



## AquaPing Acoustic Leak Detector

User Manual Version 0.9

### I. Principle of Operation

When pressurized water escapes from a seam, crack, or loose fitting on plumbing, broadband high frequency acoustics are emitted. These signals can travel an extended distance ( $> 30$  feet) in free-space and be detected by a remote sensor. This suggests a stand-off method for water leak detection that is substantially different compared to conventional contact moisture sensors and various types of smart flow meters installed on or inside of plumbing lines.

The AquaPing combines the principles of glass breakage and smoke detectors. Glass breakage occurs on a timescale of a few seconds or less, requiring continuous data acquisition at rates approaching 1 kHz or more. Almost all plumbing leaks are persistent, which allows for sub-Hz sampling rates and exceptional battery life.

Glass breakage has the advantage of being generally louder than the ambient environment so that detector sensitivity is not an issue. Small water leaks, especially when separated from the sensor by acoustic barriers such as walls, are likely to be at or below the environmental background. Much higher sensitivity and noise mitigation is essential. This can be addressed by active and passive filtering and judicious use of statistical analysis.

The acoustic approach to water leak detection presented here works well but is not capable of identifying all water issues. It will not respond to leaking roofs or melting ice, for example. The nature of the pressurized orifice is also important. A smooth hole drilled into a pipe will produce a Laminar flow jet that emits a much weaker acoustic signal – perhaps undetectable – compared to a jagged crack or loose fitting at the same backing pressure. Acoustic sensing can be used to complement standard leak detection techniques and will be especially useful when monitoring large areas.

The AquaPing is intended to be deployed as a remote, wireless network node. There are many dozens of wireless protocols, however, so it was decided to make the device agnostic to the host hardware and thus have the widest possible use cases. It will work with any controller having an I2C interface and 3V3 power bus.

The sensor has been designed to work in an environment with a stable, relatively quiet acoustic background. Human activity will often produce noise that will impair the sensor's ability to hear leaks. The formation of puddles and similar indicators of water escaping, however, are usually obvious and

quickly noticed. Best performance and reliability are attained in conditions where people are not present.

## II. Configuration

Acoustic environments vary a lot and the sensor must be configured by the host controller for different and possibly changing conditions. Two key considerations are: i) duration of the training time and ii) duration of data acquisition. Configuration parameters are set by the host on the I2C bus; see Section IV.

**1. Training time** Sensitivity is set automatically at startup and requires the AquaPing to acquire a set of background samples. Data is gathered at a 1 Hz rate with sample size that is user-configurable in the range 10–255; default is 30 seconds. The environment should be nominally quiet and leak-free. If statistical analysis of this data determines that the background acoustic level is sufficiently stable, the green LED blinks once and the firmware transitions to monitor mode. If not, the red LED blinks twice and the background acquisition repeats until stability is attained.

If the sensor cannot find a sufficiently stable background, it will loop indefinitely in the training mode. The user can try lengthening the training time by increasing the background sample size. If this doesn't work, there are two likely explanations: i) The 3V3 power line has excessive noise present. Consider adding more filter capacitance or a capacitance multiplier circuit. ii) The device is being deployed in an environment that is too noisy and/or unstable for reliable leak detection.

**2. Time sampling window** When the sensor is monitoring, it acquires a constantly updated data set. The product of event array size times the polling period determines the time sampling window. This is set by the user and requires some situational awareness. To illustrate, consider an example application that places the AquaPing in a bathroom with a shower. The sound from a normally running shower will likely be interpreted as a leak signal. To prevent false alarms, the sampling window can be set for 20 minutes; persistent signals exceeding this time would alert for a potential leak. An array size of 120 and polling period of 10 seconds will produce the desired 20 minute window (1200 seconds). To allow for acoustic irregularities, the alarm trigger count can be set lower than the event array size, eg. 100. The trigger count is not critical and is a free parameter for testing and experimentation.

It may be advantageous for the controller to put the sensor to sleep when its operating environment is known to be noisy. This will save battery power and prevent false alarms. The sensor also has an automated noise mitigation capability that helps prevent false alarms; see Section III.4.

**3. Out of box operation** The default settings can be used to demonstrate the AquaPing without a controller on the I2C bus. Simply connect a **quiet** DC power source at the positive 3V3 and GND pins. Double-check your connections as there is no reverse polarity protection. The sensor will first acquire a 30 second background and – if the background is determined to be sufficiently stable – begin polling at a 2 second period as described above. The green and red LEDs will blink in response to the acoustic environment; details are in Section III.4.

## III. Design

The sensor is fabricated on a 48 x 33 mm<sup>2</sup> PCB using standard, low cost electronic components. It is designed to work with a host controller as part of a low power wireless system with a well-filtered 1.8–3.6 VDC power source. Two-way communication takes places via I2C. All signal processing and

analysis occurs on the device; no audio is sent to the cloud. Real-time sensor information is available on-demand as 12 status bytes.

**1. Analog front-end** The acoustic transducer is a surface-mount Knowles SiSonic MEMS microphone that forms one end of a rectangular Helmholtz resonator. The resonator center frequency is  $\sim 8$  kHz with  $Q \sim 30$ . The microphone signal is amplified 40 dB and sent to a 2-stage, 4th-order Sallen-Key high-pass filter with 3 dB cutoff at  $\sim 8$  kHz. This is followed by a 23 dB post-filter gain stage. The microphone is pulse-biased and the amplifier stages are gated on-off to increase battery life. To illustrate, if a single audio acquisition occurs every 2 seconds (default), the sensor will be in sleep mode  $> 99\%$  of the time.

The microphone and resonator provide some sound collection directivity and the PCB can be oriented for this purpose. Be aware that acoustics can bounce off surfaces like walls and ceilings, which can complicate locating the signal source.

**2. Signal processing** After the filtered signal leaves the analog front-end, a 12-bit ADC in the MCU collects 256 sample points at 33.3 kS/s corresponding to a 7.68 ms duration temporal acquisition. The data is checked for ADC saturation and tagged as noisy if necessary. If no saturation has occurred, the MCU performs an FFT of the time data using a uniform window, as appropriate for broadband white noise. The LEA of the Texas Instruments MSP430FR5994 calculates the FFT  $> 10\times$  faster and more efficiently than an ARM-Cortex M0+. The spectral data is rendered as an array of complex numbers separated by 130.2 Hz frequency steps. A 34 point subset in the range 7–11.5 kHz is selected. The spectral energy at each frequency step is calculated and then summed to provide the total acoustic energy in the frequency window. The combination of mechanical, analog, and digital filtering greatly reduces the influence of background noise, dramatically reduces the potential of ADC saturation, and eliminates the possibility of intelligible conversations being discerned if the sensor network becomes compromised, i.e. eavesdropping is impossible.

**3. Analysis** Each polling cycle analyzes the acoustic signal and designates it as one of the following three events: quiet, leak, or noise. These events are accumulated in three separate data arrays. The value 1 is entered if that specific event occurs; otherwise it is counted as 0. For example: if a leak signal is detected, a 1 is added to the leak array while 0 is placed in the quiet and noise arrays. As new events are added, the oldest events are removed, i.e. the fixed-size data arrays slide in time.

All data arrays are the same size and contain elements that are either 0 or 1. The array size is configured by the host in the range 10–255; default is 20. Size and polling rate determine the time over which a full data set is acquired. The default 0.5 Hz polling rate and 20 event set size define a 40 second data accumulation window. This time sampling window will depend on the application and should be adjusted by the user as necessary. If the environment is known to always be nominally quiet, a short time window will identify a potential leak quickly. In a bathroom where a shower may be running 20 minutes or more, the time sampling window must be longer than this to avoid false alarms.

If leak events become an appreciable portion of the time sampling window, an alarm condition may exist. Ideally, all events in the window will be identified as leaks. In practice, environmental noise, irregularities in water flow, pressure fluctuations, the presence of bubbles, or other acoustic disturbances may be recorded as quiet or noise events – not leaks. To account for non-ideal signals, the alarm trigger size can be independently set below the event set size. The trigger value is not critical; testing has shown that 80–90 percent of the event set size is a good choice. The default values are 18 for the trigger size and 20 for the event size.

When a potential leak is identified, the sensor provides an immediate alert by toggling the digital ALARM line to the high state (logic 1). The host controller can monitor this line with an interrupt. The corresponding status byte is also set; see I2C communications in Section IV.

The real-time count of quiet, leak, and noise events are available at any time by querying the sensor via I2C. If noise events become significant, the noise status byte is set. This is important, as leak detection is compromised when ongoing noise is present. The count of quiet, leak, and noise events sum to the event set size.

**4. Noise mitigation** To hear the smallest leaks, the electronics must be configured to have high sensitivity to very weak signals. The trade-off is increased susceptibility to ambient environmental noise that can increase the probability of false alarms. The sensor is adept at identifying various types of noise that is ignored. Excessive noise will, however, mask the presence of a leak. By accumulating statistics from each polling cycle, the sensor can inform the host controller that it is operating in a noisy environment.

On power-up or after a reset, the sensor enters a training session to learn about the ambient acoustic environment. It collects a set of acoustic signals of duration  $\sim 8$  ms at a 1 Hz rate. The size of the background data set can be in the range 10--255; default is 30 corresponding to a 30 second total acquisition time. The data is analyzed to get an average background ( $\bar{x}_B$ ) and standard deviation ( $\sigma_B$ ). If  $2\sigma_B > \bar{x}_B$ , the background is determined to be too noisy for reliable operation. The red LED blinks twice and a new background data set is collected. To obtain additional noise rejection, the acoustic data stream is checked for ADC saturation. Excessively loud signals are discarded and the individual acquisition is repeated. This continues until the background acoustic environment is sufficiently quiet and stable. The green LED blinks once and the sensor transitions to monitoring.

The ambient acoustic level and electronic noise floor determine the sensitivity to leaks. For this reason, a well filtered supply voltage should be placed on the 3V3 terminal. Battery power does not guarantee immunity from noise originating in other electronics that can propagate into the sensor board, potentially causing oscillation of the high-gain amplifier stages.

Once the background is established, the sensor begins polling at a user defined rate; default is 0.5 Hz. Slower polling rates increase battery life. The ADC audio samples are first checked for saturation. If an excessively loud signal is present, a **noise event** is recorded, the red LED blinks twice, and the firmware enters a sleep state to wait for the next polling cycle.

The training session sets the threshold for detecting a leak signal, which is one standard deviation above the noise floor:  $\bar{x}_B + \sigma_B$ . If there is no ADC saturation, the spectral energy ( $x_1$ ) derived from an FFT of the temporal data is checked. If  $x_1 < \bar{x}_B + \sigma_B$ , no leak is detected and a **quiet event** is recorded; the green LED blinks once. If  $x_1 > \bar{x}_B + \sigma_B$ , additional processing commences: four sequential acoustic measurements are made; these are separated by 45 ms corresponding to a sampling rate of about 22 Hz. If any of these additional acquisitions saturate the ADC, further processing aborts, an environmental **noise event** is recorded, the red LED blinks twice, and the polling loop pauses for the next iteration.

If all five filtered audio signals are within the dynamic range of the ADC, their individual spectral energies ( $x_i : i=1-5$ ) are used to calculate the average energy ( $\bar{x}$ ) and standard deviation ( $\sigma$ ). If  $2\sigma > \bar{x}$ , excessive fluctuations are present in the data set. This indicates a level of environmental noise that prevents reliable leak detection. This polling cycle is recorded as a **noise event**, the red LED blinks twice, and the program waits in low-power mode for the next timed iteration.

If the acoustic energy fluctuations are sufficiently small, the next step is to check if all five spectral energies ( $x_i$ ) exceed the leak signal threshold:  $x_i > x_B + \sigma_B$ . When this occurs, a **leak event** is recorded and the red LED blinks once. Otherwise, this loop iteration is associated with impulse noise, but recorded as a **quiet event**. The green LED blinks twice.

Performing the above sequence of tests and checks ensures that only persistent, stable signals in the target acoustic frequency range are counted as **leak events**. This is a design trade-off between sensitivity and reliability that is needed to reduce false alarms.

#### IV. I2C write and read operations

This section describes how a master controller interfaces with the slave sensor on the I2C bus. Recommended clock rate: 100 kHz. External pullup resistors required (4.7k recommended). The default I2C address 0x77.

**1. Commands (write only)** All commands are sent as two-byte pairs. Send START, followed by the 7-bit slave address and the R/W bit cleared, then the desired function hex code and a single byte data field as shown in Table 1. Finally, assert NACK and STOP. Between 1 and 8 pairs (function byte + data byte) can be sent, provided there are an even number of bytes. The hex code and/or data byte can be sent as hexadecimal or decimal. With one exception, all write commands clear the counting arrays. Either a two-byte sequence to turn the LEDs on (0x6C 0x31) or off (0x6C 0x30) will *not* clear the counts.

FUNCTION	HEX CODE	DATA BYTE
System reset	0x72	0x72
Alarm	0x61	0x30 or 0x31 (Off or On) Default Off
Sleep	0x70	0x30 or 0x31 (Sleep or Active) Default Active
Background array size*	0x62	0x0A – 0xFF (10 – 255) Default: 30
Event array size*	0x65	0x0A – 0xFF (10 – 255) Default: 20
Alarm trigger size*	0x74	Less than or equal to event array size; Default: 18
Polling period*	0x6F	0x01 – 0x09 (1 – 30 sec) Default: 1 sec; see Table 2
LED*	0x6C	0x30 or 0x31 (Off or On) Default On

**Table 1.** Two-byte write commands. The five settings marked with an asterisk (\*) are written to persistent storage; the data bytes are retained with power cycling.

The function descriptions are as follows:

- System reset restores the default settings. This can be sent with other functions, but they will be ignored.
- An alarm state is cleared with 0x61 0x30. Alarm can be set by master with 0x61 0x31; use for testing.
- Sending 0x70 0x30 turns the sensor off and reduces power consumption to minimum. Previously set monitor functions in Table 1 are retained. The sensor is re-activated with 0x70 0x31.
- Background size: The sensor will automatically train to learn about the nominally quiet background, i.e. with no noise or leaks present. The background acoustic level is acquired at 1 second intervals for the specified number of counts. Default background size is 30 counts = 30 seconds. A new training

session is performed: i) at startup, ii) whenever the background count is set, eg. 0x62 0x1E, or iii) on system reset. The training session sets the sensitivity level.

- The event array size and alarm trigger size are configurable from 10 to a maximum of 255. These arrays record the number of events below and exceeding the sensitivity level found in the training session. The oldest events are discarded from the arrays as each new event is added. An alarm condition exists when the sum of events above the sensitivity level (leak\_count) reaches the alarm trigger size. Alarm trigger size must be less than or equal to event array size.
- Polling period determines the rate at which the sensor makes an acoustic measurement. The period is set with one of the data bytes in Table 2. Longer measurement intervals reduce battery power consumption. Default is 2 seconds.

0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09
1 sec	2 sec	3 sec	5 sec	10 sec	15 sec	20 sec	25 sec	30 sec

**Table 2.** Polling period data byte is set with function command 0x25

- The LED function (0x6C) disables (0x30) or enables (0x31) surface-mount LEDs on the PCB. Visual indicators can be useful for configuring and evaluating the device, but should be turned off when the sensor is in operation. Default is enabled (on).

*Example command string:* The 4-byte sequence 0x6F 0x03 0x6C 0x30 sets the monitor period at 3 seconds and turns the LEDs off.

Monitor parameters can be set while in the sleep state. The five parameters in Table 1 marked with an asterisk (\*) are written to persistent storage and will be retained with power cycling.

**2. Sensor status (read only)** The sensor status can be obtained at any time by asserting START, sending the 7-bit slave address with the R/W bit set, and then reading the 12-byte sequence shown in Table 3. The master should follow the sequential read by asserting NACK then STOP.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Alarm	Noise alert	quiet_count	leak_count	noise_count	background_size
Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
event_size	trigger_size	Period	LED	Sleep	Version

**Table 3.** 12-byte sensor status string.

- Byte 0 is set to 0x01 if an alarm exists; 0x00 otherwise. The digital output line marked ALARM is toggled when an alarm situation is detected. The host can monitor this line with an interrupt so it is not necessary to poll the status of Byte 0. The firmware can be configured to place the sensor into a sleep state when an alarm is detected or it can continue acquiring and processing data.
- Byte 1 is set when  $\text{leak\_count} < \text{trigger\_size}$  **AND**  $\text{leak\_count} + \text{noise\_count} \geq \text{trigger\_size}$ ; 0x00 otherwise. This provides a warning that the sensor is operating in a noisy environment.

- Byte 2 is the current count (in hexadecimal) of below-sensitivity level events.
- Byte 3 is the current count (in hexadecimal) of above-sensitivity level events.
- Byte 4 is the count (in hexadecimal) of noise events. The sensor ignores these events when assessing for the presence of a leak, which helps prevent false alarms.
- Bytes 5–8 are defined in Section 1.
- Byte 9 indicates if the LEDs are off (0x00) or on (0x01).
- Byte 10 indicates if the sensor is sleeping (0x00) or active (0x01).
- Byte 11 is the firmware version running on the slave.

**Note:** The status bytes are not accessed individually in designated registers as is often done with I2C slave devices. All 12 status bytes should be readout and parsed as required.