IRremote Library

IRremote, by Ken Shirriff, allows you to receive or transmit Infrared Remote Control codes. You can make your projects controlled by a remote, or make them control other devices like televisions and stereo components.

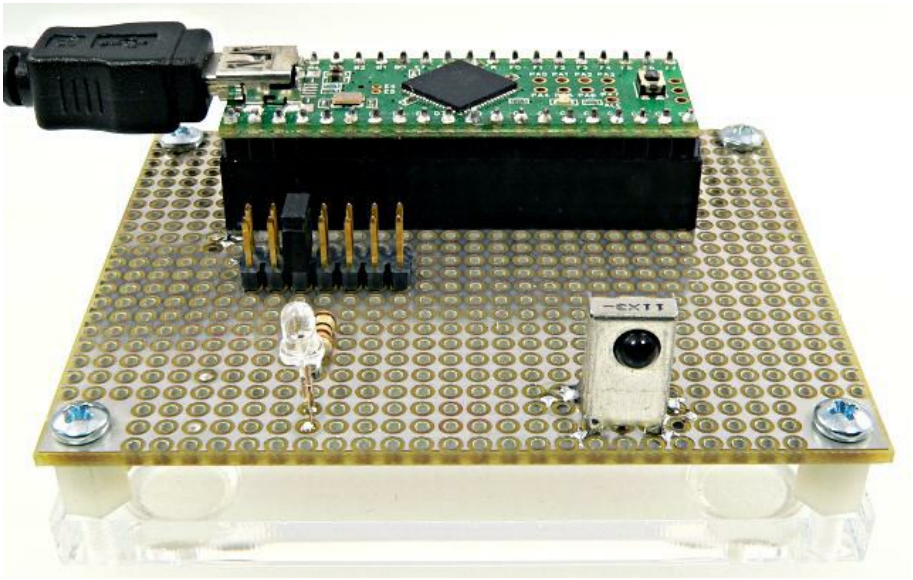
Download: Included with the Teensyduino Installer
Latest Developments on Github

Hardware Requirements

For transmitting, a single Infrared LED and resistor are needed. For receiving, an IR receiver module with internal bandpass filter is needed.

TODO: Add part numbers of known-good infrared LEDs and receivers. The LED in this photo is Lumex OED-EL-8L (Digikey 67-1000-ND) and the receiver is probably Sharp GP1UD281YK0F (now discontinued, Digikey 425-1987-ND).

TODO: Test Vishay TSOP39338 receiver (Digikey 751-1390-5-ND). It's very likely to work. Update this photo. Maybe PJRC should sell a known-good LED & receiver pair?



For transmitting, you must connect the LED to a specific pin. The receiver output may be connected to any pin.

Board	Receive Pin	Transmit Pin	Timer Used	PWM Pins Disabled
Teensy 4.1 / 4.0	Any	8	FlexPWM1.3	None
Teensy 3.6 / 3.5	Any	5	CMT	None
Teensy 3.2 / 3.1	Any	5	CMT	None
Teensy 3.0	Any	5	CMT	None
Teensy LC	Any	16	FTM1	17
Teensy 2.0	Any	10	4	12
Teensy 1.0	Any	17	1	15, 18
Teensy++ 2.0	Any	1	2	0
Teensy++ 1.0	Any	1	2	0

Basic Usage

IRremote acts like 2 libraries, one for sending and one for receiving. Usually it's easiest to find the codes to transmit by first using the receiver.

Receiving

IRrecv irrecv(receivePin)
: Create the receiver object, using a name of your choice.

irrecv.enableIRIn()
: Begin the receiving process. This will enable the timer interrupt which consumes a small amount of CPU every 50 µs.

irrecv.decode(&results)

Attempt to receive a IR code. Returns true if a code was received, or false if nothing received yet. When a code is received, information is stored into "results".

results.decode_type: Will be one of the following: NEC, SONY, RC5, RC6, or UNKNOWN.
results.value: The actual IR code (0 if type is UNKNOWN)
results.bits: The number of bits used by this code
results.rawbuf: An array of IR pulse times
results.rawlen: The number of items stored in the array

irrecv.resume()

After receiving, this must be called to reset the receiver and prepare it to receive another code.

irrecv.blink13(true)

Enable blinking the LED when during reception. Because you can't see infrared light, blinking the LED can be useful while troubleshooting, or just to give visual feedback.

Transmitting

IRsend irsend;

Create the transmit object. A fixed pin number is always used, depending on which timer the library is utilizing.

irsend.sendNEC(IRcode, numBits);

Send a code in NEC format.

irsend.sendSony(IRcode, numBits);

Send a code in Sony format.

irsend.sendRC5(IRcode, numBits);

Send a code in RC5 format.

irsend.sendRC6(IRcode, numBits);

Send a code in RC6

irsend.sendRaw(rawbuf, rawlen, frequency);

Send a raw code. Normally you would obtain the contents of rawbuf and rawlen by using the receiver many times and averaging the results. Some adjustments may be necessary for best performance. The frequency is the expected bandpass filter frequency at the receiver, where 38 is the most commonly used.

Example Program - Receive


```
#include <IRremote.h>

const int RECV_PIN = 6;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
  irrecv.blink13(true);
}

void loop() {
  if (irrecv.decode(&results)) {
    if (results.decode_type == NEC) {
      Serial.print("NEC: ");
    } else if (results.decode_type == SONY) {
      Serial.print("SONY: ");
    } else if (results.decode_type == RC5) {
      Serial.print("RC5: ");
    } else if (results.decode_type == RC6) {
      Serial.print("RC6: ");
    } else if (results.decode_type == UNKNOWN) {
      Serial.print("UNKNOWN: ");
    }
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

Example Program - Transmit

```
#include <IRremote.h>

IRsend irsend;

void setup() {
}

void loop() {
    irsend.sendSony(0x68B92, 20);
    delay(100);
    irsend.sendSony(0x68B92, 20);
    delay(100);
    irsend.sendSony(0x68B92, 20);
    delay(300000);
}
```

Timer Usage

IRremote requires a timer for both transmitting and receiving. If you wish to use another library that requires the same timer, you can edit IRremoteInt.h to make IRremote use a different timer.

Details

For more details, please refer to Ken's pages:

- Latest IRremote code at Github
- A Multi-Protocol Infrared Remote Library for the Arduino
- Understanding Sony IR remote codes, LIRC files, and the Arduino library
- Detecting an IR Beam Break with the Arduino IR Library
- Don't walk! Controlling a pedestrian sign with an Arduino
- Using arbitrary remotes with the Arduino IRremote library
- Controlling your stereo over the web with the Arduino infrared library
- IR Bubbles: Controlling a relay with an Arduino and an IR remote
- An Arduino universal remote: record and playback IR signals