Ken Shirriff's blog

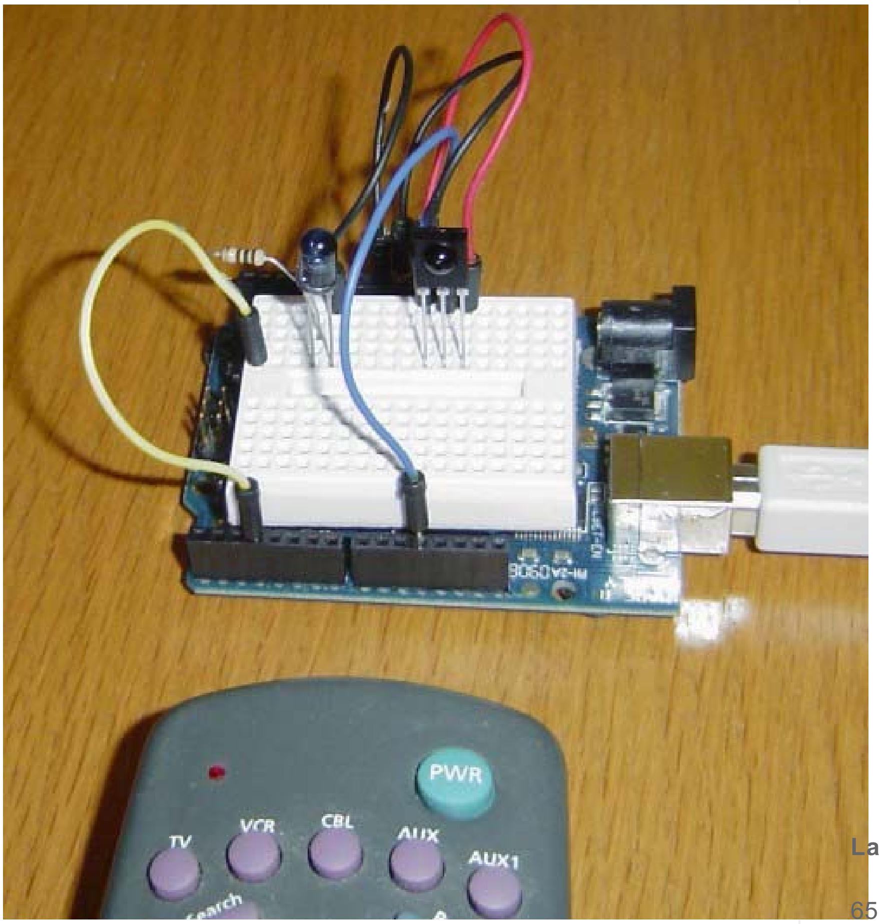
Chargers, microprocessors, Arduino, and whatever

A Multi-Protocol Infrared Remote Library for the Arduino

Code now on github

The most recent code is at github.com/shirriff/Arduino-IRremote. If you have any issues, please report them there.

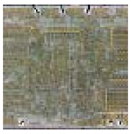
Do you want to control your Arduino with an IR remote? Do you want to use your Arduino to control your stereo or other devices? This IR remote library lets you both send and receive IR remote codes in multiple protocols. It supports NEC, Sony SIRC, Philips RC5, Philips RC6, and raw protocols. If you want additional protocols, they are straightforward to add. The library can even be used to record codes from your remote and re-transmit them, as a minimal universal remote.



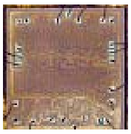
To use the library, download from github and follow the installation instructions in the readme.

How to send

Popular Posts



Die photos and analysis of the revolutionary 8008 microprocessor, 45 years old



Inside the 74181 ALU chip: die photos and reverse engineering

A Multi-Protocol Infrared Remote Library for the Arduino

Apple iPhone charger teardown: quality in a tiny expensive package

A dozen USB chargers in the lab: Apple is very good, but not quite the best

Fixing the core memory in a vintage IBM 1401 mainframe

Macbook charger teardown: The surprising complexity inside Apple's power adapter

The Texas Instruments TMX 1795: the (almost) first, forgotten microprocessor

Labels

6502 8085 alto apple arc arduino arm beaglebone bitcoin c# calculator css electronics f# fractals genome haskell html5 ibm1401 ipv6 ir java javascript math

This infrared remote library consists of two parts: IRsend transmits IR remote packets, while IRrecv receives and decodes an IR message. IRsend uses an infrared LED connected to output pin 3. To send a message, call the send method for the desired protocol with the data to send and the number of bits to send. The `examples/IRsendDemo` sketch provides a simple example of how to send codes:

```
#include <IRremote.h>
IRsend irsend;

void setup()
{
  Serial.begin(9600);
}

void loop() {
  if (Serial.read() != -1) {
    for (int i = 0; i < 3; i++) {
      irsend.sendSony(0xa90, 12); // Sony TV power code
      delay(100);
    }
  }
}
```

This sketch sends a Sony TV power on/off code whenever a character is sent to the serial port, allowing the Arduino to turn the TV on or off. (Note that Sony codes must be sent 3 times according to the protocol.)

How to receive

IRrecv uses an infrared detector connected to any digital input pin.

The `examples/IRrecvDemo` sketch provides a simple example of how to receive codes:

```
#include <IRremote.h>

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

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

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

The `IRrecv` class performs the decoding, and is initialized with `enableIRIn()`. The `decode()` method is called to

oscilloscope photo power supply random reverse-engineering sheevaplug snark spanish teardown theory unicode Z-80

Power supply posts

- iPhone charger teardown
- A dozen USB chargers
- Magsafe hacking
- Inside a fake iPhone charger
- Power supply history

Email:
kens@arcfn.com

Blog Archive

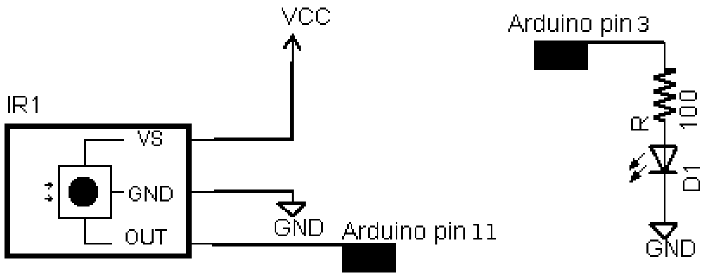
- ► 2017 (1)
- ► 2016 (34)
- ► 2015 (12)
- ► 2014 (13)
- ► 2013 (24)
- ► 2012 (10)
- ► 2011 (11)
- ► 2010 (22)
- ▼ 2009 (22)
  - ► December (2)
  - ► November (5)
  - ► September (1)
  - ▼ August (3)
    - A Multi-Protocol Infrared Remote Library for the A...
    - Arc + Arduino + ARM: Temperature monitoring
    - World's smallest Arc server
  - ► July (1)
  - ► June (3)
  - ► April (1)
  - ► March (3)
  - ► February (2)
  - ► January (1)
- ► 2008 (27)

Quick links

see if a code has been received; if so, it returns a nonzero value and puts the results into the `decode_results` structure. (For details of this structure, see the `examples/IRrecvDump` sketch.) Once a code has been decoded, the `resume()` method must be called to resume receiving codes. Note that `decode()` does not block; the sketch can perform other operations while waiting for a code because the codes are received by an interrupt routine.

Hardware setup

The library can use any of the digital input signals to receive the input from a 38KHz IR receiver module. It has been tested with the Radio Shack 276-640 IR receiver and the Panasonic PNA4602. Simply wire power to pin 1, ground to pin 2, and the pin 3 output to an Arduino digital input pin, e.g. 11. These receivers provide a filtered and demodulated inverted logic level output; you can't just use a photodiode or phototransistor. I have found these detectors have pretty good range and easily work across a room.



For output, connect an IR LED and appropriate resistor to PWM output pin 3. Make sure the polarity of the LED is correct, or it won't illuminate - the long lead is positive. I used a NTE 3027 LED (because that's what was handy) and 100 ohm resistor; the range is about 15 feet. For additional range, you can amplify the output with a transistor.

Some background on IR codes

An IR remote works by turning the LED on and off in a particular pattern. However, to prevent inteference from IR sources such as sunlight or lights, the LED is not turned on steadily, but is turned on and off at a modulation frequency (typically 36, 38, or 40KHz). The time when a modulated signal is being sent will be called a mark, and when the LED is off will be called a space.

Each key on the remote has a particular code (typically 12 to 32 bits) associated with it, and broadcasts this code when the key is pressed. If the key is held down, the remote usually repeatedly broadcasts the key code. For an NEC remote, a special repeat code is sent as the key is held down, rather than repeatedly sending the code. For Philips RC5 or RC6 remotes, a bit in the code is toggled each time a key is pressed; the receiver uses this toggle bit to determine when a key is pressed down a second time.

On the receiving end, the IR detector demodulates this signal, and outputs a logic-level signal indicating if it is receiving a

- Arduino IR library
- 6502 reverse-engineering

signal or not. The IR detector will work best when its frequency matches the sender's frequency, but in practice it doesn't matter a whole lot.

The best source I've found for details on the various types of IR codes is SB IR knowledge base.

Handling raw codes

The library provides support for sending and receiving raw durations. This is intended mainly for debugging, but can also be used for protocols the library doesn't implement, or to provide universal remote functionality.

The raw data for received IR measures the duration of successive spaces and marks in 50us ticks. The first measurement is the gap, the space before the transmission starts. The last measurement is the final mark.

The raw data for sending IR holds the duration of successive marks and spaces in microseconds. The first value is the first mark, and the last value is the last mark.

There are two differences between the raw buffers for sending and for receiving. The send buffer values are in microseconds, while the receive buffer values are in 50 microsecond ticks. The send buffer starts with the duration of the first mark, while the receive buffer starts with the duration of the gap space before the first mark. The formats are different because I considered it useful for the library to measure gaps between transmissions, but not useful for the library to provide these gaps when transmitting. For receiving, 50us granularity is sufficient for decoding and avoids overflow of the gaps, while for transmitting, 50us granularity is more than 10% error so 1us granularity seemed better.

Obtaining codes for your remote

The easiest way to obtain codes to work with your device is to use this library to decode and print the codes from your existing remote.

Various libraries of codes are available online, often in proprietary formats. The Linux Infrared Remote Control project (LIRC), however, has an open format for describing codes for many remotes. Note that even if you can't find codes for your exact device model, a particular manufacturer will usually use the same codes for multiple products.

Beware that other sources may be inconsistent in how they handle these protocols, for instance reversing the order, flipping 1 and 0 bits, making start bits explicit, dropping leading or trailing bits, etc. In other words, if the IRremote library yields different codes than you find listed elsewhere, these inconsistencies are probably why.

Details of the receiving library

The IRrecv library consists of two parts. An interrupt routine is called every 50 microseconds, measures the length of the marks and spaces, and saves the durations in a buffer. The user calls a decoding routine to decode the buffered measurements into the code value that was sent (typically 11 to 32 bits).

The decode library tries decoding different protocols in succession, stopping if one succeeds. It returns a structure that contains the raw data, the decoded data, the number of bits in the decoded data, and the protocol used to decode the data.

For decoding, the MATCH macro determine if the measured mark or space time is approximately equal to the expected time.

The RC5/6 decoding is a bit different from the others because RC5/6 encode bits with mark + space or space + mark, rather than by durations of marks and spaces. The getRClevel helper method splits up the durations and gets the mark/space level of a single time interval.

For repeated transmissions (button held down), the decoding code will return the same decoded value over and over. The exception is NEC, which sends a special repeat code instead of repeating the transmission of the value. In this case, the decode routine returns a special REPEAT value.

In more detail, the receiver's interrupt code is called every time the TIMER1 overflows, which is set to happen after 50 microseconds. At each interrupt, the input status is checked and the timer counter is incremented. The interrupt routine times the durations of marks (receiving a modulated signal) and spaces (no signal received), and records the durations in a buffer. The first duration is the length of the gap before the transmission starts. This is followed by alternating mark and space measurements. All measurements are in "ticks" of 50 microseconds.

The interrupt routine is implemented as a state machine. It starts in STATE_IDLE, which waits for the gap to end. When a mark is received, it moves to STATE_MARK which times the duration of the mark. It then alternates between STATE_MARK and STATE_SPACE to time marks and spaces. When a space of sufficiently long duration is received, the state moves to STATE_STOP, indicating a full transmission is received. The interrupt routine continues to time the gap, but blocks in this state.

The STATE_STOP is used a a flag to indicate to the decode routine that a full transmission is available. When processing is done, the resume() method sets the state to STATE_IDLE so the interrupt routine can start recording the next transmission. There are a few things to note here. Gap timing continues during STATE_STOP and STATE_IDLE so an accurate measurement of the time between transmissions can be obtained. If resume() is not called before the next transmission starts, the partial transmission will be discarded. The motivation behind the stop/resume is to ensure the receive buffer is not overwritten while it is still being processed; debugging becomes very difficult if the buffer is constantly changing.

Details of the sending library

The transmission code is straightforward. To ensure accurate output frequencies and duty cycles, I use the PWM timer, rather than delay loops to modulate the output LED at the appropriate frequency. (See my Arduino PWM Secrets article for more details on the PWM timers.) At the low level, enableIROut sets up the timer for PWM output on pin 3 at the proper frequency. The mark() method sends a mark by enabling PWM output and delaying the specified time. The space() method sends a space by disabling PWM output and delaying the specified time.

The IRremote library treats the different protocols as follows:

NEC: 32 bits are transmitted, most-significant bit first. (protocol details)

Sony: 12 or more bits are transmitted, most-significant bit first. Typically 12 or 20 bits are used. Note that the official protocol is least-significant bit first. (protocol details) For more details, I've written an article that describes the Sony protocol in much more detail: Understanding Sony IR remote codes.

RC5: 12 or more bits are transmitted most-significant bit first. The message starts with the two start bits, which are not part of the code values. (protocol details)

RC6: 20 (typically) bits are transmitted, most-significant bit first. The message starts with a leader pulse, and a start bit, which is not part of the code values. The fourth bit is transmitted double-wide, since it is the trailer bit. (protocol details)

For Sony and RC5/6, each transmission must be repeated 3 times as specified in the protocol. The transmission code does not implement the RC5/6 toggle bit; that is up to the caller.

Adding new protocols

Manufacturers have implemented many more protocols than this library supports. Adding new protocols should be straightforward if you look at the existing library code. A few tips: It will be easier to work with a description of the protocol rather than trying to entirely reverse-engineer the protocol. The durations you receive are likely to be longer for marks and shorter for spaces than the protocol suggests. It's easy to be off-by-one with the last bit; the last space may be implicit.

Troubleshooting

To make it easier to debug problems with IR communication, I have optional debugging code in the library. Add `#define DEBUG` to the beginning of your code to enable debugging output on the serial console. You will need to delete the `.o` files and/or restart the IDE to force recompilation.

Problems with Transmission

If sending isn't working, first make sure your IR LED is actually transmitting. IR will usually show up on a video camera or cell phone camera, so this is a simple way to check. Try putting the LED right up to the receiver; don't expect a lot of range unless you amplify the output.

The next potential problem is if the receiver doesn't understand the transmitter, for instance if you are sending the wrong data or using the wrong protocol. If you have a remote, use this library to check what data it is sending and what protocol it is using.

An oscilloscope will provide a good view of what the Arduino or a remote is transmitting. You can use an IR photodiode to see what is getting transmitted; connect it directly to the oscilloscope and hold the transmitter right up to the photodiode. If you have an oscilloscope, just connect the oscilloscope to the photodiode. If you don't have an oscilloscope, you can use a sound card oscilloscope program such as xoscope.

The Sony and RC5/6 protocols specify that messages must be sent three times. I have found that receivers will ignore the message if only sent once, but will work if it is sent twice. For RC5/6, the toggle bit must be flipped by the calling code in successive transmissions, or else the receiver may only respond to a code once.

Finally, there may be bugs in this library. In particular, I don't have anything that receives RC5/RC6, so they are untested.

Problems with Receiving

If receiving isn't working, first make sure the Arduino is at least receiving raw codes. The LED on pin 13 of the Arduino will blink when IR is being received. If not, then there's probably a hardware issue.

If the codes are getting received but cannot be decoded, make sure the codes are in one of the supported protocols. If codes should be getting decoded, but are not, some of the measured times are probably not within the 20% tolerance of the expected times. You can print out the minimum and maximum expected values and compare with the raw measured values.

The `examples/IRrecvDump` sketch will dump out details of the received data. The dump method dumps out these durations but converts them to microseconds, and uses the convention of prefixing a space measurement with a minus sign. This makes it easier to keep the mark and space measurements straight.

IR sensors typically cause the mark to be measured as longer than expected and the space to be shorter than expected. The code extends marks by 100us to account for this (the value MARK_EXCESS). You may need to tweak the expected values or tolerances in this case.

The library does not support simultaneous sending and receiving of codes; transmitting will disable receiving.

Applications

I've used this library for several applications:

- Controlling a pedestrian sign with a remote.
- I extend the library to arbitrary remotes.
- Controlling my stereo over the web.
- I've used this library to implement a "Universal remote" to record and playback IR codes.
- I demonstrate turning something on and off via remote in my infrared bubble maker project.
- Using the library to detect IR beam breaks

Other projects that use this library

- electrosthetics used the library to control a home theater receiver.
- Arduino: Redefining the TV Remote - using the library and an ultrasonic sensor to control a TV by waving your hand.

Other Arduino IR projects

I was inspired by Building a Universal Remote with an Arduino; this doesn't live up to being a universal remote, but has a lot of information. The NECIRrcv library provided the interrupt handling code I use.

Labels: arduino, ir

858 comments:

1 – 200 of 858 Newer› Newest»

Jack said...

Wow this is great, I have been looking for something like this for awhile now.

Thanks so much for posting!!!

I am having trouble sending the values from my remote through the send function.

Specifically I recorded the "volume down" signal from my macbook remote (77E130DA).

I then tried sending that with this code:

irsend.sendRaw(0x77E130DA, 12, 36);

But that would not compile.

My programming skills are low to medium so I could be missing something completely obvious here. Any help would be much appreciated.

My project is to turn my iPhone into my television remote using safari on the iphone > iobridge > arduino > the TV

Your code seems perfect for this so I am really excited to get the codes transmitting properly.

Thanks in advance for your help

September 1, 2009 at 3:42 PM

Ken Shirriff said...

Hi Jack! Thanks for your comments. sendRaw only works when you explicitly give an array of on and off durations. I think the macbook uses the NEC protocol, so try using irsend.sendNEC(...).

September 1, 2009 at 4:24 PM

Jack said...

Thanks Ken I'll tinker around with that.

Is there a way to capture the raw signal to resend with sendRaw?

Basically so you could copy and paste the codes

September 2, 2009 at 12:24 PM

Harrison said...

This is great!

Could you please tell me how I can change the output pin? (I have an arduino clone with pin 13 as LED and IR-Tx)

September 5, 2009 at 2:42 PM

Ken Shirriff said...

Jack: try the IRdump example to dump out the raw data.

Harrison: you have to use pin 3 as the IR LED output because that's the PWM pin. (You could use a different PWM pin, but it would take substantial modification to the code.)

September 5, 2009 at 3:36 PM

alsterg said...

Great work! Does this library support also operation with multiple IR receivers?

September 6, 2009 at 5:29 AM

Arvin said...

I've been using a picaxe to tx\rx ir but that only works with sony. Your's would allow me to do other remote controls.

So I put the code you show in the IRrecv into a arduino sketch and it errors on "lessthanresults". Can't use lessthan!

I've tried * and nothing there and get errors both ways.

What should it be?

September 13, 2009 at 8:06 AM

Arvin said...

Back again. Read the code in one of your example files. It used & in front of the "result". That gives me a "redefinition of 'int RECV_PIN'" error.

September 13, 2009 at 8:26 AM

Ken Shirriff said...

This comment has been removed by the author.

September 13, 2009 at 9:08 AM

Ken Shirriff said...

Arvin: Please look at the code in the zip file; I think I messed up the HTML formatting in the code snippet in the article.

September 13, 2009 at 9:38 AM

Giorgio O said...

Thanks for the library!
I'm trying to use the example IRrecord.pde.
Everything works fine on a superficial level: the elctronic circuit is ok, I'm receiving the infrared data from the remote control of a sat decoder box and I'm printing it on the serial window... Then when I try to play it back nothings happens (I'm not able to trigger the satellite decoder with my IR led from the arduino). If I look at the ir led with my camera I see it pulsing... so my question is: since the library is sampling the ir signal from the remote control with a resolution of 50microseconds, is it possible that the raw data recorded is not usable for the playback of the same signal? Would it be possible to increase the sampling resolution?

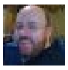
September 16, 2009 at 6:35 PM

australopitecus said...

Hi.
I really like this project. Thank you very much. It helps me a lot.
But I have one problem. Everything works, except sending the IR pulse.Neither in the example IRrecord nor with IRsendDemo.
There´s no output at the PWM pin 3. I´ve tested it with a camera, a LED instead of the IR-diode and the oscilloscope. There´s definitely no signal.
First i mentioned there´s something wrong with the sourcecode. But it´s programmed really sophisticated

and i couldn´t find a mistake, though i consider myself a quite acceptable programmer.Sorry for self-praise.
But Giorgio O wrote, the diode is sending. So there´s another problem.
Is there anything I could have forgot or has anybody else had a similar problem? I´m trying for 4 days....
I use the Arduino mega board with the 0017 software.
Excuse my mistakes in English. I´m not a native speaker.
Thanks for help..

September 17, 2009 at 12:47 AM



Wolfgang said...

I added the Panasonic IR protocol.

Constants:

```
#define PANASONIC_HDR_MARK 3502
#define PANASONIC_HDR_SPACE 1750
#define PANASONIC_BIT_MARK 502
#define PANASONIC_ONE_SPACE 1244
#define PANASONIC_ZERO_SPACE 370
```

Sending function:

```
void IRsend::sendPanasonic(unsigned long address,
unsigned long data) {
enableIROut(38);
mark(PANASONIC_HDR_MARK);
space(PANASONIC_HDR_SPACE);
for (int i=0; i < 32; i++) {
mark(PANASONIC_BIT_MARK);
if (address & 0x80000000) {
space(PANASONIC_ONE_SPACE);
} else {
space(PANASONIC_ZERO_SPACE);
}
address <<= 1;
}
for (int i=0; i < 16; i++) {
mark(PANASONIC_BIT_MARK);
if (data & 0x8000) {
space(PANASONIC_ONE_SPACE);
} else {
space(PANASONIC_ZERO_SPACE);
}
data <<= 1;
}
mark(PANASONIC_BIT_MARK);
space(30000);
space(30000);
space(14000);
}
```

Receiving funtion:

```
long IRrecv::decodePanasonic(decode_results *results) {
unsigned long data = 0;
int offset = 1;

if (!MATCH_MARK(results->rawbuf[offset],
PANASONIC_HDR_MARK)) {
return ERR;
}
offset++;
if (!MATCH_MARK(results->rawbuf[offset],
PANASONIC_HDR_SPACE)) {
return ERR;
}
offset++;

// decode address
for (int i = 0; i < 32; i++) {
if (!MATCH_MARK(results->rawbuf[offset++],
PANASONIC_BIT_MARK)) {
return ERR;
}
if (MATCH_SPACE(results-
>rawbuf[offset],PANASONIC_ONE_SPACE)) {
data = (data << 1) | 1;
} else if (MATCH_SPACE(results-
>rawbuf[offset],PANASONIC_ZERO_SPACE)) {
data <<= 1;
} else {
return ERR;
}
offset++;
}
results->address = data;
data = 0;
for (int i = 0; i < 16; i++) {
if (!MATCH_MARK(results->rawbuf[offset++],
PANASONIC_BIT_MARK)) {
return ERR;
}
if (MATCH_SPACE(results-
>rawbuf[offset],PANASONIC_ONE_SPACE)) {
data = (data << 1) | 1;
} else if (MATCH_SPACE(results-
>rawbuf[offset],PANASONIC_ZERO_SPACE)) {
data <<= 1;
} else {
return ERR;
}
offset++;
}
results->value = data;

results->decode_type = PANASONIC;
results->bits = 48;
return DECODED;
}
```

September 17, 2009 at 12:42 PM

Ken Shirriff said...

Giorgio: there are several reasons your sat decoder box may not respond. At the low level, it might be using a different modulation frequency than 38KHz, or a different IR wavelength. Higher level, it may be expecting more than just the played-back signal. For instance, RC5 alternates between two different codes, so simple playback won't work. Sony codes require the code to be transmitted at least twice. Your box may have its own strange requirements, such as different codes with particular timings. It could be the 50us resolution, but I've found that generally isn't a problem. More likely is IR sensors typically extend the on time and shrink the off time; my code tries to correct for this, but the correction factor could be different for your sensor. Unfortunately, there are a lot of variables to investigate. Also, try the circuit with your TV or stereo, to see if it works there. If you let me know what model satellite box you're trying to control, I could investigate a bit.

australopitecus: My code won't work with an Arduino mega as it has a different processor and the pins are all different. (Sorry I didn't mention that in my original article.) You could see if there's anything on PWM pin 9; I believe that's where the OC2B output connects on the mega. Probably you'll need to study the atmega 1280 datasheet and see how the PWM flags are different for the mega's processor, and change the code appropriately. Unfortunately I don't have a mega, so I can't try this myself.

Wolfgang: thanks for the code update.

September 17, 2009 at 11:28 PM

australopitecus said...

Hi.
That was the problem, yes.
Today I found in the o-file "pins_arduino.c.o" the path to the "cores" folder and a file named "pins_aruino.c". According to this, in the digital_pin_to_timer, which is a part of the program memory, I think, Timer2B is related to PWM9. I tried it and it worked. I had only to change the output of the pinMode.
I don´t know if that´s exactly the same you mentioned, but I have a signal now. So I can receive and send.
Thanks to you, this saved my weekend.
I love the internet.
And also thanks to Wolfgang for the Panasonic protocol.

September 18, 2009 at 2:55 AM

reconnnn said...

Hi
I try to control my LG tv using a arduino and I found you lib and i think it would work great. I also found my tv ir codes on LIRC but I have no idea how im going to use the information at LIRC to controle my tv.

Thanks in advance for your help!!

September 26, 2009 at 5:17 AM

Ken Shirriff said...

Hi reconnnn: I'm in the process of writing an explanation of LIRC data, but for your case LG usually uses 32-bit NEC protocol. Take the pre_data bytes in the LIRC file and join them to the bytes for the button you want to control to make a 4-byte hex number, and use sendNEC. Power on is probably 0x20df10ef; some LG devices use 0x897641be or 0x34347887. If you have an

IR detector, you can read out what your remote sends, rather than looking in the LIRC files.

September 26, 2009 at 3:17 PM

reconnnn said...

Hi Thanks for that quick answer Ken Shirriff

I found this
http://lirc.sourceforge.net/remotes/lg/MKJ40653802
document and it say that power should be 0x10EF am i right then?
And should i use 0xD02F for input?

Thanks again for the help!!
//Reconnn

September 26, 2009 at 4:03 PM

Ken Shirriff said...

reconnn: The NEC protocol uses constant first two bytes to indicate the device (lirc pre_data 0x20df in your case), and different last two bytes (the lirc button code) to indicate the action.

So power on/off for your LG TV would probably be irsend.sendNEC(0x20df10ef, 32), volume up would be irsend.sendNEC(0x20df40bf, 32), volume down would be irsend.sendNEC(0x20dfc03f, 32), etc.

September 26, 2009 at 4:31 PM

Todd Treece said...

Thanks for posting this! I have been looking for info on how to do this, but I thought I needed an oscilloscope to decode the IR signals.

I have gotten almost all of my devices to work with your code, but I can't seem to figure out what protocol my Sharp LCD TV is using.

I believe that this LIRC remote file is close to my model:
http://lirc.sourceforge.net/remotes/sharp/GA538WJSA

Do you happen to know how I could control the TV using your library?

October 15, 2009 at 9:48 PM

MicMac said...

Thanks for this great great Library.
A few observations I made:
Receiving does not work with official Ethernet-Shield, but I had no problems sending. The Problem (I guess) is that SPI (pins 10 - 13) is needed for commucation with the ethernet shield and also used for the ir communication. But it might also be interrupt or timer related.
I will take a deeper look at it, but atm I'm covered in

"look-at's", so no promises.
Again thanks and keep the good work flowing ;)

November 9, 2009 at 7:44 PM

Ken Shirriff said...

MicMac, the Ethernet shield uses pins 10-13, so the IR library should work if you use any other pins for input. The library requires pin 3 for output, but can use any digital I/O pin for input.

November 10, 2009 at 7:54 AM

Anonymous said...

Hi,

Thanks for the library. I read through the code and understand its functions, sending and receiving. But just one question:

Why "MARK" is defined as 0 and "SPACE" is defined as 1? I thought it would be the other way around, since the sending functions turn ON the PWM pin when sending a MARK and OFF when sending a SPACE.

Thanks.

November 24, 2009 at 8:36 PM

Anonymous said...

Never mind. I got it now why the MARK is defined as 1 and SPACE is as 0 for the receiving side. It's because the the receiver is active low. (It should be as named. Don't pay attention to the actual values for easier understanding.)

I had refactored the code into little tiny C modules for sending and receiving, so that you can choose to link a particular sender and receiver with very small flash size that fits to 2K ATTINIES, such as, the ATTINY25.

Also, the code is now Arduino-free, meaning running on any Atmel chip, as long as you chnage interrupt part of the code. (I had to write one simple module that support digitalWrite()/digitalRead()/pinMode() functions.)

I had also optimized to code to do bit packing of the port info into 16 bits, as well as misc info, for example, when using uint8_t instead of uint16_t.

The core functionality is exactly the same as the original code, without change.

I frankly haven't tested the original code by Ken, the author, because the compiled hex file won't fit into an ATMEGA8/ATTINY25.

But I tested out my stuff.

I do have one question in general.

The moment the receiver receives sufficent IR data to get it into the STATE_STOP state, I have noticed that it took almost 4 secs (SECS with an ATMEGA8 at 16Mhz) to actually to have irparams->timer > GAP_TICKS, which is calculated out to be 100, thus, 100 * 50us = 5000us = 5 ms, but it really shouldn't be 4 secs between the moment I finish sending some IR data (via a remote control) to the time STATE_STOP is set. Why?

Thanks for any tips.

November 26, 2009 at 7:34 PM

edward said...

I compiled the demo (Sony TV power), and modified it to send a on/off every 1 second. Only one in about 60 work and I'm 3 feet from the TV. I took a peek at the LED output via a digital camera and it looks very dim. I've removed the resistor on the LED and it's still noticably less bright than a regular remote (viewed through the camera, and it only works at all this way). Swapped in another LED - same thing.

Ideas?

December 6, 2009 at 3:01 AM

Ken Shirriff said...

Edward: two ideas on your problem. First, Sony codes need to be sent at least twice or the receiver will ignore it. Try something like:

```
irsend.sendSony(code, 12);
delay(50);
irsend.sendSony(code, 12);
delay(50);
irsend.sendSony(code, 12)
```

Second, add a driver transistor to increase the IR power, because the Arduino chip outputs have limited current. See, for example the TV-B-Gone design.

December 6, 2009 at 8:25 AM

edward said...

Ah ha!

You indirectly revealed the "bug". I was sending the code 3x (your Sony power code snippet) with the 100ms delay between pulses, every one second. I reduced your 100ms to 50ms and it worked. So you might consider changinge your sample snippet to 50ms delays between pulses.

Thanks a ton for the quick response. I'm now unblocked.

Cheers from Seattle, WA

Edward

December 6, 2009 at 2:18 PM

Randy Wallace said...

I've been using this library for a while; I'm doing a project that involves using the Arduino as a webserver for controlling IR output. Some interesting things I'm implementing are the webduino libraries, EEPROM for storing received values, and a 8x2 LCD.

One really important thing I've noticed is that the Arduino cannot provide enough power from the Digital Out pins. I'm using a Stock IR LED from Radio Shack, which can operate at up to 100mA at 1.2 Volts. For more power, I am using a 2N2222 transistor: the base is in series with a 2K resistor and the digital output pin; the emitter goes to ground; and, the collector is in series with a 47 ohm resistor, the IR LED, and the 5V supply. I haven't ran this through a ammeter; regardless, on paper there is more than enough power, and it works well for me, regardless of how much power i'm drawing from the power supply.

Let me report, also, that I haven't experienced a code, received and sent by these libraries, that doesn't work. Kudos, my friend, Kudos!

December 12, 2009 at 8:35 AM

Anonymous said...

I was receiving codes and then sending codes and I couldn't figure out why it would hang after sending a code. The reason is that I didn't know that you needed to use the irrecv.enableIRIn() after every send before you can receive again. Spent many hours trying to figure out what was going on .

Thanks for the great library !

December 15, 2009 at 11:35 AM

Jeko said...

Yesterday i finally discovered your amazing IRLibrary for Arduino... I would like to thank you for sharing your job... it's by far best IRLibrary for Arduino, it works very very well and it's very easy to use, too.

After playing with it a bit, controlling leds, switches and anything that could be attached to my arduino, I tried to use it for controlling some DMX devices... it would be amazing for me to control my house illumination with an IRRemote and some standard DMX equipment (I've got a lot of DMX devices and i love to deal with them, DMX works very very well in many environments)

I use this DMX Library
http://code.google.com/p/tinkerit/wiki/DmxSimple

Sadly, i soon discovered that both the IRLibrary and the DMX library use Timer2, in a different, different way, and as far as I've seen they won't work together

my programming skills are not advanced enough to get out of this issue... and i was wondering if you have some ideas

I think it would be very useful to use arduino to control DMX devices via IRRemote, because DMX controllers are usually very expensive and not so customizable, and there are hundreds of DMX devices that asks only to be controlled :)

Thank you again for sharing your great job, and happy new year!

December 30, 2009 at 8:38 AM

Michael Gold said...

Love this library! I ran into a little hitch that caused me some confusion, but I eventually found a work-around that I wanted to mention here. While using the library to enable an IR receiver (salvaged from an old VCR) while elsewhere in my code also driving an RGB LED using other PWM pins (on pin 9,10,11 for r,g,b) I got some odd behavior: I couldn't control 'green' on pin 11). I'm a bit of a newbie at this, but it seemed like a timing issue affecting the PWM.

Looking at the library code, I could see that it manipulates one of the microcontroller's timers (timer2, via TCCR2A and TCCR2B) in order to properly sense the incoming IR signal. Through Googling I came across the PWM cheatsheet on the arduino playground (http://www.arduino.cc/playground/Main/TimerPWMCheatsheet) which mentions that timer2 handles PWM timing for pin 3 and 11. Since I originally had my 'Green' connection on pin 11, the cheatsheet seemed to indicate that it was affected by the library's use of timer2. Sure enough, moving the Green conenction to to another PWM pin (not handled by timer2) seemed to solve my issue.

So, if one uses this library, looks like PWM on pin11 is not possible, given the library's use of timer2. A small limitation on an otherwise incredibly useful library!

Many thanks,
Michael

ps: I saw the comment above regarding the DMX controller library conflict because it uses timer2. Wondering: In order to solve the conflict - or, as above, if one absolutely needs PWM on pin11 - could one modify the IR library to just use timer1 (ex: TCCR1A,TCCR1B, OCR1A, OCR1B etc) or timer0 (ex: TCCR0A,TCCR0B, OCR0A, OCR0B etc)?

January 6, 2010 at 10:55 AM

insulated copper wire said...

I really like this project. Thank you very much. It helps me a lot.
But I have one problem. Everything works, except sending the IR pulse.Neither in the example IRrecord

nor with IRsendDemo.There´s no output at the PWM pin 3. I´ve tested it with a camera, a LED instead of the IR-diode and the oscilloscope. There´s definitely no signal.

January 6, 2010 at 11:28 AM

Jon said...

Awesome library!

I've confirmed that the Panasonic code posted above works but there are a few quirks.

You need to modify the raw buffer size in IRremote.h to accommodate the long input as follows:

```
#define RAWBUF 100
```

After that it worked fine in debug mode.

Then I had to remove the match macros and add standard functions in IRremote.cpp as follows:

```
#ifndef DEBUG
int MATCH(int measured, int desired) {return measured
>= TICKS_LOW(desired) && measured <=
TICKS_HIGH(desired);}
int MATCH_MARK(int measured_ticks, int desired_us)
{return MATCH(measured_ticks, (desired_us +
MARK_EXCESS));}
int MATCH_SPACE(int measured_ticks, int desired_us)
{return MATCH(measured_ticks, (desired_us -
MARK_EXCESS));}
#endif
```

Before the modification it would only successfully decode a panasonic signal in debug mode. I assume this is because Macros are not type safe nor do they handle expressions entered as arguments perfectly in all cases. I didn't look too deeply into the failure but this fix worked for me.

January 9, 2010 at 2:19 PM

Jon said...

I added the JVC protocol:

Constants:

```
#define JVC_HDR_MARK 8000
#define JVC_HDR_SPACE 4000
#define JVC_BIT_MARK 600
#define JVC_ONE_SPACE 1600
#define JVC_ZERO_SPACE 550
#define JVC_RPT_LENGTH 60000
```

Sending function:

```
void IRsend::sendJVC(unsigned long data, int nbits, int
repeat)
{
enableIROut(38);
data = data << (32 - nbits);
if (!repeat){
```

```
mark(JVC_HDR_MARK);
space(JVC_HDR_SPACE);
}
for (int i = 0; i < nbits; i++) {
if (data & TOPBIT) {
mark(JVC_BIT_MARK);
space(JVC_ONE_SPACE);
}
else {
mark(JVC_BIT_MARK);
space(JVC_ZERO_SPACE);
}
data <<= 1;
}
mark(JVC_BIT_MARK);
space(0);
}
```

Receiving funtion:

```
long IRrecv::decodeJVC(decode_results *results) {
long data = 0;
int offset = 1; // Skip first space
// Check for repeat
if (irparams.rawlen - 1 == 33 &&
MATCH_MARK(results->rawbuf[offset], JVC_BIT_MARK)
&&
MATCH_MARK(results->rawbuf[irparams.rawlen-1],
JVC_BIT_MARK)) {
results->bits = 0;
results->value = REPEAT;
results->decode_type = JVC;
return DECODED;
}
// Initial mark
if (!MATCH_MARK(results->rawbuf[offset],
JVC_HDR_MARK)) {
return ERR;
}
offset++;
if (irparams.rawlen < 2 * JVC_BITS + 1 ) {
return ERR;
}
// Initial space
if (!MATCH_SPACE(results->rawbuf[offset],
JVC_HDR_SPACE)) {
return ERR;
}
offset++;
for (int i = 0; i < JVC_BITS; i++) {
if (!MATCH_MARK(results->rawbuf[offset],
JVC_BIT_MARK)) {
return ERR;
}
offset++;
if (MATCH_SPACE(results->rawbuf[offset],
JVC_ONE_SPACE)) {
data = (data << 1) | 1;
}
else if (MATCH_SPACE(results->rawbuf[offset],
JVC_ZERO_SPACE)) {
data <<= 1;
}
else {
```

```
return ERR;
}
offset++;
}
//Stop bit
if (!MATCH_MARK(results->rawbuf[offset],
JVC_BIT_MARK)){
return ERR;
}
// Success
results->bits = JVC_BITS;
results->value = data;
results->decode_type = JVC;
return DECODED;
}
```

*Note instead of sending the REPEAT constant if you want the JVC repeat signal sent, send the original code value and change the repeat argument from 0 to 1. JVC protocol repeats by skipping the header NOT by sending a separate code value like NEC does.

January 9, 2010 at 2:39 PM

Anonymous said...

Hi Just a quick note to say I used your lib to implement a protoype ethernet based remote for my TV & DVDR which I access from my iPhone via wifi. I also make use of a local web server between my Iphone & the IRremote arduno/ethernet shield. Hopefully, I will be able to do without the web server in-between by using ajax on the iPhone. The IR receiver component was very useful for decoding my DVD remote. The TV and DVDR are from philips and both use RC6. The TV also accepts a base set of RC5 codes as well.

Thanks for your great work from Dublin, Ireland :)

January 13, 2010 at 6:17 PM

Ken Shirriff said...

As some people mentioned above, the library uses timer2 so it will conflict with other software that uses timer2. Timer0 is used by the Arduino framework for millis() and delay(), so the library can't easily use timer0. Timer1 is a 16-bit timer, so it's not directly compatible with timer2. I'm looking into modifying the library to use timer1.

January 18, 2010 at 9:57 AM

Michael Gold said...

(From the have-my-cake-and-eat-it-too department): If you were to undertake adding support for timer1, might it be worthwhile to make it user-configurable so the lib could use either timer1 or timer2 based on some sort of configuration directive/variable setting?

Thinking that if you substituted timer1 support *in place of* support for timer2, that might fix the issues on pin 3 and 11 but it would probably negatively affect those using PWM on pins 9 and 10 (since timer1 controls PWM on 9 and 10)...

again - thanks for creating this library!
Michael

January 19, 2010 at 7:16 PM

Ken Shirriff said...

Michael, thanks for your comment. I was planning to use a compile-time #ifdef to select the timer (assuming I get support for timer0 and timer1 working).

January 19, 2010 at 7:23 PM

Todd Treece said...

I am not sure if this will help anyone, but using an oscilloscope I was able to create sending functions for a Sharp LCD TV and a DISH Network receiver.

The Dish send function needs to be repeated 4 times and the Sharp function has the necessary repeats built in. I know that it's not consistent, but I don't have the time to update my code.

Here are the LIRC files that I found that seem to match the remote codes from the oscilloscope:

Sharp LCD TV:
http://lirc.sourceforge.net/remotes/sharp/GA538WJSA

DISH NETWORK (echostar 301):
http://lirc.sourceforge.net/remotes/echostar/301_501_3100_5100_58xx_59xx

For the DISH codes, only send the last for characters of the hex.
i.e. use 0x1C10 instead of 0x0000000000001C10 which is listed in the linked LIRC file.

If anyone can help create the receiving functions, that would be great! Keep in mind that the Dish remote operates at 56hz.

Here are the relevant mods to the library (sorry if it's sloppy code):

```
//IRremote.h
#define DISH 5
#define SHARP 6

void sendDISH(unsigned long data, int nbits);
void sendSharp(unsigned long data, int nbits);

//IRremoteInt.h
#define SHARP_BIT_MARK 245
#define SHARP_ONE_SPACE 1805
#define SHARP_ZERO_SPACE 795
#define SHARP_GAP 600000
#define SHARP_TOGGLE_MASK 0x3FF
```

```
#define SHARP_RPT_SPACE 3000

#define DISH_HDR_MARK 400
#define DISH_HDR_SPACE 6100
#define DISH_BIT_MARK 400
#define DISH_ONE_SPACE 1700
#define DISH_ZERO_SPACE 2800
#define DISH_RPT_SPACE 6200
#define DISH_TOP_BIT 0x8000

#define SHARP_BITS 15
#define DISH_BITS 16

//IRremote.cpp
void IRsend::sendSharp(unsigned long data, int nbits) {
unsigned long invertdata = data ^
SHARP_TOGGLE_MASK;
enableIROut(38);
for (int i = 0; i < nbits; i++) {
if (data & 0x4000) {
mark(SHARP_BIT_MARK);
space(SHARP_ONE_SPACE);
}
else {
mark(SHARP_BIT_MARK);
space(SHARP_ZERO_SPACE);

}
data <<= 1;
}

mark(SHARP_BIT_MARK);
space(SHARP_ZERO_SPACE);
delay(46);
for (int i = 0; i < nbits; i++) {
if (invertdata & 0x4000) {
mark(SHARP_BIT_MARK);
space(SHARP_ONE_SPACE);
}
else {
mark(SHARP_BIT_MARK);
space(SHARP_ZERO_SPACE);

}
invertdata <<= 1;
}
mark(SHARP_BIT_MARK);
space(SHARP_ZERO_SPACE);
delay(46);
}


void IRsend::sendDISH(unsigned long data, int nbits)
{
enableIROut(56);
mark(DISH_HDR_MARK);
space(DISH_HDR_SPACE);
for (int i = 0; i < nbits; i++) {
if (data & DISH_TOP_BIT) {
mark(DISH_BIT_MARK);
space(DISH_ONE_SPACE);
}
else {
```

```
mark(DISH_BIT_MARK);
space(DISH_ZERO_SPACE);
}
data <<= 1;
}
}
```

January 21, 2010 at 9:05 AM

ArduEve (Maple=ARM Arduino) said...

Simply fabulous!
I was searching for an Arduino IR library & found your blog only a couple days after you 1st posted it. Been using it ever since, of course. I've seen other IR code here & there but nothing as good as yours.

Thank you very much for sharing this & your continued work on it.

Also thanks to all who have added more protocols.
I'm sure many people haven't found this yet. Would you mind if I add a link in the Arduino Playground wiki?
http://www.arduino.cc/playground/Main/GeneralCodeLibrary

January 22, 2010 at 9:54 PM

Ken Shirriff said...

I've put the IRremote source on github at http://github.com/shirriff/Arduino-IRremote. I'm hoping this will help integrate fixes and extensions from other people.

January 22, 2010 at 10:46 PM

simonjtaylor212 said...

Hi Ken!

Love this, I have so far managed to compile the example and upload to my brand new arduino and turn my Sony TV off!

So.... could you help me, and explain what I need to do to find more codes from LIRC and use them with your code? I'm a bit stuck now.. I can only turn my TV off! :)

Thanks

Simon Taylor, London, UK

January 27, 2010 at 6:57 AM

Ken Shirriff said...

Simon: there are two ways you can get the codes. First, you can look at the Sony LIRC files and try to find one with codes that work on your TV. One complication: the Sony LIRC codes are a bit ambiguous about the last bit. A lot drop the last bit, so they are 11 bits long instead of 12, and you need to multiply them by 2. Others end in 1 and you need to subtract 1 (try the RM-S609 codes and subtract one).

Alternatively, if you have an IR receiver module, you can run the IRrecvDump example to print out the codes from your existing remote.

January 28, 2010 at 9:00 PM

dario said...

Hi guys! how can I use the panasonic code posted here? do i have to paste it in the h file library?

January 29, 2010 at 7:58 AM

Juan said...

I LOVE this library,I have been playing the entire week with it. I have one issue. The first time I tried the code I run all the samples, the first sample IRSendDemo was not working for me, I tried the IRRecord and it worked perfecltly, pressed a button, saved the code and then submitting that code to my TV. Great! The thing is that I'm not able to JUST send a code. I try to use the sendNec method but no output on pin3, what I'm missing here? Thanks!!!

January 30, 2010 at 7:36 PM

Juan said...

mmmm sorry for the previous post, I tried again and remove the first if on IRSendDemo....and ir worked PERFECTLY...sadly, I was using the entire block on ALL my tests...removing the IF for the Serial.read solved it... I'M VERY HAPPY!! :)

(posting this in case somebody else ran across the same problem!)

January 30, 2010 at 7:50 PM

Anonymous said...

Total Arduino newbie:
When I try to compile IRSendDemo, I get the error:
22: error: IRremote.h: No such file or directory In function 'void loop()':
Anyone can answer...thanks

January 31, 2010 at 12:35 AM

Anonymous said...

Newbie:
I didn't follow the install instructions...

January 31, 2010 at 9:44 AM

Anonymous said...

Hello Ken, this library is really great. I just like to have the Panasonic Option but I don´t know how to integrate the Codes above (Wolfgang and next). I get many Errors and persistenly "in decode there is no address" The decoding function maybee different from the others. Sorry I´m a Newbie and there should be some adapptions I don´t understand for now. Could someone explain the integration of the Panasonic-Code (Step by Step in Detail with every associated File:)
thanks Christopher

February 2, 2010 at 3:15 AM

Anonymous said...

Ok, I fixed the Problem above (just the variable address to add). But now I´ve got serious trouble. I am trying to implement the RECS-80 (a very old, but cheap-chip protocol) and got stuck. I modified the NEC-Code (for the RECS-80 is pulsewidth modulated) but can´t get a reaction. The Marks are very short (158us) and the spaces very long (4900us = 0, 7432us = 1). The Duration of the Signal may be to long? Could it be a problem with irparam.timer or RAWBUFFER?

Thanks for reply Christopher

February 3, 2010 at 8:47 AM

irkgreen said...

I'm a n00b here, but is this easily portable to the ATMEGA168 using AVR studio 4? I have an STK500 and see that I can make an ardunio dev kit, but it requires a 16 MHz xtal (i have 7.3728 and 12).

February 4, 2010 at 12:09 PM

simonjtaylor212 said...

Be wary of laptop IR!

I was getting alot of random

" 0,
could not decode message "

results on the IRDump example.

The reason?

My laptop's IR port was sending random data, which the IR receiver was picking up!

I've taped over it for now. Must have a look in the control panel!

Cheers

Simon

February 7, 2010 at 7:00 AM

Anonymous said...

I can't seem to compile this library for some reason, I keep getting all these errors:

IRremote.cpp: In member function 'void IRsend::mark(int)':

IRremote.cpp:172: error: 'TCCR2A' was not declared in this scope

IRremote.cpp:172: error: 'COM2B1' was not declared in this scope

plus about 15 more... any idea whats causing this?

February 12, 2010 at 6:11 AM

Glen H said...

Hi There. I'm wanting to use this brilliant library to control my Acer projector. I have received the IR codes from Acer and they look like this:

Power On OKOKOKOKOK
Power Off * 0 IR 001
Keystone * 0 IR 002
Mute * 0 IR 004
Freeze * 0 IR 006
Menu * 0 IR 007
Up * 0 IR 008
Down * 0 IR 009
Right * 0 IR 010
Left * 0 IR 011

Anyy hints on how I can add these into the library?

Many thanks!

February 15, 2010 at 2:05 PM

CrYoPyRe said...

Hi, I was recently trying to use your library on my AtMega8 based NG Freeduino board.

It seems your library was made keeping the Atmega 168 in mind and hence gives errors in compilation for my board type.

Would you know of an alternative for Atmega 8 based boards?

February 19, 2010 at 3:14 AM

kens said...

Glen H: I don't know what the codes for an Acer projector would be. I'd suggest using IRDump and see if you get lucky enough that Acer uses an encoding that the library handles.

CrYoPyRe: I don't have an Atmega 8 so I can't port the library. You'd have to change the code to use the appropriate registers. An earlier comment said the code is too big to fit in an Atmga 8 in any case, so it might need to be cut down to size too.

February 21, 2010 at 6:05 PM

anshul said...

I am working on a project in which I want to control home appliances using any TV/DVD remote. I am using Atmega8 for this purpose and I dont have any Arduino Board. I have downloaded Aurduino Software and ur IR decoding libraries are working on it.
Now the problem is that how I burn micrcontroller with hex file generated from ur libraries ? In which directory the Arduino Software save these hex files ?

February 27, 2010 at 12:16 PM

Anonymous said...

Thanks for this awesome IR library.
I was able to use it for an IR translator application.

I did want to mention that one problem that I encountered was that I needed to do receive IR and send IR for my application, and it took me a while to realize that the send IR routine DISABLES the receive IR function.
So anyone who need to send and receive IR, make sure to do "irrecv.enableIRIn();" right after you finish sending IR. Otherwise our Arduino would stop receiving IR.

March 4, 2010 at 11:10 AM

Buks said...

Brilliant!, I have Vately Air-conditioner remote which is still working if you bend the PCB in just the right way. I want to see if can read the codes and then build a replacement remote.

I have a totally unrelated question though... What are those nice looking connecting wires you used in your project? I live in South Africa, and getting local suppliers for hobby electronics stuff can be rather frustrating at times.

Thanx for a great blog.

March 7, 2010 at 1:32 PM

Ken Shirriff said...

Buks: the wires are Adafruit's breadboarding wire bundle.

March 7, 2010 at 8:58 PM

Mike S said...

Very nice work. I post this in case someone else has been struggling.

This allows you to control a Samsung TV using OBC to provide additional functions not on the remote (OBC's can be found online).

First change 9000 to 4500.

Then for example OBC 190 switches to HDMI#2.
Convert 190 to binary and REVERSE the order (I got this idea from looking at NEC examples), THEN convert to hex, e.g this becomes 7D. Invert the binary and and convert to hex 82. Thus web page value becomes NE0E07D82.

March 9, 2010 at 3:56 PM

Anonymous said...

Hi Mike S,

Could you please do another example for OBC code converion to the IR library hex code?
For my NEC IR code, I have 8 hex digits. If I have the JP1 OBC code, how do I translate it to the IR library hex code for the send function?
Thanks in advance.

March 9, 2010 at 5:04 PM

Buks said...

The busted Aircon-Remote Saga:

1) When I put the circuit together as explained it would at first not work. After reading the datasheet for my particular IR receiver, I discovered that pins 1, 2 and 3 are not the same as the ones described at beginning of the blog. Once I got that sorted out, the RcvDump example just streamed random junk the whole time. Back to the datasheet. It seems like its always good idea to use the sample circuit your manufacturer suggests in the datasheet. 1 cap and 2 resistors later and it worked like a charm (the extra components filters out power supply noise).

2) The next weird thing I noticed that my aircon remote was always sending the same sequence no matter what settings I used on it. After digging in the code of IRremote.cpp I found this little section of code:

```
if (irparams.rawlen >= RAWBUF) {
// Buffer overflow
irparams.rcvstate = STATE_STOP;
}
```

IRremote.h had RAWBUF set to 76, which just happened to be exactly the same number of values I was getting in the dump. I had to increase it all the way to 200 to fit in the whole message from the air-conditioner remote. It seems like air-conditioner remotes send all the settings (temperature, fan speed, mode, swing etc) all in one go every time you press a button.

Next: I have to reverse engineer the codes so that I can build a replacement remote. I have already figured out which 4 bits is used for temperature.

I am already envisioning voice control for my aircon... muahahaha

March 10, 2010 at 10:22 AM

Ioan said...

Hi,
I'm curious if I can use this library to make a beam-break detector, so the IR LED transmits continuous and the receiver detects when the beam is broken, when an object is between the two. Is this possible?

March 11, 2010 at 3:59 PM

Mike S said...

Re another example, I just used an online convert at mathisfun com. Try it with the example I've provided and I think you can make it work. By reversing the order I mean for example 10110000 becomes 00001101, then convert this to hex. The second hex is the inverted 11110010 convert to hex. It helps to google the NEC code format. Hope this helps.

March 11, 2010 at 4:16 PM

Anonymous said...

HELP!

I can't get the Panasonic decode to work. This is what I get:

```
Attempting PANASONIC decode
Testing mark 3500 vs 3502: 54 <= 70 <= 91
Testing mark 1700 vs 1750: 27 <= 34 <= 47
Testing mark 500 vs 502: 9 <= 10 <= 16
Testing space 400 vs 1244: 17 <= 8 <= 29
Testing space 400 vs 370: 4 <= 8 <= 7

Could not decode message
Raw (100): -1324 3500 -1700 500 -400 450 -1250 500
-400 450 -400 450 -450 450 -400 450 -400 500 -400
450 -400 450 -450 450 -400 500 -350 500 -400 450
-1250 500 -400 450 -400 500 -400 450 -400 450 -400
500 -400 450 -400 450 -450 450 -400 450 -1300 450
-400 450 -450 450 -400 450 -400 500 -400 450 -400
450 -450 450 -400 450 -1300 450 -400 450 -450 450
-400 450 -400 500 -1250 500 -400 450 -400 450 -1300
450 -400 450 -450 450 -400 450 -400 500 -1250 500
-400 450 -1250 500
0
```

Can anyone tell why the Panasonic code can't decode this signal from my Panasonic remote?

March 11, 2010 at 8:47 PM

Ken Shirriff said...

Reply to Ioan: for a beam break detector, you could use this library to generate the modulated signal for the IR source. I wouldn't recommend using the library to detect the breaks; you could just read the detector directly on a digital input.

Panasonic decode: it looks like it's almost working:
Your log shows "Testing space 400 vs 370: 4 <= 8 <= 7"
That means your value is 400us (8 ticks), and it's expecting 370us and will accept anything between 4 and 7. Your value is just a bit too big (8 vs 7), so the decode fails.

Try increasing PANASONIC_ZERO_SPACE from 370 to 400 or 450. You can also loosen up the comparison by increasing TOLERANCE from 25% to maybe 35%.

I should write up a posting about how this all works, but hopefully this is enough to get you going.

March 13, 2010 at 9:29 PM

Anonymous said...

Hi Ken,

Thanks for the hints for the Panasonic code, but it's still not working...
I did increase the space to 450, and change the tolerance to 35%.
It's getting further, but still couldn't decode. This is what I get now:

```
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 400 vs 1244: 14 <= 8 <= 31
Testing space 400 vs 450: 4 <= 8 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 400 vs 502: 7 <= 8 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 400 vs 1244: 14 <= 8 <= 31
Testing space 400 vs 450: 4 <= 8 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 1300 vs 1244: 14 <= 26 <= 31
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 400 vs 502: 7 <= 8 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 1300 vs 1244: 14 <= 26 <= 31
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 400 vs 1244: 14 <= 8 <= 31
Testing space 400 vs 450: 4 <= 8 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 400 vs 502: 7 <= 8 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 400 vs 1244: 14 <= 8 <= 31
Testing space 400 vs 450: 4 <= 8 <= 10
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 1300 vs 1244: 14 <= 26 <= 31
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 450 vs 1244: 14 <= 9 <= 31
Testing space 450 vs 450: 4 <= 9 <= 10
Testing mark 400 vs 502: 7 <= 8 <= 17
Testing space 1350 vs 1244: 14 <= 27 <= 31
Testing mark 450 vs 502: 7 <= 9 <= 17
Testing space 0 vs 1244: 14 <= 0 <= 31
Testing space 0 vs 450: 4 <= 0 <= 10
```

Any chance you can spot what's wrong? Thanks!

March 14, 2010 at 8:57 PM

Ken Shirriff said...

Anonymous Panasonic: it looks like the code is designed to receive 48 bits and you're receiving about 16 bits. (You're running out of data too soon and getting the 0 time value.) You could try cutting out all the decode address code. Or you could replace the "return ERR" with "break" inside the loops, and then it would succeed rather than fail when it runs out of data.

March 14, 2010 at 9:22 PM

Ioan said...

Hi Ken,
Can you post a short example on how use this library to generate the modulated signal for the IR source at 38kHz (continuous send for object detection).
TIA!

March 15, 2010 at 12:10 PM

Anonymous said...

Ken,

Thanks for the tips. I finally go the Panasonic decode working. I had accidentally modified the original Panasonic code during the course of debugging. The original Panasonic code with tweaks that you suggested (space, and tolerance) made it work.
Thanks again for the help and this awesome IR library.

Hey Loan, I would not suggest using the send functions from the library to do the IR break detection. The IR send function DISABLES the receive function. So you will not be able to receive and when you are sending IR.
I guessing that you can do a simple analogwrite on a pwm pin to get a constant 38 khz IR output.
But maybe Ken has better ideas.

March 15, 2010 at 11:35 PM

Ken Shirriff said...

Loan: I wrote up details on using the library to detect an IR beam break. See my latest posting at arcfn.com.

March 16, 2010 at 10:45 PM

Ben Norling said...

Ken,

This all looks great. Is there any reason I couldn't use a production quality Ir Receiver? I've hooked one up (Niles m220) and I get the flashing light, but it can't decode the IR. I ran the dump program and it just says "can't decode message" and a bunch of negative numbers.... Thoughts?

Thankds

April 13, 2010 at 10:08 PM

Ken Shirriff said...

Ben: I can think of a couple reasons why your M220 receiver might not work:
a) Maybe the receiver is active-high instead of active-low?
b) Maybe the code your remote is transmitting isn't one the library understands? In that case, look at http://www.arcfn.com/2010/01/using-arbitrary-remotes-with-arduino.html
c) Maybe the tolerances of your receiver are a bit different than the ones I use?

Can you send me the output from the dump program? I can take a look and see what's happening.

April 13, 2010 at 10:40 PM

Ben Norling said...

Ken,

Below are outputs from the dump program and the raw program.

I'm using the M220 in an attempt to be able to cover all remotes. This receiver is not limited to a 38.1 kHz remote. I'm a bit unclear on the active high/low thing. To clarify, the output from the m220 is normally high and goes low as it receives impulses from a remote. As a result I had to cobble together a not-gate from a 3 input nand gate with both other inputs forced high.. With the output from the m220 going directly to the input pin on the arduino, I got absolutely no response.

I am a home theater contractor and have been amazed at how much the remote companies charge to do fairly simple things. If I were more of a programmer, I could probably use an arduino and a few cents worth of other bits and replace a 600 dollar remote processor. It's a dream..

Thanks for your help,

Ben

```
'real' decode: 0, hash decode: F3FED56D

'real' decode: 0, hash decode: AC9BB4FA

'real' decode: 0, hash decode: 8E46808

'real' decode: 0, hash decode: C439C7D8

'real' decode: 0, hash decode: F692C022

'real' decode: 0, hash decode: F3FED56D

'real' decode: 0, hash decode: 2D9CE207

0

Could not decode message

Raw (42): -10526 100 -2900 100 -1750 100 -1100 100
-1100 100 -1750 100 -1700 150 -1700 150 -1700 100
-1100 100 -1750 100 -1100 100 -1700 150 -1700 150
-1700 100 -1100 150 -1050 150 -1700 100 -1100 150
-1050 150 -1700 100

0

Could not decode message

Raw (42): -7328 100 -2900 100 -1750 100 -1100 100
-1100 100 -1750 100 -1750 100 -1700 150 -1700 100
-1100 100 -1750 100 -1100 100 -1750 100 -1700 150
-1700 150 -1050 150 -1050 150 -1700 150 -1050 150
-1050 150 -1700 150
```

April 14, 2010 at 7:46 AM

Ben Norling said...

Ken,

Additionally, I am pressing the same button on a sony dvd remote several times to get this output.

Ben

April 14, 2010 at 8:03 AM

Ken Shirriff said...

Ben: I see two reasons that your codes aren't decoding.

First, I can tell from the positive and negative pattern that on and off are reversed. My library expects an active-low signal, which your M220 provides, so remove the inverter.

Second, your off durations are ~100ms when they should be ~600us, and your on durations are ~500us too long. In other words, your receiver is taking about 500us to switch from on to off. My code allows 100us for this transition (based on the IR receivers I've used), but 500us will probably throw off the decoding. In the file IRremote.h, change the constant to #define MARK_EXCESS 500

I think with these changes you should decode successfully. I could manually decode the dump output and see that you're pressing the UP button on your Sony DVD remote, right?

April 14, 2010 at 9:08 PM

Anonymous said...

Many thanks for providing this for free - it has saved me a lot of time and trouble.
I think that I have a bug fix for you though regarding the NEC protocol.
In IRremoteInt.h you have defined NEC_ONE_SPACE as 1600. I think that this should be 1690 (2250ms - 560uS = 1690ms not 1600ms). The original value was not recognised by an InFocus projector, but the amended value did work.

April 16, 2010 at 9:23 AM

DrNeon said...

Hi there, I'm trying to use this library to reset the lamp hours timer on my Plus U2-1130 projector (which, for some stupid reason requires the remote to reset the timer, and the remote is lost). I got the Arduino board and the 950nm IR LED from RobotShop, and have started playing. The LED blinks as verified with a camera, but I'm not getting any response from the projector. From the manufacturer, I have a file that says custom code 18E9 (though I've been told on RC forums it should be 1816), and On code 08F7. I've tried sending (with sendNEC, since the manufacturer says it is NEC protocol) 0x18E908F7, (ie irsend.sendNEC(0x18e908f7, 32);) but I'm not seeing any response on the projector. Am I missing something? Does the bit order need to be reversed, or does something need to be complemented? I don't have the remote, so I can't use a receiver to get the true code. I've held the LED up against the projector in case it is a low power issue, still no response. Once I can get it to turn on, I will need to simulate the On button held down for 10 sec, which I imagine involves something like irsend.sendNEC(REPEAT, 32); being repeated every some number of ms. Thanks for the awesome library, and the help!

April 17, 2010 at 10:06 AM

Vishal said...

Hi Ken!

Thanks for posting this library!

I'm trying to send the power-on code for an Xbox360...I'm using the LIRC file from here: http://lirc.sourceforge.net/remotes/microsoft/Xbox360

Using the data in this file, I figured that I needed to send the following code:

irsend.sendRC6(0x37ff086f3, 37);

This comes from the fact that the pre_data is listed as 0x37ff0 with 21 bits, and the power on/off code is 0x86f3, with 16 bits. I clearly have something wrong, because the Arduino software spits out the error: "integer constant is too large for 'long' type." I understand that the code I'm trying to send is too long, but what should I do to get this to work correctly?

April 17, 2010 at 6:43 PM

Anonymous said...

This comment has been removed by a blog administrator.

April 23, 2010 at 3:08 AM

Anonymous said...

This comment has been removed by a blog administrator.

April 23, 2010 at 3:08 AM

spl23 said...

Just wanted to say thank you very much for the library - it's enabled me to set up an IR controller for my Sonos system. It all just worked first time - many thanks!

April 23, 2010 at 5:12 AM

Anonymous said...

I've just about thrown all this out the window in frustration so I was excited to find a promising library. (Thank you, btw)

Unfortunately seems the library isn't working for me either so I must be missing something somewhere. Hope you can provide some ideas.

When I run the send and receive demos I only get three 0's for output. The IR appears to be sending (seen via camera) but not getting anything valid. I tried a Sony remote also but all I get is 0's for any button.

I'm using a sharp IR receiver (38KHz) (http://media.digikey.com/pdf/Data%20Sheets/Sharp%20PDFs/GP1UE261X,267X,26X.pdf). Testing with two Arduino Pro Mini's. Not sure if it matters but these do run at 3.3v and 8 MHz. Does that change the PWM stuff?

Before trying this library I was getting somewhat consistent (although noisy) results using the classic oscillationWrite and pulseIn functions so I'm pretty sure the wiring is correct. However, it's far from reliable so I'm going crazy.

Thanks in advanced!

April 23, 2010 at 8:57 PM

Anonymous said...

I tried the receive dump with the send demo and this is what I come up with:

```
0
Could not decode message
Raw (26): 7620 1200 -350 600 -300 300 -300 600 -300
300 -300 600 -300 350 -250 350 -300 600 -300 300
-300 300 -300 300 -300 300
0
Could not decode message
Raw (26): 14686 1200 -300 600 -300 300 -300 600 -300
300 -300 650 -300 300 -300 300 -300 600 -350 250
-350 300 -300 300 -300 300
0
Could not decode message
Raw (26): -3524 1200 -350 600 -300 300 -300 600 -300
300 -300 600 -300 350 -300 300 -300 600 -300 300
-300 300 -300 300 -350 250
```

Thanks (again) in advanced!

April 23, 2010 at 9:10 PM

Ken Shirriff said...

Anonymous: before you throw everything out the window, try changing the clock speed in in IRremoteInt.h to: #define SYSCLOCK 8000000. I think the problem is your clock speed is 8MHz, and my code is designed for 16MHz.

Vishal: if you need 37 bit NEC codes, try changing sendNEC to take a uint64_t type, since long is limited to 32 bits.

Anonymous: yes, you are right; NEC_ONE_SPACE should be 1690.

DrNeon: I don't have any good suggestions for you. Maybe try Anonymous's change to NEC_ONE_SPACE above.

April 26, 2010 at 8:44 PM

Bobrik said...

Hi I have problem with library, first press is good decoded, second is bad, next good, bad, repeately

```
Decoded RC5: 5 (12 bits)
Raw (24): 746 950 -800 1850 -800 950 -800 1000 -750
950 -800 1000 -750 1000 -750 1000 -750 1000 -750
```

```
1000 -1600 1900 -1600 950
'real' decode: 5, hash decode: 33157B67
Decoded RC5: 805 (12 bits)
Raw (24): 3390 1000 -750 1000 -750 1900 -750 1000
-750 1000 -750 1000 -750 1000 -750 1000 -800 950
-800 950 -1650 1850 -1650 950
'real' decode: 805, hash decode: 6E6E8FBB
Decoded RC5: 5 (12 bits)
Raw (24): -7536 900 -850 1800 -850 900 -850 900 -850
900 -850 900 -850 900 -900 850 -900 900 -850 900
-1700 1800 -1650 1000
'real' decode: 5, hash decode: 33157B67
Decoded RC5: 805 (12 bits)
Raw (24): 7466 900 -800 950 -800 1850 -800 1000 -750
1000 -750 1000 -800 950 -800 950 -800 950 -800 950
-1650 1850 -1650 950
'real' decode: 805, hash decode: 6E6E8FBB
```

May 1, 2010 at 2:55 AM

Simon Long said...

Hi Ken,

Your library is great, and works very well in my Sonos control project. However, I do have one recommendation for you.

In IRremoteInt.h, you set the tolerance for pulse width using the code

```
#define TOLERANCE 25 // percent tolerance in
measurements
#define LTOL (1.0 - TOLERANCE/100.)
#define UTOL (1.0 + TOLERANCE/100.)
```

This is the only floating-point code used in the library, and it means that the compiler pulls in the Arduino floating-point libraries, which are rather large, and it is wasteful of memory to use them for this simple calculation.

If you delete these three lines, and modify the definitions of TICKS_LOW and TICKS_HIGH (the only lines that use them) to

```
#define TICKS_LOW(us) ((((us) * 3)/(4 *
USECPERTICK)))
#define TICKS_HIGH(us) ((((us) * 5)/(4 *
USECPERTICK) + 1))
```

you get exactly the same behaviour, but you don't need the FP libraries any more. On my sketch, this one change reduced the memory used from 16404 bytes to 15220 bytes - a saving of well over 1kbyte, which is very worthwhile in the Arduino's limited memory footprint.

In general, on embedded systems, you should use integer maths wherever possible - floating-point is invariably expensive in terms of memory and processing power.

May 1, 2010 at 3:43 AM

Ken Shirriff said...

Bobrik: that's how RC5 and RC6 codes work. They have a toggle bit that is set every other time you press the button. You can subtract (AND) off the toggle bit if you don't want it. For details see Philips RC5 Protocol.

Simon Long: that's a good point about using floating point. I'll replace that with integer arithmetic in my next release (whenever that is).

May 1, 2010 at 10:48 AM

;artin said...

Hi Ken, I just tried it and found your library really useful and well documented. I'm using it to gain control over an LG DVX482H DVD player for an arts installation (replace its AKB35840202 remote). The Remote seems to use 2 different versions of NEC Protocols for TV control and DVD control. TV Protocol uses a MARK of 9000, the DVD part works when I change NEC_HDMARK to 4500 (tried this value by guessing from raw value readings). Since I don't use the TV part this change is fine for me. Maybe this is of use for someone else. I'm both successfully receiving and transceiving right now.
Thanks!

May 5, 2010 at 5:40 PM

Scott said...

awesome project! any update on when the new library will be released?

May 5, 2010 at 8:57 PM

Anonymous said...

Great Library. I'm developing a remote control for watching TV. I set a home channel, then when a commercial comes on, I can surf all over for 2 minutes or so, then the Arduino sends the signal to return to the home channel. I call it the AdBlaster.

For those who want to measure IR LED current, don't forget that the Arduino has 6 ADC inputs. I turn the IR LED on, and measure voltage at both ends of the series resistor. The IR LED voltage can also be measured.

The typical IR LED's are rated for 100 mA+ continuous current, so by making the duty cycle 25 to 30 percent, higher currents are easily tolerated.

May 6, 2010 at 11:46 PM

TiStyle said...

Hello Ken,

I'm an Interaction Student on the HKU in Hilversum, the Netherlands. I'm
currently busy with a project that requires constant IR-

communication and I
have a question about the IRremote library.
I'm using an Yamaha remote. Now I want to use the volume + button(5EA158A7),
that the remote is sending, to power a led and the volume - button(5EA1D827)
to disable the led, but that doesn't work..

```
if(results.value == '5EA158A7') {
Serial.println("+");
digitalWrite(ledPin, HIGH);
}
else if(results.value == '5EA1D827') {
Serial.println("-");
digitalWrite(ledPin, LOW);
}
```

Could you help me with the
problem?

Kind regards,

Tibor Dujmovic
Interaction Design 2
HKU Hilversum, the Netherlands

May 7, 2010 at 12:43 PM

Halo said...

Hi TiStyle
Thought I would try to help.
irrecv.resume();
is at the end of your code right?

IRrecvDump decodes your remote without issue every time?

May 7, 2010 at 6:17 PM

Halo said...

Oh... try

```
if(results.value == 0x5EA158A7) {
Serial.println("+");
digitalWrite(ledPin, HIGH);
}
else if(results.value == 0x5EA1D827) {
Serial.println("-");
digitalWrite(ledPin, LOW);
}
```

May 7, 2010 at 6:26 PM

TiStyle said...

YES!!! Thank you so much man!That's all I needed! =D

May 8, 2010 at 6:02 AM

Vishal said...

Hi Ken!

I've been playing with your IRRecvDump example...the timing data makes my life SO much easier. Before I got your code, I was measuring pulse widths from my receiver chip using a oscilloscope in the lab...

I'm still having trouble with the Xbox360 remote. I have ascertained that the remote uses the RC6 IR Protocol, and I have used a photodiode/opamp setup in that lab with an oscilloscope to verify that the timing data your program receives is accurate, and that the modulation frequency is in spec for RC6 (36kHz).

When I press the power button on the Xbox360 remote, I got the following response:

```
800FF40C
Decoded RC6: 800FF40C (36 bits)
Raw (68): 27496 2750 -850 500 -350 550 -350 550 -800
500 -850 1400 -800 500 -400 500 -400 500 -350 550
-350 550 -350 500 -400 500 -350 550 -350 550 -350
550 -350 950 -350 550 -350 550 -350 500 -400 500
-400 500 -350 550 -350 550 -800 950 -800 550 -350
550 -350 500 -400 500 -400 500 -350 1000 -350 500
-850 500 -350 550
```

Then, I noticed I got a different response when I pressed it again:

```
800F740C
Decoded RC6: 800F740C (36 bits)
Raw (66): -14044 2750 -800 550 -350 500 -400 500
-800 550 -800 1400 -800 550 -350 550 -350 500 -400
500 -400 500 -350 550 -350 550 -350 500 -400 500
-400 500 -350 1000 -350 500 -400 500 -350 550 -800
950 -400 500 -400 500 -800 1000 -800 500 -400 500
-350 550 -350 550 -350 500 -400 950 -400 500 -800
550 -350 500
```

My assumption is that this was the toggle bit that I was seeing. Noting that your code doesn't call the toggle bit, I figured that if I put the following code in my sketch, the Arduino would indeed turn on the Xbox:

```
Serial.print("Sending XBOX Power...\n");
for(int i = 0; i <4; i++)
{
irsend.sendRC6(0x800FF40C, 36);
delay(50);
irsend.sendRC6(0x800F740C, 36);
delay(50);
}
Serial.print("Sent.\n");
```

It didn't. =(

I've tested the LED circuit I'm using (IR LED with forward voltage 1.2V, and forward current 100mA, in series with 39 Ohm resistor) with IR code I've written for a different (and much easier to understand) remote protocol, and it works. Of course, I have connected the LED to digital pin 3, as you said, so I'm certain it's not a hardware failure.

I know you've stated that you do not have any hardware that supports RC5/RC6, and are thus unable to test the code yourself. However, you appear to be quite knowledgeable on this subject, and it's my hope that you can point out something I've overlooked.

Thanks again for taking the time to help people use your software, and continuing to respond to comments like mine. Your code is a great help to tinkerers (and college EE majors trying to finish their design projects) everywhere. =D

Vishal Kotcherlakota
B.S. Electrical Engineering, 2010
UC San Diego

May 9, 2010 at 4:57 PM

Ken Shirriff said...

Hi Vishal! Let me try to explain the RC6 toggle bit. If you press and hold a button on a remote, the code will be transmitted multiple times without the toggle bit as long as you hold down the button. If you then release the button and press and hold it again, the code will be transmitted multiple times with the toggle bit set. If you release and hold the button again, it will be transmitted multiple times without the toggle bit set. The point of this is that if you're holding down the power button, the TV (or other receiver) can tell if you've pressed it once or twice, even if the signal is interrupted (e.g. someone walks in front of the TV).

Your code fragment will flip the toggle bit over and over, which won't work.

If the toggle bit is the issue, you should be able to send a code once, but then it won't work again unless you use the remote (which will flip the receiver's expected toggle bit).

I don't know if the toggle bit is your problem or not; the 36-bit Xbox codes are outside what I've dealt with, so lots of things could go wrong with the code. Maybe if someone loans me an Xbox :-)

You said you're using an oscilloscope; you should be able to compare the codes the Arduino transmits to the code the remote transmits and see what's different. A couple things to look for: some receivers expect the code to be transmitted multiple times with a particular spacing (expecially Sony), so see how many times the remote transmits the code if you press the button once. Also, make sure the transmitted codes are the same, especially at the end. I wouldn't be surprised if there's an off-by-one problem and the Arduino is transmitting one last pulse too many or too few.

Good luck!
Ken

May 9, 2010 at 9:01 PM

TiStyle said...

Hello guys!

I'm back, but with an other question this time... I want to use multiple IRreceivers and I thought it would be a piece of cake.. When I wanted to test the code, only 1 of the 2 receivers responded. It's not the receiver because I've already switched it with the pins... I guess that the code isn't correct.

****

```
#include

int RECV_PIN = 2;
int RECV_PIN2 = 3;

IRrecv irrecv(RECV_PIN);
IRrecv irrecv2(RECV_PIN2);

decode_results results;

void setup()
{
Serial.begin(9600);

irrecv.enableIRIn();
irrecv2.enableIRIn();
}

void loop() {

if (irrecv.decode(&results)) {

if(results.value == 0xFF0AF5) {
Serial.println("1");
}

Serial.println(results.value, HEX);

irrecv.resume();
}

if (irrecv2.decode(&results)) {

if(results.value == 0xFF0AF5) {
Serial.println("2");
}

Serial.println(results.value, HEX);

irrecv2.resume();
}
```

May 12, 2010 at 4:00 AM

Todd Treece said...

@TiStyle

I have just connected the data pin from multiple IR receivers to the same Arduino receive pin in the past. I'm not sure if this could cause problems, but it seems to work for me.

May 12, 2010 at 10:39 AM

TiStyle said...

@Todd Treece

I'm not sure if I understand you.. What I've done is put 2 receivers in 2 different pins and only one of them works, but I can't figure out why.

May 13, 2010 at 6:58 PM

Todd Treece said...

@TiStyle

I'm saying I have connected multiple IR receivers to one Arduino pin, and I just told the Arduino to listen on that one pin for both receivers. My receivers are in separate rooms, so they never operate at the same time.

May 13, 2010 at 7:12 PM

greenembrace said...

Hi...
I'm try out your IR stuff, Can get a reponse but it's always "0" zero, mutch to my frustration.

The Dump output is:

```
0
Could not decode message
Raw (68): -9050 4500 -550 1650 -650 500 -600 500
-600 550 -600 500 -600 550 -600 500 -650 500 -550
550 -600 1650 -600 1650 -650 1600 -600 1650 -600
1650 -650 1600 -600 1650 -600 500 -600 1650 -600
550 -600 500 -650 1600 -600 550 -600 1600 -650 500
-650 1600 -600 500 -650 1600 -600 1650 -600 550 -600
1650 -600 500 -600 1650 -600 39850
```

and always the same no matter what button i press, and i have tryed a number of different controls.

Any thoughts where i should start looking.

regards
greenembrace

May 14, 2010 at 1:20 AM

Ken Shirriff said...

TiStyle: The code doesn't currently support two receivers on different pins, mainly because I never thought of implementing that. The interrupt handler would need to be modified slightly to check the status of each pin, not just one pin. Maybe in the mythical version 2.0 :-)

greenembrace: it looks like your input is inverted; the code expects an active-low input, so it can't interpret the data it's receiving from your (NEC?) remote. Are you using some strange type of IR detector? Stick in an inverter and it should work.

May 14, 2010 at 10:45 AM

TiStyle said...

@Todd Treece

I've did that before, but that's not what I need.. I'm going to use a maximum of 30 to 50 receivers and they'll all need an other id for my project. Please help..

May 14, 2010 at 10:59 AM

Ken Shirriff said...

TiStyle: running 30-50 independent receivers sounds like a lot! It sounds like an interesting project; do you want to give more details?

To outline what you'd need to do to support multiple independent IR receivers...

The code currently uses a single irparams global structure to keep track of the received code. The timer 2 interrupt code checks the status of the input pin and updates the irparams structure.

You'd need a separate irparams structure for each pin. You could have an array of them, or maybe have irparams part of the IRrecv class. The array would make it easier for the interrupt handler, but putting it in the class would be architecturally cleaner. Then the interrupt code would need to loop through all of them, updating each one according to the input.

Then the decoding routines would need to process a particular irparams structure, rather than the global one.

I'm not sure how many inputs you could process before you'd run out of time in the interrupt handler.

Hopefully this all makes sense; let me know if there's anything else I can help explain.

Are you planning to use an Arduino Mega to support all these inputs?

May 14, 2010 at 12:10 PM

TiStyle said...

This comment has been removed by the author.

May 15, 2010 at 4:20 AM

TiStyle said...

Hey Ken,

Thanks for the advice, I'm going to try it out as far as I could.

We'll probably use a Mega or maby 2 seperated, we still need to test it out. We'll be making a big light that has about 50 power LEDS. You'll have beacons that will emmit a pulse and the receivers will be used as direction trackers. There where the beacon is, the light will shine uppon(Sounds a bit mythical)...

I'm not sure if I get what you mean... So in the IRremote.cpp/.h I'll need to make an array for the "irparams"?

May 15, 2010 at 4:21 AM

Ken Shirriff said...

I've implemented a new version of the library with a whole bunch of changes including multiple input support (TiStyle, this is for you), support to send and receive codes for arbitrary space-encoded remotes (not just NEC and Sony), and some other features and cleanup.

This is an alpha version of the code, so you get to find out about it here first. If you want to try it out, go to http://github.com/shirriff/Arduino-IRremote/tree/dev (the development branch) and click on Download Source at the upper right.

Let me know how it goes...

May 16, 2010 at 9:16 PM

TiStyle said...

Ken... It works like butter! Thank you so much! =D

I'll send a video of the project when we're finished.

May 17, 2010 at 5:06 AM

greenembrace said...

Hi, I know that i'm a bit boring to reply to compared with the others, but you have a small error it the IRremote.h

```
class decode_results {
public:
int decode_type; // SPACE_ENC, RC5, RC6, NEC_REPEAT, UNKNOWN
unsigned long long value; // Decoded value
int bits; // Number of bits in decoded value
volatile unsigned int *rawbuf; // Raw intervals in .5 us ticks
int rawlen; // Number of records in rawbuf.
class space_enc_data spaceEncData;
};
```

value is a long long

May 18, 2010 at 8:11 PM

tilluigi said...

we are trying this here with an arduino mega and could get it to work (physically, meaning we have output on the LED) setting the pinMode in the library (as described

here in the comments).
however no matter which hex value we're trying to send, it will always output the same bit sequence (we are using an oscilloscope to monitor it). we are using the RC5 protocol and (meanwhile) the latest source from github. has anybody encountered the same problem or any idea what the cause could be?

May 19, 2010 at 12:35 PM

Ken Shirriff said...

tilluigi: the latest github code (dev) is really untested; if you're having problems, I'd suggest using the original code at http://arcfn.com/files/IRremote.zip . Also, I haven't tried any of this on a mega, so I would not be surprised if there are problems. The only board I have is the Arduino Duemilanove, so it's pure luck if the code works with anything else.

greenembrace: thanks for letting me know about the type issue. I hope your previous problem with inverted codes got solved.

May 19, 2010 at 8:42 PM

Cameron said...

I am having problems getting the irrecvdemo sample to run. It seems to be crashing runtime at irrecv.enableIRIn(); in the setup method. I have only modified "int RECV_PIN = 6;" Do you have any ideas what might be causing this? I am running windows 7 with arduino 0018 (but I have also tried 0017 with no success).

May 22, 2010 at 6:02 PM

Cameron said...

I am having problems getting the irrecvdemo sample to run. It seems to be crashing runtime at irrecv.enableIRIn(); in the setup method. I have only modified "int RECV_PIN = 6;" Do you have any ideas what might be causing this? I am running windows 7 with arduino 0018 (but I have also tried 0017 with no success).

May 22, 2010 at 6:02 PM

TiStyle said...

Ken:

Hey Ken, I'm back with an other problem (I'm so sorry to bother you like this)... The new code works fine, but as soon as I define 5 or more receivers the arduino starts to lagg or something like that.. I really can't figure out what it is...

I've checked your library and I think our project won't be needing as much as your library has in it..

Our project is about a lamp that can be controlled bij a beacon. The beacon is the sender and the bulb of light has receivers in it.. All the receivers will be looking at other directions. The receiver that picks up a IRsignal, turns on a 1watt power LED in that direction.
The beacon can also control the intensity of the LEDS trough a potentiometer. As soon as the potentiometer-value changes, so does the HEX that the beacon is sending...

That's pretty much it.. I hope that you could help us, this is the final thing we need to fix.

Thanks in advance!

June 2, 2010 at 10:44 AM

TiStyle said...

Ken:

Hey I think I've found a solution, but I'm not sure...In IRremoteHw.cpp I've changed the:

#define SYSCLOCK 16000000 // main Arduino clock
To:
#define SYSCLOCK 100 // main Arduino clock

And so far it works with 7 sensors, I'm going to test it with more tommorow..

June 3, 2010 at 8:17 AM

Paul Shmakov said...

Hi Ken! Thanks a lot for a library.

Just wanted to inform you it works great on Teensy 2.0 after a couple of modifications (like moving to timer1).

June 4, 2010 at 5:04 AM

Jerry Zheng said...

thx for the great work.
recommend for a bigger RAWBUF,as most aircondition controller send a IR message >100bits even 200bits.

June 9, 2010 at 7:03 AM

Larry said...

Hi,

Thanks for the library - check out y usage of it here:

http://luckylarry.co.uk/2010/06/arduino-redefining-the-tv-remote/

made te work a lot easier :)

Larry.

June 13, 2010 at 2:32 PM

Miles said...

Hi Ken, fantastic work decoding and writing up an excellent library.

I'm trying to use it to control a Skybox in the UK but I thought I'd take a slightly different take on it.

First of all, the Sky remote is actually RC6-Mode 6 (Not that I know much about that).

I've taken out the IR receiver and attached it to the arduino and run a couple of wires where the IRC receiver was and plugged them into the "TX" pin of the Arduino.

Before:
Remote >>>> SkyBox

Now:

Remove >>>> Arduino >>> SkyBox.

Your code recognises one RC6 code (channel up/down) but the rest come through as unknown.

I tried the code of yours which reads in a code then lets you send it via TX but unfortunately that didn't work.

Do you have any hints as to why this may be failing?

June 14, 2010 at 7:12 AM

Ken Shirriff said...

Thanks for all the comments!

@Miles: You could try the IRrecvDump sketch and see what's getting received. I don't have a RC-6 remote so unfortunately the RC-6 code isn't tested too well.

@Larry: interesting project; I've added a link to it.

@Jerry: thanks for the info. I want to keep the buffer small, though, so it doesn't use too much memory.

@TiStyle: I don't know how many sensors one Arduino can support before running out of cycles. Keep me informed :-)

@Cameron: Sorry, no idea why you'd be getting crashes. Are you using an Arduino duemilanove or something else?

June 14, 2010 at 9:38 PM

silver.smith said...

Hi Ken,

Thanks for the library.

I have managed to send and recieve commands with it, but I have a problem with the only command I really need to send. I would like to send the record/stop code of my camcorder as a raw ir-code. I measured the the raw ir-code with both arduino and usb ir-toy. After that I was also able locate my remote (canon wl-80, not any regular protocol)commands at the lirc-site, but I can't figure out how to use the send.raw command. Perhaps you would like to write an example?

Because my coding is really beginners level and I need only one ir-command, I'm not trying to write the whole protocol.

June 15, 2010 at 4:56 AM

silver.smith said...

Forget my last comment, I just noticed the link to Lucky Larrys site, where I found the information I needed. Idiot me...

June 15, 2010 at 10:58 AM

Miles said...

@Ken

Yeah, for the most part it cannot decode the message. I think that stands to reason as the controller is RC6 Mode 6. Sadly documentation on that protocol is lacking at best.

From my understanding it's just RC6 but accepts 32bit packets.

Here's what the IR Dump receives:

```
0
Could not decode message
Raw (42): 25934 1800 -300 400 -1050 300 -600 350
-500 350 -1000 400 -950 800 -550 300 -550 300 -1550
300 -550 800 -900 750 -650 300 -600 300 -1000 250
-700 700 -550 350 -1000 350 -600 750 -500 250 -2050
300
0
Could not decode message
Raw (44): 7922 2700 -900 400 -500 350 -500 400 -950
400 -950 850 -500 350 -500 400 -500 400 -550 300
-550 850 -900 850 -500 350 -550 400 -900 400 -500
850 -450 450 -900 400 -550 800 -500 350 -950 400
-500 400
0
Could not decode message
Raw (42): 18046 2700 -900 450 -500 350 -500 400 -950
400 -900 850 -500 400 -500 350 -550 400 -450 400
-500 850 -950 800 -500 400 -500 400 -950 400 -500
350 -500 800 -1000 800 -500 400 -950 400 -500 800
0
Could not decode message
Raw (42): -11498 2650 -900 400 -500 350 -550 400
-950 350 -950 800 -550 400 -500 350 -500 350 -550
400 -450 900 -900 850 -550 300 -550 350 -950 400
-500 400 -500 800 -1000 800 -500 400 -950 350 -550
800
```

```
0
Could not decode message
Raw (44): 3866 2650 -950 400 -500 400 -500 350 -950
400 -950 800 -550 350 -550 350 -550 350 -550 350
-500 850 -900 850 -550 300 -550 400 -900 400 -550
350 -500 800 -950 850 -500 400 -500 350 -1000 400
-450 400
0
Could not decode message
Raw (44): -29792 2600 -1050 350 -550 350 -500 400
-950 350 -950 850 -500 400 -550 350 -500 350 -500
400 -500 850 -950 800 -500 400 -550 350 -900 450
-450 850 -950 350 -550 350 -550 350 -500 400 -550
800 -500 350
0
Could not decode message
Raw (44): -25962 2650 -900 450 -450 400 -500 350
-950 450 -900 850 -550 300 -550 350 -500 400 -500
400 -500 850 -900 850 -550 350 -500 350 -1000 350
-550 800 -950 350 -550 350 -550 350 -550 300 -550
850 -500 350
0
Could not decode message
Raw (44): 4252 2650 -950 400 -500 400 -450 400 -950
400 -950 850 -450 400 -550 350 -700 100 -600 350
-500 850 -900 850 -500 450 -450 400 -950 400 -500
850 -950 350 -500 400 -500 400 -500 350 -500 850
-500 400
0
Could not decode message
Raw (68): 21548 4450 -4500 550 -1700 500 -1750 550
-1700 550 -600 500 -600 550 -600 500 -600 500 -600
500 -1750 550 -1700 500 -1750 550 -600 500 -650 500
-600 500 -600 550 -550 500 -1750 550 -1700 500 -650
500 -1750 500 -650 450 -600 550 -600 500 -650 500
-600 500 -600 500 -1750 550 -600 500 -1750 500 -1750
500 -1750 500 -1700 550
0
Could not decode message
Raw (68): -7214 4450 -4500 550 -1700 550 -1700 550
-1700 550 -600 500 -600 500 -650 500 -600 550 -550
500 -1750 550 -1700 500 -1750 550 -550 550 -600 550
-550 550 -600 500 -600 550 -1700 550 -1700 550 -600
500 -1700 550 -600 500 -600 550 -600 500 -600 500
-650 500 -600 500 -1750 550 -600 500 -1700 550 -1700
550 -1750 450 -1750 550
0
Could not decode message
Raw (68): -7214 4450 -4500 550 -1700 500 -1750 550
-1700 500 -650 500 -600 550 -600 500 -600 550 -550
550 -1700 500 -1750 550 -1750 450 -650 500 -600 500
-600 550 -650 500 -550 550 -1700 550 -1700 550 -600
500 -1750 500 -600 550 -550 550 -600 500 -600 500
-650 500 -600 550 -1700 500 -650 500 -1750 500 -1700
550 -1700 500 -1750 550
0
Could not decode message
Raw (40): -11526 2550 -1000 300 -550 400 -550 300
-1000 350 -1000 750 -1450 300 -550 300 -650 350 -550
750 -1050 700 -1450 350 -1000 250 -650 750 -650 300
-950 350 -600 250 -1100 250 -1000 250 -700 300
0
Could not decode message
```

```
Raw (38): 7922 2600 -950 250 -650 350 -550 300 -2550
600 -700 200 -500 400 -600 250 -600 300 -550 800
-950 800 -600 300 -550 350 -1000 250 -650 750 -1900
350 -650 250 -950 300 -1000 350
0
Could not decode message
Raw (6): -5650 200 -2750 350 -4700 250 0
```

June 16, 2010 at 8:42 AM

Ken Shirriff said...

@Miles: Those codes look mighty strange. It looks like there are at least three different coding schemes in there. You have signals that start with on for 1800us, off for 300; ones that start on for 2700us, off for 900us; and ones that start on for 4450us, off for 4500us. (Looking at your dumped values, starting with the second number.) The third set look like 32-bit NEC codes, except the first on value is 4450us instead of 9000us. Try changing NEC_HDR_MARK to 4450 and see if those ones decode. The first code is mystifying. It definitely doesn't look like the documented RC-6 protocol; see http://www.sbprojects.com/knowledge/ir/rc6.htm and notice the timings don't match. The first code has timings all over the place, especially the 1550us and 2050us values. The second code looks like RC-6 with 24 bits of data. However, the marks (positive numbers) are a bit short and the spaces (negative numbers) are a bit long; the sensors I've seen are all the other way around. Try changing MARK_EXCESS to 0 or maybe -80 or -100 and see if that helps. (The RC-6 timing should be 444us, but you're seeing on for 350-400us and off for 500-550us.)

June 18, 2010 at 12:18 AM

CaptainTuna said...

This library is great. Thank you!!

now to the question :=) :
I was thinking about using both the sending and the receiving functions in a single sketch (project which involves both sending and receving of IR) and i was asking myself: won't the interrupts called by the receiving function corrupt the code being sent with irsend? That is, when i call the function irsend, do the interrupts (called when timer1 overflows) get called also while i'm sending the data?

June 18, 2010 at 2:12 AM

Miles said...

Thanks Ken, I'll take a look at it and see what I can come up with.

What's the best way to simply replay the message? rawSend?

Once again, great work!

June 20, 2010 at 2:24 AM

Craig said...

I built an Arduino steering wheel to IR control circuit this weekend to control an xpressRCi XM satellite radio receiver in my car. Your fabulous library and my previous experience hacking JP1 remotes allowed me to get my project up and running in no time.

I did run into one issue with your library; there is no way to send RC5 extended codes. The fix would be to comment out the two lines that say "Second start bit" in IRsend::sendRC5() but doing that breaks any existing code so a better solution is to create IRsend::sendRC5X().

I'd also suggest tweaking the comments to IRsend::sendRC5():

```
-// Note: first bit must be a one (start bit)
+// Caller needs to take care of flipping the toggle bit
+// (which is 1st passed bit)
void IRsend::sendRC5(unsigned long data, int nbits)
```

BTW, the current comment is wrong; the code clearly sends both start bits.

Something like this would be appropriate for IRsend::sendRC5X():

```
+// RC5 extended
+// 1st passed bit is the 2nd start bit/field bit/6th
command bit
+// 2nd passed bit is the toggle (caller handles flipping)
+void IRsend::sendRC5X(unsigned long data, int nbits)
```

Here's a link to a context diff for IRremote.cpp/IRremote.

June 20, 2010 at 8:17 PM

Craig said...

I almost forgot; here's the macro I came up with to build the data bytes and send a RC5X commmand. (I actually expected to find something like this in IRremote.h!)

```
/*
* Construct 13 of the 14 bits of a RC5 extended
command
*
* (the 1st start bit is provided by irsend::sendRC5X())
* 2nd start bit/aka field bit/aka 7th command bit
* toggle bit
* 5 bits of device
* 6 bits of command (OBC format)
*/
#define RC5XCODE(dev, obc, toggle) \
(((((((obc) & 0x20) >> 4) | (toggle)) << 5) | \
((dev) & 0x1F)) << 6) | ((obc) & 0x3F))

/* XDPR1 XM Satellilte Radio remote RC5 device code */
#define MY_DEVICE 27

unsigned char toggle;
```

```
/* Send a RC5 extended command to the XM receiver */
void
sendrc5(unsigned char obc)
{
irsend.sendRC5X(RC5XCODE(MY_DEVICE, obc,
toggle), 13);
toggle = !toggle;
}
```

June 20, 2010 at 8:34 PM

CaptainTuna said...

Sorry for the above comment, i just noticed that you have written that it doesn't support simultaneuos receiving and transmitting.

But that makes me want to ask you something else: what does that exactly mean? Does it mean that when i use "irsend.sendSony(0xa90, 12)" the interrupt won't be called while transmission is being sent, or does that mean that just writing "IRsend irsend;" disables reception for the whole sketch? I guess it's the first case (=reception disabled WHILE transmitting) even because of your other project which receives codes from other remotes and duplicates them.

again, thank you for the great work!

June 21, 2010 at 1:27 PM

TomTom said...

what am I doing wrong? Trying to add my viewsonic remote. I added this code

CPP/H :::::::::::::::::::::::::::::

```
#define VIEWSONIC_HDR_MARK 9150
#define VIEWSONIC_HDR_SPACE 4400
#define VIEWSONIC_BIT_MARK 700
#define VIEWSONIC_ONE_SPACE 1550
#define VIEWSONIC_ZERO_SPACE 400

void IRsend::sendViewSonic(unsigned long data, int
nbits)
{
enableIROut(40);
mark(VIEWSONIC_HDR_MARK);
space(VIEWSONIC_HDR_SPACE);
for (int i = 0; i < nbits; i++) {
if (data & TOPBIT) {
mark(VIEWSONIC_BIT_MARK);
space(VIEWSONIC_ONE_SPACE);
}
else {
mark(VIEWSONIC_BIT_MARK);
space(VIEWSONIC_ZERO_SPACE);
}
data <<= 1;
}
space(0);
}
```

Arduino:::::::::::::::::::::::

```
#include

IRsend irsend;

void setup()
{
Serial.begin(9600);
}

void loop() {

if (Serial.read() != -1) {
for (int i = 0; i < 3; i++) {
irsend.sendViewSonic(0xC18F50AF, 32); // Viewsonic TV power code
delay(100);
}
}
}
```

Problem is it'll work once every 10 tries or so (and only turns off my tv not on). I put it up to a oscilloscope (solar cell going to my computer) and I get about 21hz from my viewsonic remote, but my arduino program changes between 10-889hz. Your sony code stays at about 19-20hz (sorry don't know if its hz, mhz, or khz, too lazy to turn on my other computer to see).

I changed enableIROut() int but still doesn't work. Am I missing something?

I have the following from the dump program:

```
Decoded NEC: C18F50AF (32 bits)
Raw (68): -7728 9150 -4400 700 -1500 700 -1600 700
-400 700 -400 700 -450 700 -400 750 -350 750 -1550
650 -1550 700 -450 700 -450 650 -450 700 -1550 700
-1550 700 -1550 700 -1550 700 -400 700 -1550 700
-400 700 -1550 700 -450 650 -450 750 -350 750 -400
700 -1550 700 -400 700 -1550 700 -450 650 -1600 650
-1600 700 -1550 700 -1550 700
```

June 22, 2010 at 10:04 PM

CaptainTuna said...

One last thing: above it says that the receiver interrupt code is called when TIMER1 overflows. I have checked the .cpp file and there is no sign of timer1 while there clearly is something related to timer2. Did i miss something?

June 23, 2010 at 6:21 AM

Ken Shirriff said...

@TomTom: I looked at the LIRC file for Viewsonic. I think the main thing is you need to send a mark(VIEWSONIC_BIT_MARK) at the end of your code. You could also try tweaking your timings a bit to match

the LIRC file:
http://lirc.sourceforge.net/remotes/viewsonic/RC00070P
but I think the missing mark at the end is the problem.

@CaptainTuna: yes, the code uses TIMER2. No simultaneous sending and receiving: when you send a code it disables IR input, and you need to call enableIRIn to start receiving again. (I do this because I kept running into trouble with the board receiving the code it was sending, so it seemed better to just do one thing at a time.)

@Craig: I didn't know there *were* RC5 extended codes; I'll look into adding them to the next version.

June 23, 2010 at 5:53 PM

TomTom said...

Thanks Ken!!! Adding mark(VIEWSONIC_BIT_MARK) at the end of my code worked (I also had it loop from 3 to 6) but now it works everytime! Should have noticed it from the data dump. Thanks again!

Now I have everything I need to complete my project!

June 23, 2010 at 9:12 PM

Craig said...

@Ken Shirriff: I didn't know there were RC5 extended codes; I'll look into adding them to the next version.

This is the first RC5 device I've encountered and I learned about the RC5 extended protocol from the Phillips RC-5 Protocol page you reference above.

@edward: I reduced your 100ms to 50ms and it worked. So you might consider changinge your sample snippet to 50ms delays between pulses.

I second this. In fact, your Understanding Sony IR remote codes, LIRC files, and the Arduino library page gives the minimum packet gap as 45076 us and I've found that with the Sony equipment I have, 100ms does not work but 45ms does.

June 24, 2010 at 11:05 AM

Craig said...

Regarding the next version, I have some ideas about handing sony12/sony15. I wanted a macro that could hand device and command (OBC) values. However thanks to the "LSB first" feature of the Sony protocol, this is a hassle. I started by writing code that used a reverse bit routine:

```
unsigned long
revb(unsigned long v, int s)
{
unsigned long r;
unsigned long m;
int i;

m = 0;
for (i = s - 1; i >= 0; --i) {
m <<= 1;
m += 1;
}

r = v;
--s;
for (v >>= 1; v; v >>= 1) {
r <<= 1;
r |= v & 1;
--s;
}
r <<= s;
return (r & m);
}

void
sendsony12(unsigned char dev, unsigned char obc)
{
unsigned long code;

/* sony12: 7 command bits and 5 device bits */
code = (revb(obc & 0x7F, 7) << 5) | revb(dev & 0x1F, 5);
for (int i = 0; i < 3; ++i) {
irsend.sendSony(code, 12);
delay(45);
}
}

void
sendsony15(unsigned char dev, unsigned char obc)
{
unsigned long code;

/* sony15: 7 command bits and 8 device bits */
code = (revb(obc & 0x7F, 7) << 8) | revb(dev & 0xFF, 8);
for (int i = 0; i < 3; ++i) {
irsend.sendSony(code, 15);
delay(45);
}
}
```

This all works but it's not very optimal. I considered writing macro's to do the bit reversing but cpp is fairly limited so you'd have to have one define per dev/cmd code, e.g.:

```
#define SONY8DEV_1 0x80
#define SONY8DEV_2 0x40
[...]
#define SONY8DEV_48 0x0C

#define SONY7OBC_20 0x14

/* sony15: 7 command bits and 8 device bits */
#define SONY15CODE(dev, obc) \
((((obc) & 0x7F) << 8) | ((dev) & 0xFF))

irsend.sendSony(SONY15CODE(SONY8DEV_48,
SONY7OBC_20), 12);
```

I think that's workable but it sure involves a lot of monkey motion.

My next idea was to make a send sony routine that clocks the bits out backwards from the other IR routines. That way the user doesn't have to worry about reversing the bits. He can pass the data in "normally" and the routine automatically reverses the bits. That works great:

```
#define SONY_REPEAT_MS 45

/* "Reverse" sendSony() */
void
sendSonyR(unsigned long data, int nbits)
{
irsend.enableIROut(40);
irsend.mark(SONY_HDR_MARK);
irsend.space(SONY_HDR_SPACE);
for (int i = 0; i < nbits; ++i) {
if (data & 1) {
irsend.mark(SONY_ONE_MARK);
irsend.space(SONY_HDR_SPACE);
} else {
irsend.mark(SONY_ZERO_MARK);
irsend.space(SONY_HDR_SPACE);
}
data >>= 1;
}
}

/* sony12: 7 command bits and 5 device bits (reversed)
*/
#define SONY12CODER(dev, obc) \
((((dev) & 0x1F) << 7) | ((obc) & 0x7F))

void
sendsony12(unsigned char dev, unsigned char obc)
{
unsigned long code;

code = SONY12CODER(dev, obc);

PORTB = 0x20;
sendSonyR(code, 12);
PORTB = 0x00;

delay(SONY_REPEAT_MS);

PORTB = 0x20;
sendSonyR(code, 12);
PORTB = 0x00;
}

/* sony15: 7 command bits and 8 device bits (reversed)
*/
#define SONY15CODER(dev, obc) \
((((dev) & 0xFF) << 7) | ((obc) & 0x7F))

void
sendsony15(unsigned char dev, unsigned char obc)
{
unsigned long code;
code = SONY15CODER(dev, obc);

PORTB = 0x20;
sendSonyR(code, 15);
PORTB = 0x00;

delay(SONY_REPEAT_MS);

PORTB = 0x20;
sendSonyR(code, 15);
PORTB = 0x00;
}

/* A/V mute */
sendsony15(48, 20);
```

sendSonyR() is smaller than sendSony() and it's straight forward to assemble codes.

(Note: I tested the "send thrice" but not the "send twice" version of the code yet -- my first/only dev board is currently in the car! But more parts arrive today...)

June 24, 2010 at 11:07 AM

Ken Shirriff said...

@Craig: I have some bit-reversing logic and logic to break apart Sony codes in the dev version: see http://github.com/shirriff/Arduino-IRremote . But that's the opposite direction from what you're doing; assembling the Sony codes from the component parts. I'll merge your code in when I have time.

Also, for anyone interested in Sony codes, see my blog post Understanding Sony IR remote codes.

June 24, 2010 at 8:59 PM

Anonymous said...

I'm trying this on my photoframe,
and it's working partialy.. The IRrecord sketch works, and it recognizes it as NEC.
Recieved NEC: 1FE7887.
When i push the button it sends it out, perfect..
But what do i do with this recieved code.. i tried to paste it in IRsendDemo, i get an error invalid suffix.

Other than my mistakes, its perfect for what i want, thanks

June 26, 2010 at 8:46 AM

Ken Shirriff said...

@anonymous: you need to put 0x before a hex value in C++. I.e. 0x1FE7887

June 26, 2010 at 5:26 PM

Anonymous said...

thanks ken,

for all your work and your feedback..

Now it's working great.

I've got a Photoframe, without the original remote, but found out, by accident, that all those cheap remote's (fotoframe, usb-dbv-t, and the 3$ remote nuelectronics sells, etc) have a few buttons in common.. So with the help of your program, and a lot of different remote's i testet, i made the control my photoframe.. Digital digital controlled..

thanks

mies,
the netherlands

June 27, 2010 at 2:16 AM

wingster said...

hi!

the library works really fine with some of my remotes, thank you very much! unfortunatly, most of my stuff is controlled with the panasonic protocol... and my skill seems not good enough to bring the code pieces together.

is there a "full" version of the library available including the panasonic protocoll?

if not, i will dive deeper in the code and will figure out why it's saying

/Applications/Arduino.app/Contents/Resources/Java/libraries/IRremote/IRremote.cpp:625: error: 'class decode_results' has no member named 'address'

at the moment.

thank you!

bye

wingster

June 28, 2010 at 12:43 PM

CaptainTuna said...

Thanks for the answers Ken. I got some more though (hope i'm not bothering you much with these):

1) there's a constant in your library called GAP_TICKS. If i well understood it is the min time before two consecutive transmissions. Why did you introduce it? Why do you want to have a minimum time between 2 transmissions and not make the receiver pick consecutive ones?

2) why are you using a time interrupt? I see that the ISR routine gets called when the TIMER2 overflows (set at 50us). Why not use the attachInterrupt offered by the arduino so that interrupt routine is called only when the arduino detects a change in the state of the input pin connected to the IR receiver (goes LOW from HIGH(idle)), and then maybe use pulsein to measure the lenghts of marks and spaces? It sounds much easier to me, but i'm not an expert like you so i am surely missing some CONs about this method.

June 29, 2010 at 3:20 AM

Anonymous said...

I've created some tv remote pranks using your library and an IR LED to send out data. For example: if they hit channel up, the arduino will send out channel down, and vice versa. Also, if they try to turn the TV off the arduino will turn it back on.
It's posted at the arduino website:
http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1277840151

June 29, 2010 at 1:10 PM

Andrew Rosenblum said...

I'm not able to get 2 receivers to work. I tried code simlar to TiStyles. If the pins assigned to RECV_PIN and RECV_PIN2 are reversed, only the receiver attached to RECV_PIN works. Please take a look at the below code. Is there a sample multiple receiver code?

```
/*
* IRremote: IRrecvDemo - demonstrates receiving IR codes with IRrecv
* An IR detector/demodulator must be connected to the input RECV_PIN.
* Version 0.1 July, 2009
* Copyright 2009 Ken Shirriff
* http://arcfn.com
*/

#include

int RECV_PIN = 4;
int RECV_PIN2 = 8;
int ledPin = 13;

IRrecv irrecv(RECV_PIN);
IRrecv irrecv2(RECV_PIN2);

decode_results results;

void blinkOnce() {
digitalWrite(ledPin, HIGH);
delay(400);
digitalWrite(ledPin, LOW);
}

void blinkTwice() {
digitalWrite(ledPin, HIGH);
delay(100);
digitalWrite(ledPin, LOW);
delay(300);
digitalWrite(ledPin, HIGH);
delay(100);
digitalWrite(ledPin, LOW);
}

void setup()
{
Serial.begin(9600);
irrecv.enableIRIn(); // Start the receiver
irrecv2.enableIRIn(); // Start the receiver

pinMode(ledPin, OUTPUT);
}

int successCount;
int errorCount;

void loop() {
boolean gotOne = false;
if (irrecv.decode(&results)) {
Serial.println("ir0");
gotOne = true;
} else if (irrecv2.decode(&results)) {
Serial.println("ir1");
gotOne = true;
}

if(gotOne) {
Serial.print("type=");
Serial.println(results.decode_type);
if (results.decode_type != -1) {
Serial.print("value=");
Serial.println(results.value, HEX);
Serial.print("number of bits=");
Serial.println(results.bits);
}

if (results.decode_type == -1) {
blinkTwice();
errorCount++;
} else {
blinkOnce();
successCount++;
}

Serial.print("success count=");
Serial.print(successCount); Serial.println("");
Serial.print("error count="); Serial.print(errorCount);
Serial.println("");

irrecv.resume(); // Receive the next value
irrecv2.resume(); // Receive the next value
}
}
```

June 30, 2010 at 8:47 PM

Andrew Rosenblum said...

Ignore last post. The problem was hardware. The second IR receiver was missing its 5V wire. (Yeah, I know it is a pretty simple circuit.)

Thanks for the great library. It makes my project much easier.

My project is tank war: 2 remote control bots that move around and can 'shoot' at each other. There would be 2-4 receivers to detect shots from any direction. The bots themselves would be remote controlled. The human commander can drive the tank and shoot.

Technologically, this is pretty similar to Polulo's beacon, http://www.pololu.com/catalog/product/701, but I got the idea from laser tag: http://www.ibm.com/developerworks/edu/os-dw-os-arduino1.html

The laser tag game sounds like fun too.

What are other people working on?

July 1, 2010 at 6:58 PM

CaptainTuna said...

Cool! I am also working on a laser tag game (http://www.laserforums.com/forum/showthread.php/4669-Arduino-Lasertag). We could join forces :=)

..and currently trying to get the transmission up to 56KHz

July 2, 2010 at 12:05 PM

Andrew Rosenblum said...

I tried 5 different 38 kHz receivers. I picked 38 kHz for compatibility with common remotes. They all work pretty well. The more expensive ones have better noise suppression, a wider reception angle and can pickup some reflections.

The 56 kHz with a short signal might be better for laser tag to minimize the dead time during transmission. I'm not sure if it matters. I have not put it all together to test usability.

I'm not sure which 38 kHz receiver is best for laser tag. Noise suppression seems good, but don't actually want to pick up reflections.

Current test circuits are just bread boards with exposed receiver and led. Next step would be too restrict hits to well aimed shots.

I will add a tube around the led. A lens would be better. In the IBM link a few posts back, a lens was added the doubled the range.

The receiver has 2 Vishay receivers. I'm not sure how many to use and of what quality for best results. One suggestion on arduino fourms was to use a single receiver with an acrylic lens. The lens would be a piece of an acrylic tube with the receiver in the hole. Not really sure how this works.

Any thoughts on either of these optical factors:
- the led tube and lens
- the receiver lens

July 2, 2010 at 4:30 PM

arduEve said...

@Andrew Rosenblum
Would you have the part numbers to the 5 different 38 kHz receivers you tried & could you comment on which turned out better?
Perhaps post on http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1251570417.
Also if anyone wants more room to discuss Ken's IR Library head on over there too.

@Ken might be good to add a prominent to the forum so that some of these comments might land there instead.

July 2, 2010 at 6:08 PM

arduEve said...

@Ken might be good to add a prominent **link** to the forum so that some of these comments might land there instead.

(& I thought i proof read)

July 2, 2010 at 6:11 PM

Jerry Zheng said...

```
#define RAWBUF 237 // a bigger buffer was modified
```

get a raw code as:

```
Raw (194): -7300 650 -1600 600 -1600 600 -1550 650
-1550 700 -1500 700 -1500 650 -1550 700 -1500 700
-450 600 -500 650 -500 600 -500 650 -500 600 -500
650 -500 600 -550 600 -1600 600 -1600 600 -1550 550
-1650 650 -1550 650 -1550 550 -1650 700 -1500 700
-450 600 -500 650 -500 600 -500 650 -500 550 -600
500 -600 550 -600 600 -1600 650 -1550 650 -1550 650
-1500 700 -1500 700 -1500 700 -1550 500 -1650 700
-450 600 -500 600 -550 550 -550 650 -500 650 -500
600 -500 650 -500 650 -1500 700 -1550 700 -1500 650
-450 650 -1550 650 -1550 550 -1650 700 -1500 700
-450 650 -450 650 -500 650 -1550 650 -450 650 -500
650 -450 650 -500 600 -500 550 -600 550 -1650 700
-450 600 -1600 650 -450 650 -1550 700 -1500 550
-1650 700 -1500 650 -450 650 -1600 650 -450 650
-1550 500 -600 600 -550 650 -500 600 -1550 550 -600
550 -1650 700 -450 550 -1650 650 -450 650 -500 600
-1600 700 -400 650 -1550 700 -450 600 -1550 700 -500
550 -1600 700 -1500 650
```

then a array was clared as

```
unsigned int powerOn[194] =
{650,1600,600,1600,600,1550,650,1550,700,1500,700,
1500,650,1550,700,1500,700,450,600,500,650,500,600,
500,650,500,600,500,650,500,600,550,600,1600,600,16
00,600,1550,550,1650,650,1550,650,1550,550,1650,70
0,1500,700,450,600,500,650,500,600,500,650,500,550,
600,500,600,550,600,600,1600,650,1550,650,1550,650,
1500,700,1500,700,1500,700,1550,500,1650,700,450,6
00,500,600,550,550,550,650,500,650,500,600,500,650,
500,650,1500,700,1550,700,1500,650,450,650,1550,65
0,1550,550,1650,700,1500,700,450,650,450,650,500,65
0,1550,650,450,650,500,650,450,650,500,600,500,550,
600,550,1650,700,450,600,1600,650,450,650,1550,700,
1500,550,1650,700,1500,650,450,650,1600,650,450,65
0,1550,500,600,600,550,650,500,600,1550,550,600,550
,1650,700,450,550,1650,650,450,650,500,600,1600,700
,400,650,1550,700,450,600,1550,700,500,550,1600,700
,1500,65};
```

so,the following codes were uploaded to arduino

```
############################################
######
#include

IRsend irsend;

// just added my own array for the raw signal
unsigned int powerOn[194] =
{650,1600,600,1600,600,1550,650,1550,700,1500,700,
1500,650,1550,700,1500,700,450,600,500,650,500,600,
500,650,500,600,500,650,500,600,550,600,1600,600,16
00,600,1550,550,1650,650,1550,650,1550,550,1650,70
0,1500,700,450,600,500,650,500,600,500,650,500,550,
600,500,600,550,600,600,1600,650,1550,650,1550,650,
1500,700,1500,700,1500,700,1550,500,1650,700,450,6
00,500,600,550,550,550,650,500,650,500,600,500,650,
500,650,1500,700,1550,700,1500,650,450,650,1550,65
0,1550,550,1650,700,1500,700,450,650,450,650,500,65
0,1550,650,450,650,500,650,450,650,500,600,500,550,
600,550,1650,700,450,600,1600,650,450,650,1550,700,
1500,550,1650,700,1500,650,450,650,1600,650,450,65
0,1550,500,600,600,550,650,500,600,1550,550,600,550
,1650,700,450,550,1650,650,450,650,500,600,1600,700
,400,650,1550,700,450,600,1550,700,500,550,1600,700
,1500,65};


void setup()
{
Serial.begin(9600);
}

void loop() {

// altered the code just to send/test my raw code
irsend.sendRaw(powerOn,194,38);
}
############################################
#########
```

but it dosen't work.neither "static unsigned int powerOn[194]"
then a led was used to instead of the IR led,it dosen't blink.

however,while use a smaller array such as "unsigned int powerOn[10] = {650,1600,600,1600,600,1550,650,1550}",the led blinked.
so i guess that the array powerOn[194] was too big.
the question:how to tackle the problem rised by those big raw codes?

jerry
great thx

July 3, 2010 at 10:58 AM

Anonymous said...

thank you very much!
thats a great library :-)
But I don't see any information about the license of the source code. Can i change the library, use it in another opensource project etc. ?!

July 3, 2010 at 1:00 PM

Andrew Rosenblum said...

Choice of receiver depends on the application.

My application is a tank bot with an ir gun. I'd like range of 10-20', but they shooter must be fairly accurate. The receiver should work 360 degrees.

I'm fairly enamored with the tsop 2438 and 1138. Both are wonderful receivers in terms of robustness and angle of acceptance. They seem to accept echos. The 2438 could read a signal from 31' away, the length of my living room. It would be a good choice for a remote control tv.

Today, I worked on restricting the shooting angle. The LED will be mounted in a cardboard tube. I'm not sure how much cardboard shields IR, but it seems to work.

There a several great posts about laser tag. Here they are looking for 100-300' range. I actually bought a magnifying glass to try this. But I could not get this to work.

Laser tag is a bit of a distraction, but the technology is so similar I felt like trying it. Besides using 56 kHz signals, they mount 3 sets of 3 sensors on a helmet. I imagine they just tie the signal pin together in each set to simplify the wiring. I tried both ways (connecting the signal pins to separate arduino inputs and connecting them to the same input), but did not reach any conclusions. I think I will use their design as it seems mature.

July 3, 2010 at 6:11 PM

CaptainTuna said...

You are right. Every lasertag system i've seen out there has the receivers connected in parallel. I am curious about having different pins for reception though, as it would be quite cool. It surely needs some modification of the library though.

July 4, 2010 at 1:44 AM

CaptainTuna said...

Hey guys. I modified the library a little, so that it can send IR in the background. Still have to test it out. Here is what i've done so far:

```
void IRsend::sendSony(unsigned long data, int nbits) {

if(irparams.bitstx == 0) { //if IDLE then transmit
enableIROut(40);
irparams.timertx=0; //reset timer
irparams.resettx=48; //initial header pulse is 2400us
long. 2400/50us ticks =

48
irparams.bitstx=nbits + 1; //bits left to transmit, including
header
irparams.datatx=data << (32 - bitstx); //unsigned long is
32 bits. data gets

shifted so that the leftmost (MSB) will be the first of the
32.
irparams.spacetx = false;
TCCR2A |= _BV(COM2B1); // Enable pin 3 PWM output
for transmission of header

}
}
```

```
ISR(TIMER2_OVF_vect)
{
RESET_TIMER2;

//TRANSMISSION

if(irparams.bitstx != 0) { //if we have got something to
transmit

irparams.timertx++;
if(irparams.timertx == irparams.resettx) { //if reset value
is reached
if(irparams.spacetx){ //it was a space that has just been
sent thus a

total "set" (bit + space) so..
irparams.spacetx = false; //we are not going to send a
space now
irparams.bitstx = irparams.bitstx - 1; //we got 1 less bit to
send
irparams.datatx <<= 1; //shift it so MSB becomes 1st digit
if(irparams.datatx & TOPBIT) {
irparams.resettx = 24;
TCCR2A |= _BV(COM2B1); //activate pin 3 PWM
(MARK)
}
else {
irparams.resettx = 12;
TCCR2A &= ~(_BV(COM2B1)); //disable pin 3 PWM (SPACE)
}
}
else { //we sent the bit, now we have to "send" the space
irparams.spacetx = true; //we are sending a space
irparams.resettx = 12; //600us/50us = 12
TCCR2A &= ~(_BV(COM2B1));
}
}
}
```

....continues with the rest of the ISR routine.

----
this goes in the IRremoteInt.h

```
// information for the interrupt handler
typedef struct {
uint8_t recvpin; // pin for IR data from detector
uint8_t rcvstate; // state machine
uint8_t blinkflag; // TRUE to enable blinking of pin 13 on IR processing
unsigned int timer; // state timer, counts 50uS ticks.
unsigned int timertx; // timer for counting 50us ticks (Tx)
unsigned int resettx; // reset value for above timer (Tx)
unsigned int bitstx; // number of bits left to send (Tx)
unsigned long datatx; // data to be transmitted (Tx)
boolean spacetx // true if space has been sent (Tx)
unsigned int rawbuf[RAWBUF]; // raw data
uint8_t rawlen; // counter of entries in rawbuf
}
irparams_t;
```

What do you think about it? Could it work. I'll test it out tomorrow to let you know.

July 5, 2010 at 10:18 AM

CaptainTuna said...

This comment has been removed by a blog administrator.

July 5, 2010 at 10:19 AM

CaptainTuna said...

This comment has been removed by a blog administrator.

July 5, 2010 at 10:20 AM

andrucha said...

Woww, I think this is just what I was looking for. However, I´m trying to send and receive ir signals in the same applicattion, and when I use the irsend.sendSony, it blocks the irrecv.decode(&results), and then it doesn´t work any more...

Do you have any idea which is my mistake?

```
#include

int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;

IRsend irsend;
const int buttonPin = 4; // the number of the pushbutton
pin
int buttonState; // current state of the button

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

void loop() {
buttonState = digitalRead(buttonPin);

if (buttonState == HIGH){
for (int i = 0; i < 3; i++){
irsend.sendSony(0xa50, 9); // Sony TV power code
Serial.println("a90");
delay(100);
}
}

if (irrecv.decode(&results)) {
Serial.println(results.value, HEX);
irrecv.resume(); // Receive the next value
tag = true;
}
}
```

July 19, 2010 at 2:20 PM

Rabiul said...

Thank you for the IR library. I have downloaded and tried it. I am getting the "Could not decode message" error. I read the messages by some other users who are also getting this and it is advised to use an inverter to fix this.

Can anyone please tell me how to use an inverter? Or any link having tutorial on using inverter with IR receiver module would be nice.

Thank you in advance.

July 21, 2010 at 2:55 PM

Rabiul said...

Referring to my above message the following messages was received by pressing 1 and 2 on Sony TV remote:

```
0

Could not decode message

Raw (76): -19820 300 -100 50 -100 50 -100 50 -100 50
-100 50 -100 50 -100 50 -100 150 -50 100 -50 100 -50
100 -50 100 -3450 300 -50 100 -100 50 -100 50 -100 50
-100 50 -100 50 -100 50 -100 150 -50 100 -50 100 -50
100 -50 100 -3450 300 -50 100 -50 100 -50 100 -50 100
-50 100 -50 100 -50 100 -50 200 -50 100 -50 100 -50
100

0

Could not decode message

Raw (76): -3050 300 -100 150 -50 100 -50 100 -50 100
-50 100 -50 100 -50 100 -50 200 -50 100 -50 100 -50
100 -50 100 -3350 300 -100 150 -50 100 -50 100 -50
100 -50 100 -50 100 -50 100 -50 150 -100 50 -100 50
-100 50 -100 100 -3350 300 -100 150 -50 100 -50 100
-50 100 -50 100 -50 100 -50 100 -50 150 -100 50 -100
50 -100 50
```

July 21, 2010 at 3:09 PM

Ribeiro Santos said...

I have a Samsung remote, and as this Excellent library IRemote hasn´t (yet!) a decode/send samsung. Using that also Excellent post http://www.maartendamen.com/?p=257, is simple to add that funcionallity. The send method is basically the same as NEC.

After follow the instructions in maartendamen (wich decodes the samsung ir),

add in IRemote.cpp:

```
void IRsend::sendSamsung(unsigned long data, int
nbits)
{
enableIROut(38);
mark(SAMSUNG_HDR_MARK);
space(SAMSUNG_HDR_SPACE);
for (int i = 0; i < nbits; i++) {
if (data & TOPBIT) {
mark(SAMSUNG_BIT_MARK);
space(SAMSUNG_ONE_SPACE);
}
else {
mark(SAMSUNG_BIT_MARK);
space(SAMSUNG_ZERO_SPACE);
}
data <<= 1;
}
mark(SAMSUNG_BIT_MARK);
space(0);
}
```

Add in IRemote.h

```
class IRsend
{
public:
IRsend() {}
void sendNEC(unsigned long data, int nbits);
//add this line
void sendSamsung(unsigned long data, int nbits);
```

After that, just use, in your sketch;
irsend.sendSamsung(, 32);

by example, next program
irsend.sendSamsung(0xE0E048B7, 32);

July 23, 2010 at 12:53 PM

---

weihang said...

Thanks Ken for the great library and Wolfgang for the Panasonic IR protocol. I managed to capture the signal from my remote control after tweaking the mark & space and moving the inline offset++ out to a line on its own, i.e.

```
for (int i = 0; i < 32; i++) {
if (!MATCH_MARK(results->rawbuf[offset++],
PANASONIC_BIT_MARK)) {
return ERR;
}
```

becomes

```
for (int i = 0; i < 32; i++) {
if (!MATCH_MARK(results->rawbuf[offset],
PANASONIC_BIT_MARK)) {
return ERR;
}

offset++;
```

Before modification, the inline offset++ adds 2 to offset. When offset++ is on its own, offset is incremented by 1. Can anyone explain why the unary increment operator behaves this way?

August 3, 2010 at 8:11 PM

Ken Shirriff said...

@weihang: you're getting the double increment because when the MATCH_MARK macro expands out, it evaluates the expression with the increment twice. You don't want to do anything with a side effect (e.g. ++) in a C++ macro. For details, see http://gcc.gnu.org/onlinedocs/cpp/Duplication-of-Side-Effects.html

August 5, 2010 at 10:54 PM

Jessarel said...

hi Ken, im trying to change the pwm from pin 3 to pin 11 with this mods:

```
// TCCR2A |= _BV(COM2B1);
TCCR2A |= _BV(COM2A1);

// TCCR2A &= ~(_BV(COM2B1));
TCCR2A &= ~(_BV(COM2A1));

// pinMode(3, OUTPUT);
// digitalWrite(3, LOW);

pinMode(11, OUTPUT);
digitalWrite(11, LOW);
```

But doesn't seems to be working,

Can you help me

August 16, 2010 at 3:09 PM

Jessarel said...

Solved

```
TCCR2B = _BV(CS20);
```

thanks :)

August 16, 2010 at 4:37 PM

Ken Shirriff said...

@Jessarel: I'm glad you figured out how to use the other timer. If you're looking for more information on how to use other boards or timers see http://www.pjrc.com/teensy/td_libs_IRremote.html where Paul Stoffregen has ported IRremote to multiple platforms. When I have time, I'll merge this in with my code, but for now, you can pick up the modified library there.

August 21, 2010 at 3:45 PM

afaucher said...

I used your library to make a laser tag gun compatible with the Phoenix LTX series (Nerf & Tiger). It did however required bumping the minimum gap from 5000 to 7000

http://executedata.blogspot.com/2010/08/phoenix-ltx.html

August 23, 2010 at 10:14 PM

ben said...

Awesome code, Ken. Thanks!
I thought I would add my twist: I need to send IR to 2

different applicances (TV and cable set top box). So I need to send IR out on pin 3 (as you do) and another pin. This means of course using another timer (Timer 1 for me).
My result is below for pin 9. It works, but it's hacked together and I would appreciate anyone improving it!
Ben

```
void IRsend::enableIROutNine(int khz) {
TIMSK1 &= ~_BV(TOIE1); // clears the timer (Timer1) overflow interrupt enable bit

pinMode(9, OUTPUT); // New output pin nine
digitalWrite(9, LOW); // When not sending PWM, we want it low

// Alternative, hacked together from the arduino eg (http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1199535681)
TCCR1A = _BV(WGM13) | _BV(WGM11) | _BV(WGM10);
TCCR1A = _BV(COM1A0); //start the timer
TCCR1A &= ~(_BV(COM1A0)); // Disable pin 9 PWM output
TCCR1B = ( _BV(WGM12) | 1); //set the predivide, in this case 1 = 001 means no prescaling

OCR1A = SYSCLOCK / 2 / khz / 1000;
OCR1B = OCR1A / 3; // 33% duty cycle
}


void IRsend::markNine(int time) {
// Sends an IR mark for the specified number of microseconds.
// The mark output is modulated at the PWM frequency.
//TCCR2A |= _BV(COM2B1); // Enable pin 3 PWM output
TCCR1A |= _BV(COM1A0); // Enable pin 9 PWM output
delayMicroseconds(time);
}

/* Leave pin off for time (given in microseconds) */
void IRsend::spaceNine(int time) {
// Sends an IR space for the specified number of microseconds.
// A space is no output, so the PWM output is disabled.
//TCCR2A &= ~(_BV(COM2B1)); // Disable pin 3 PWM output
TCCR1A &= ~(_BV(COM1A0)); // Disable pin 9 PWM output
delayMicroseconds(time);
}
```

August 29, 2010 at 3:20 AM

Vlad said...

Would anybody be kind to write an example line of code for sending a key for this remote:
http://lirc.sourceforge.net/remotes/dell/MR425

I have tried RC6 and raw (with the "IRrecvDump" received array), and no luck. I can see the IR led sending with a photo camera, but I get no reaction from the receiver.

September 2, 2010 at 12:27 PM

Ken Shirriff said...

@ben: For more info on using different timers, see Paul's extensions.

@Vlad: It looks like the Dell MR425 remote uses 37 bits, while my code only supports 32 bits. You could probably change the library to use "long long" integers. Are you trying to receive codes from this remote, or use the Arduino to send codes? To receive, see my article Using arbitrary remotes with the Arduino IRremote library. If you're trying to send using the raw codes, note that the RC6 protocol alternates between two different codes for each button, so you'll need to do that too.

September 4, 2010 at 11:58 AM

Vlad said...

Thanks for the reply Ken. I am trying to send codes. Receiving seems to work just fine. From all the remotes I have around in my place, I only managed to send keys to a cheap DVD player that uses NEC protocol. Four other remotes are not recognized. I've tried sending raw codes from all, but failed.
I've noticed that the raw codes recorded from all of the remotes begin with two random codes. For example, here's the same key of the same remote pressed a few times (once every two seconds):

```
Could not decode message
Raw (76): 11826 3400 -1650 500 -350 .....
Could not decode message
Raw (76): 20004 3400 -1600 500 -350 .....
Could not decode message
Raw (76): 5686 3400 -1600 500 -350 ......
Could not decode message
Raw (76): 3112 3400 -1650 450 -350 ......
```

They always start differently and sometimes, the third and forth number is (+) or (-) 50. Is this normal?

September 7, 2010 at 3:58 AM

Seth said...

can anybody help me send Raw codes?

Ive used IRrecvDump to get the following code:

```
Could not decode message
Raw (76): -9200 4450 -650 500 -650 1600 -650 1600
-650 1600 -650 450 -650 1650 -650 1600 -650 500 -700
450 -650 500 -650 450 -700 500 -650 500 -650 1600
-650 500 -650 450 -650 550 -650 1600 -600 550 -650
450 -650 1600 -650 1600 -650 1650 -600 1650 -650
1600 -650 1600 -650 450 -650 500 -650 450 -700 500
-650 500 -650 450 -650 500 -700 500 -650 1600 -700
```

1550 -650 1600

How do I send this code using irsend.sendRAW ?

Also, this remote is listing in the LIRC database, but it is obviously not recognized by this library. Can I use the LIRC information to produce a protocol for the library?

September 18, 2010 at 8:55 AM

arunreddy said...

First of all i would like to thank you for such a wonderful library.

Am a student trying to build a arduino clock with 7 segment displays driven by 74hc595. Trying to set time with IR Remote. The existing code uses a clock to set display. IR Remote is not getting detected or rather can say not working while sending the serial data.

I have kept the code here. http://pastebin.com/c8BYpeUr ...

Is it because of serial data push / ShiftOut causing the issue ? If yes, any help on how to rectify it. ?

Thanks.

September 18, 2010 at 11:56 AM

arunreddy said...

When shiftOut() function is being used in the code along with IR Remote library. The Reciever doesnt detect the key press actions.

ShiftOut : http://arduino.cc/en/Reference/ShiftOut

September 18, 2010 at 12:05 PM

lafs said...

Please can you show me a IR LED to buy in order the IRRecord example from IR arduino library to work.

I also have the PANASONIC IR Receiver and it works but I need also a IR LED to send the signal.

I Have Arduino Dueminalove 328. Please show me a IR LED for this purpose.

Also, I want the IR LED to work with NEC, RC5/6, SONY, etc..

Thanks!

September 23, 2010 at 4:53 AM

Gus Grubba said...

This is my very first Arduino program/sketch. My intention is to control my AV system. I've tried the

Logitech remotes, and lately a Phlips Pronto. In the end, none of them work. At least not in a way my wife or the children can use. I looked around and found the Arduino! Great. Now I can write my own and add controls/sensors to see if the things are actually working the way they are supposed to and report back so corrective actions can be taken. The idea (like many others here) is to provide a web interface to all this so I can access it all through an iPad.

This code will parse a Pronto Hex code set and send it through Ken's library.

Ken, thanks a lot! You saved me a lot of work. Feel free to do as you please with this code. As I said, it's my very first so it may not be all that. The famous: it works for me...

I tried appending the code to the comment but it was rejected for being too large (beyond the 4k limit). Instead, I've uploaded to my server. You can download it here: IRProntoSend.tgz.

Thanks!

g

September 24, 2010 at 11:48 AM

Gus Grubba said...

Oops... the link got messed up. Try this (no tags this time): http://www.xcreet.com/IRProntoSend.tgz

Thanks!

g

September 24, 2010 at 11:53 AM

WhiteHexagon said...

Wow! after wasting half a day trying to decode some IR signals with my own Arduino code, I'm so glad I came across your blog!! I'm using SEN-08554 from sparkfun, and your library seems to work perfectly :)

The whole reason for my project is I have an obscure remote for my A/C unit which I'm trying to 'record' a control for 'play back' at a later date via an 950 IR led (COM-09349).

IRRecvDump produces: Raw (76): -4348 3550 -1650 500 -1200 550 -1200 550 -300 550 [etc]

So my first question is what the best way to play back this raw data. Can I just put it back into an array and call SendRaw? or will the MARK_EXCESS errors cause me trouble.

Q2. I would have just tried the above, but I only have a single Arduino. So I tried a push button on an interrupt to trigger a send sony control (code from IRSendDemo).

However this seems to block and stop the receive functionality working. I'm know this isn't the multi-threaded world I'm used to with Java :) but is there any way to make the send and receive co-exist? I want to see if what I send is similar to what the original remote sent.

September 25, 2010 at 6:41 AM

Anonymous said...

Hey Guys,
Where should i unpack this on a MAC? I don't see any file structure for the arduino/hardware/libraries I am using an Arduino Mega and I can upload and run sketches on it fine but when I try to run an example from this project, it obviously doesn't know where the includes should be pointing to. I am new to this so I apologize in advanced for being stupid.

September 28, 2010 at 9:09 PM

Carl said...

This is a really great project!

Has anyone managed to get an Xbox 360 to power on? I know the code is RC6: 800F740C (36 bits), but that doesn't work using the command "irsend.sendRC6(0x800F740C, 36);". All my other remotes work perfectly using this.

Thanks a lot for your hard work Ken.

October 11, 2010 at 9:10 AM

Ken Shirriff said...

Carl: there's some discussion of the Xbox360 remote earlier in the comments; look for "Vishal". Unfortunately I don't think Vishal got it working. If anyone was successful with the Xbox360, please let me know.

October 12, 2010 at 8:33 PM

Carl said...

Hi again Ken,

I still haven't got it working, but I did manage to get a look at it under an oscilloscope today. Bear in mind, I have absolutely no idea how to work an oscilloscope, this was the first time I touched one, but I did get some sort of output (excuse the fact that this machine looks like it was built in the stone age).

What I tried to do was get the entire packet into the screen at once, so it's really tiny, but maybe you'll be able to spot something. I shortened the code to just loop the Xbox code with a delay(30), that seemed to match the output.

The following images are the best I could capture (how do you capture data on these old things?).

http://imgur.com/a0EF6.jpg -ARDUINO

http://imgur.com/hhAD5.jpg -XBOX REMOTE

When you look at these, they look pretty similar, but blurry and small as hell. I'll try to get something better to you soon. We have some digital scopes, but they were being used today, so I'll have a look again tomorrow if I can.

Thanks Ken.

October 14, 2010 at 9:37 AM

Ken Shirriff said...

Carl: The oscilloscope traces are extremely helpful. It looks like the code you want is 37 bits and your code is 36 bits. (I don't know how you got 36 bits since my code only supports 32 bits, so you must be doing something sneaky.) The other problem is the XBOX remote's code is 0x1001fe8180 (if I manually decoded the oscilloscope trace correctly), which is entirely different from the code you're sending. The timing all looks good, so I think you're close.

October 14, 2010 at 11:39 PM

Carl said...

Hello again,

I didn't get to use an oscilloscope today, but I have got a solution (sort of). I gave up on RC6 and went with Raw.

So, if you decode the raw file, assign the values to a string, and call that in irsend.sendRaw(), it will work. Now, the Xbox 360 sends 2 different codes for on/off, so this will only do one or the other. I'll decode the other one tomorrow, maybe.

Assign this outside of your loop:

```
unsigned int XboxOn[66] = {2700,850,500,400,450,400,500,850,450,850,1400,850,450,400,500,400,500,400,500,350,500,400,500,400,450,450,450,400,500,400,500,400,900,450,450,400,500,400,450,850,950,400,500,400,450,850,950,800,500,400,500,400,500,400,500,400,450,400,950,400,450,850,500,400,500};
```

Then call it with this command:

```
irsend.sendRaw(XboxOn,66,36);
```

That should either switch your xbox on, or switch it off, so you'll need a remote to send the other code.

Ken, what do you think would be the best way of implementing this so when the command is called, this code and the other corresponding raw code is sent? Just send them both fairly quickly and hope the Xbox doesn't just switch on and off again?

Thanks,

Carl

October 15, 2010 at 12:22 PM

Anonymous said...

Hi there Ken, I'm having a few problems getting this to work with my TV + DTV box. I think it's because of a weird protocol used by both. They're both manufactured by Thomson.
Here's a dump of the TVs Vol+ button:

```
0
Could not decode message
Raw (26): -26298 700 -1800 700 -1800 650 -4350 700
-4350 650 -1850 650 -1850 650 -4400 600 -1900 600
-4400 650 -1850 650 -1850 650 -4400 600
```

As you can see, it appears the marks and spaces are inverted. Wondering if it was someting wrong with my wiring, I checked a sony remote, and it decoded fine.

So, is there a way to invert the output/IRsend marks and spaces in code.
I want to do it in code because the dongle I'm looking at making will send out these non-standard codes alongside sony codes, so don't want to invert the output wholesale.

Cheers,
The Cageybee

October 16, 2010 at 3:36 AM

Anonymous said...

Hi again Ken. Forget my previous comment.

Sending the command as RAW is working. Just turns out I'm not getting much power from my IR LED.

Regards,
The Cageybee

October 16, 2010 at 7:51 PM

bleik said...

Hey
I have a mac and I downloaded your library and dumped the file under arduino/libraries next to the file arduino.jar
Then I tried to open the IrRecivdump and compile the program but it tells me Variable or field 'Dump' declared void and the first error line message is IRrecvDump.cpp:9:22: error: IRremote.h: No such file or directory

did i put the zip file in the wrong place? and if it is can someone can please tell me where to put it in a mac thx!!!

October 19, 2010 at 10:48 AM

Andrew Rosenblum said...

Hey Bleik,
The folder tree is: Documents, Arduino, libraries, IRremote. If he folder libraries does not exist, create it.

The bit about the Arduino.jar sounds fishy, but it could just be a jar that is not used.

You have to bounce the Arduino IDE to pickup this library.

Good luck!

October 20, 2010 at 4:55 PM

Saurabh said...

I am using the library for SAMSUNG Remote (http://www.maartendamen.com/?p=257) and it gives me output as OFFSET 3 using the IRRecvDump :(

October 20, 2010 at 6:30 PM

Saurabh said...

I am using the library for SAMSUNG Remote (http://www.maartendamen.com/?p=257) and it gives me output as OFFSET 3 using the IRRecvDump :(

October 20, 2010 at 6:30 PM

bleik said...

Hey Andrew,
Thanks so much it worked! woop!

October 21, 2010 at 8:46 AM

Nick E. said...

Hey Ken,

Just wanted you to know that I'm using your library for a robot that plays laser tag!

It takes commands from my SONY TV remote and fires an IR LED "Cannon" at the other guy.

It's working great!

Your library is very easy to use, saved me no doubt countless hours, and hasn't caused me any trouble all quarter long!

Thank you so much for putting your work out there for everyone, and even more for publicly helping all these people troubleshoot!

November 3, 2010 at 7:52 PM

Ken Shirriff said...

Hi Nick E! Thanks for your comment on your laser tag robot. It sounds like a really cool project - do you have a link to it?

For any of you who have done interesting things with the library, please send me a link to your project and I'll make a list of projects.

November 3, 2010 at 11:49 PM

Rotceh_dnih said...

Hay ken,

im very new to arduino but have picked up your code and have maneged to dump all my remote code's and play them back with the IRSendDemo with a push button one at a time but i wish to have 5 push button's each sending a differnt code,,,i have tryed to wright one script with 2 button's (one for power one for chan-up) but when i upload it only the first one work's
[code]

```
*/

#include

IRsend irsend;

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2; // the number of the pushbutton pin
const int buttonPin1 = 4; // the number of the pushbutton pin
// the number of the LED pin

// variables will change:
int buttonState = 0; // variable for reading the pushbutton status

void setup() {
// initialize the LED pin as an output:
Serial.begin(9600);
// initialize the pushbutton pin as an input:
pinMode(buttonPin, INPUT);
pinMode(buttonPin1, INPUT);
}

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

// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
if (buttonState == HIGH) {
// turn LED on:
(Serial.read() != -1); {
for (int i = 0; i < 3; i++) {
```

```
irsend.sendNEC(0x2FD48B7, 32); // MY TV power code


// read the state of the pushbutton value:
buttonState = digitalRead(buttonPin1);

// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
if (buttonState == HIGH) {
// turn LED on:

(Serial.read() != -1); {
for (int i = 0; i < 3; i++) {
irsend.sendNEC(0x2FDF807, 32); // MY tv Chan-up
}
}
}
}
}
}
}
[/code]
```

thnx in advance :)

November 5, 2010 at 7:49 AM

Ken Shirriff said...

@Rotceh_dnih: it looks like you are closing the for loop and the if statement in the wrong place, so the second signal will only be sent if both buttons are pressed. You need to move a couple of the closing braces up. Go to Tools -> Auto Format, and the code structure will be clearer. Let me know if a more detailed explanation would help.

November 6, 2010 at 12:32 PM

Nick E said...

Here's a link to a YouTube video demoing my project - the robot that played laser tag relying heavily on your library:

November 9, 2010 at 8:07 PM

Nick E said...

Sorry, this link:

http://www.youtube.com/watch?v=F8QwoZgwDbw

November 9, 2010 at 8:07 PM

Rick said...

Ken, is it possible to put an IR LED on each of the 6 PWM pins on an Arduino Pro Mini? I'm looking to address each IR output individually. The idea is to say something like: "3,stb,power_on" where 3=IR LED #3, stb=the remote, power_on=the power on command.

BTW - Great work! Thanks for the effort.

November 18, 2010 at 7:52 AM

Todd Treece said...

Here is a link to a page describing my use of the library with an iPhone for anyone who is interested.

http://unionbridge.org/design/ircommand

Thanks for the great library Ken!

November 18, 2010 at 7:56 AM

evan said...

im trying to decode a room locater that sends IR signals...

i cant tell if it is NEC or sony or what exactly ..

can someone help !

this is my dump values

```
Could not decode message
Raw (76): 14654 950 -300 350 -250 350 -550 400 -200
350 -250 400 -300 250 -300 4
00 -200 300 -300 350 -400 550 -250 300 -400 250 -600
300 -550 450 -300 250 -250
350 -250 300 -650 300 -250 350 -300 300 -250 400
-250 400 -200 350 -450 100 -350
550 -500 150 -300 350 -600 300 -750 50 -50 700 -300
250 -300 350 -500 450 -350
200 -350 350 -200 350 -250 400
0
Could not decode message
Raw (76): 15654 850 -300 200 -400 300 -550 350 -250
300 -350 250 -250 400 -350 2
50 -300 300 -400 200 -400 400 -400 400 -200 400 -550
300 -600 850 -500 250 -250
300 -600 350 -300 300 -250 350 -450 100 -350 350
-300 300 -200 350 -1250 350 -25
0 350 -700 150 -600 350 -900 350 -200 400 -650 200
-250 450 -300 200 -450 200 -3
50 250 -250 450 -850 600 -400 250
0
Could not decode message
Raw (76): 15654 1000 -450 100 -400 300 -550 350 -300
300 -300 300 -300 300 -350
250 -250 400 -200 400 -200 350 -50 200 -400 400
-1100 400 -550 600 -650 300 -300
250 -650 300 -300 350 -250 300 -400 200 -300 350
-250 350 -250 350 -250 650 -10
00 250 -600 350 -750 750 -250 300 -350 250 -650 300
-300 300 -450 100 -350 350 -
350 200 -300 350 -300 300 -400 450
0
Could not decode message
Raw (76): 15254 950 -250 350 -300 300 -550 350 -300
300 -900 250 -400 250 -300 4
```

```
00 -200 400 -300 300 -600 300 -300 300 -600 300 -550
850 -450 250 -300 350 -550
400 -300 300 -250 350 -850 400 -200 300 -350 300
-1200 350 -350 250 -600 300 -60
0 250 -400 300 -200 400 -300 300 -1200 300 -950 300
-350 300 -250 350 -400 50 -4
50 550 -350 250 -350 350 -550 350
0
Could not decode message
Raw (76): 15504 950 -200 400 -250 350 -600 300 -250
400 -350 250 -300 300 -300 3
50 -300 200 -400 250 -250 650 -350 300 -300 300 -550
350 -600 900 -300 400 -250
250 -600 400 -300 300 -250 350 -300 300 -250 350
-900 300 -250 700 -300 300 -250
400 -1450 950 -350 250 -350 250 -650 200 -300 400
-250 350 -250 350 -900 350 -3
00 300 -300 550 -350 300 -350 250
0
Could not decode message
Raw (76): 15404 950 -250 400 -250 300 -600 350 -350
250 -250 400 -200 350 -250 3
50 -350 250 -250 400 -300 550 -300 350 -200 450
-1500 900 -300 300 -300 350 -650
200 -350 300 -250 300 -350 300 -400 100 -400 300
-350 300 -250 650 -250 350 -35
0 250 -600 350 -650 900 -250 300 -300 300 -650 350
-200 350 -250 350 -250 400 -3
00 300 -350 150 -450 250 -300 600
0
Could not decode message
Raw (76): 15454 900 -200 350 -250 400 -650 250 -250
450 -200 300 -250 300 -400 3
00 -300 250 -950 650 -200 400 -300 200 -650 400 -650
850 -200 400 -250 300 -600
350 -450 150 -300 300 -350 300 -200 400 -300 300
-250 350 -300 650 -400 150 -350
300 -1450 950 -900 350 -600 350 -250 350 -250 350
-250 350 -350 100 -500 300 -3
00 250 -350 550 -450 150 -300 350
0
Could not decode message
Raw (76): 14654 1000 -250 350 -250 350 -1200 350
-200 400 -250 400 -250 300 -300
350 -900 600 -250 450 -1050 400 -600 900 -300 300
-350 300 -650 250 -250 350 -3
00 300 -450 150 -300 350 -300 300 -400 200 -250 700
-300 300 -1150 350 -600 950
-200 400 -1150 350 -300 350 -250 300 -350 300 -250
350 -900 350 -250 600 -350 10
0 -550 250 -1450 1000 -1000 150 -600 350
0
Could not decode message
Raw (76): 14654 1000 -200 400 -300 250 -600 350 -250
400 -250 350 -200 400 -350
150 -400 300 -250 350 -250 650 -350 300 -200 400
-600 350 -550 900 -450 50 -400
350 -550 350 -300 300 -350 300 -300 250 -400 200
-300 350 -250 350 -300 400 -500
350 -1200 350 -500 600 -650 350 -400 150 -600 400
-200 400 -250 350 -350 250 -2
50 350 -300 350 -300 250 -300 600
0
```

```
Could not decode message
Raw (76): 15604 1000 -200 450 -400 200 -600 300 -250
400 -250 300 -250 400 -300
250 -300 350 -250 350 -200 750 -800 350 -600 350
-650 850 -900 350 -550 300 -300
350 -350 300 -200 300 -350 300 -300 350 -250 400
-350 350 -400 400 -400 150 -60
0 350 -550 650 -650 250 -300 350 -600 300 -300 300
-250 400 -400 100 -350 350 -3
00 300 -300 300 -350 600 -250 300
0
```

November 18, 2010 at 3:21 PM

Post a Comment

1 – 200 of 858 Newer› Newest»

Links to this post

Create a Link

Newer Post Home Older Post