

# Pump-eye Project Entry FZ1606

## Background

Many buildings, including my house, have a basement. Buildings like this are constructed with a system of weeping tiles or drainage pipes to route water from around the foundation to a place where it can drain. In many areas you are not allowed to let this water drain into the domestic sewer system. In these cases a sump pit is used to collect the water and a sump pump periodically pumps the water back outside away from the building.

The problem with sump pumps is they don't have a long life span and don't tend to give much warning that they are about to fail. There are other points of failure as well. For instance, most are installed with a check valve to prevent the water from draining back into the sump pit. These valves can fail, blocking the pump outlet. If you live in a cold climate the outlet can also be blocked by frozen water in the drain pipe outside the building. Then there are electrical failures such as a bad float switch, tripped circuit breaker, open thermal cutout or power failure. These factors were the driving force for developing the Pump-Eye system. The purpose of the system is to monitor the sump pump to ensure it is continuing to protect your property against flooding.

## System Description

The Pump-Eye monitoring system is a flexible system comprised of three different types of electronic units. There is a Sensor Unit, Base Unit and Ethernet Unit. A simple system could consist of just one unit, the Sensor Unit. This system would monitor the water level in the sump pit and display this on a 10 segment LED bar graph. It will sound an alarm if the water exceeds the normal maximum height by 10%. There are also LED's that come on when the AC power is off or the 9-Volt Sensor Unit backup battery needs to be replaced.

The next step up would be to add the Base Unit. This Unit has the same indicators as the Sensor Unit. Since the connection between these units is wireless it can be remotely located to allow people to know the state of the system without having to go into the basement to view the front panel of the Sensor Unit. It also sounds the same alarm signal as the Sensor Unit. To help distinguish the alarm from other devices I wanted to make the sound unique so I would know it is not the smoke detector, microwave oven, bread maker, washer, dryer, etc. The sound I chose is old but very recognizable (at least to some of us), S-O-S in Morse code.



Front view of the Base and Sensor Units

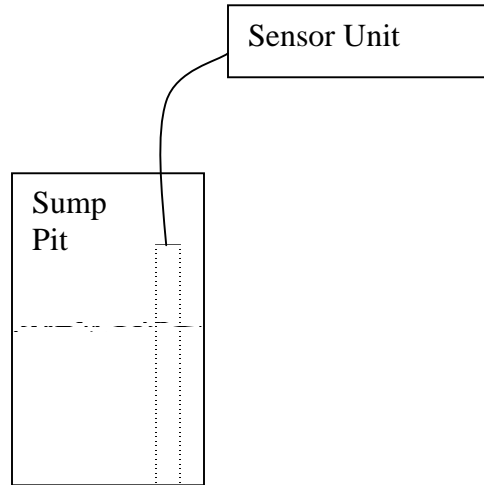


Rear view of the Base and Sensor Units

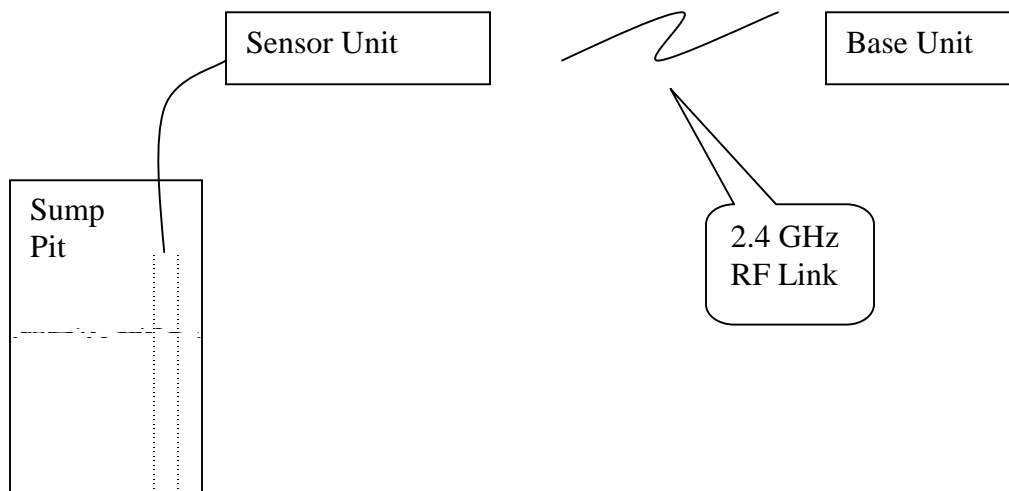
The third component of the system that can be added is the Ethernet Unit. The system is flexible in that the Ethernet Unit can connect to either the Sensor Unit or the Base Unit via an RS232 connection. This lets you place the Ethernet Unit where it is most convenient to connect to an Uninterruptible Power Supply (UPS) and network.

The Ethernet Unit receives commands from the unit it is attached to. Acting on these commands, it sends syslog messages to a syslog server so that pump cycles can be time stamped, counted and a record kept of pump run time. The syslog server also captures other event messages which are detailed later. The Ethernet Unit sends an email to alert the owner about conditions that require immediate attention, such as higher than normal water levels in the sump pit or a loss of AC power for the pump.

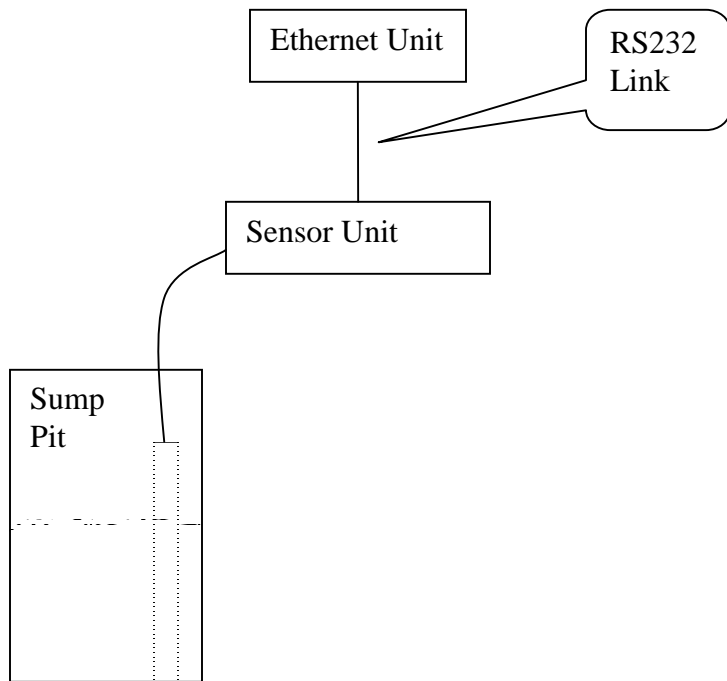
The following diagrams depict the possible configurations the Pump-Eye system can be used in:



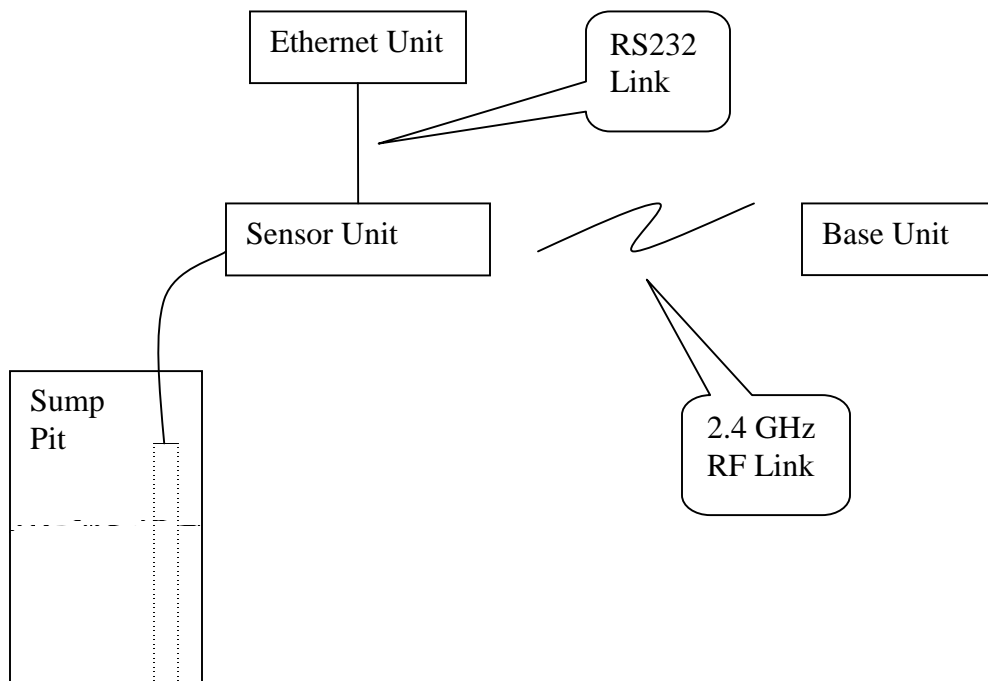
Sensor Unit Only



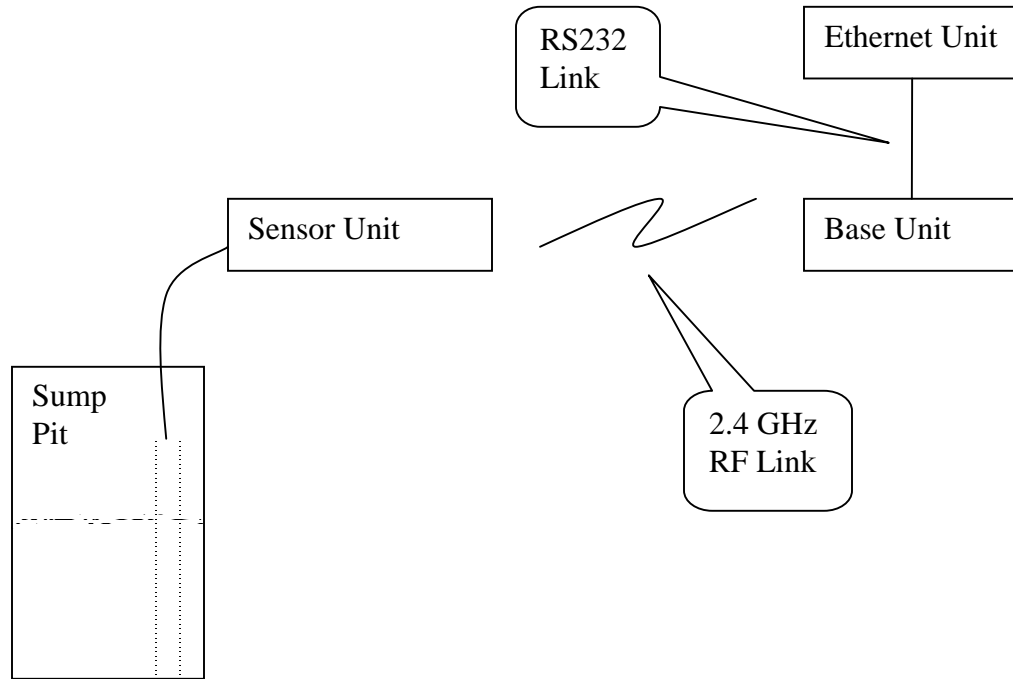
Sensor Unit and Base Unit



Sensor Unit plus Ethernet Unit



## Sensor Unit plus Ethernet Unit and Base Unit



## Sensor Unit and Base Unit plus Ethernet Unit

### **Sensor Unit**

The prototype of the Sensor Unit uses a modified Freescale SARD board along with two wire-wrapped boards and a MPXM2010GS sensor from the break-away sensor board supplied by Freescale. This sensor board also needed a modification since the trace for pin 1 on the board is missing. A 26 conductor ribbon cable connects the SARD board to the front panel board. A 10 conductor ribbon cable connects the front panel board to the rear panel board where the signal conditioning circuitry resides.

A Hammond 1598CSGY plastic enclosure with aluminum end panels was used for the prototype. This enclosure was chosen because of the internal slots for holding the circuit boards. The spacing is just right for the 10 segment LED package, a 20 pin solder tail DIP socket and a 20 wire-wrap socket. This places the display right against some red 3M tape applied to the inside of the front panel.

The slot for the 10 segment LED package was cut with a milling machine with a 1/8 inch bit. All other holes were made with a hand punch after the labels were applied. The labels were made with a program called Micrografx Designer and printed on Avery permanent white I.D. labels (part number 06573). The top of the label was overlaid with a glossy transparent film. The following are pictures of the Sensor unit.

### **Base Unit**

The prototype of the Base Unit also uses a modified Freescale SARD board along with one wire-wrapped board. A 26 conductor ribbon cable connects the SARD board to the front panel board. The only connection from the front panel to the rear panel is two wires for the piezo buzzer.

A Hammond 1598BSGY enclosure was used for the prototype. This enclosure is a little smaller than the 1598CSGY that was used for the Sensor Unit. It was chosen for the same reasons mentioned earlier and for the fact that extra room was not required for a battery and sensor signal conditioning circuitry.

The same techniques mentioned earlier were used for constructing the front and rear panels. The front panel board of the Base Unit is almost the same as the front panel board of the Sensor Unit so I did not include pictures of it.

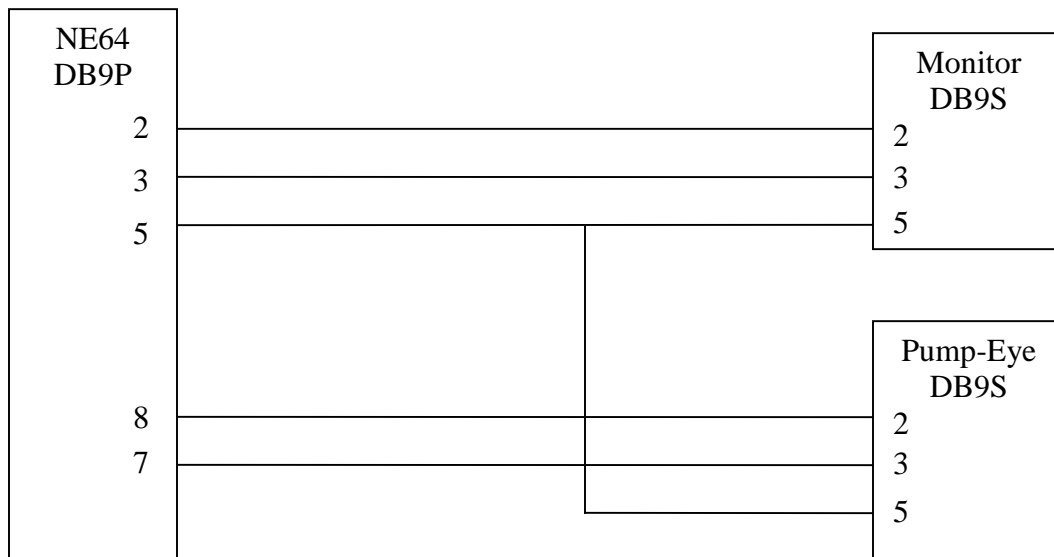
The following pictures show the modifications made to the SARD boards, which are identical for both the Sensor and Base Unit. The modifications involve adding wires to the board as outlined in the schematic. Additional GPIO pins, +3V, +5V and ground were added to J3.

## Ethernet Unit

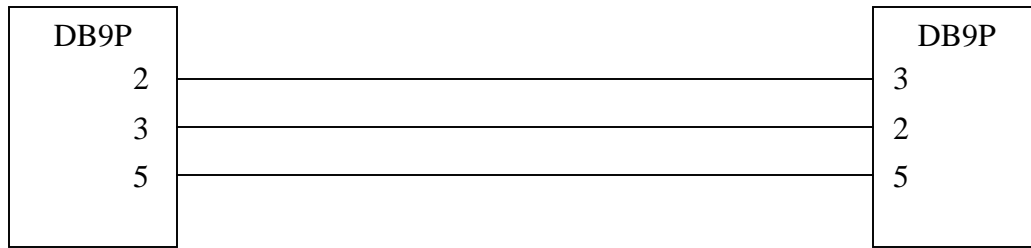
The Ethernet Unit required the least amount of work to construct. I took a standard Freescale DEMO9S12NE64 unit and modified it to add a second serial port on the existing DB9 connector. This involved adding 4 jumper wires to the circuit board and making a special male to double female Y cable. I chose the pins such that no damage will occur if you use a 1:1 male to female serial cable to connect the unit directly to a PC COM port instead of using the Y cable.

The Y cable connects two RS232 ports to the single DB9 connector that was already on the unit. I used pins 7 & 8 for the 2<sup>nd</sup> set of RXD and TXD lines that connect to a Pump-Eye Sensor or Base Unit. Pins 2 & 3 are still used for downloading software and communicating with the monitor when debugging code.

The schematics for the Ethernet Unit's Y cable and Crossover cable are outlined below:



Ethernet Unit Y cable schematic



Ethernet Unit to Pump-Eye Unit Crossover Cable

The Sensor Unit ADC readings for the water level and battery voltage are both filtered using a very fast integer math filter shown in this code snippet:

```

/* Set filter constants such that new values are weighted 5%, running average 95% */
const filter_c_data_t f_cdata = {bit(13) - (0.05 * bit(13)),(0.05 * bit(13))};
filter_data_t water_level_fdata;
filter_data_t battery_level_fdata;

uint16_t filter(filter_data_t *fd, uint16_t new_val)
{
    uint16_t retval;
    uint32_t numerator;

    // New values get a minor weighting and the existing values
    // get a major weighting. This minimizes the effect of noise spikes.
    numerator = (uint32_t)fd->fcd->c_major * fd->last_result +
                (uint32_t)fd->fcd->c_minor * new_val + fd->remainder;

    retval      = (uint16_t)(numerator >> 13) ; // numerator / 2^13
    fd->remainder = (uint16_t)(numerator & 0x1FFF) ; // numerator % 2^13
    fd->last_result = retval;

    return retval ;
}

```

Another interesting piece of code is the piecewise linear lookup technique used to calculate the percentage of water level based on the calibration values recorded for the low and high water marks which are dynamically populated into lines 2 and 3 of the lookup table. It works on the rise over run principle as you can see in this code snippet:

```
LUT_data_t ADC_to_Percent_LUT[] =
{
  {MIN_LUT_XP, (0 << 8)},
  { 0x0000, (0 << 8)},
  { 0x03FF, (100 << 8)},
  { 0x03FF, (200 << 8)},
  {MAX_LUT_XP, (200 << 8)}
};

uint16_t piecewiseLinearLookup(const LUT_data_t table[] , uint16_t xval)
{
  uint8_t i;
  uint16_t retval;
  uint32_t delta;
  uint32_t rise;
  uint32_t run;

  i = 0;
  while (xval > table[i].xpoint) ++i;

  if (xval == table[i].xpoint)
  {
    retval = table[i].ypoint;
  }
  else
  {
    delta = (uint32_t)xval - table[i-1].xpoint;
    rise = (uint32_t)table[i].ypoint - table[i-1].ypoint;
    run = (uint32_t)table[i].xpoint - table[i-1].xpoint;

    retval = (uint16_t)(table[i-1].ypoint + (delta * (rise / run)));
  }
  return retval;
}
```