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.
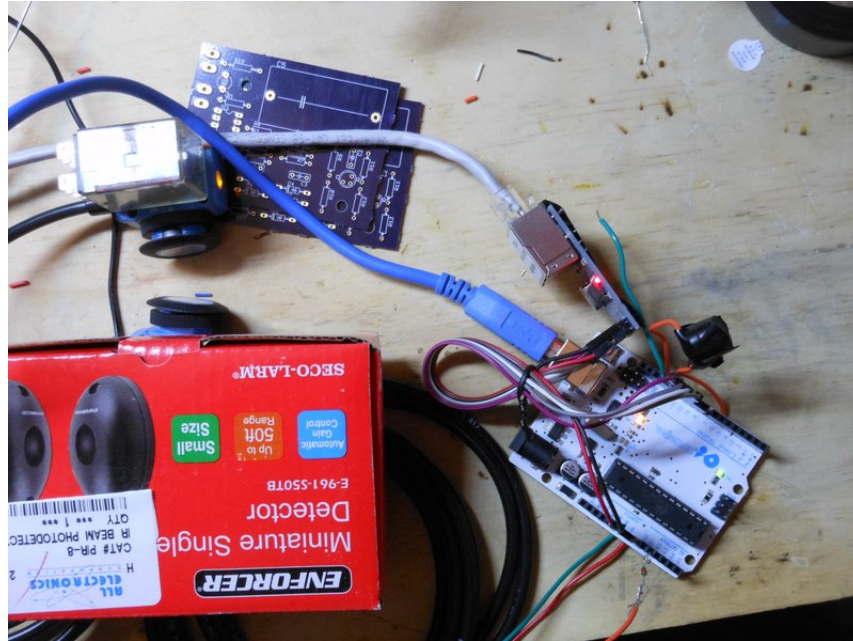
Figure 1: Testing the Sick IR Diode Tripwire

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

---

[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.

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.

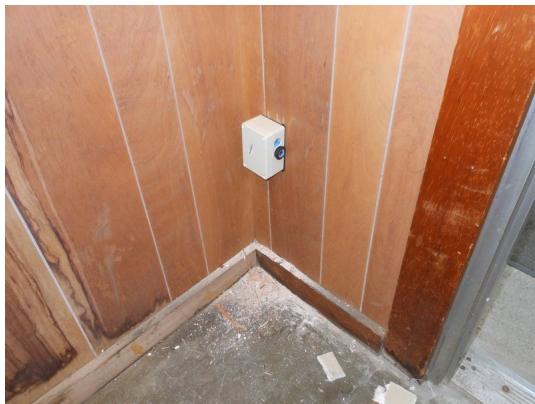Figure 2

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]

---
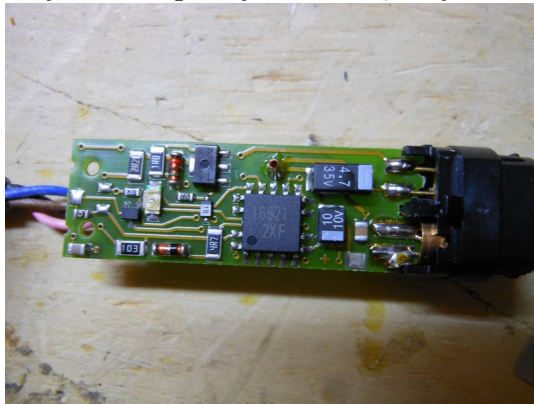
[3] Repairs may be difficult.

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.
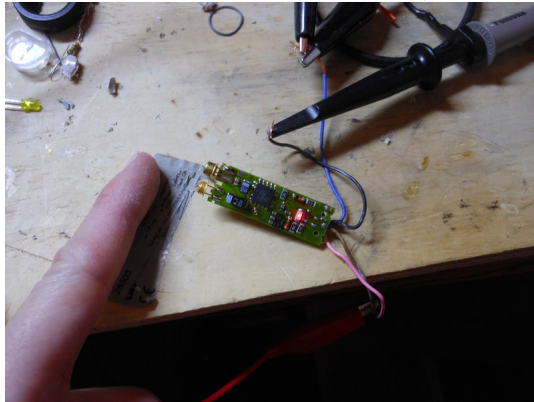

Figure 8: IR in action. (without a reflective sticker, it doesn't go very far!)

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, and libraries that print through serial likely won't work.

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, as well as saving multiple copies as you go, or alternatively version control.

Figure 9: First capture is correct. Second one, the lack of RAM causes the
ENC to fail to send data to ZMTrigger.

6 Deployment

Today I deployed both sensor boxes on site, connecting them to the ZM system. They both work[4]. 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.
However I have seen some issues with these, in the following situations.

- Mice in the ceiling, setting off the alarm

- Flourescent lights being within 2 feet of the installation causing false alarms

Keep in mind, that if anything moves, it can set off the alarms. Including small animals.
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...
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 in the repo to show this process.
There's some things I'd like to add to the prototypes. 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.

6.1 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 DFA, or design for assembly 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

---

[4]And have continued to work for about a year now

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 Uno will do. Unos also are as future proof as you can get.

6.2 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. Back to the drawing board.

6.3 HB100 are Sensitive to Fluorescent Lighting and Mains

Do not install these HB100-based sensors near either mains or tube lights. They are sensitive, and will not operate correctly. I found that being within roughly 2 feet was too close. In my case, the logic high of the HB100 would go off randomly.

6.4 Switcher Shield

As much as I have a need for POE, I also have a need for speed, so I've decided to add a new shield that manages power, as a quicker solution to the poe problem. I put two different switchers on board. Both jelly bean parts per eevblog forums [5]. I'll try each.

6.5 ENC28J60 Power Consumption

The Uno has an LP2985, which is a 3.3v reg, with 150mA max output (per data sheet). The ENC28J60 is spec'd at typical 160mA active ethernet, 120 mA inactive ethernet (Max active can be 180mA).

---

[5]https://www.eevblog.com/forum/beginners/jelly-bean-switching-regulator-alternative-to-7805/msg369867/#msg369867

What to do? I've never had an actual problem with my Uno + ENC modules yet, but this is of concern. I could include onboard 3.3v vreg on the switcher board, or after peeking through the ENC data sheet, I found a sleep mode. I checked online, and someone has covered it here:
http://blog.derouineau.fr/2011/07/putting-enc28j60-ethernet-controler-in-sleep-mode/
So this might be worth adding to the code. I don't need the internet active until an alarm is detected. Otherwise the ethernet can be off. Will save some power. And, this is a reasonable compromise for the low power requirements, assuming that the sleep and wake from sleep mode works.
Let's see if the sleep modes are already included in the library I'm using for the ENC... It does. Not called sleep though, it's powerOff, and powerOn. See:

```
Enc28J60Network::powerOff(void)
{
  #if ACTLOGLEVEL>=LOG_DEBUG_V3
    LogObject.uart_send_strln(F("Enc28J60Network::powerOff(void) DEBUG_V3:Function st
  #endif
  writeOp(ENC28J60_BIT_FIELD_CLR, ECON1, ECON1_RXEN);
  delay(50);
  writeOp(ENC28J60_BIT_FIELD_SET, ECON2, ECON2_VRPS);
  delay(50);
  writeOp(ENC28J60_BIT_FIELD_SET, ECON2, ECON2_PWRSV);
}

void
Enc28J60Network::powerOn(void)
{
  #if ACTLOGLEVEL>=LOG_DEBUG_V3
    LogObject.uart_send_strln(F("Enc28J60Network::powerOn(void) DEBUG_V3:Function sta
  #endif
  writeOp(ENC28J60_BIT_FIELD_CLR, ECON2, ECON2_PWRSV);
  delay(50);
  writeOp(ENC28J60_BIT_FIELD_SET, ECON1, ECON1_RXEN);
  delay(50);
}
```

Note that the PowerOff function does not follow the data sheet precisely.

The data sheet mentions that after RXEN is cleared, to wait for RXBUSY and TXRTS to be clear. It looks like a 50millisecond delay is considered enough here. Don't like the look of that though.