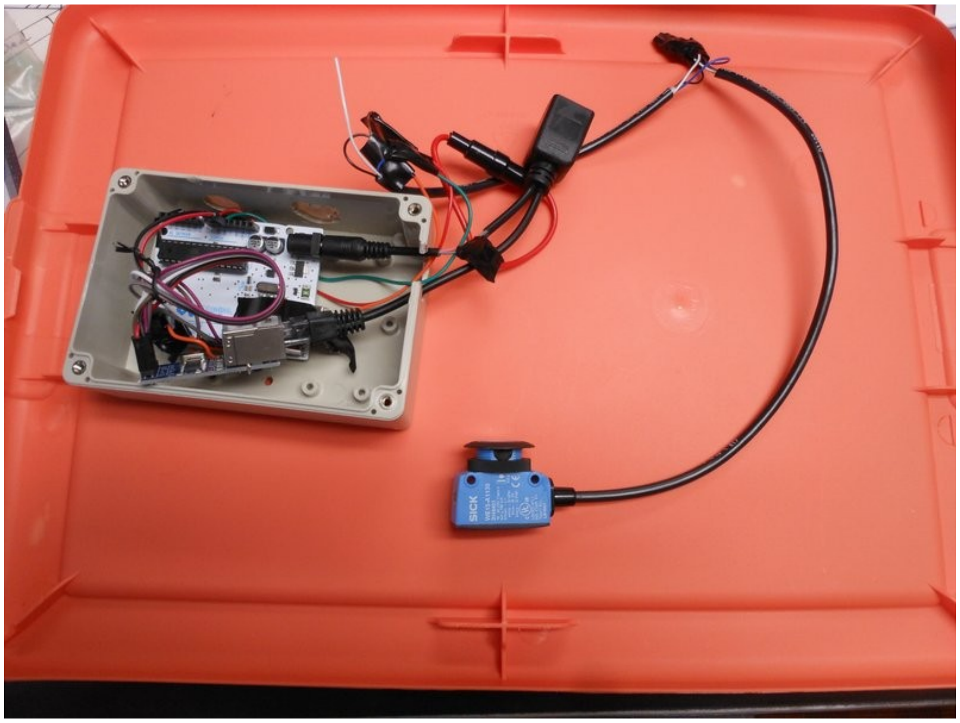
ZMHW Project

Infrared Diode Laser Motion Sensor

Objective: To make a motion sensor that acts as motion detection for Zoneminder cameras. As the cameras often have false alarms, an external sensor is a possible solution. This uses a Laser Diode Infrared Sensor.



Main half of the sensor before putting in enclosure.

Parts List:

Arduino Uno (official recommended)(DIP recommended)
ENC28J60 ethernet module
Passive PoE adaptors for IP Cameras
Series 1A fuse
Sick WS15-D1130 Infrared Laser Diode Motion Sensor
General Purpose Diode (I used 1N4818 diode) (may also use transistor, per data sheet for Sick)
Jumper Wires
Copper Wire (22-26 gauge)
Enclosure
Ethernet Wire
(optional) Low Profile one and two gang wall outlet
(optional) Blank cover plate, for one and two wall gang wall outlet
(optional) Electrical tape (I prefer halfway decent electrical tape)
(optional) piezo speaker
(optional) extras of everything, in case anything fails

Work Log:

This work log will be pictures with some notes thrown in. I'll try to make note of all important parts.

Device was assembled and using the ZMHW Project source code. This is simply an Arduino sketch with UIPEthernet (to use the ENC28J60) (make sure CS is pin 10 on Uno). For more details see source code. Explaining the details is out of the spec of this doc. Simply put, the ENC28J60 is connected, the Sick sensor black wire is connected to Analog input 1, and a speaker is connected. See source code. I will try to put a fritzing diagram in the git repo.

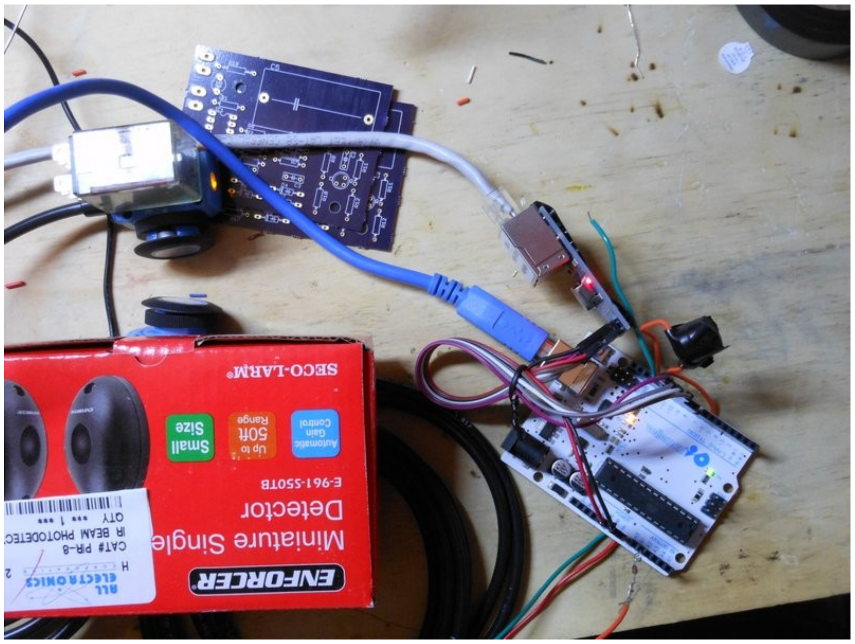


Figure 1: Orange LED on top of sensor when link detected. Series diode on output of sensor to cheat the need for transistor.

Of importance, **Figure 1** shows two things, first off a diode connected in series with the output of the Sick sensor, and also the orange LED on the top of the sensor. The orange led will be green when there is no connection between the diodes and orange when the Laser Diodes (or LEDs) are lined up correctly. When someone moves across the field of their vision, the orange LED will change to green.

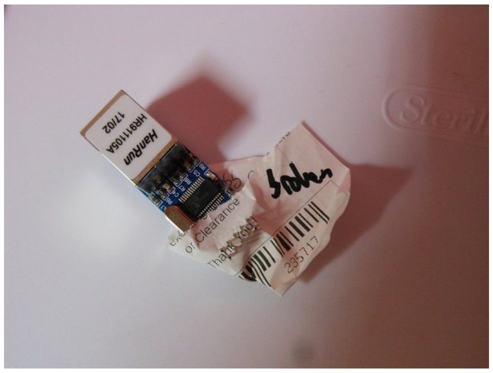
Diode on output of Sick sensor
Some laser sensors output a high or low. Some, like the Sick sensor, output a high or low (depending on whether you connect to white or black wire), however they are meant to be connected to a transistor, and thus if you connect it directly to a micro expecting it to go high or low, it will not. I dont want to deal with a transistor as I am lazy, so instead I put a 1N4819 in series with the output of the Sick sensor.
TODO: pictures showing waveforms
Edit: This is possibly an issue of output impedance.

Using the black wire, it will be normally low and go high when motion is detected (the white wire is the opposite). If you connect to a micro it will fail to go high (why?). If you put a diode on the end in series, it will turn the normally low to a noisy normally low, and sometimes it will go between 2.5-5 volts in spikes. This allows us to use the ADC to read the Sick sensor, and avoid the use of adding a transistor in. The transistor would allow for a digitalRead to be used, but we have plenty of Analog inputs to use, so let's use one of those.

It's very important to line up these sensors. If they are not lined up precisely, they will not get a sync, and the motion detection will fail. This will become important later, when we install.

Broken ENC28J60
During my testing, I suddenly was unable to get an IP address. I checked the example sketches, then began tearing down my setup, testing another Arduino and ENC module.

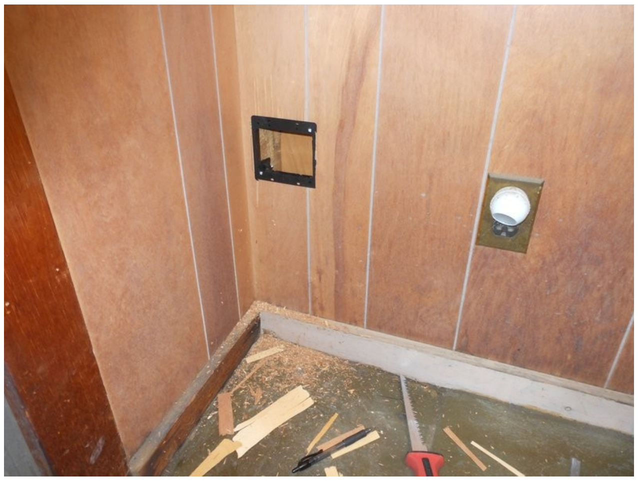
It turned out, the ENC28J60 module failed on me. Make sure to buy backups.

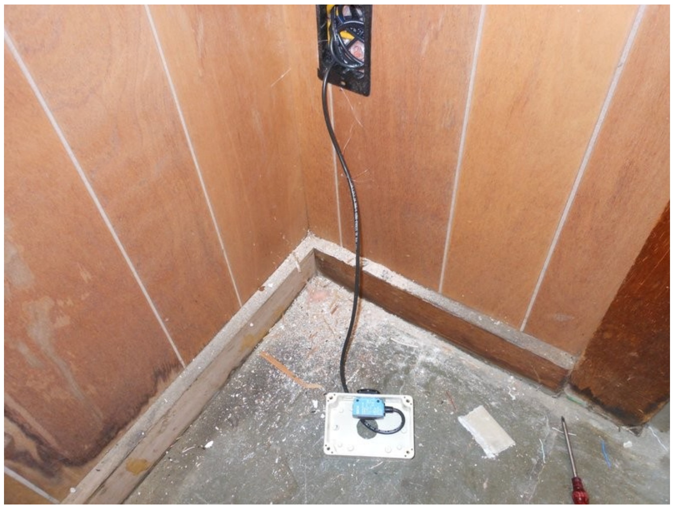


Picture Log:

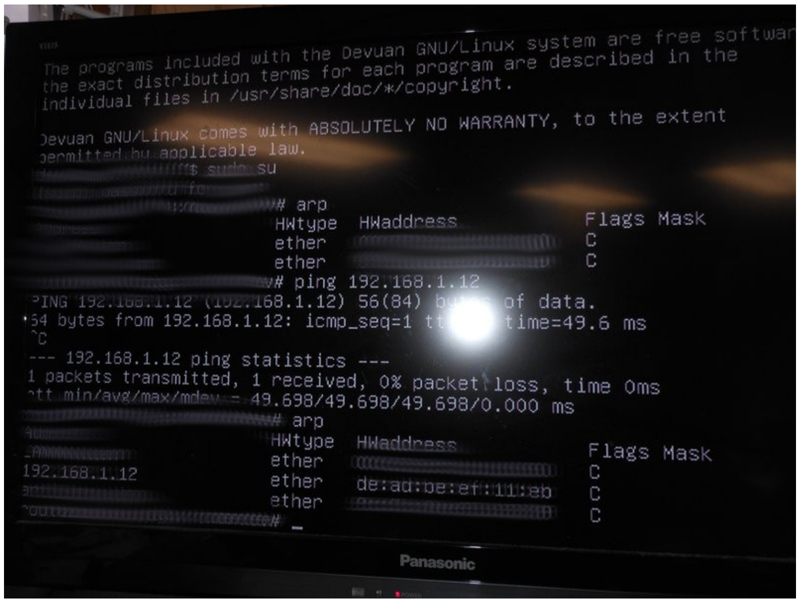
Choosing location for the sensors. Keep in mind, that the laser diode path must remain open. This is where the main board and sensor will be.

Feeding nylon string, and a wire up from the bottom of the wall to the ceiling. This is where the


Low profile two gang wall plate installed.


Installing the single sensor. I used ethernet as the power line. I also used a passive PoE adapter on both sides to transfer the power onto the ethernet, and back out to a 5.5, 2.1mm barrel plug

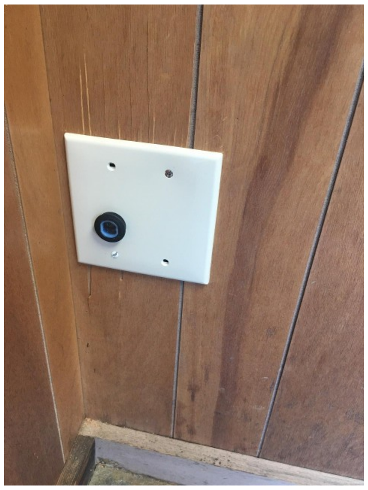
Box installed on the wall. This is a temporary box I used for testing purposes. The additional hole at the top is an error, but allows viewing the LED (though this sensor does not change its LED colour, it's always green).



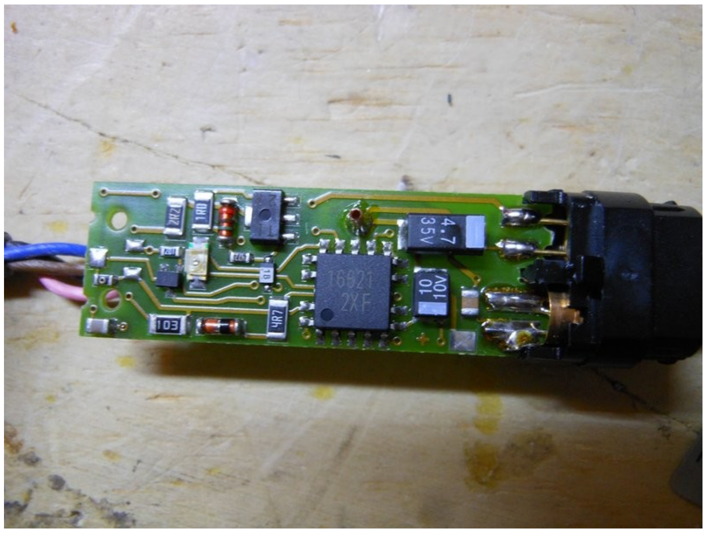
```
The programs included with the Devuan GNU/Linux system are free softwar
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Devuan GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
[redacted]$ sudo su
[redacted]
[redacted]# arp
[redacted]      HWtype  HWaddress           Flags Mask
[redacted]      ether   [redacted]          C
[redacted]      ether   [redacted]          C
[redacted]# ping 192.168.1.12
PING 192.168.1.12 (192.168.1.12) 56(84) bytes of data.
64 bytes from 192.168.1.12: icmp_seq=1 tt[illegible] time=49.6 ms
^C
--- 192.168.1.12 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 49.698/49.698/49.698/0.000 ms
[redacted]# arp
[redacted]      HWtype  HWaddress           Flags Mask
[redacted]      ether   [redacted]          C
192.168.1.12    ether   de:ad:be:ef:11:eb   C
[redacted]      ether   [redacted]          C
[redacted]# _
```

Checking the device is on the Camera LAN by pinging it, then reviewing the arp tables to make sure it's the right device (IP is static, so it's possible a conflict could arise)

This mangled web of wires, is a temporary testing ground, while I go to the store to purchase a two gang blank plate. In this setup, I will calibrate the diodes to point correctly at each other by setting this double gang receiver diode to be fixed, and then adjusting the opposite laser diode. For testing though, some electrical tape, and the power wire pulled out will do temporarily. When the lights are off, the red diode light is more visible and may be easier to calibrate.



Here's how the outlet looks just before putting the final machine screws in. It was important to buy a "non breakable" wall plate (essentially a bendable more rubberized one), as the normal wall plate, simply shattered when drilled into. The non breakable version worked well with the drill.



Omrom Motion Sensor
All electronics is also currently selling used Omrom photoelectric sensors, they are model: e3f2-r2c4. These types of photoelectric sensors are from a large catalog of different types. Some AC some DC powered. Different reaches, etc...
See resources in this git repository for some PDFs.

I tested one but had poor results. I was only able to get the light to flash when I dismantled the device. Teardown pictures are in the photos folder. Here's an example. The devices were not easy to dismantle, and can't really be put back together as well as they were originally. However, they did seem otherwise well made. More testing will be needed on these.

An example of a low cost photo electric sensor from ebay.

Testing Attempt #2
After finding some documentation on these in the reviews, I figured out why it wasn't working for me. These opto electronics absolutely require some type of reflector opposite them. Now, not all photoelectric sensors are like this, but this Omrom model is. I've also purchased low cost ebay photoelectric sensors in the past, and those require no kind of reflector. Here is the review which details how to use them:

Good quality product. It has reverse polarity protection, etc. This has the emitter and receiver built-in. Best to use a retroreflector like a bicycle reflector or retroreflector tape. The indicator light will come on to show the state. The logic output is open collector and you'll get whatever the supply voltage is. To use with 5v ttl (using a second 5v source) wire as such:

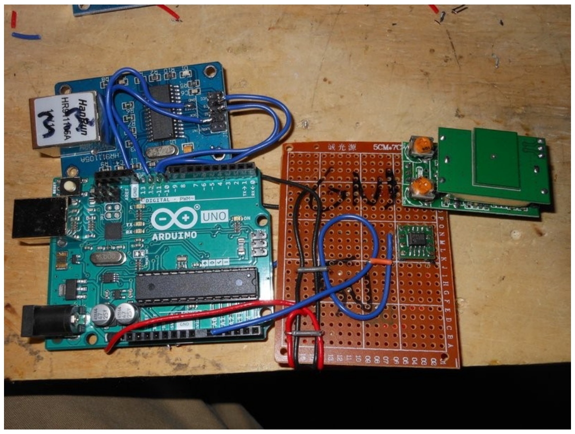
Brown to +12V; Blue to ground; Pink to either +12 or ground depending whether you want Light-ON or Dark-ON mode;

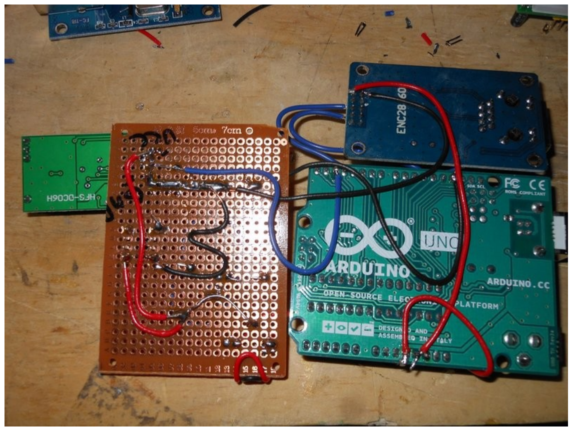
Black to a 4.7K resistor with the other side of the resistor connected to a separate +5V source (the arduino). The 5v ttl signal is at the point where the black wire connects to the resistor.

After trying with a component bag, which is slightly reflective, I was able to find a point where the emitter was consistently able to get a high result. This comment also tells you where to obtain the TTL signal, we need for a micro to register a high. Therefore, I purchased some reflective stickers available from all electronics, as well as some iron-on reflective material from ebay. I will test both of these out next.

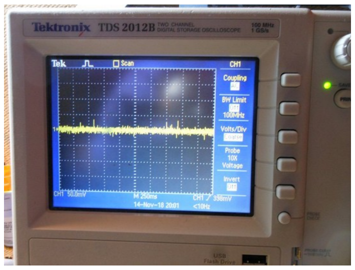
One thing I also noticed, was that used photo electric sensors from brand names can be obtained for discounts on the auction site. I saw a sick motion sensor for $10. It may be wise in the future to look at auction sites, to see if a good deal can be had.

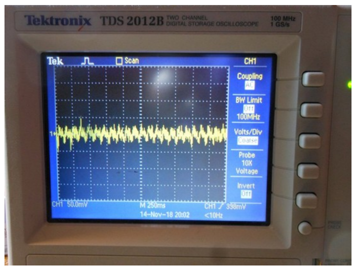
Revision 2
Using the HFS-DC06H Microwave Sensor
As I've tried with the HB100, without success, likely due to no included op amp (I tried throwing a cheapo one at it, but results were inconsistent), I've moved to the HFS microwave sensor. The code in the repo has been updated. I've also made a arduino uno shield layout to speed up assembly, which will be in this repo at some point. For now, perf board will do, while I wait for the boards to arrive.

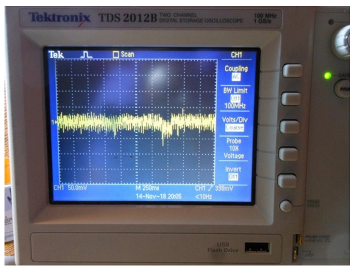


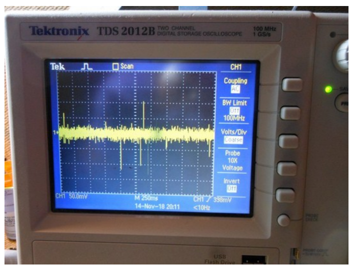


During testing, I found that the HFS sensor would not work correctly with my laptops usb 2.0 power supply. I thought it might be RF interferance from the metallic perf board – something I've seen before with the FM bug radio – but it was not. The solution was to use external power. See the four pictures (50mV/div). *1) with HFS powered direct from a bench PSU, 2) Arduino USB powered without HFS, but with ENC28J60 3) HFS from Arduino powered by USB (ENC+HFS), 4) from Arduino Uno with external 12V PSU (ENC+HFS).*









The HFS draws negligible power, in this application, <5mA. The ENC on the other hand... (ethernet or wifi are around 200mA each, 3.3v). I suspect noise not capacity was the problem. Simply adding a capacitor to the HFS (I tried a 220uF name brand), did not resolve the issue. A worthy test would be an

HFS with Arduino and no ENC. The usb power has limits that may be the culprit.

The HFS seems sensitive and responsive No false alarms upon initial testing. I use it at the shortest time setting on the potentiometer.

Uno Memory Limitations
Using ethernet with the Uno is always touchy. Version control is important, to have a functional version to work off of.

When writing my code, I found errors creep in due to using too much dynamic memory. You can see how much dynamic memory is used in Arduino by hitting verify (not upload but verify). You can also use a tool to see how much SRAM is used (code is online:https://jeelabs.org/2011/05/22/atmega-memory-use/) the following function:

“Here’s a small utility function which determines how much RAM is currently unused:

```
int freeRam () {
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}
```

And here’s a sketch using that code:

```
void setup () {
    Serial.begin(57600);
    Serial.println("\n[memCheck]");
    Serial.println(freeRam());
}

void loop () {}
```

The result will be:

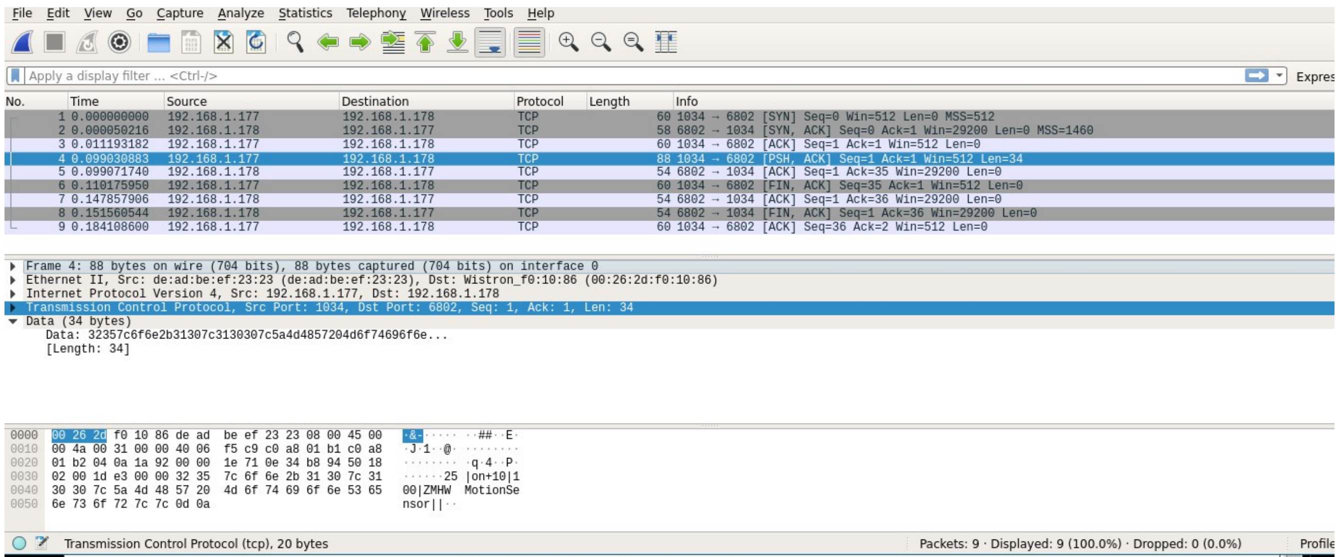
```
[memCheck]
1846
```

“

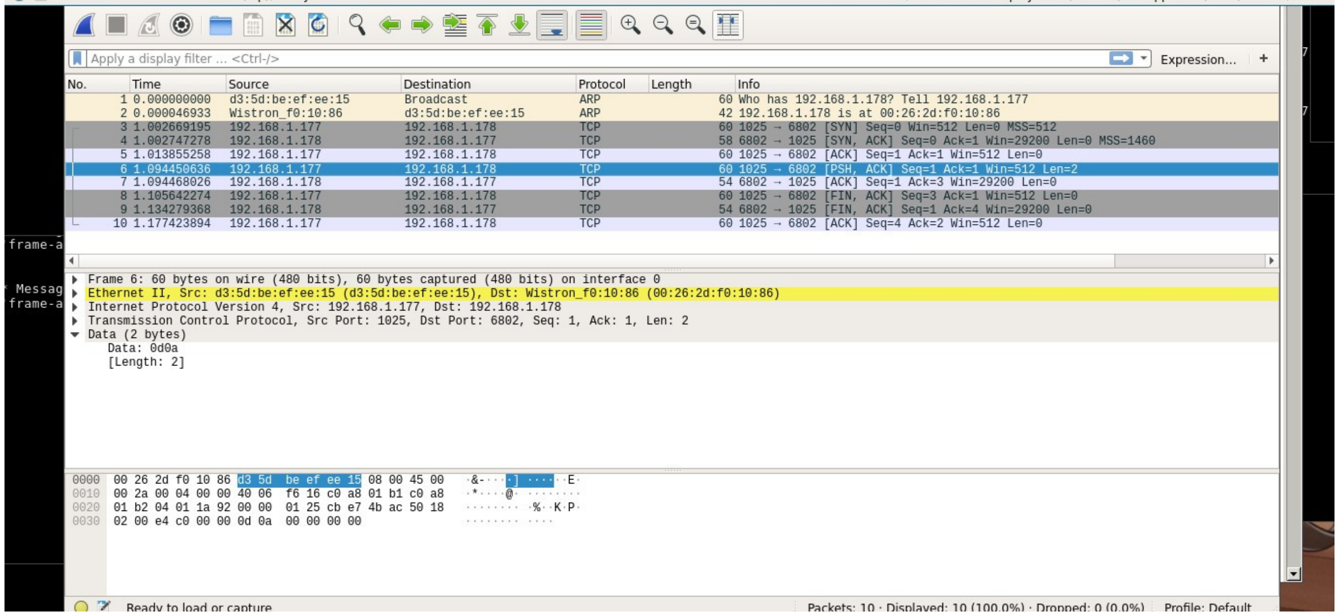
This is not a new problem for me, but it rears its ugly head again. However, this is a good thing. Limits are good.

An easy resolution for this is to put all serial.print lines into flash memory. You can verify this helps, by taking a serial.print, and commenting it out, and comparing the before and after dynamic memory used in verify. To put serial print lines in flash:
(https://www.arduino.cc/reference/en/language/functions/communication/serial/write/)
(https://www.arduino.cc/reference/en/language/functions/communication/serial/print/)
As I recall, there may be limitations to what you can do with Serial.print(F());

Low RAM errors can creep into strange places. For example, see these two wiresharks, where my code was running, equally as well, but the new code revision simply didn't work:
As you can see the data packet is mangled in the new rev. I've seen this enough to know, it was low SRAM. Otherwise, the code worked without major error.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expres

```
No.   Time         Source          Destination     Protocol  Length  Info
    1 0.000000000  192.168.1.177   192.168.1.178   TCP          60 1034 → 6802 [SYN] Seq=0 Win=512 Len=0 MSS=512
    2 0.000050216  192.168.1.178   192.168.1.177   TCP          58 6802 → 1034 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
    3 0.011193182  192.168.1.177   192.168.1.178   TCP          60 1034 → 6802 [ACK] Seq=1 Ack=1 Win=512 Len=0
    4 0.099030883  192.168.1.177   192.168.1.178   TCP          88 1034 → 6802 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=34
    5 0.099071740  192.168.1.178   192.168.1.177   TCP          54 6802 → 1034 [ACK] Seq=1 Ack=35 Win=29200 Len=0
    6 0.110175950  192.168.1.177   192.168.1.178   TCP          60 1034 → 6802 [FIN, ACK] Seq=35 Ack=1 Win=512 Len=0
    7 0.147857906  192.168.1.178   192.168.1.177   TCP          54 6802 → 1034 [ACK] Seq=1 Ack=36 Win=29200 Len=0
    8 0.151560544  192.168.1.178   192.168.1.177   TCP          54 6802 → 1034 [FIN, ACK] Seq=1 Ack=36 Win=29200 Len=0
    9 0.184108600  192.168.1.177   192.168.1.178   TCP          60 1034 → 6802 [ACK] Seq=36 Ack=2 Win=512 Len=0

▸ Frame 4: 88 bytes on wire (704 bits), 88 bytes captured (704 bits) on interface 0
▸ Ethernet II, Src: de:ad:be:ef:23:23 (de:ad:be:ef:23:23), Dst: Wistron_f0:10:86 (00:26:2d:f0:10:86)
▸ Internet Protocol Version 4, Src: 192.168.1.177, Dst: 192.168.1.178
▸ Transmission Control Protocol, Src Port: 1034, Dst Port: 6802, Seq: 1, Ack: 1, Len: 34
▾ Data (34 bytes)
    Data: 32357c6f6e2b31307c3130307c5a4d4857204d6f74696f6e...
    [Length: 34]

0000  00 26 2d f0 10 86 de ad  be ef 23 23 08 00 45 00   ·&-·····  ··##··E·
0010  00 4a 00 31 00 00 40 06  f5 c9 c0 a8 01 b1 c0 a8   ·J·1··@·  ········
0020  01 b2 04 0a 1a 92 00 00  1e 71 0e 34 b8 94 50 18   ········  ·q·4··P·
0030  02 00 1d e3 00 00 32 35  7c 6f 6e 2b 31 30 7c 31   ······25  |on+10|1
0040  30 30 7c 5a 4d 48 57 20  4d 6f 74 69 6f 6e 53 65   00|ZMHW   MotionSe
0050  6e 73 6f 72 7c 7c 0d 0a                            nsor||··

Transmission Control Protocol (tcp), 20 bytes          Packets: 9 · Displayed: 9 (100.0%) · Dropped: 0 (0.0%)    Profile
```



```
No.   Time         Source             Destination        Protocol  Length  Info
    1 0.000000000  d3:5d:be:ef:ee:15  Broadcast          ARP          60 Who has 192.168.1.178? Tell 192.168.1.177
    2 0.000046933  Wistron_f0:10:86   d3:5d:be:ef:ee:15  ARP          42 192.168.1.178 is at 00:26:2d:f0:10:86
    3 1.002669195  192.168.1.177      192.168.1.178      TCP          60 1025 → 6802 [SYN] Seq=0 Win=512 Len=0 MSS=512
    4 1.002747278  192.168.1.178      192.168.1.177      TCP          58 6802 → 1025 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
    5 1.013855258  192.168.1.177      192.168.1.178      TCP          60 1025 → 6802 [ACK] Seq=1 Ack=1 Win=512 Len=0
    6 1.094450636  192.168.1.177      192.168.1.178      TCP          60 1025 → 6802 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=2
    7 1.094468026  192.168.1.178      192.168.1.177      TCP          54 6802 → 1025 [ACK] Seq=1 Ack=3 Win=29200 Len=0
    8 1.105642274  192.168.1.177      192.168.1.178      TCP          60 1025 → 6802 [FIN, ACK] Seq=3 Ack=1 Win=512 Len=0
    9 1.134279368  192.168.1.178      192.168.1.177      TCP          54 6802 → 1025 [FIN, ACK] Seq=1 Ack=4 Win=29200 Len=0
   10 1.177423894  192.168.1.177      192.168.1.178      TCP          60 1025 → 6802 [ACK] Seq=4 Ack=2 Win=512 Len=0

▸ Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: d3:5d:be:ef:ee:15 (d3:5d:be:ef:ee:15), Dst: Wistron_f0:10:86 (00:26:2d:f0:10:86)
▸ Internet Protocol Version 4, Src: 192.168.1.177, Dst: 192.168.1.178
▸ Transmission Control Protocol, Src Port: 1025, Dst Port: 6802, Seq: 1, Ack: 1, Len: 2
▾ Data (2 bytes)
    Data: 0d0a
    [Length: 2]

0000  00 26 2d f0 10 86 d3 5d  be ef ee 15 08 00 45 00   ·&-···]  ······E·
0010  00 2a 00 04 00 00 40 06  f6 16 c0 a8 01 b1 c0 a8   ·*····@·  ········
0020  01 b2 04 01 1a 92 00 00  01 25 cb e7 4b ac 50 18   ········  ·%··K·P·
0030  02 00 e4 c0 00 00 0d 0a  00 00 00 00               ········  ····

Ready to load or capture          Packets: 10 · Displayed: 10 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

Here are some pictures of the build. As I mentioned in the code this perf board was NOT the fast way of doing things. I've already got a PCB in the mail to make this faster in the future, but the customer needed boards sooner rather than later, so perf board was used for a quick build.

Hammond cases (relatively large) with hammond cable glands. Purchased from the local Needham, MA You do it electronics (the benefit of shopping there, is that you can see the boxes before you buy them). (box is 1591ESBK. 1427CG7 – gland – Just big enough for one cat6 cable – Fits snug). I like large boxes so I don't have to worry about space. There's also expansion room. Only one cable gland going into the box (ethernet with passive PoE adapters). I also made a little plywood platform from some thin plywood inside the box. Sanding the edges with 80 grit sandpaper (see WoodWorking for Mere Mortals if you have not used sand paper much before).