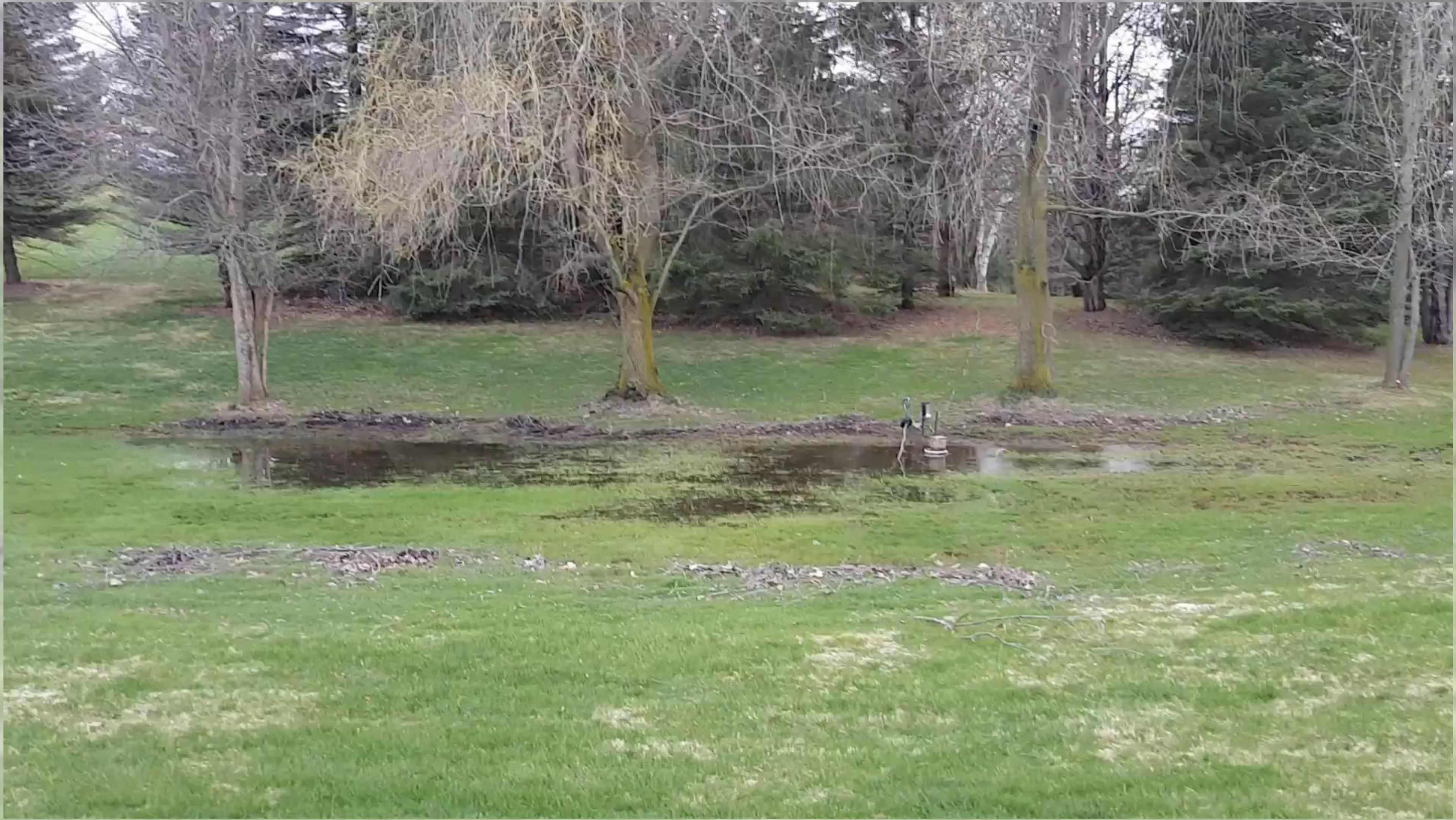
Cool Microcontroller Projects

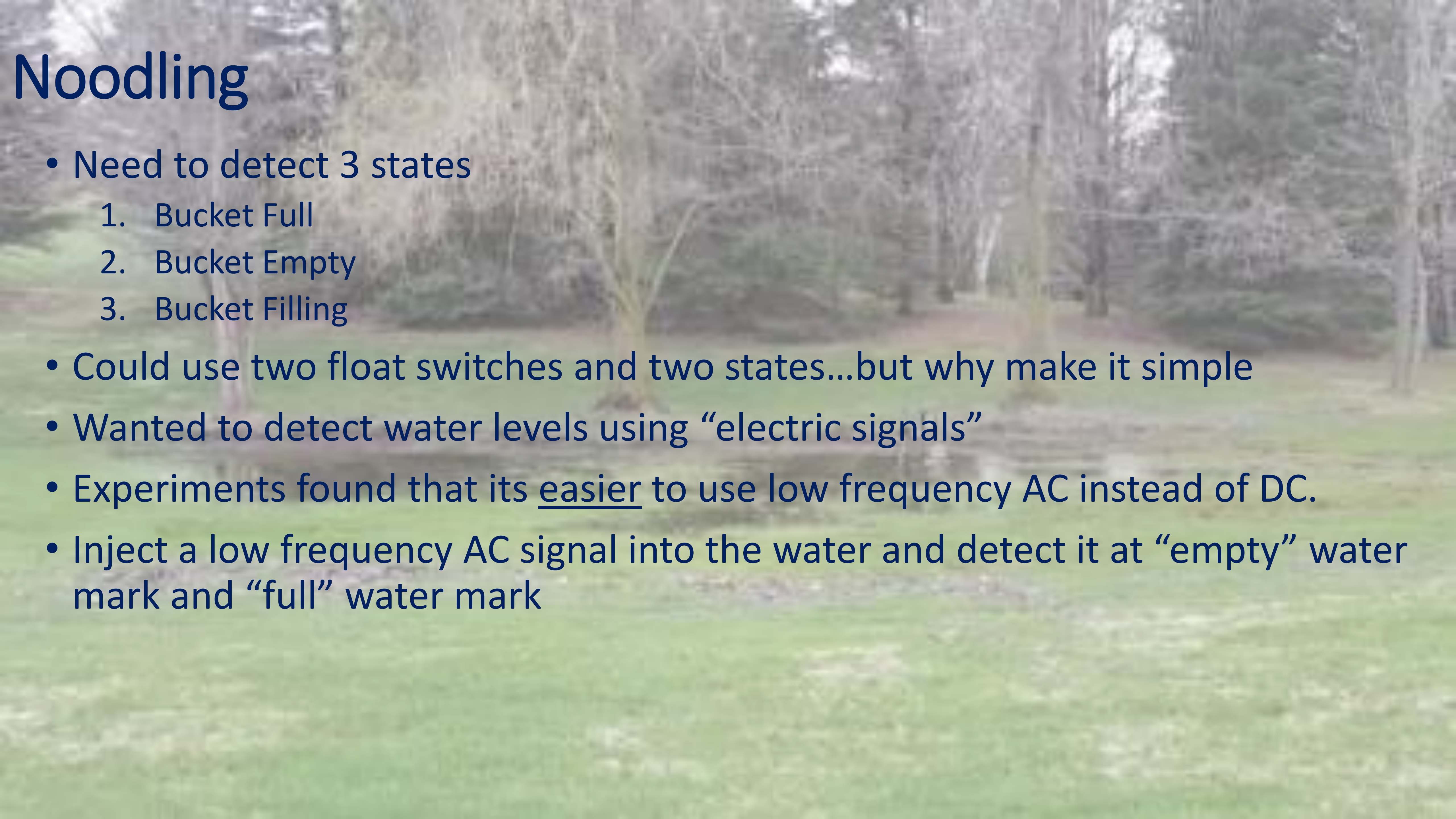
Water Pump Controller

Dave VE3OOI

Feb 2021

BACKGROUND



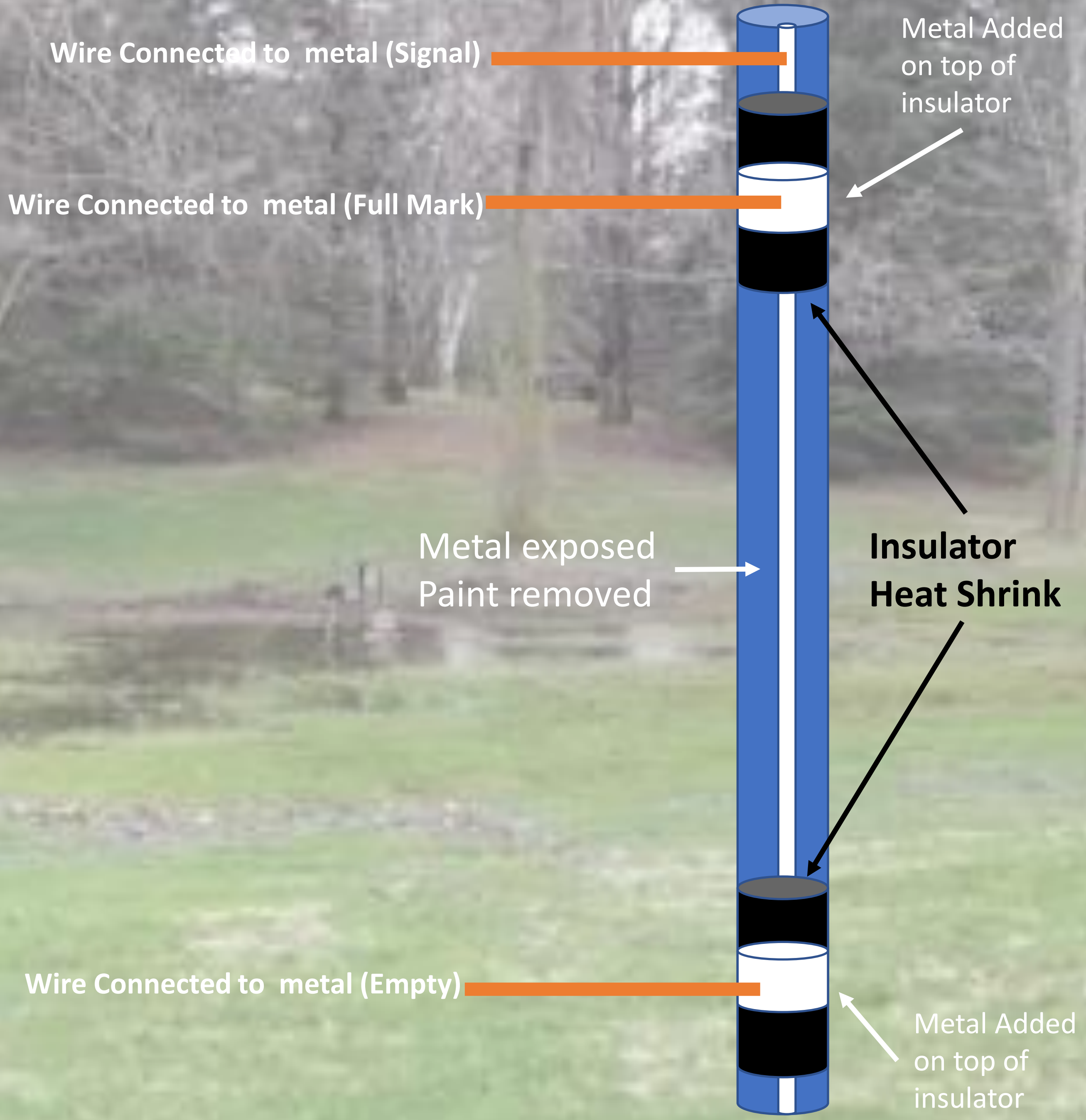


Noodling

- Need to detect 3 states
  1. Bucket Full
  2. Bucket Empty
  3. Bucket Filling
- Could use two float switches and two states…but why make it simple
- Wanted to detect water levels using “electric signals”
- Experiments found that its easier to use low frequency AC instead of DC.
- Inject a low frequency AC signal into the water and detect it at “empty” water mark and “full” water mark

LEVEL SENSOR

- Painted Metal Tube
- Paint stripped from small section of tube
- Insulator added to full water mark and empty water mark
- Bare metal added on top of insulator
- Wires soldered to bare metal and connected to controller
- After 1 season, metal strip coated with mud and guck and had to be cleaned



Wire Connected to metal (Signal)

Metal Added on top of insulator

Wire Connected to metal (Full Mark)

Metal exposed Paint removed

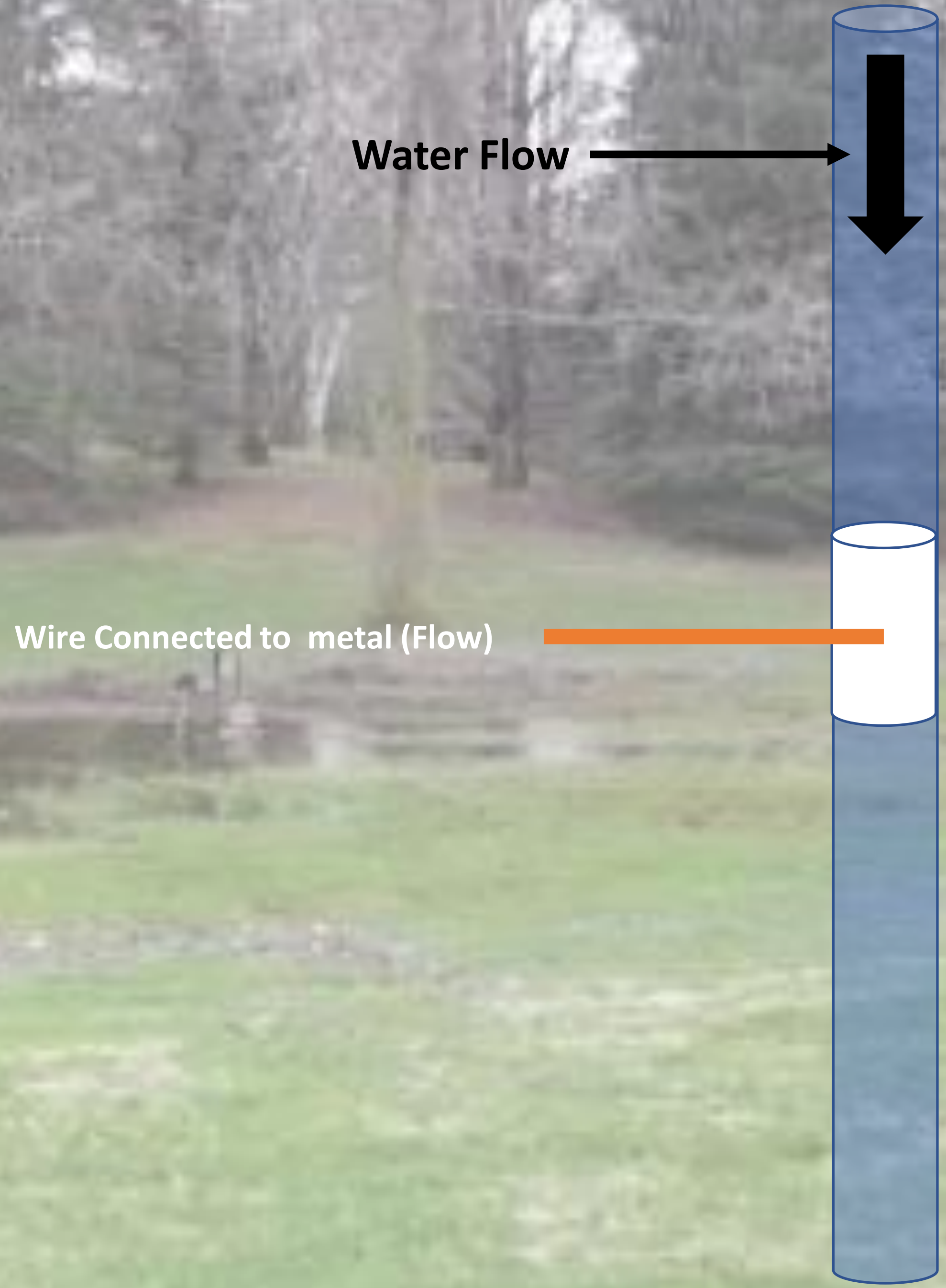
Insulator Heat Shrink

Wire Connected to metal (Empty)

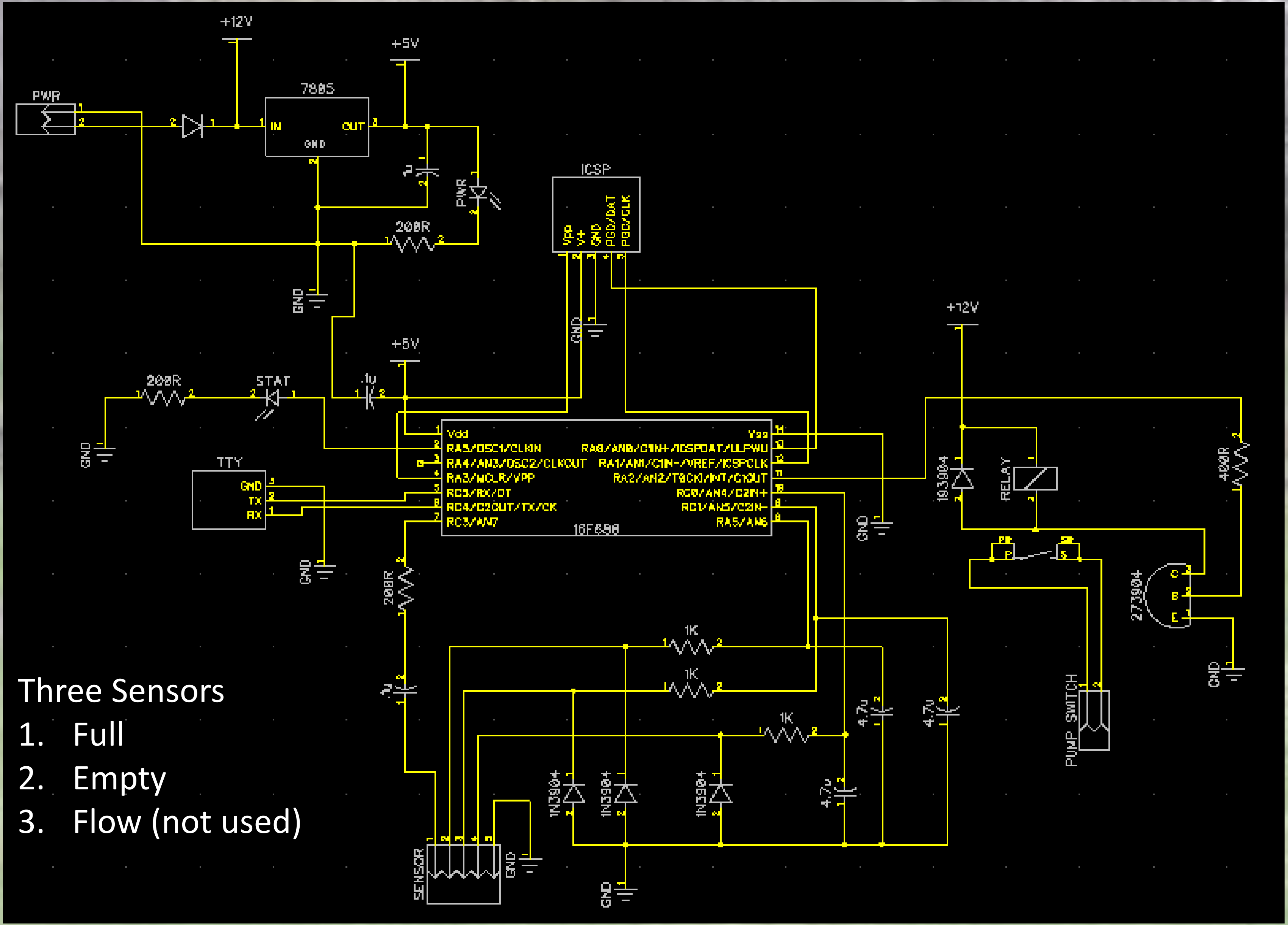
Metal Added on top of insulator

FLOW SENSOR

- Needed to identify pump on and flow present
- Thought metal in hose could detect AC signal
- Did not work reliably and ambandoned



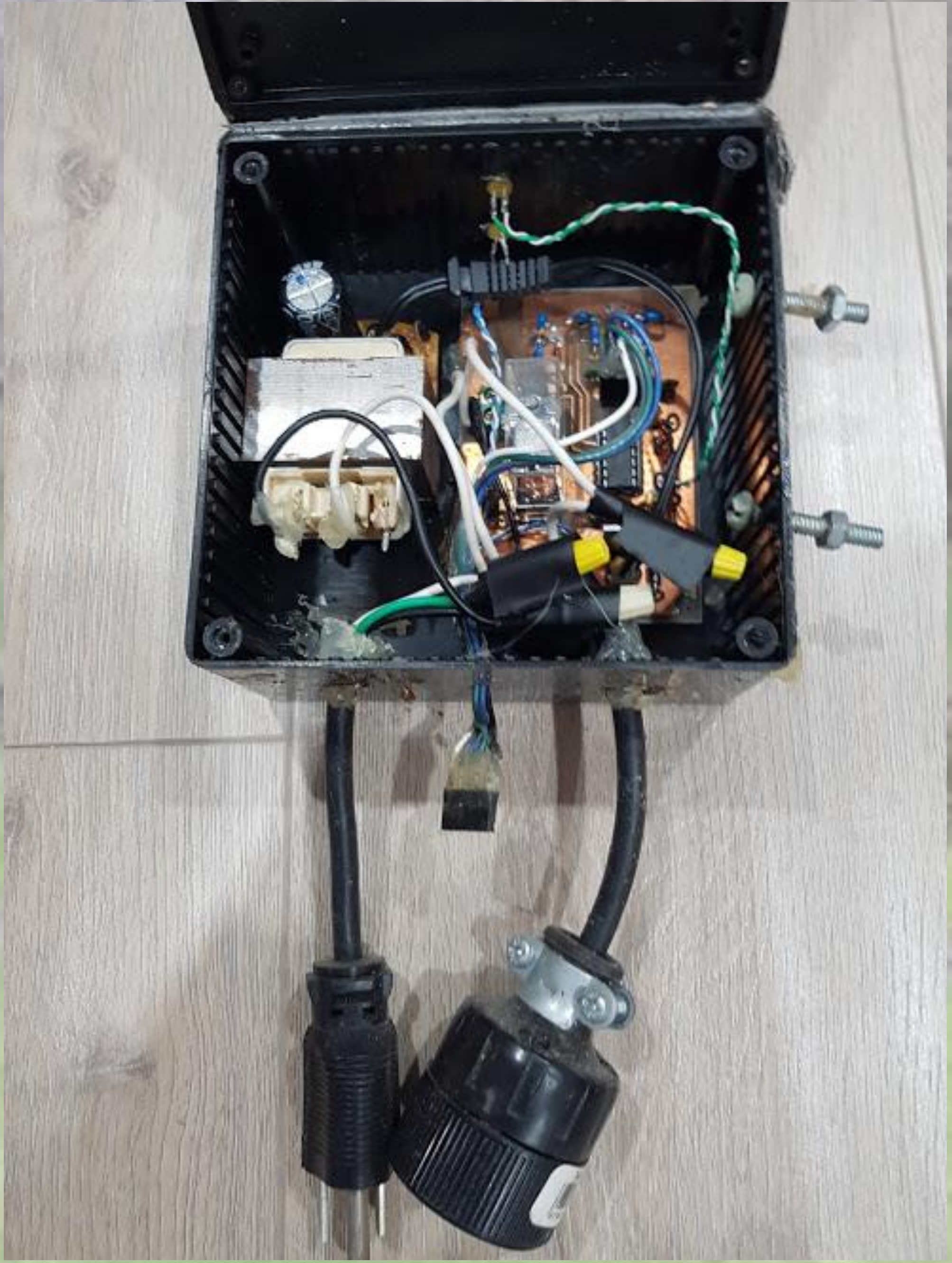
INITIAL CIRCUIT

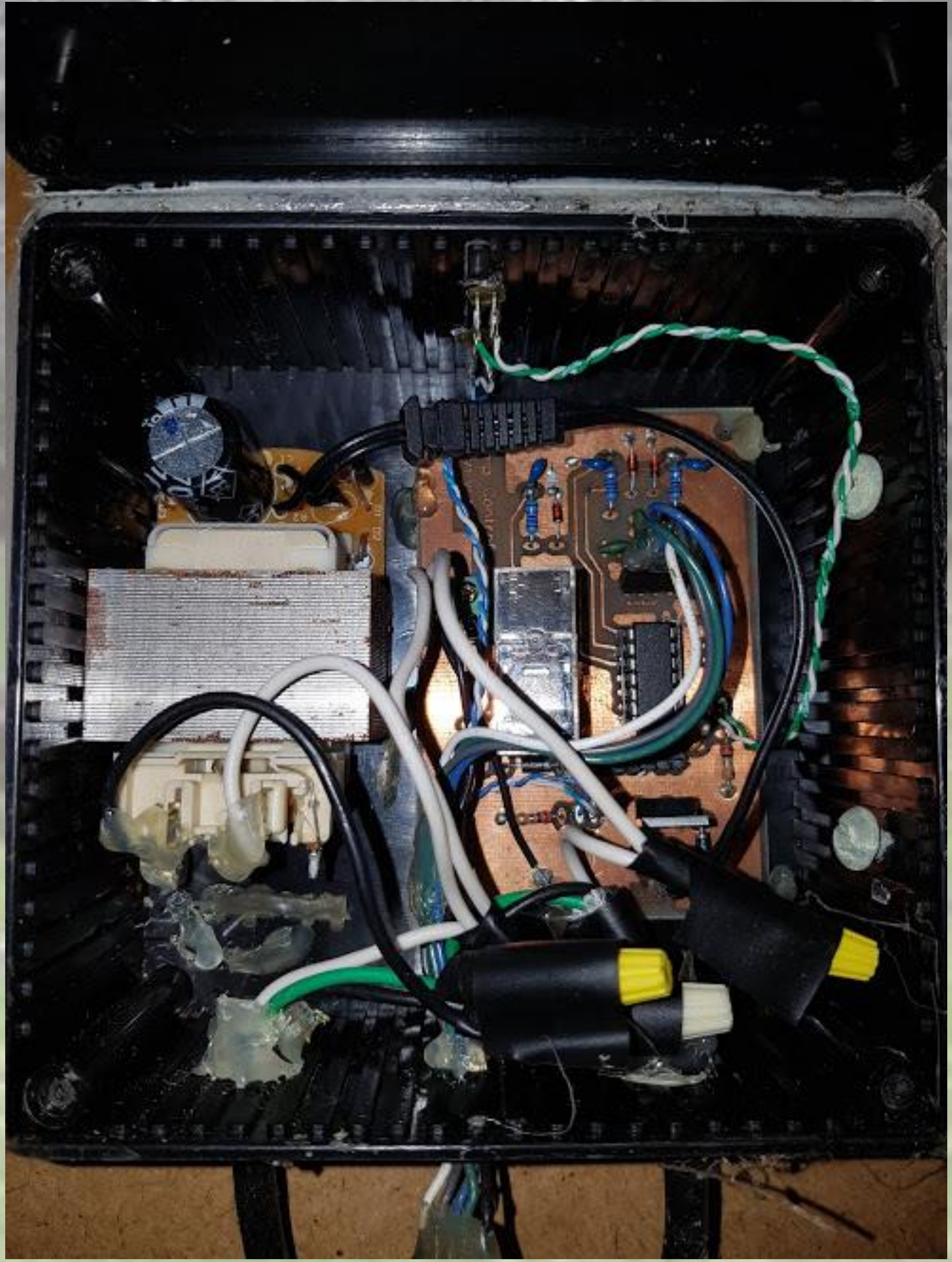


Three Sensors

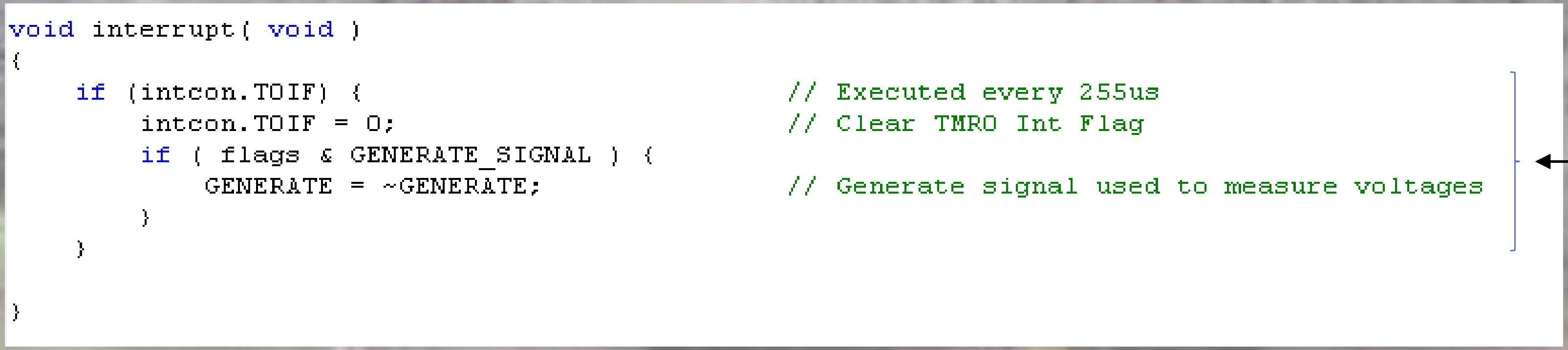
1. Full
2. Empty
3. Flow (not used)

BUILD



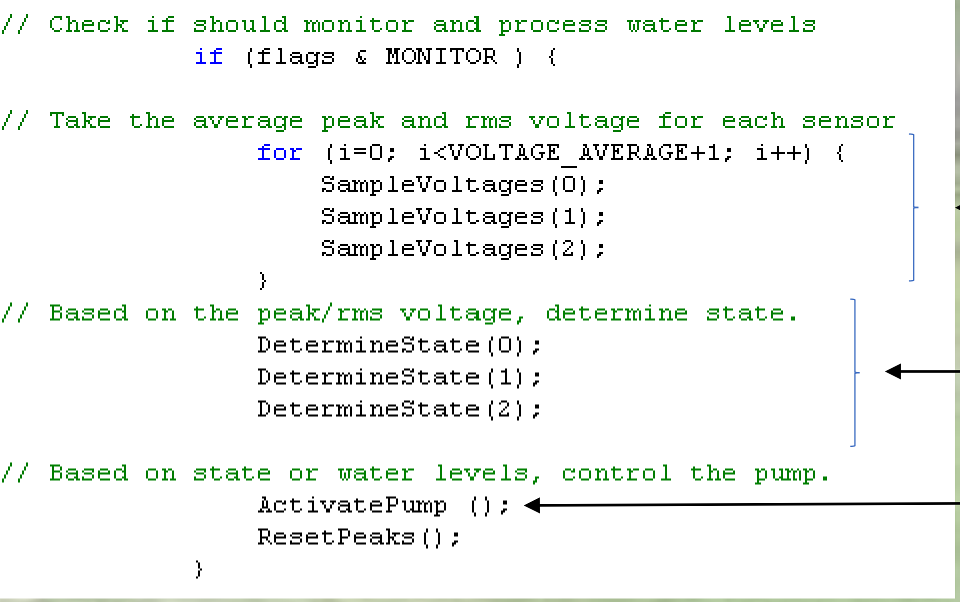


SOFTWARE - MAIN



```
void interrupt( void )
{
    if (intcon.TOIF) {                              // Executed every 255us
        intcon.TOIF = 0;                            // Clear TMRO Int Flag
        if ( flags & GENERATE_SIGNAL ) {
            GENERATE = ~GENERATE;                   // Generate signal used to measure voltages
        }
    }

}
```

Generate 4 KHz AC Signal



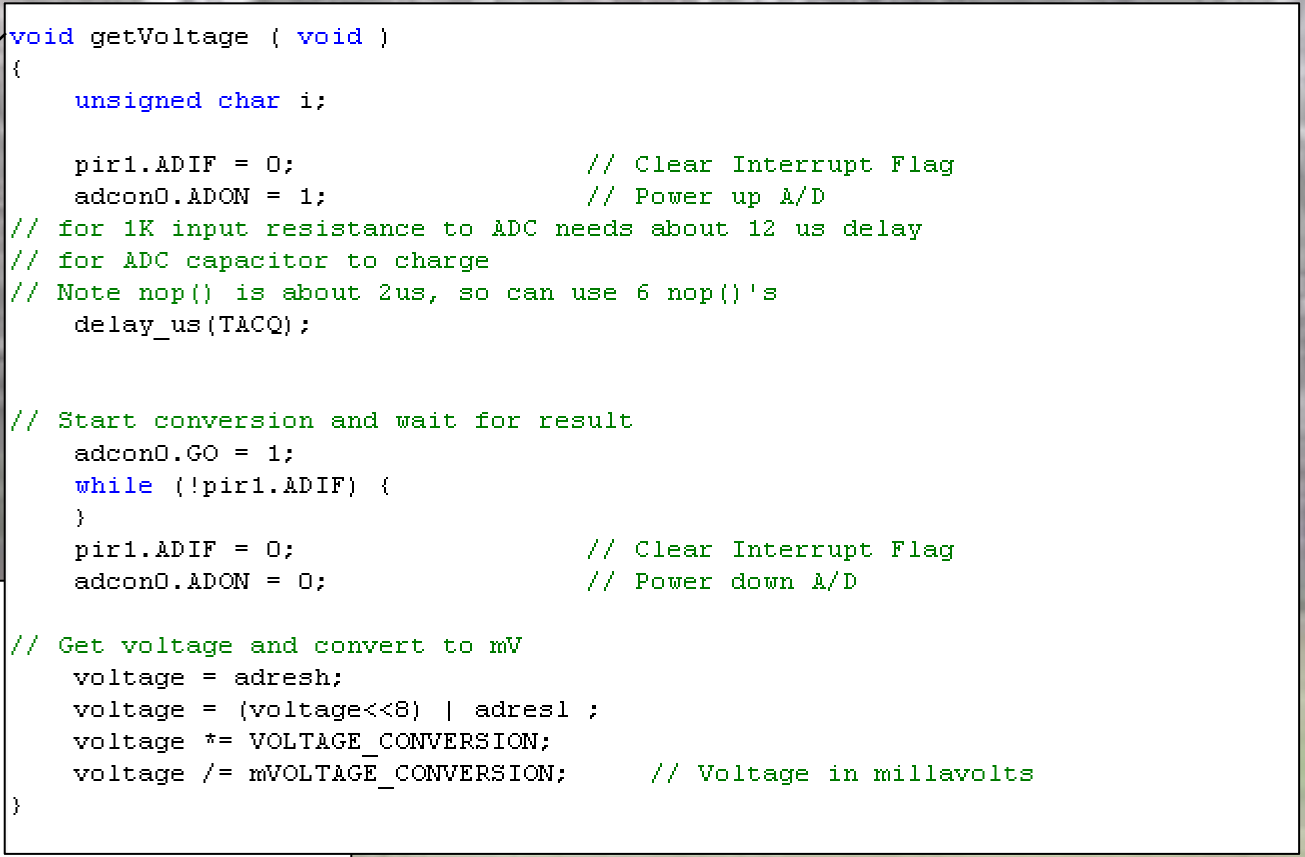
```
// Check if should monitor and process water levels
            if (flags & MONITOR ) {

// Take the average peak and rms voltage for each sensor
                for (i=0; i<VOLTAGE_AVERAGE+1; i++) {
                    SampleVoltages(0);
                    SampleVoltages(1);
                    SampleVoltages(2);
                }
// Based on the peak/rms voltage, determine state.
                DetermineState(0);
                DetermineState(1);
                DetermineState(2);

// Based on state or water levels, control the pump.
                ActivatePump ();
                ResetPeaks();
            }
```

Detect peak voltage and RMS voltage

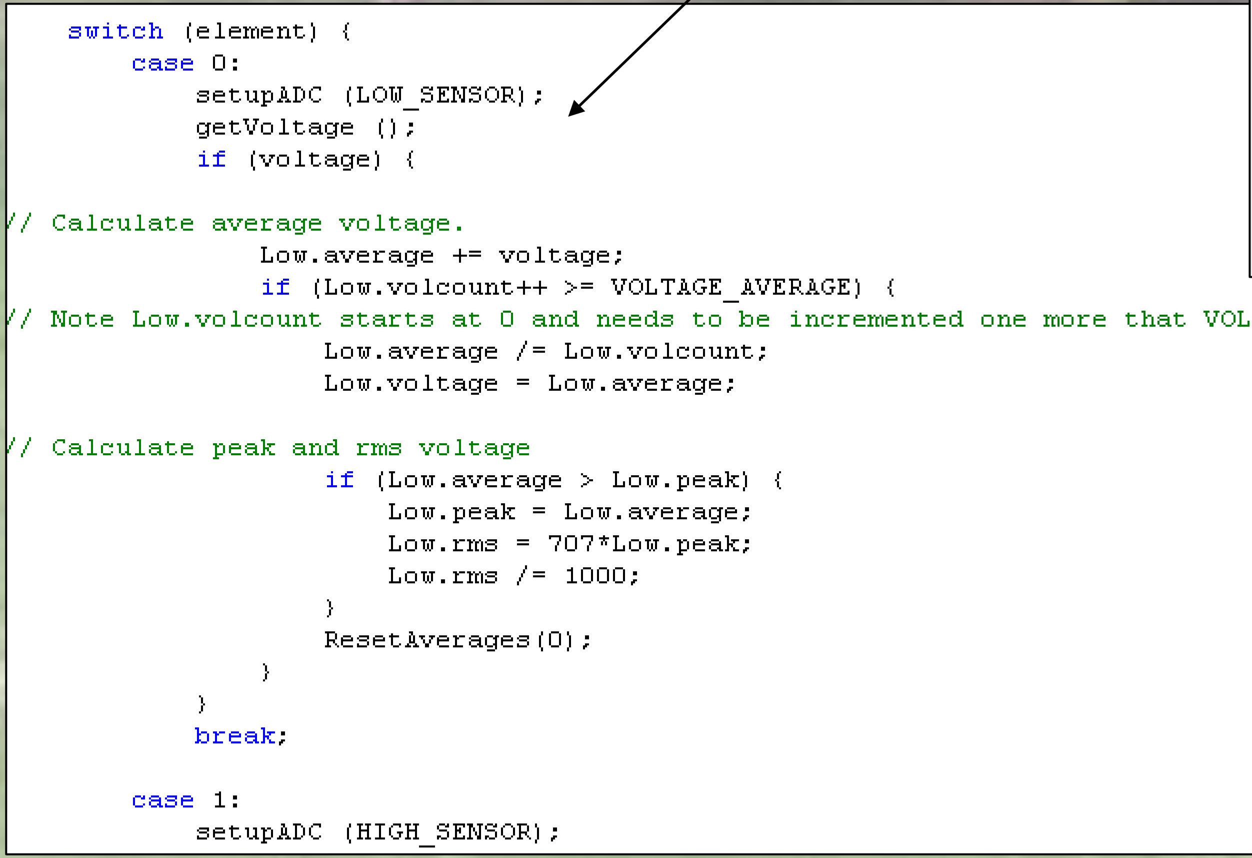
Identify if bucket is “**empty**”, “**filling**” or “**full**”

Turn on pump if bucket is “**full**”

SOFTWARE - VOLTAGE

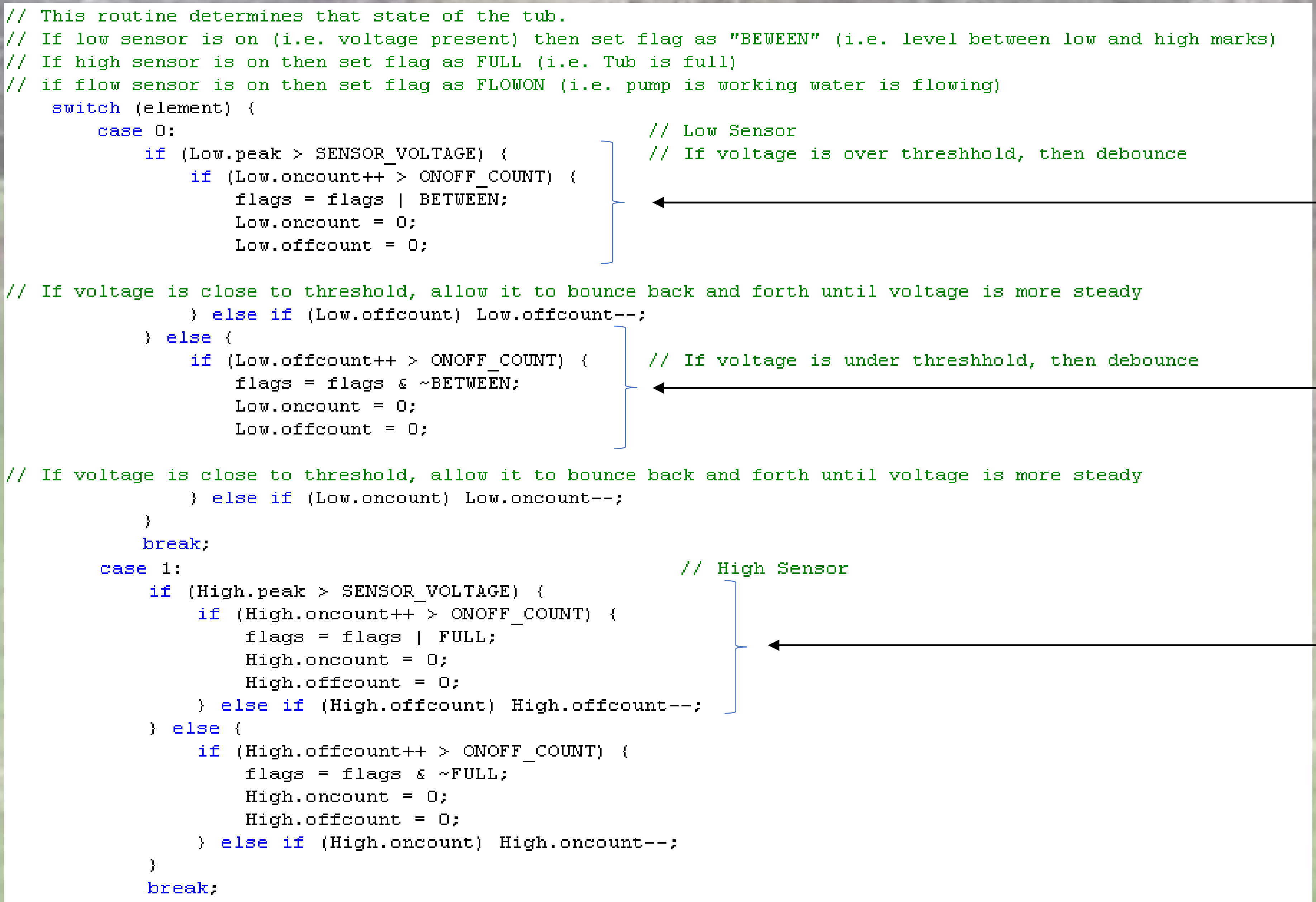


```
void getVoltage ( void )
{
    unsigned char i;

    pir1.ADIF = 0;                      // Clear Interrupt Flag
    adcon0.ADON = 1;                    // Power up A/D
// for 1K input resistance to ADC needs about 12 us delay
// for ADC capacitor to charge
// Note nop() is about 2us, so can use 6 nop()'s
    delay_us(TACQ);


// Start conversion and wait for result
    adcon0.GO = 1;
    while (!pir1.ADIF) {
    }
    pir1.ADIF = 0;                      // Clear Interrupt Flag
    adcon0.ADON = 0;                    // Power down A/D

// Get voltage and convert to mV
    voltage = adresh;
    voltage = (voltage<<8) | adresl ;
    voltage *= VOLTAGE_CONVERSION;
    voltage /= mVOLTAGE_CONVERSION;     // Voltage in millavolts
}
```



```
    switch (element) {
        case 0:
            setupADC (LOW_SENSOR);
            getVoltage ();
            if (voltage) {
// Calculate average voltage.
                Low.average += voltage;
                if (Low.volcount++ >= VOLTAGE_AVERAGE) {
// Note Low.volcount starts at 0 and needs to be incremented one more that VOLTAGE_AVERAGE
                    Low.average /= Low.volcount;
                    Low.voltage = Low.average;

// Calculate peak and rms voltage
                    if (Low.average > Low.peak) {
                        Low.peak = Low.average;
                        Low.rms = 707*Low.peak;
                        Low.rms /= 1000;
                    }
                    ResetAverages(0);
                }
            }
            break;

        case 1:
            setupADC (HIGH_SENSOR);
```

SOFTWARE - STATES



```
// This routine determines that state of the tub.
// If low sensor is on (i.e. voltage present) then set flag as "BEWEEN" (i.e. level between low and high marks)
// If high sensor is on then set flag as FULL (i.e. Tub is full)
// if flow sensor is on then set flag as FLOWON (i.e. pump is working water is flowing)
    switch (element) {
        case 0:                                             // Low Sensor
            if (Low.peak > SENSOR_VOLTAGE) {                // If voltage is over threshhold, then debounce
                if (Low.oncount++ > ONOFF_COUNT) {
                    flags = flags | BETWEEN;
                    Low.oncount = 0;
                    Low.offcount = 0;

// If voltage is close to threshold, allow it to bounce back and forth until voltage is more steady
                } else if (Low.offcount) Low.offcount--;
            } else {
                if (Low.offcount++ > ONOFF_COUNT) {         // If voltage is under threshhold, then debounce
                    flags = flags & ~BETWEEN;
                    Low.oncount = 0;
                    Low.offcount = 0;

// If voltage is close to threshold, allow it to bounce back and forth until voltage is more steady
                } else if (Low.oncount) Low.oncount--;
            }
            break;
        case 1:                                                // High Sensor
            if (High.peak > SENSOR_VOLTAGE) {
                if (High.oncount++ > ONOFF_COUNT) {
                    flags = flags | FULL;
                    High.oncount = 0;
                    High.offcount = 0;
                } else if (High.offcount) High.offcount--;
            } else {
                if (High.offcount++ > ONOFF_COUNT) {
                    flags = flags & ~FULL;
                    High.oncount = 0;
                    High.offcount = 0;
                } else if (High.oncount) High.oncount--;
            }
            break;
```

Water at low-water mark so must be at least filling.

Water BELOW low-water mark so turn pump **off** if its on

Water ABOVE high-water mark and ABOVE low-water mark so turn pump **on** if its off

FIN