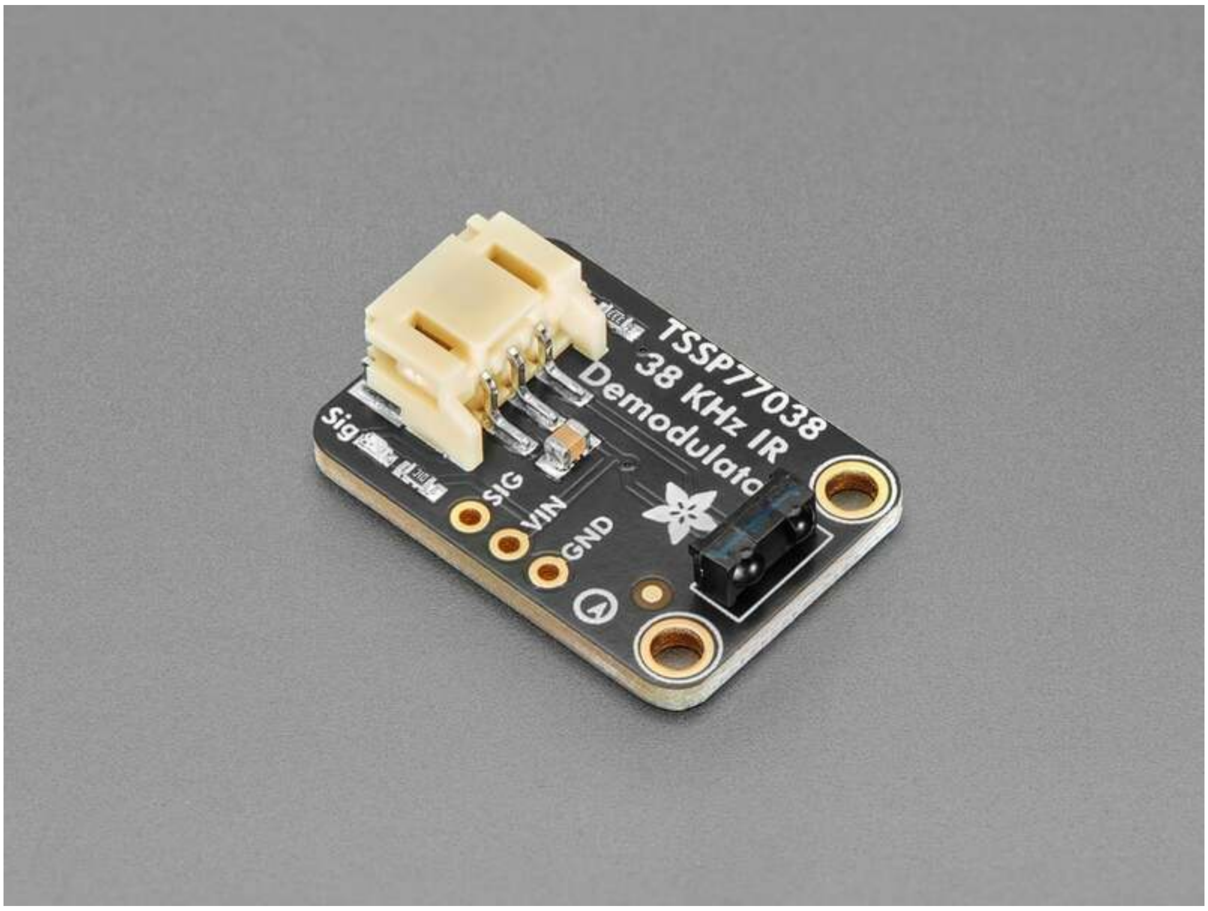
Adafruit TSSP77038 38KHz Infrared IR Demodulator Breakout

Created by Liz Clark

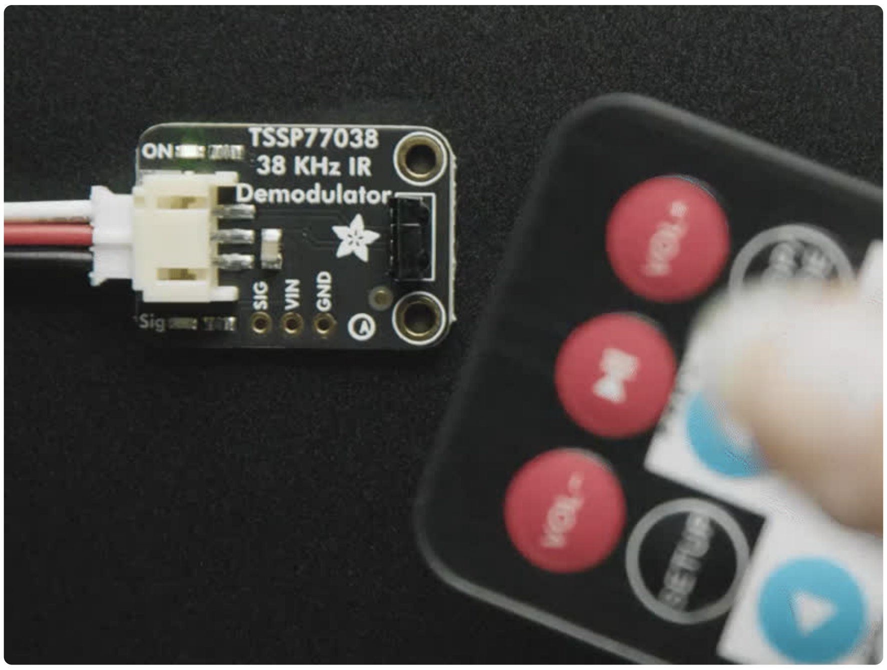


https://learn.adafruit.com/adafruit-tssp77038-38khz-infrared-ir-demodulator-breakout

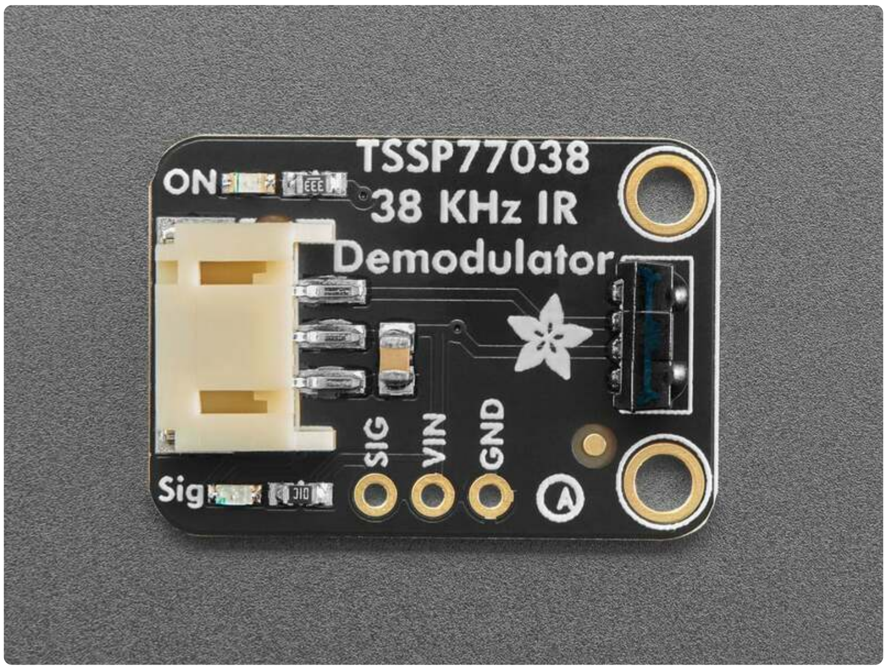
Last updated on 2024-06-11 07:37:23 PM EDT

Table of Contents

Overview

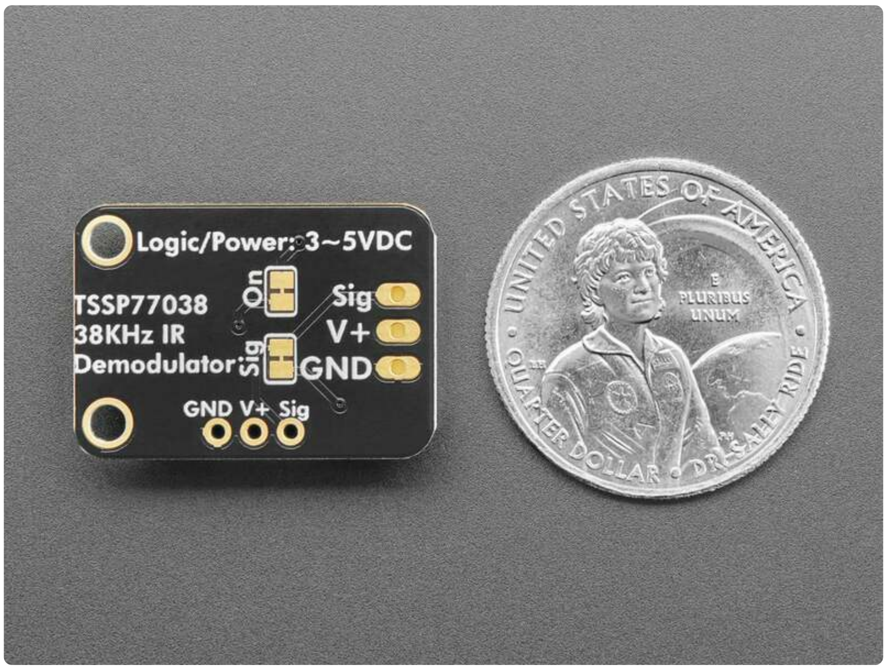


While designing our IR decoding breakout (http://adafru.it/5939), we noticed the TSSP series of chips from Vishay (https://adafru.it/19UA). These are simpler than more 'remote control' receivers in that they don't do any filtering on the codes received: you really just get the demodulated-from-38KHz-signal output.

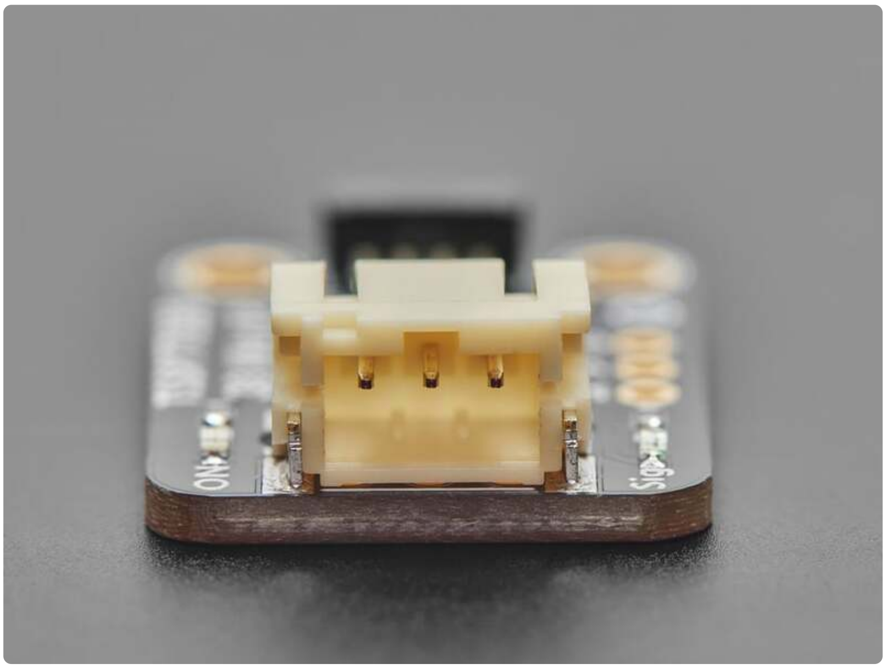


According to Vishay, they're designed for "Reflective Sensing, Light Barriers, and Fast Proximity Applications". These sensors could be interesting for use when you want a less-'intelligent' IR decoder, or for proximity projects where you don't want to go all-out for a VCNL I2C sensor (http://adafru.it/4161).

These are good for creating break-beams where the IR light is modulated rather than just solid on, because you won't be affected by other IR signals or ambient light changes.



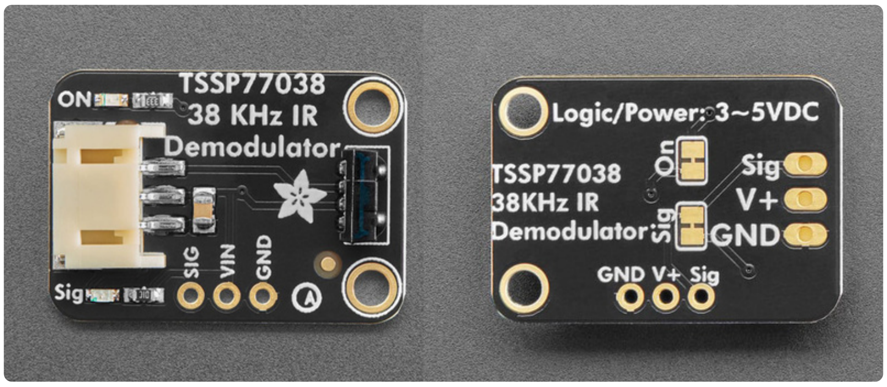
This board will work nicely for a variety of IR detecting projects, and with mounting holes and a cable, a lot easier to mount in enclosures and on devices. Using a 2mm pitch STEMMA JST PH cable with headers or alligator clips on the end, you can easily wire this board without any soldering.



Note that this board is specifically for proximity sensing or break-beam projects. While it can receive 38KHz IR remote control signals - there isn't a filter system so you'll get a lot of spurious signals.

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, 3-pin STEMMA JST PH cable (https://adafru.it/18cS). Alternatively, if you do want to solder, there's a 0.1" spaced header for power/ground/signal.

Pinouts



Power Pins

- **VIN/V+** - this is the power pin. 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.
- **GND** - common ground for power and logic.

Signal Output

- **SIG/Sig** - this is the output signal from the infrared demodulator. When a 38KHz signal is received, the IR signal is piped out the Signal pin into your microcontroller.

This board is specifically for proximity sensing or break-beam projects. It can receive 38KHz IR remote control signals but there isn't a filter system so you'll get a lot of spurious signals.

STEMMA JST PH

- **STEMMA JST PH** (https://adafru.it/Ft4) - 2mm pitch STEMMA JST port for use with 3-pin STEMMA JST PH cables (https://adafru.it/JRA). It has connections for:
  - **GND** - common ground for power and data. It is the black wire on the JST PH cable.
  - **V+** - power input for the infrared demodulator. It is the red wire on the JST PH cable.
  - **Sig** - signal to your microcontroller. It is the white wire on the JST PH cable.

Signal LED and Jumper

- **Signal LED** - to the left of the JST PH connector is the signal LED, labeled **Sig**. It is the red LED. It will light up when an IR signal is read by the demodulator.
- **LED jumper** - in the center of the back of the board is a jumper for the signal LED. It is labeled **Sig** on the board silk. If you want to disable the signal LED, cut the trace on this jumper.

Power LED and Jumper

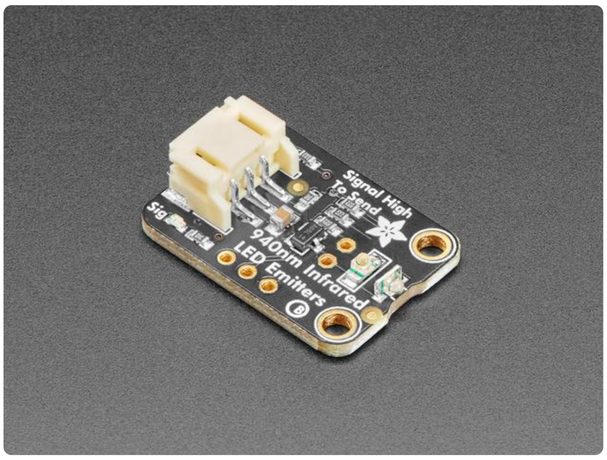
- **Power LED** - to the right of the JST PH connector is the power LED, labeled **ON**. It is the green LED.
- **LED jumper** - in the center of the back of the board is a jumper for the power LED. It is labeled **On** on the board silk. If you want to disable the power LED, cut the trace on this jumper.

CircuitPython and Python

It's easy to use the **Infrared IR Demodulator Breakout** with CircuitPython and the pulseio (https://adafru.it/18cT) core module. This module allows you to easily write Python code for sending and receiving pulse signals.

You can use this driver with any CircuitPython microcontroller board or with a computer that has GPIO, pulseio (https://adafru.it/19Sf) support and Python thanks to Adafruit_Blinka, our CircuitPython-for-Python compatibility library (https://adafru.it/BSN). Not all single board computers (SBCs) have pulseio support. Make sure to check if it is supported (https://adafru.it/18Ye) on your board.

You'll need an IR LED or IR remote controller to use this example with the demodulator:



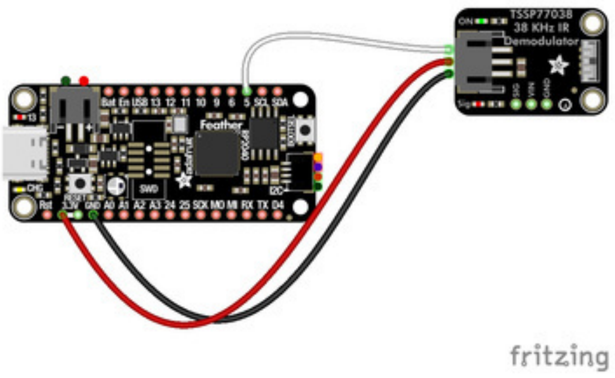
Adafruit High Power Infrared IR LED Emitter - STEMMA JST PH 2mm
pew *pew*! This board is like a little ray gun for infrared light, with two high powered LED outputs. When controlled with the onboard N-Channel FET driver,...
https://www.adafruit.com/product/5639

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 board is specifically for proximity sensing or break-beam projects. It can receive 38KHz IR remote control signals but there isn't a filter system so you'll get a lot of spurious signals.

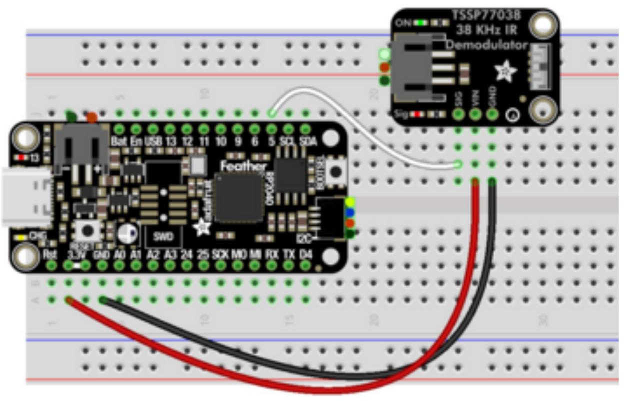
CircuitPython Microcontroller Wiring

First wire up the sensor to your board exactly as follows. The following is the demodulator wired to a Feather RP2040 with a JST PH cable.



Board 3V to **demodulator JST PH V+ (red wire)**
Board GND to **demodulator JST PH GND (black wire)**
Board pin 5 to **demodulator JST PH Sig (white wire)**

The following is the demodulator wired to a Feather RP2040 using a solderless breadboard:

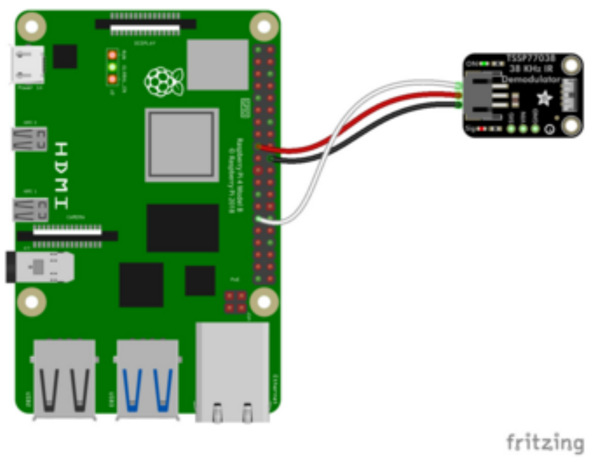


Board 3V to **demodulator VIN (red wire)**
Board GND to **demodulator GND (black wire)**
Board pin 5 to **demodulator SIG (white wire)**

Python Computer Wiring

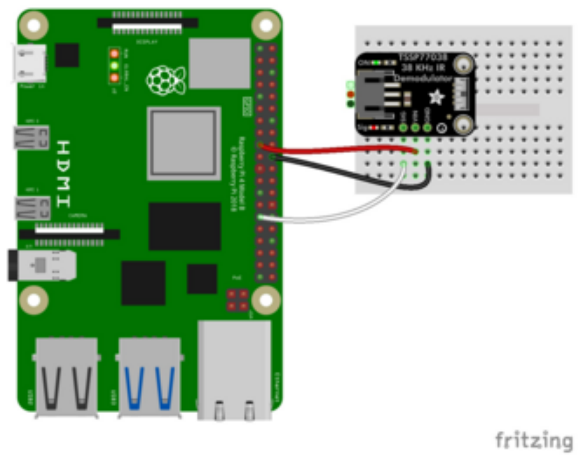
Since there are dozens of Linux computers/boards you can use, we will show wiring for Raspberry Pi. For other platforms, please visit the guide for CircuitPython on Linux to see whether your platform is supported (https://adafru.it/BSN).

Here's the Raspberry Pi wired to the demodulator using a JST PH cable:



Pi 3V to **demodulator JST PH V+ (red wire)**
Pi GND to **demodulator JST PH GND (black wire)**
Pi GPIO5 to **demodulator JST PH Sig (white wire)**

Here is how you'll wire the demodulator to a Raspberry Pi with a breadboard:



Pi 3V to **demodulator VIN (red wire)**
Pi GND to **demodulator GND (black wire)**
Pi GPIO5 to **demodulator SIG (white wire)**

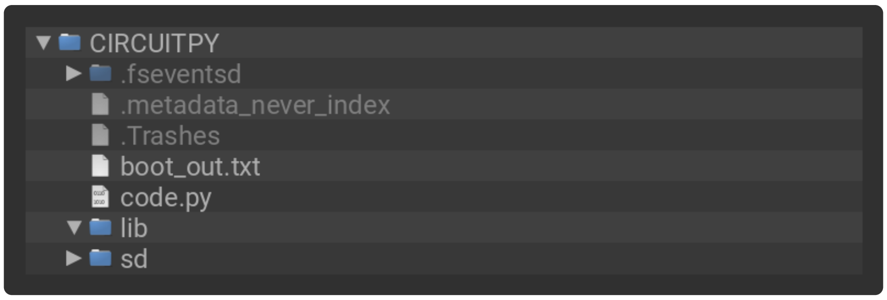
Python Installation

You'll need to install the **Adafruit_Blinka** library that provides the CircuitPython support in Python. This may also require enabling I2C on your platform and verifying you are running Python 3. Since each platform is a little different, and Linux changes often, please visit the CircuitPython on Linux guide to get your computer ready (https://adafru.it/BSN)!

CircuitPython Usage

To use with CircuitPython, 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 **code.py** file in a zip file. Extract the contents of the zip file, and the **code.py** file to your **CIRCUITPY** drive.



Only core modules are used for this example. No additional libraries need to be added to the /lib folder.

Python Usage

Once you have the library `pip3` installed on your computer, copy or download the following example to your computer, and run the following, replacing **code.py** with whatever you named the file:

`python3 code.py`

Example Code

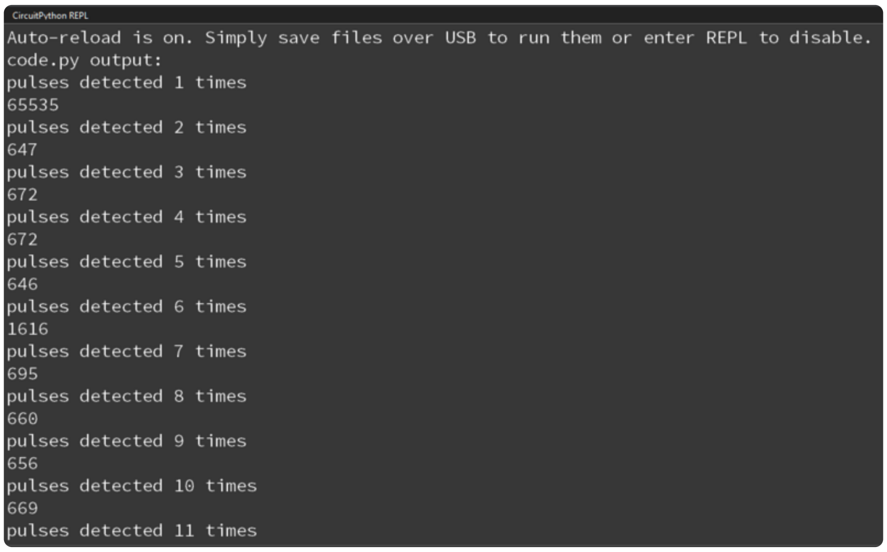
If running CircuitPython: Once everything is saved to the **CIRCUITPY** drive, connect to the serial console (https://adafru.it/Bec) to see the data printed out!

If running Python: The console output will appear wherever you are running Python.

```
# SPDX-FileCopyrightText: Copyright (c) 2024 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import pulseio
import board

pulses = pulseio.PulseIn(board.D5)
pulse = False
pulse_count = 0

while True:

    # Wait for an active pulse
    while len(pulses) == 0:
        if pulse:
            pulse = False

    if len(pulses) != 0 and not pulse:
        pulse_count += 1
        print(f"pulses detected {pulse_count} times")
        pulse = True
    # Pause while we do something with the pulses
    pulses.pause()
    # Print the pulses. pulses[0] is an active pulse unless the length
    # reached max length and idle pulses are recorded.
    print(pulses[0])
    # Clear the rest
    pulses.clear()
    # Resume with an 80 microsecond active pulse
    pulses.resume(80)
```

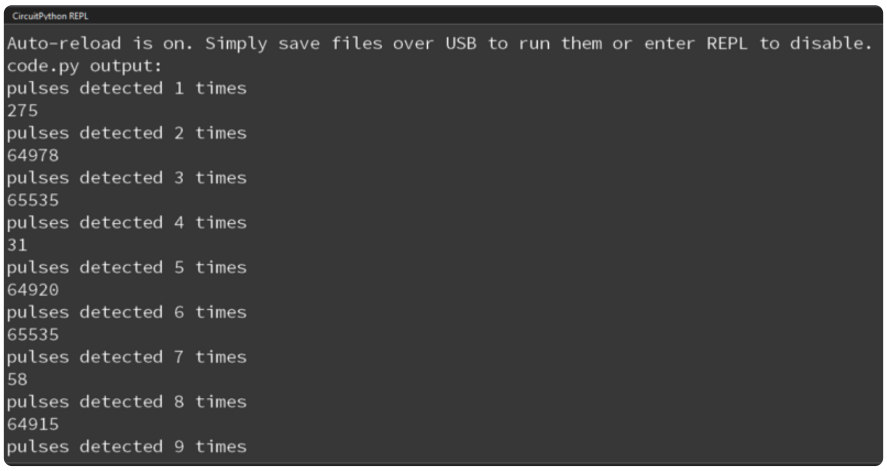
In the code, if an IR pulse is detected, its pulses are printed to the serial monitor. An additional variable `pulse_count` keeps track of how many times a new pulse is detected. The `pulse` state tracks whether a pulse input is detected or not by the demodulator. You can use this code as a template for using the demodulator as a proximity sensor or break-beam.

Here is the output using an IR remote with the demodulator:



```
CircuitPython REPL
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
pulses detected 1 times
65535
pulses detected 2 times
647
pulses detected 3 times
672
pulses detected 4 times
672
pulses detected 5 times
646
pulses detected 6 times
1616
pulses detected 7 times
695
pulses detected 8 times
660
pulses detected 9 times
656
pulses detected 10 times
669
pulses detected 11 times
```

Here is the output using an IR LED emitter with the modulator:



```
CircuitPython REPL
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
pulses detected 1 times
275
pulses detected 2 times
64978
pulses detected 3 times
65535
pulses detected 4 times
31
pulses detected 5 times
64920
pulses detected 6 times
65535
pulses detected 7 times
58
pulses detected 8 times
64915
pulses detected 9 times
```

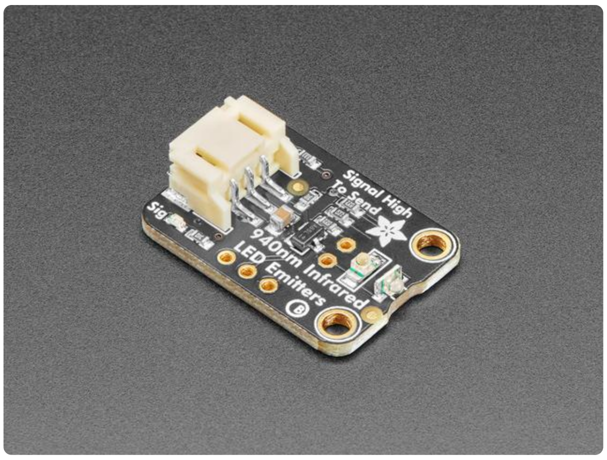
Python Docs

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

Arduino

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

You'll need an IR LED or IR remote controller to use this example with the demodulator:



Adafruit High Power Infrared IR LED Emitter - STEMMA JST PH 2mm
pew *pew*! This board is like a little ray gun for infrared light, with two high powered LED outputs. When controlled with the onboard N-Channel FET driver,...
https://www.adafruit.com/product/5639

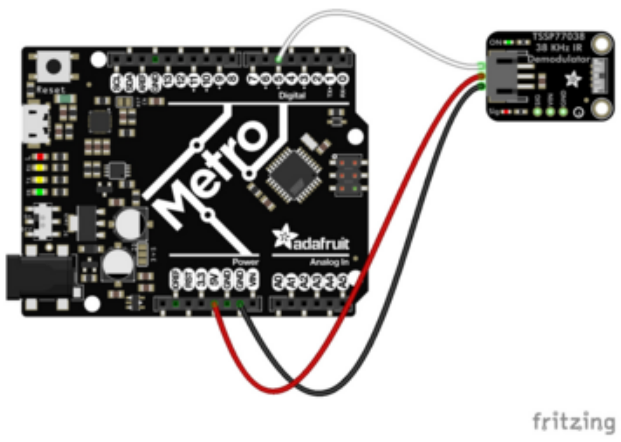
Mini Remote Control
This little remote control would be handy for controlling a robot or other project from across the room. It has 21 buttons and a layout we thought was handy: directional buttons and...
https://www.adafruit.com/product/389

This board is specifically for proximity sensing or break-beam projects. It can receive 38KHz IR remote control signals but there isn't a filter system so you'll get a lot of spurious signals.

Wiring

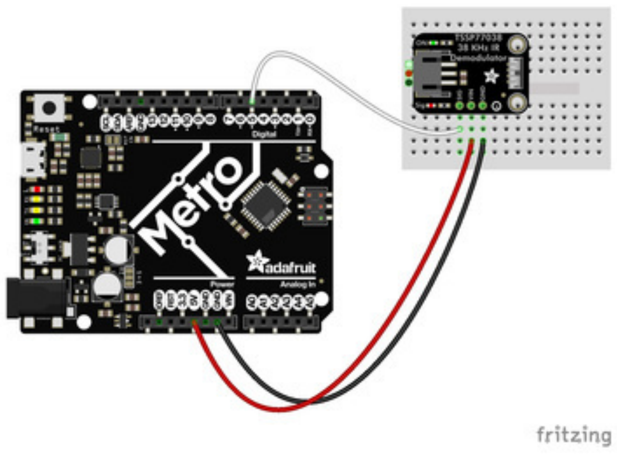
Wire as shown for a **5V** board like an Uno. If you are using a **3V** board, like an Adafruit Feather, wire the board's 3V pin to the breakout VIN.

Here is an Adafruit Metro wired up to the demodulator using a JST PH cable.



Board 5V to **demodulator JST PH V+ (red wire)**
Board GND to **demodulator JST PH GND (black wire)**
Board pin 5 to **demodulator JST PH Sig (white wire)**

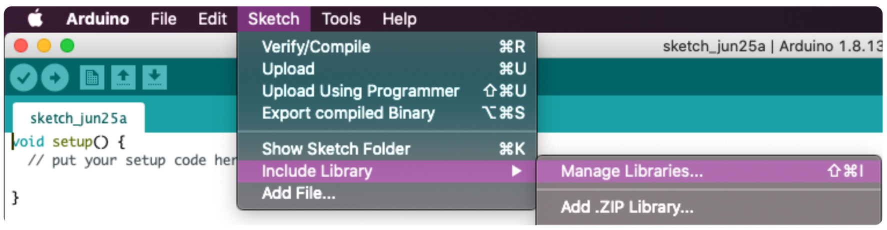
Here is an Adafruit Metro wired up using a solderless breadboard:



Board 5V to **demodulator VIN (red wire)**
Board GND to **demodulator GND (black wire)**
Board pin 5 to **demodulator SIG (white wire)**

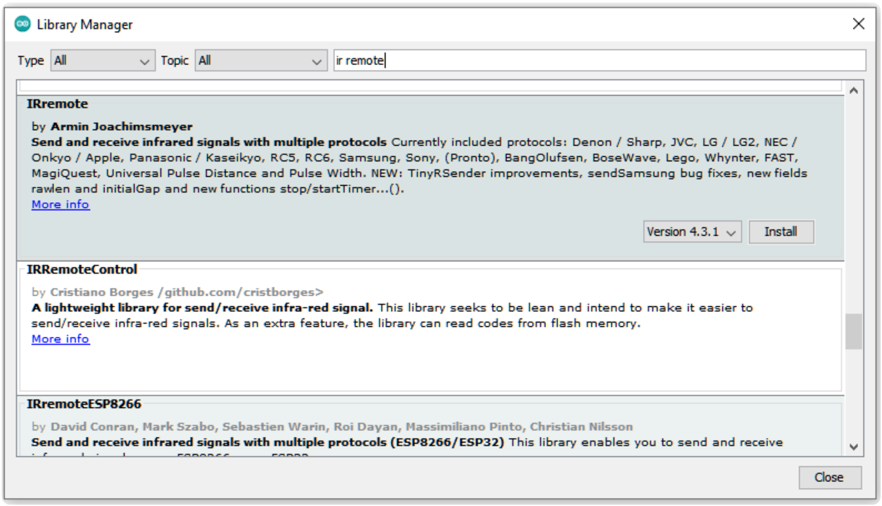
Library Installation

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



```
void setup() {
  // put your setup code her
}
```

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



IRremote

by **Armin Joachimsmeyer**
Send and receive infrared signals with multiple protocols Currently included protocols: Denon / Sharp, JVC, LG / LG2, NEC / Onkyo / Apple, Panasonic / Kaseikyo, RC5, RC6, Samsung, Sony, (Pronto), BangOlufsen, BoseWave, Lego, Whynter, FAST, MagiQuest, Universal Pulse Distance and Pulse Width. NEW: TinyRSender improvements, sendSamsung bug fixes, new fields rawlen and initialGap and new functions stop/startTimer...().
More info

IRRemoteControl

by Cristiano Borges /github.com/cristborges>
A lightweight library for send/receive infra-red signal. This library seeks to be lean and intend to make it easier to send/receive infra-red signals. As an extra feature, the library can read codes from flash memory.
More info

IRremoteESP8266

by David Conran, Mark Szabo, Sebastien Warin, Roi Dayan, Massimiliano Pinto, Christian Nilsson
Send and receive infrared signals with multiple protocols (ESP8266/ESP32) This library enables you to send and receive

There are no additional dependencies needed for this library.

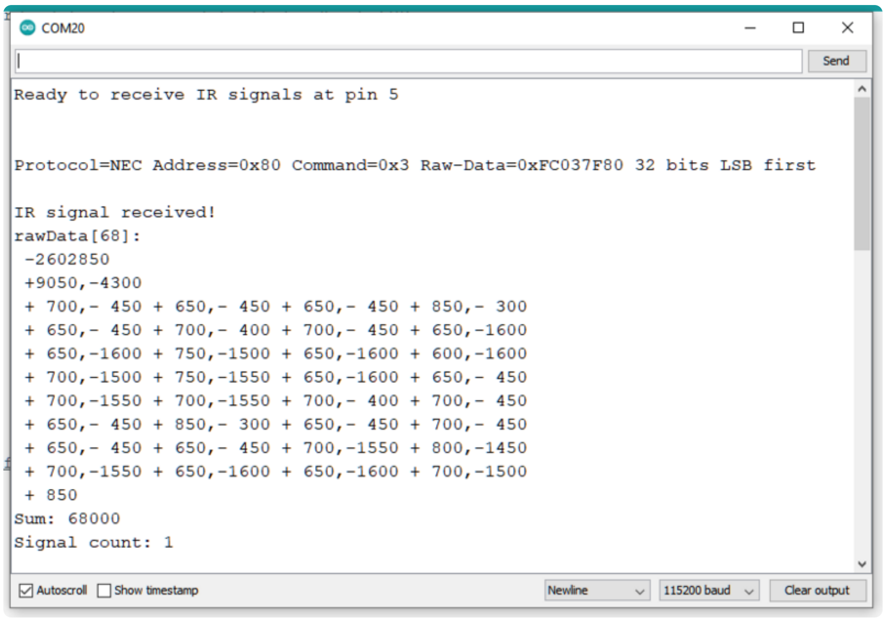
Example Code

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

/*
 * Based on the ReceiverDump.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>

#define IR_RECEIVE_PIN      5
#define MARK_EXCESS_MICROS    20    // Adapt it to your IR receiver module. 20 is
recommended for the cheap VS1838 modules.

int ir_count = 0;
bool ir_state = false;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);

  Serial.begin(115200);
  // Start the receiver and if not 3. parameter specified, take LED_BUILTIN pin
from the internal boards definition as default feedback LED
```

```
  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);
  Serial.print(F("Ready to receive IR signals "));
  Serial.print("at pin ");
  Serial.println(IR_RECEIVE_PIN);

}

void loop() {
  // put your main code here, to run repeatedly:
  if (IrReceiver.decode()) {  // Grab an IR code
        // At 115200 baud, printing takes 200 ms for NEC protocol and 70 ms for NEC 
repeat
        ir_state = true;
        Serial.println(); // blank line between entries
        Serial.println(); // 2 blank lines between entries
        IrReceiver.printIRResultShort(&Serial);
        if (IrReceiver.decodedIRData.flags & IRDATA_FLAGS_WAS_OVERFLOW) {
            Serial.print("Try to increase the \"RAW_BUFFER_LENGTH\" value of ");
            Serial.println(RAW_BUFFER_LENGTH);
            // see also https://github.com/Arduino-IRremote/Arduino-
IRremote#compile-options--macros-for-this-library
        } else {
          Serial.println();
          Serial.println("IR signal received!");
          IrReceiver.printIRResultRawFormatted(&Serial, true);  // Output the 
results in RAW format
          ir_count += 1;
          Serial.print("Signal count: ");
          Serial.println(ir_count);
          Serial.println();
        }
   IrReceiver.resume();
  }
  else {
    ir_state = false;
  }

}
```

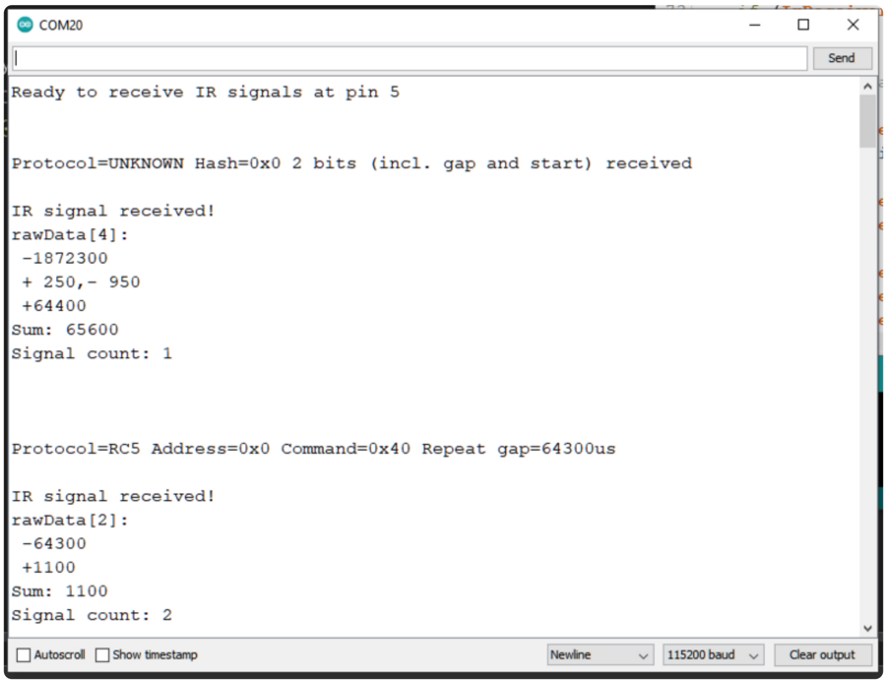
Upload the sketch to your board and open up the Serial Monitor (**Tools -> Serial Monitor**) at 115200 baud. As you send IR signals to the demodulator, you'll see the raw data packets print to the Serial Monitor. You'll also see the `signal_count`, which increases in value by `1` every time a new signal is detected. A boolean state called `ir_state` is `true` when a signal is detected and `false` when a signal is not detected. You can use this code as a template for using the demodulator as a proximity sensor or break-beam.

Here is the output using an IR remote with the demodulator:



```
Ready to receive IR signals at pin 5


Protocol=NEC Address=0x80 Command=0x3 Raw-Data=0xFC037F80 32 bits LSB first

IR signal received!
rawData[68]:
 -2602850
 +9050,-4300
 + 700,- 450 + 650,- 450 + 650,- 450 + 850,- 300
 + 650,- 450 + 700,- 400 + 700,- 450 + 650,-1600
 + 650,-1600 + 750,-1500 + 650,-1600 + 600,-1600
 + 700,-1500 + 750,-1550 + 650,-1600 + 650,- 450
 + 700,-1550 + 700,-1550 + 700,- 400 + 700,- 450
 + 650,- 450 + 850,- 300 + 650,- 450 + 700,- 450
 + 650,- 450 + 650,- 450 + 700,-1550 + 800,-1450
 + 700,-1550 + 650,-1600 + 650,-1600 + 700,-1500
 + 850
Sum: 68000
Signal count: 1
```

Here is the output using an IR LED emitter with the demodulator:



```
Ready to receive IR signals at pin 5


Protocol=UNKNOWN Hash=0x0 2 bits (incl. gap and start) received

IR signal received!
rawData[4]:
 -1872300
 + 250,- 950
 +64400
Sum: 65600
Signal count: 1



Protocol=RC5 Address=0x0 Command=0x40 Repeat gap=64300us

IR signal received!
rawData[2]:
 -64300
 +1100
Sum: 1100
Signal count: 2
```

Arduino Docs

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

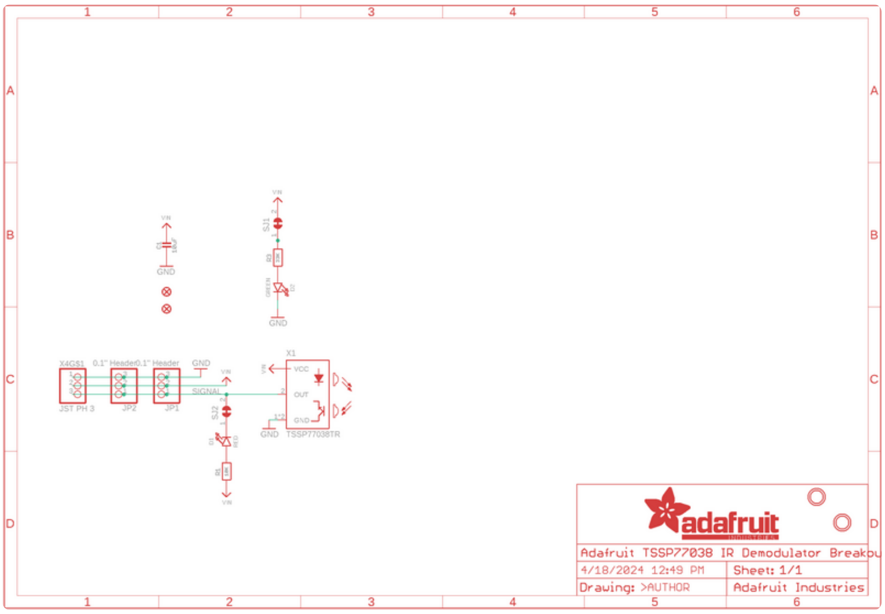
Downloads

Files

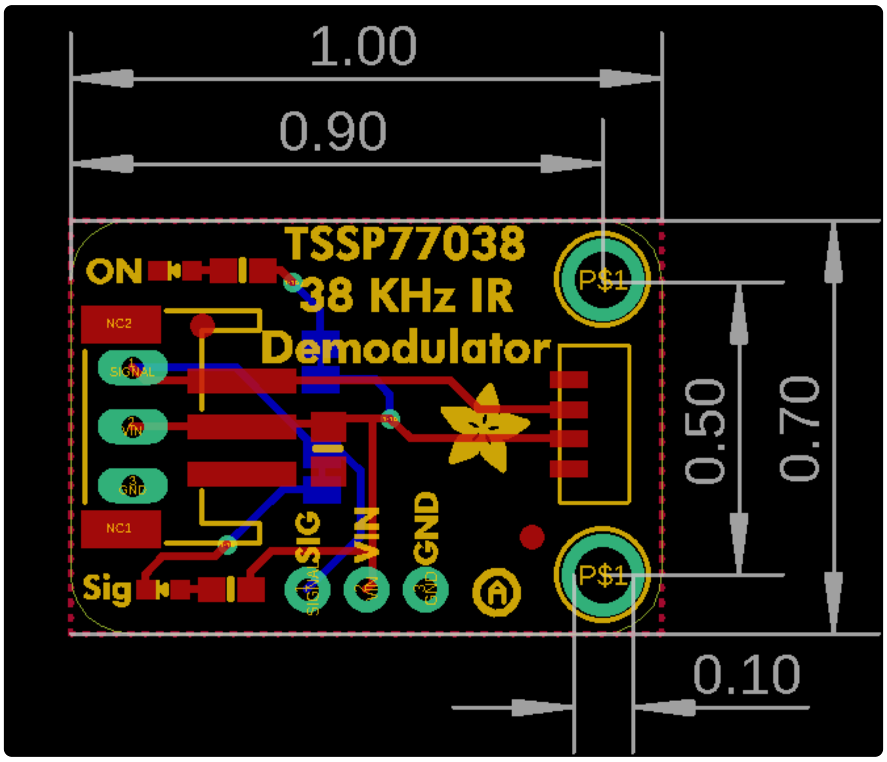
- TSSP77038 Datasheet (https://adafru.it/19UB)

- EagleCAD PCB Files on GitHub (https://adafru.it/19UC)
- Fritzing object in the Adafruit Fritzing Library (https://adafru.it/19UD)

Schematic and Fab Print



Adafruit TSSP77038 IR Demodulator Breakou
4/18/2024 12:49 PM
Sheet: 1/1
Drawing: >AUTHOR
Adafruit Industries
TSSP77038TR
JST PH 3



1.00
0.90
0.50
0.70
0.10
TSSP77038
38 KHz IR
Demodulator
ON
Sig
SIG
VIN
GND
NC2
NC1