ZMHW Modector

Steak Electronics

Contents

1 Overview

Making and deploying a Motion Sensor for Zoneminder CCTV software installations. These sensors use ZMTrigger.pl (wiki.zoneminder.com/ZMTrigger) to activate an alarm on a camera for a period of time. The advantage of hardware motion sensors over the software detection of Zoneminder, is that the hardware motion sensors avoid some of the problems inherent in software detection, such as false positives from day-to-night, bugs, missed detections, and others.

I've tried different motion sensors. Let's start with the Infrared Laser Diode.

2 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

Later on we will try a different sensor. The HFS-DC06H. This sensor is a combination of an HB100 radio, with a decoding board that will read the signal and output a logic high or low. You may also want to try PIR sensors.

2.1 Other Sensors

- HFS-DC06H
- PIR Sensor
- Any other Laser Diode Sensor you like
- Reflective tape

3 Work Log

3.1 Sick Motion Sensor

The first tests were with the Sick diode sensor and receiver. This device is good for a doorway, where the door must be opened in order for people to pass. Putting it in the way of the door ensures that it will activate. It has a distance of at a max 15 feet or 3 meters. It is a laser type tripwire, which means it can be avoided, if someone knows where it is.

Device was assembled and using the ZMHW Modector 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.

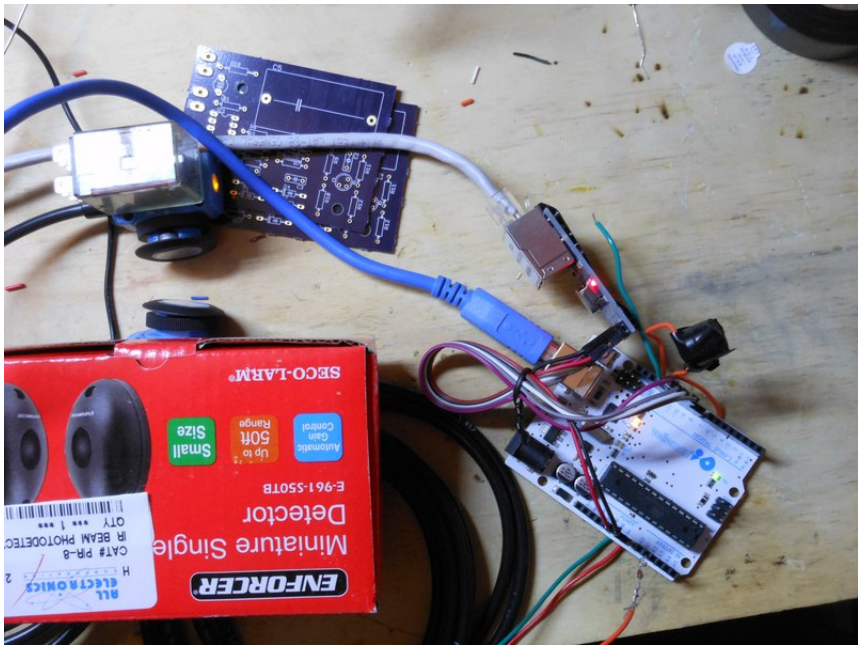

Figure 1: Testing the Sick IR Diode Tripwire

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 Diodes (or LEDs) are lined up correctly. When someone moves across the field of their vision, the orange LED will change

to green.[1]

3.2 Diode on Output of Sick Sensor instead of Transistor (Hack)

Some IR diode / receiver pairs 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. Being lazy, and seeking a quick solution, I put a 1N4819 in series with the output of the Sick sensor. TODO: pictures showing waveforms [2]
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 important to line up the emitter and receiver. If they are not lined up precisely, they will not get a sync, and the motion detection will fail. Thankfully, the diode outputs more of a cone, and less of a straight line, so some buffer is there. When the lights are dark, it is possible to see the red IR emitted if the distance is not too much.

3.3 Broken ENC28J60 Module

During my testing, I suddenly was unable to get an IP address. I checked the testsuite sketches, which didn't work, 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.

3.4 Installation Log in Pictures

Here is an overview of what installing this in the wall might look like. One side is emitter, the other the receiver.

[1] This will later become important when installing the IR diode and receiver, as they must be lined up correctly.

[2] This is possibly an issue of output impedance, but I will admit, at the time, I didn't bother to check.

Figure 2

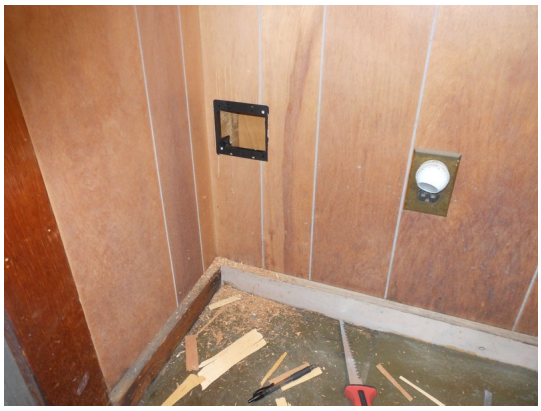

Figure 3

Figure 4

Figure 5: Part 1 of sensor. By mounting it on the right side of a project box, we can get a 90 degree angle.

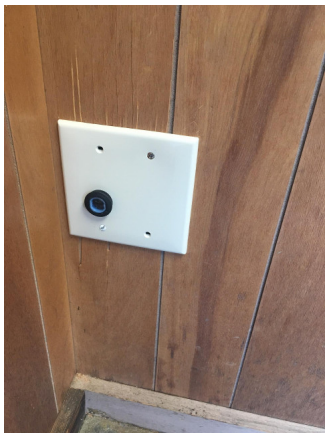

Figure 6: Arduino and Part 2 of sensor. Lined up with the other part.

4 Omrom Photoelectric IR Emitter/Receiver

All electronics is 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 max distance, etc... See resources in this git repository for some PDFs.
I tested one without knowing how to use them, but had poor results. I was only able to get the light to flash when I dismantled the device, and put my hand very close to the IR. Teardown pictures are in the photos folder. 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. [3]

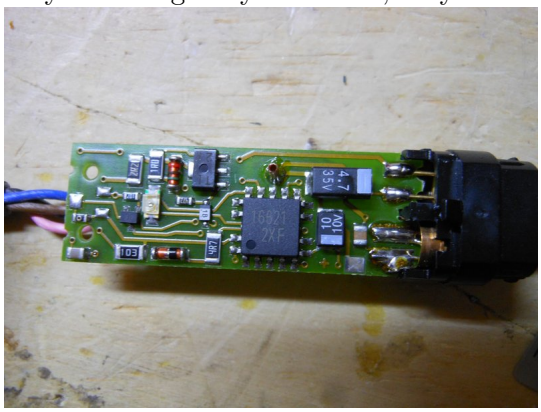

Figure 7: Omrom 'Photoelectric' IR Emitter and Receiver Pair. Notice the two diodes behind the black cover on the right side.

After finding some documentation on these in the reviews, I found out that the IR emitters require a reflective sticker in order to 'see' their IR beam reflect back. Not just a white surface, but the type of reflector you might see on a construction or night worker orange vest.

Hookup Instructions from All Electronics Comments:

```
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.
```

One thing I also noticed, was that used photo electric sensors from brand names can be obtained for discounts on the 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.

[3] Repairs may be difficult.

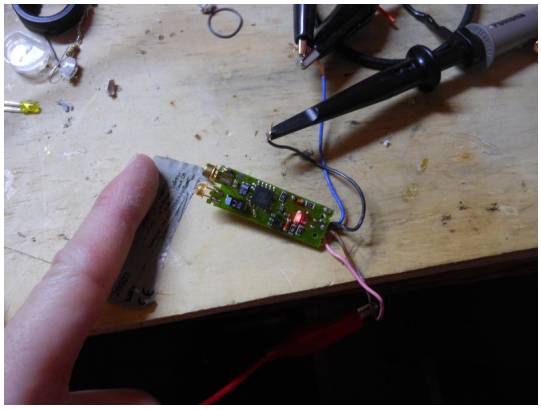

Figure 8: IR in action. (without a reflective sticker, it doesn't go very far!)

5 Using the HFS-DC06H Microwave Sensor

As I've tried with the HB100 (just the RF part of this HFS-DC06H), without success, I've moved to the HFS-DC06H microwave sensor which includes the Op Amp and accompanying circuitry (it simply outputs a digital high or low).
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 an FM bug radio but it was not. The HFS and the ENC require external power that is greater than my laptop can output.

5.1 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 of what the Arduino IDE calls dynamic memory. You can see how much dynamic memory is used in Arduino by hitting verify (not upload but verify)and reading the output from the toolchain. 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:

```
Heres a small utility function which determines how much RAM iscurrentlyunused:
int freeRam () {
  extern int __heap_start, *__brkval;
```

```
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}
And heres 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
```

The UIPEthernet code requires significant RAM for the Uno. 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: