRResultRawFormatted(&Serial, true);
            IrReceiver.resume();
        } else {
            IrReceiver.resume();
            IrReceiver.printIRResultShort(&Serial);
            IrReceiver.printIRSendUsage(&Serial);
            delay(1000);
            Serial.println("Sending received command..");
            IrSender.sendNEC(IrReceiver.lastDecodedProtocol,
IrReceiver.lastDecodedCommand, IrReceiver.repeatCount);
            delay(1000);
            Serial.print("Sent!");
            //Serial.println(IrReceiver.lastDecodedProtocol,
IrReceiver.lastDecodedCommand, IrReceiver.repeatCount);
        }
        Serial.println();
    }
}
```

Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. As you press buttons on your IR remote, you'll see the protocol, address, command, raw data and repeat gap print to the Serial Monitor. Then, the breakout will send the received IR code via the IR LEDs.



Arduino Docs

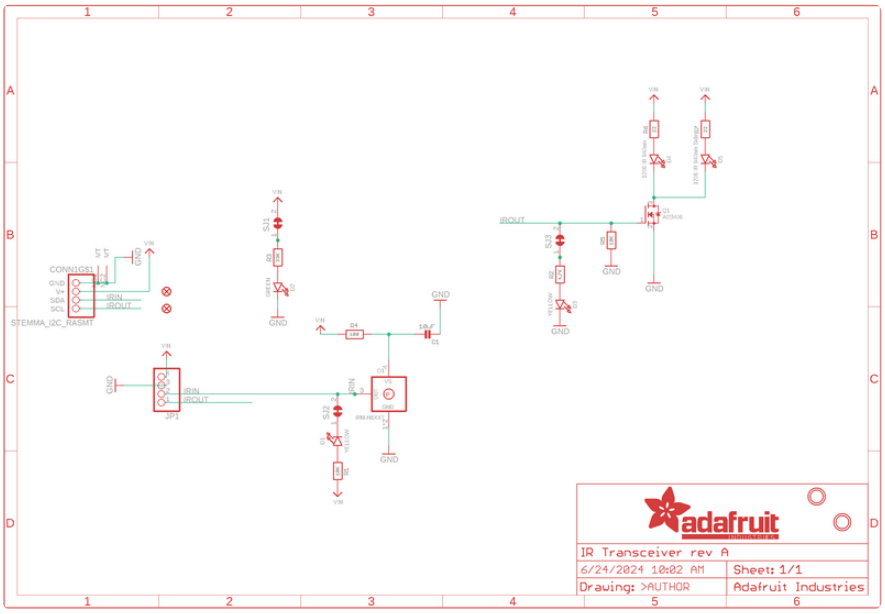
Arduino Docs (https://adafru.it/19Sa)

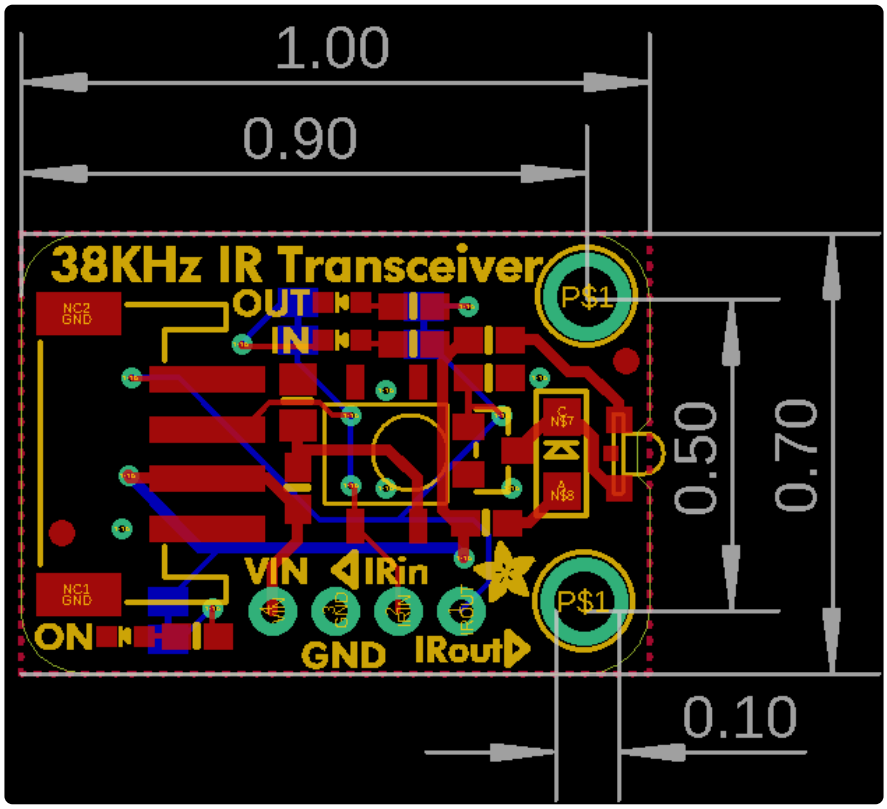
Downloads

Files

- EagleCAD PCB Files on GitHub (https://adafru.it/1a3Z)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/1a40)

Schematic and Fab Print





Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Adafruit:

5990