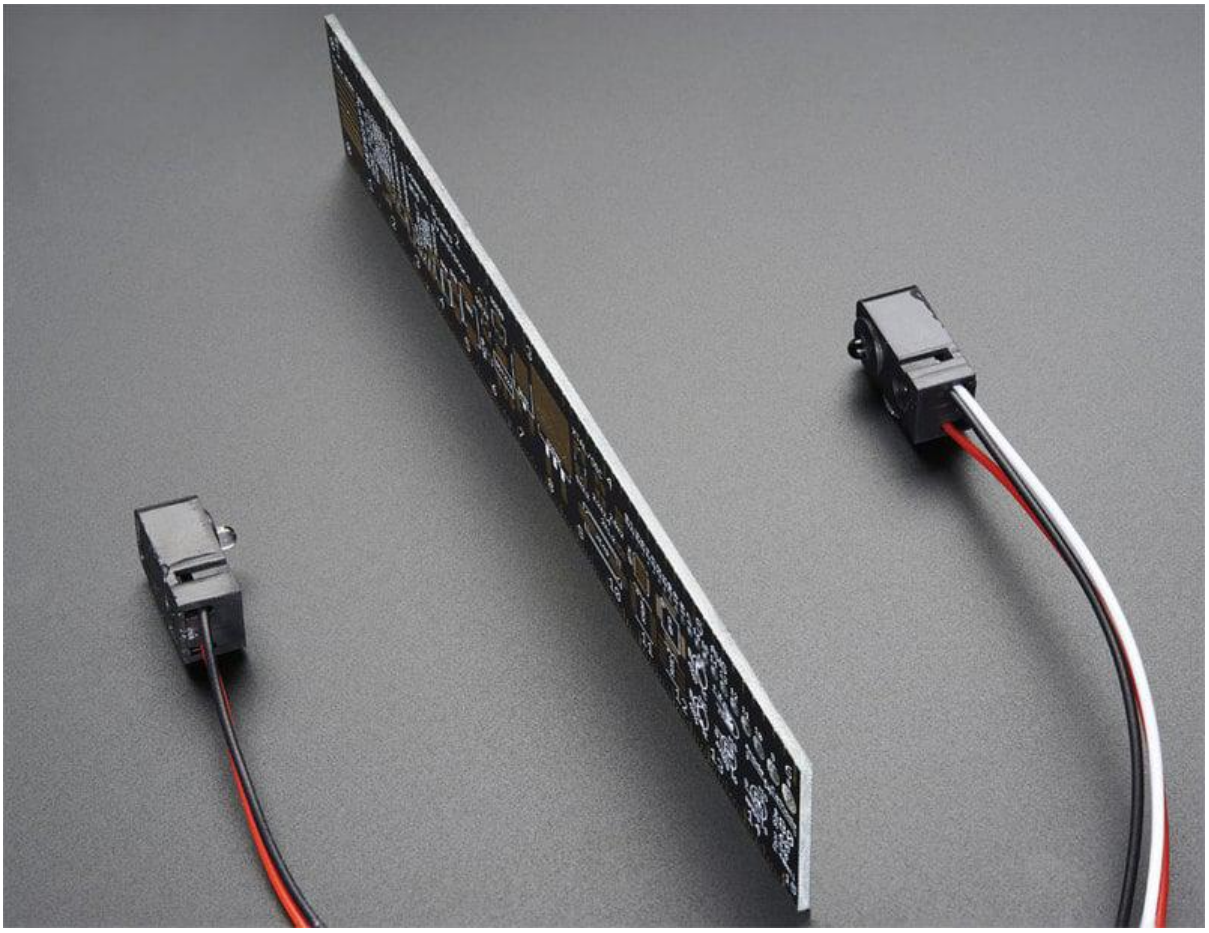
IR Breakbeam Sensors

Created by lady ada

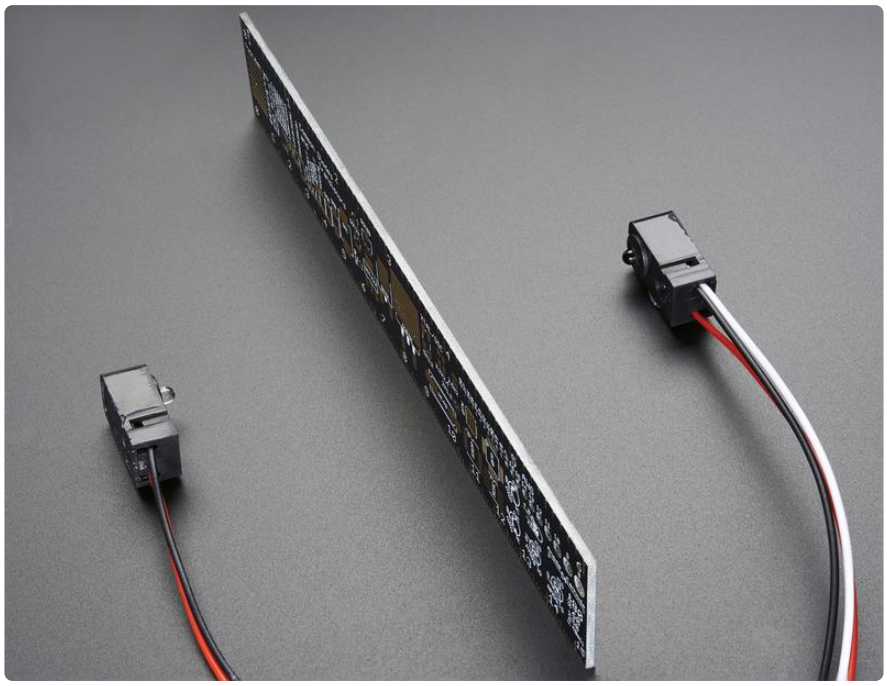


https://learn.adafruit.com/ir-breakbeam-sensors

Last updated on 2022-12-01 02:20:48 PM EST

Table of Contents

Overview



Infrared (IR) break-beam sensors are a simple way to detect motion. They work by having an emitter side that sends out a beam of human-invisible IR light, then a receiver across the way which is sensitive to that same light. When something passes between the two, and its not transparent to IR, then the 'beam is broken' and the receiver will let you know.

Compared to PIR sensors, breakbeams are faster and allow better control of where you want to detect the motion. Compared to Sonar modules, they're less expensive. However, you do need both emitter and receiver

The receiver is on the left, it has three wires. The transmitter is on the right, it has two wires

---

Arduino

Wiring these sensors for Arduino use is really easy.

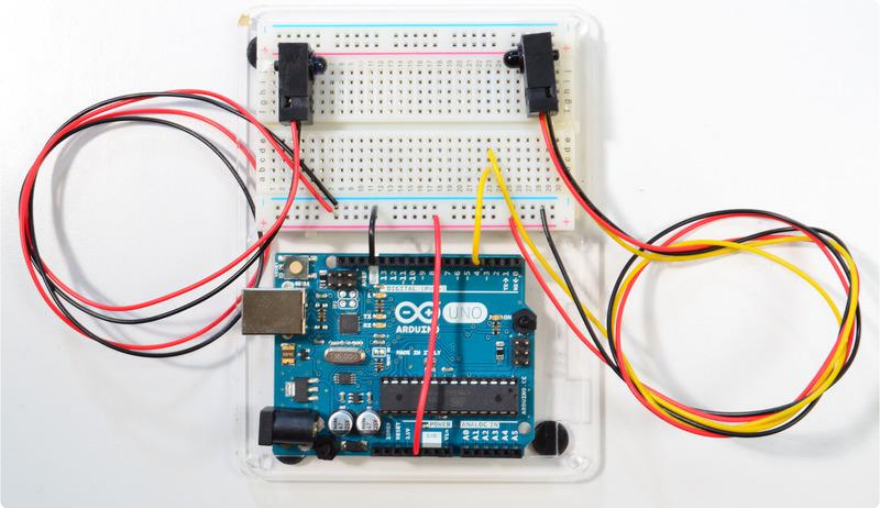
First up you'll need to power the transmitter. Connect the black wire to ground and the red wire directly to 3.3V or 5V power. It will draw 9mA from 3.3V (lower power) and 20mA from 5V (better range)

Next up you'll want to connect up the receiver. Connect the black wire to ground, the red wire to 3.3V or 5V (whichever logic level you like) and then the white or yellow wire to your digital input.

Note that you do not have to share power supply ground or power between the two, the 'signal' is sent optically.

The receiver is open collector which means that you do need a pull up resistor. Most microcontrollers have the ability to turn on a built in pull up resistor. If you do not, connect a 10K resistor between the white wire of the receiver and the red wire.

On an Arduino, we'll connect the signal (yellow/white) pin to Digital #4

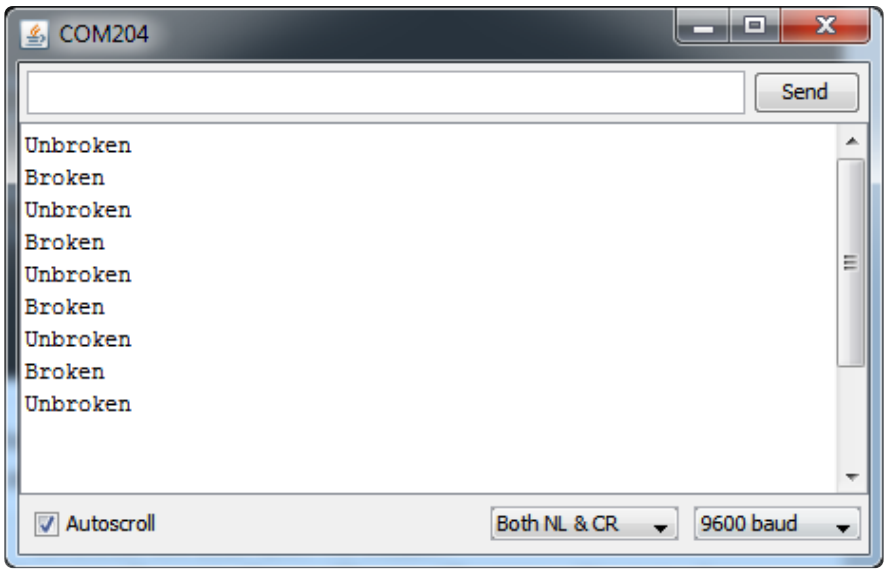


Run this demo code on your Arduino

```
/*
  IR Breakbeam sensor demo!
*/

#define LEDPIN 13
  // Pin 13: Arduino has an LED connected on pin 13
  // Pin 11: Teensy 2.0 has the LED on pin 11
  // Pin  6: Teensy++ 2.0 has the LED on pin 6
  // Pin 13: Teensy 3.0 has the LED on pin 13

#define SENSORPIN 4

// variables will change:
int sensorState = 0, lastState=0;         // variable for reading the pushbutton
status

void setup() {
  // initialize the LED pin as an output:
  pinMode(LEDPIN, OUTPUT);
  // initialize the sensor pin as an input:
  pinMode(SENSORPIN, INPUT);
  digitalWrite(SENSORPIN, HIGH); // turn on the pullup

  Serial.begin(9600);
}

void loop(){
  // read the state of the pushbutton value:
  sensorState = digitalRead(SENSORPIN);

  // check if the sensor beam is broken
  // if it is, the sensorState is LOW:
  if (sensorState == LOW) {
    // turn LED on:
    digitalWrite(LEDPIN, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(LEDPIN, LOW);
  }

  if (sensorState &amp;&amp; !lastState) {
    Serial.println("Unbroken");
  }
  if (!sensorState &amp;&amp; lastState) {
```

```
    Serial.println("Broken");
  }
  lastState = sensorState;
}
```

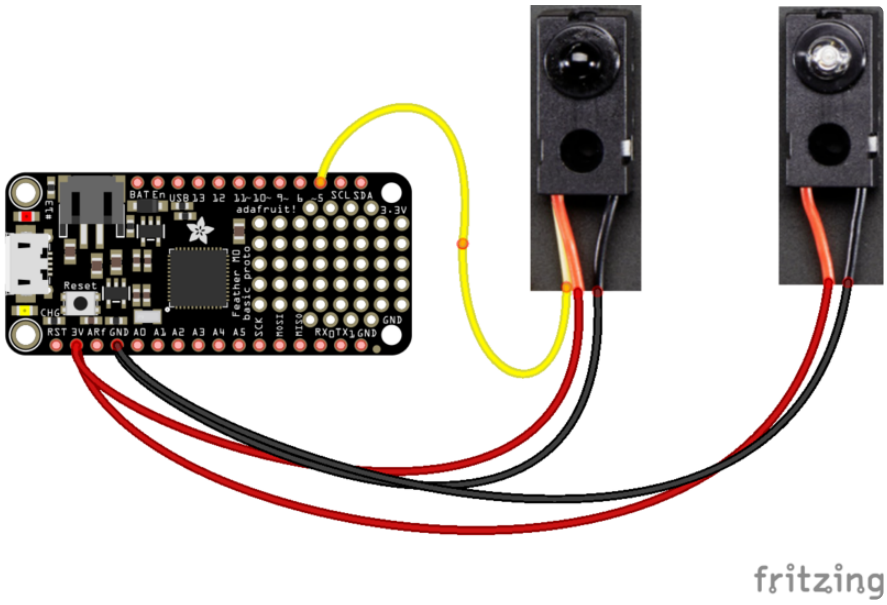
With the above wiring, when you put you hand between the sensor pair, the onboard LED will turn on and the serial console will print out messages:



CircuitPython

It's easy to read a break beam sensor from CircuitPython code using built-in digital input/output capabilities ().

First wire up a break beam transmitter and receiver just like you would for an Arduino. Here's an example of wiring to a Feather M0:



- Board 3V (or 5V if your board has it) to both receiver and transmitter red wire.
- Board GND to both receiver and transmitter black wire.
- Board D5 (or any other digital input) to receiver yellow wire.

Next connect to the board's serial REPL ()so you are at the CircuitPython >>> prompt.

Now import the board and digitalio modules that allow you to create a digital input. Be sure you've read the CircuitPython digital I/O guide () for more background too!

```
import board
import digitalio
```

Create a digital input for the pin connected to the receiver, D5 in this case:

```
break_beam = digitalio.DigitalInOut(board.D5)
break_beam.direction = digitalio.Direction.INPUT
break_beam.pull = digitalio.Pull.UP
```

Notice you set the direction property to input, and the pull property to a pull-up (just like the digital I/O guide mentions ()). This is necessary to configure the digital input with an internal pull-up resistor so it always reads a good value from the break beam sensor.

Checking if the sensor detects a break is as easy as reading the value property of the digital input. When value is true it means the input is at a high logic level which occurs when the receiver can see the transmitter and the beam is not broken. However if you get a value of false the input is at a low logic level which means the receiver cannot see the transmitter and the beam is broken!

Try reading the value with nothing blocking the transmitter and receiver:

```
break_beam.value
```

```
>>> break_beam.value
True
```

Now put something large and opaque, like your hand, in front of the transmitter to block the light. Read the value again:

```
break_beam.value
```

```
>>> break_beam.value
False
>>>
```

Awesome! Notice the digital input value was true, or at a high logic level, when nothing was blocking the beam. As soon as your hand covered the beam the input value turned false, or low logic level, to indicate an obstruction.

You can put all of this together into a complete program that prints a message whenever the beam is blocked. Save this as main.py on your board and examine the serial monitor for output, a message is printed when the beam is blocked:

```
import time

import board
import digitalio


# Create digital input with pull-up resistor on pin D5
# for break beam sensor.
break_beam = digitalio.DigitalInOut(board.D5)
break_beam.direction = digitalio.Direction.INPUT
break_beam.pull = digitalio.Pull.UP

# Main loop runs forever and prints a message once a second
# while the sensor is blocked/broken.
while True:
    if not break_beam.value:
        # Break beam input is at a low logic level, i.e. broken!
        print('Beam is broken!')
    time.sleep(1.0)  # Delay for 1 second and repeat again.
```

That's all there is to reading a beam break sensor with CircuitPython!