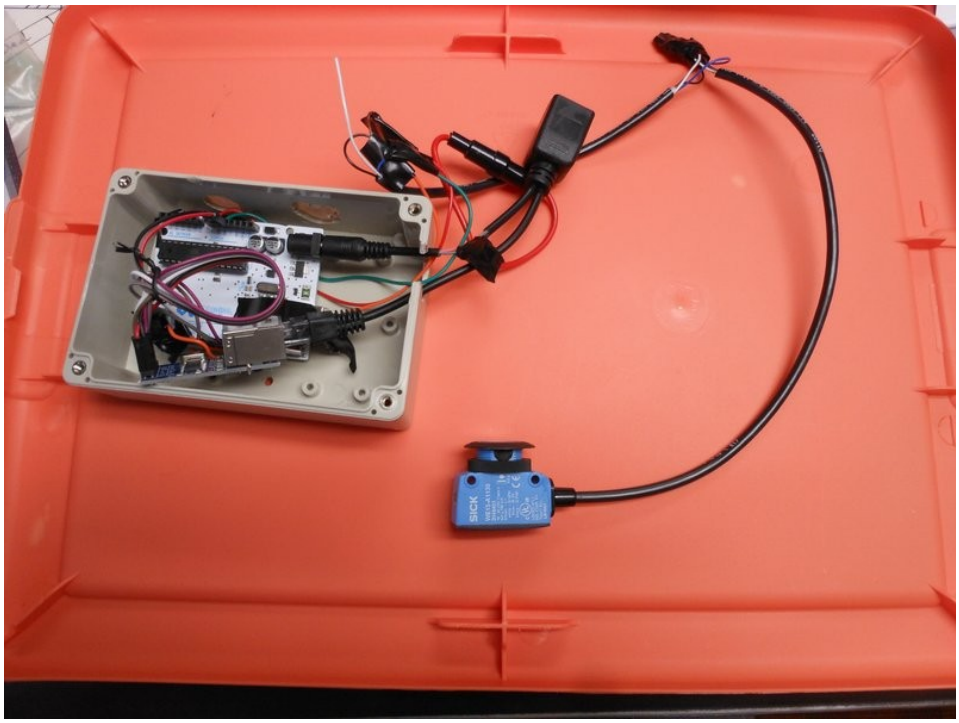
ZMHW Project

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.

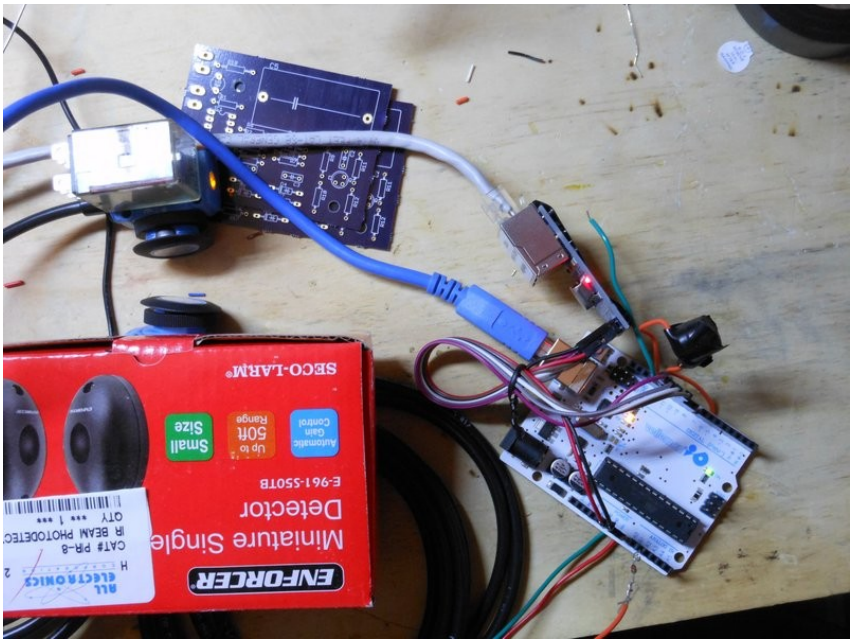


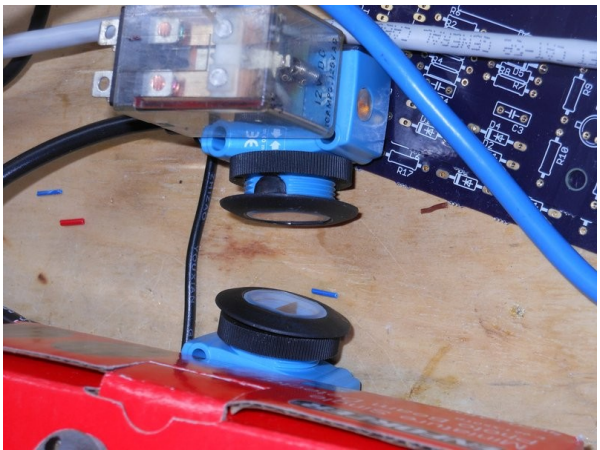
Figure 1: Orange LED on top of sensor when link detected. Series diode on output of sensor to cheat the need for transistor.

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

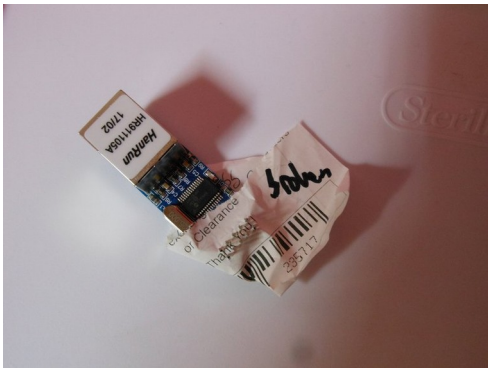
Some laser sensors output a high or low. Some, like the Sick sensor, output a high or low (depending on whether you connect to white or black wire), however they are meant to be connected to a transistor, and thus if you connect it directly to a micro expecting it to go high or low, it will not. I dont 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.

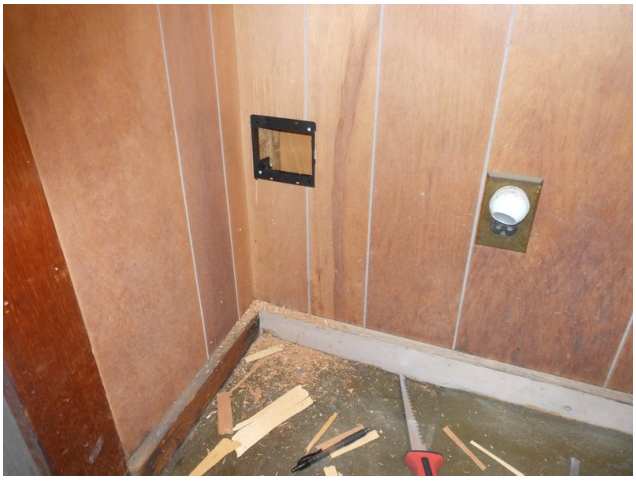


It turned out, the ENC28J60 module failed on me. Make sure to buy backups.

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.

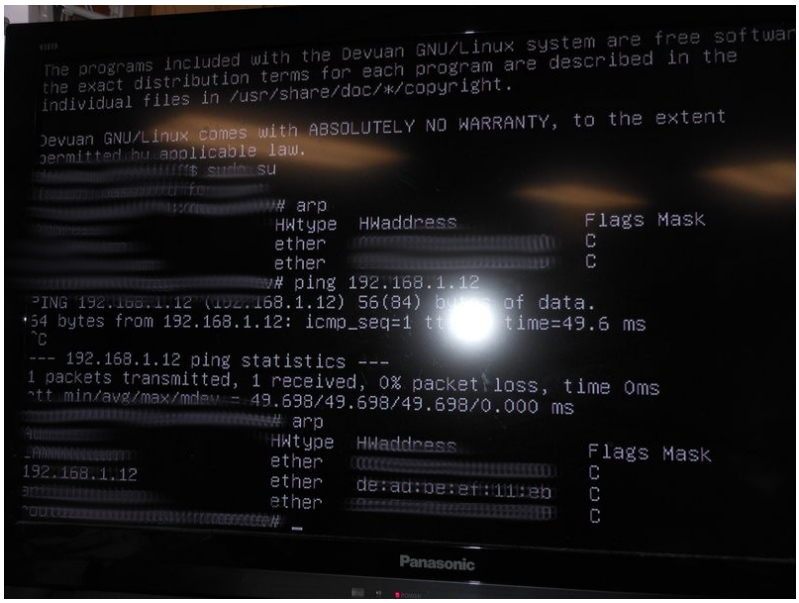
Feeding nylon string, and a wire up from the bottom of the wall to the ceiling. This is where the single sensor will be.



Low profile two gang wall plate installed.

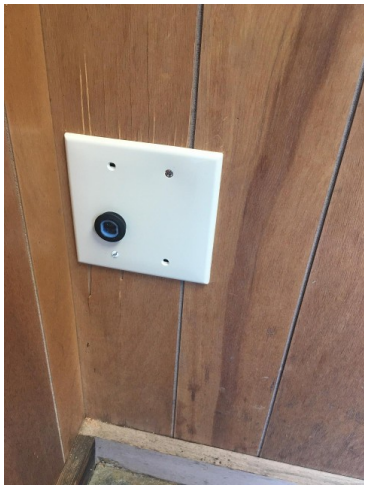
Installing the single sensor. I used ethernet as the power line. I also used a passive PoE adapter on both sides to transfer the power onto the ethernet, and back out to a 5.5, 2.1mm barrel plug

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).

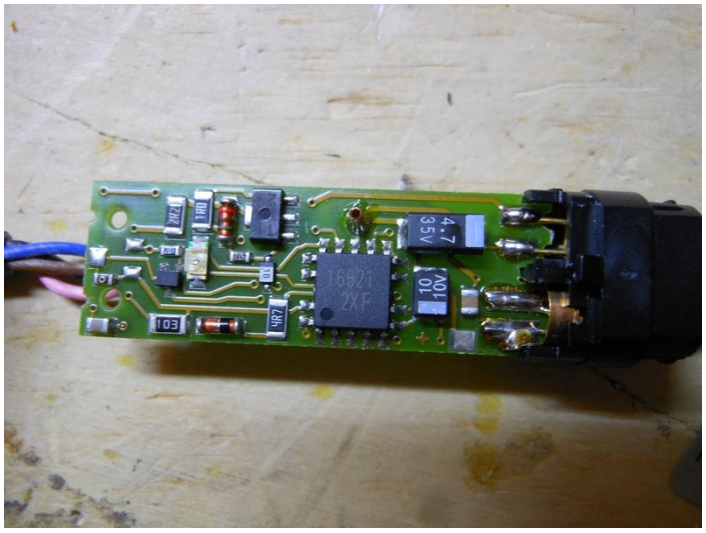


Checking the device is on the Camera LAN by pinging it, then reviewing the arp tables to make sure it's the right device (IP is static, so it's possible a conflict could arise)

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 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

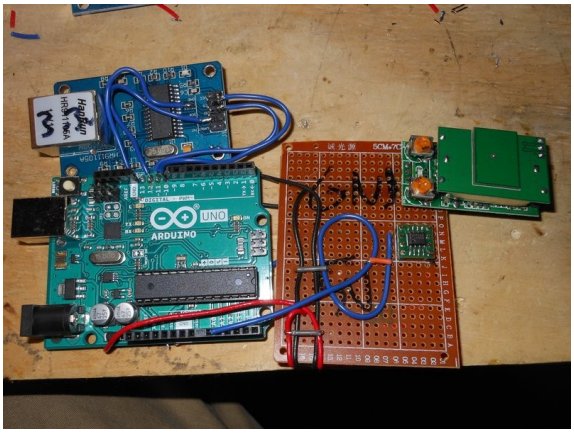
After finding some documentation on these in the reviews, I figured out why it wasn't working. These opto electronics absolutely require some type of reflector opposite them. Now, not all photoelectric sensors are like this, but this Omrom model is. I've also purchased low cost ebay photoelectric sensors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

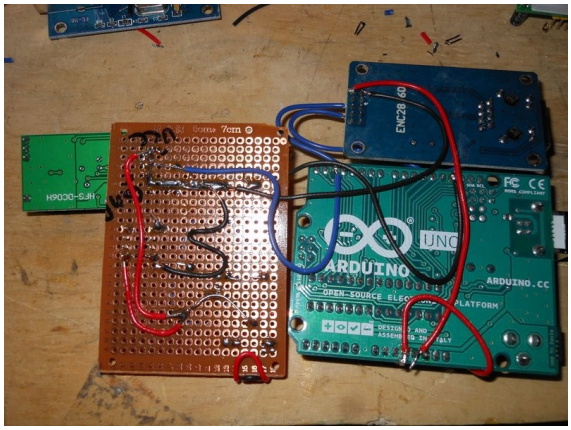
> *Good quality product. It has reverse polarity protection, etc. This has the emitter and receiver built-in. Best to use a retroreflector like a bicycle reflector or retroreflector tape. The indicator light will come on to show the state. The logic output is open collector and you'll get whatever the 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.

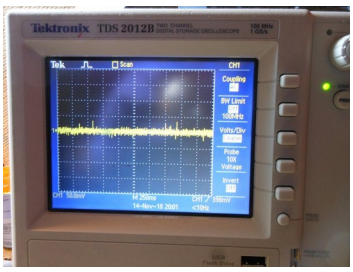
One thing I also noticed, was that used photo electric sensors from brand names can be obtained for discounts on the auction site. I saw a sick motion sensor for $10. It may be wise in the future to look at auction sites, to see if a good deal can be had. When buying them new, they can be relatively expensive for a hobbyist working out of his/her garage.

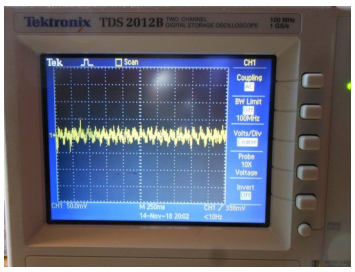
Revision 2
Using the HFS-DC06H Microwave Sensor
As I've tried with the HB100, without success, likely due to no included op amp (I tried throwing a cheapo one at it, but results were inconsistent), I've moved to the HFS-DC06H microwave sensor which includes the Op Amp and accompanying circuitry (simply outputs a digital high or low). 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.

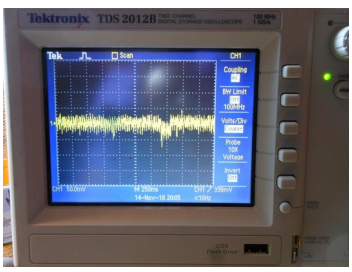


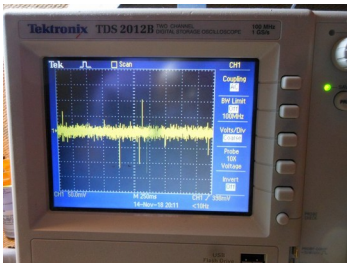


During testing, I found that the HFS sensor would not work correctly with my laptops usb 2.0 power supply. I thought it might be RF interferance from the metallic perf board – something I've seen before with the FM bug radio – but it was not. The solution was to use external power. See the four pictures (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 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
```

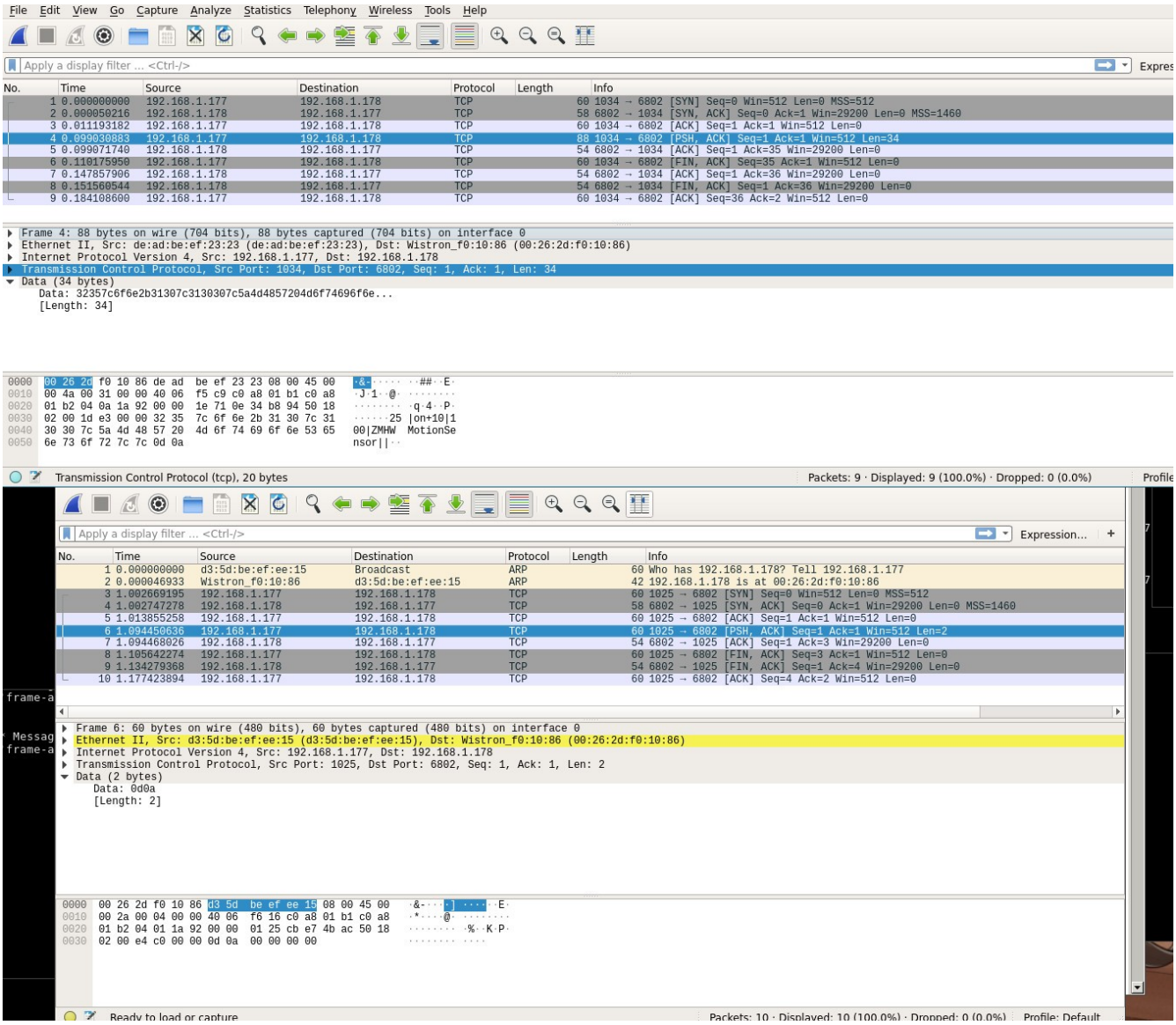
“

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()), fotr example, converting variables into it will likely not work without further finesse.

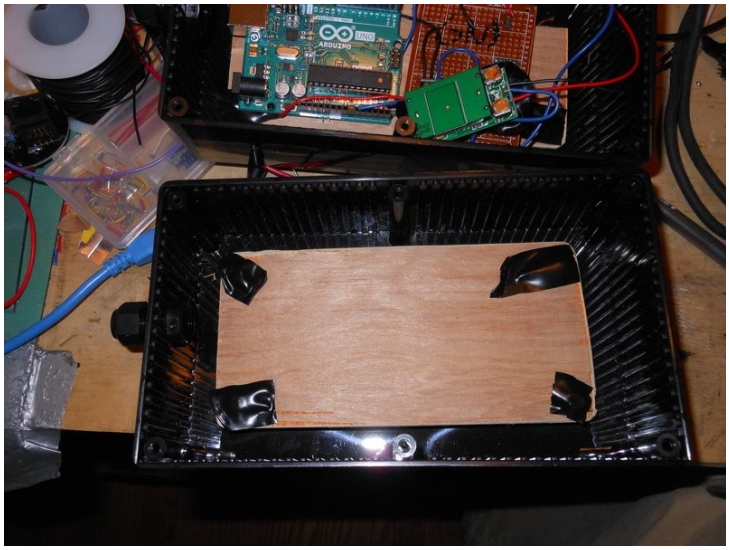
Low RAM errors can creep into strange places. For example, see these two wiresharks, where my code was running, equally as well, but the new code revision simply didn't work:

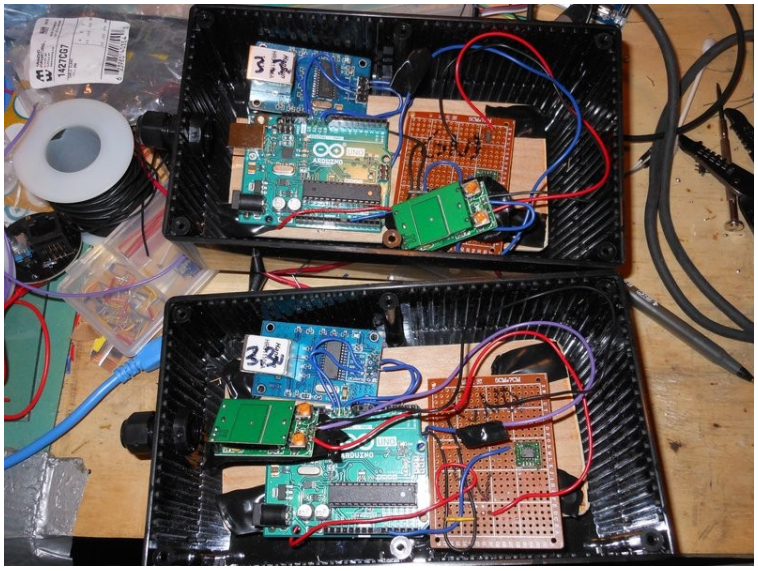
As you can see the data packet is mangled in the new rev. I've seen this enough to know, it was low SRAM. Otherwise, the code worked without major error. This small error effectively broke the program! Testing is always important.



Here are some pictures of the build. As I mentioned 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) before finishing with 220. I neglected to use urethane here, but in future stands, I will finish the wood with a protective layer.





The final build, had also a passive PoE adapter inside the box.

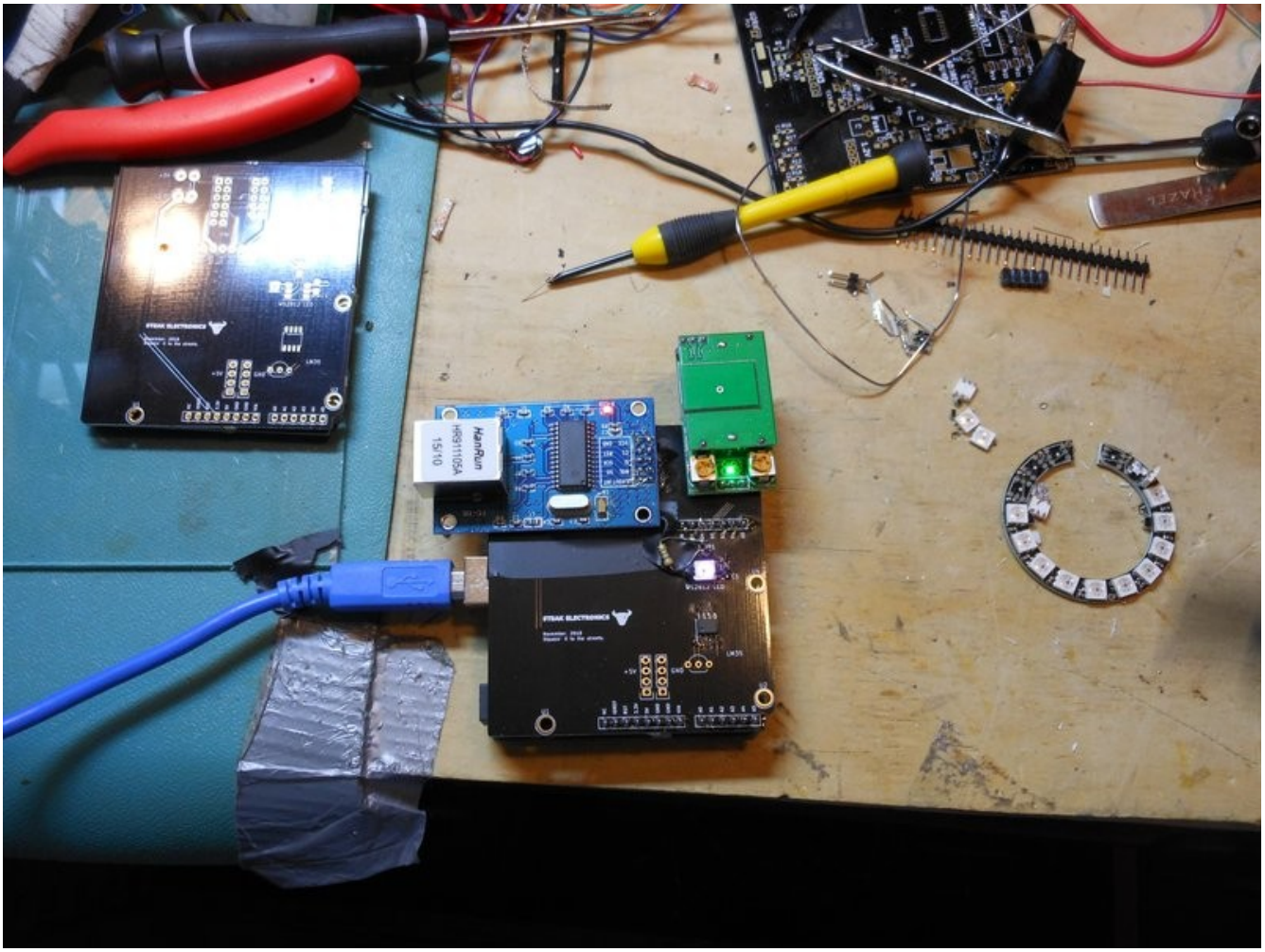
Deployment

Today I deployed both sensor boxes on site, connecting them to the ZM system. They both work. I found that these HFS sensors appear to be fairly directional. I've been unable to get them to fire, when behind a wall, or up a floor, which is good – I don't want them to do that. The metal shield they have, seems to work well – blocking radio waves from going behind it. I mounted one on the ceiling and one on a wall, setting the cameras to nodect. Success.

I'm not sure exactly what frequency they are, but I think it is around 5GHz, comparable to 5GHz wifi (not going through walls well). I'm also considering putting some copper tape on the inside of the box, to help block anything through the wall it's mounted to, possibly...

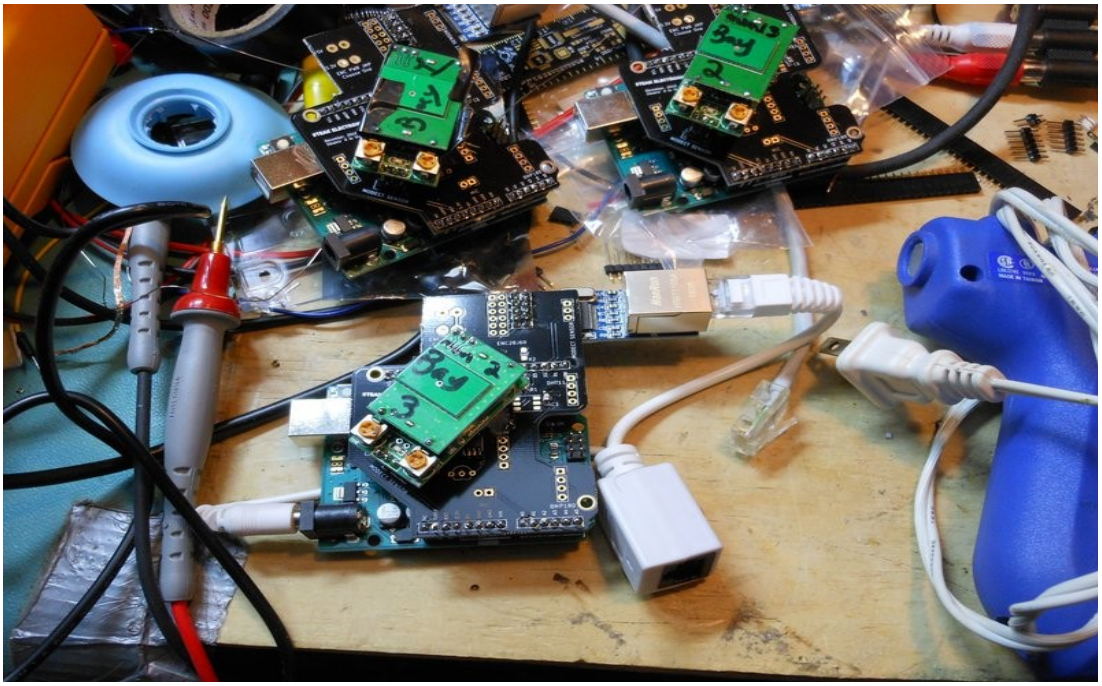
Deployment 2
I've deployed another sensor to a location, this one where Mice are occasionally in the ceiling. Believe it or not, the mice will set off the sensor. I've yet to move the sensor to a less sensitive location. I've also made a shield, which will be in the final ZMHW repo, when I get something polished.

When building this new shield I've made for the motion sensor, you can use flush cut pliers (the ones that cut really close to a board, if you are not familiar) and cut the WS2812 RGB LEDs right off of a $3 RGB ring. You can also buy individual WS2812 LEDs. Cutting them off, is easier, and cheaper (and for those who run on a shoe-string budget) thus what I did here. There is a video I've made to show this process.



Shield Rev1, and RGB LEDs cut off of ring.

TODO
There's one thing I'd like to add. First off, a light pipe, from the RGB LED to the box. Perhaps, 3D printed. Second, the RGB LED should change colours, very slightly when motion is detected. Not so much colour change that a casual glance will notice, but enough that if you are watching for it, you will see it.



PCB Revision 2: A few notes on this. The Ethernet pinout was reversed by accident, so it was necessary to bend the passive POE adapter to the other side. The boards are labeled with the location of each sensor. Each one is programmed specifically for each monitor, so it's important to match them.

Work Log 1/31/19
I'm assembling three of these for a company today. I can see right away some of the mistakes I made, and it's all DFM or design for manufacture errors.

I made a whole Arduino UNO shield, and that was a bad idea. I don't want to solder all of the 0.1” headers. There's a 10, two 8s, and a 6. It takes say 3-5 minutes per board. Too slow. I need to make the next shield with as few 0.1” headers as possible. Also, possibly a different connecter besides these pin headers would be better. Something easier to solder, perhaps reflow possible.

Another option would be to use a nano. Make assembly easy. That is very important. I also might want to consider a micro with built in ethernet, but for simplicity sake, for now Arduino will do.

NOTE: I made the PCB too small for the 2x5 Pin ENC mini module. Need to shorten the end....

Motion sensor on bottom needs to be flipped. Pinout is wrong. I got it correct on the ZMHW map, but

for some reason this board was done earlier.

RGB LED capacitor should be moved one side over, so it's not near pins when soldering.

Mice in the Ceiling
So I've deployed a few of these devices in two different sites. One site, the sensors work flawlessly. In the other site, the sensors give off false readings, due to mice in the walls (technically, not false readings). This makes mice in the walls a serious concern for these motion sensors. If you are in a building that has such a problem, you should use either PIR, or laser, or both.

The need for POE
I deployed three of these in one company, and power wiring was an issue for one of them. One of them is going to require it's own 12V wiring (the other two worked with passive POE). Based on this, and knowing that there is a limit to what passive 12V POE can do, I've decided that I will use an 802.11 48v POE board for this application going forward. I don't want to take any chances with power being an issue. POE Injectors are relatively cheap.