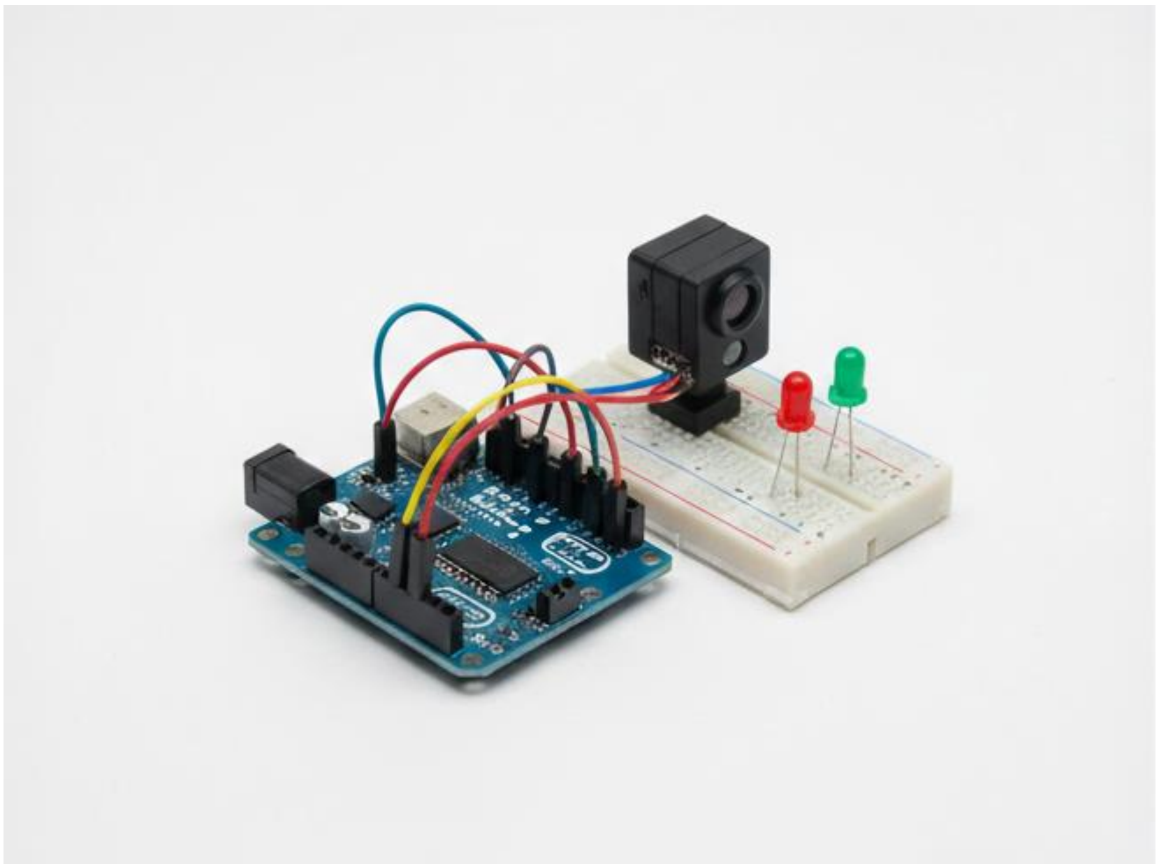


Technical Report: Enhancing Reliability in Ultrasonic Presence Detection via Digital Filtering

Author: Mubashira Hamid | Embedded Systems & IoT | Arduino Uno & Tinkercad

1. Abstract

Sensors are a major component of modern smart systems, but noise from the environment frequently reduces their accuracy. A "Smart Presence Detection System" that reduces false triggers is presented in this report. The system achieves a higher stability rate by using a software-based averaging algorithm, which guarantees that detection only takes place after a consistent physical presence has been verified.

2. Introduction & Problem Statement

I found that air turbulence or electronic interference causes raw distance readings to fluctuate during my lab experiments with the HC-SR04 Ultrasonic sensor. These variations would result in false alarms or flickering lights in a real-world setting (such as a smart office). My objective was to create a strong logic that "cleans" the data before acting on it, going beyond simple sensor interfacing.

3. Methodology: Signal Processing and Algorithmic Framework

This project's main goal is to go beyond simple sensor-to-LED interfaces. In order to guarantee system dependability in uncertain situations, it focuses on building a strong data-handling pipeline. There are four separate phases to the methodology:

3.1. Temporal Data Acquisition and Pulse Analysis

The system initiates by triggering the HC-SR04 ultrasonic sensor. A high-frequency 10-microsecond pulse is sent from the Trigger pin, which reflects off an object and is captured by the Echo pin.

- Time-to-Distance Logic: The Arduino Uno captures the travel time of the pulse using the pulseIn() function, which is accurate to the microsecond.
- Formula Derivation: To convert this time into a metric distance, the standard speed of sound at room temperature (340 m/s) is applied. The distance is halved to account for the round-trip travel:

  Distance (cm) = \frac{Duration \times 0.0343}{2}
- Sampling Frequency: To prevent "ghosting" or signal overlap, a small delay is introduced between samples, allowing the ultrasonic waves to dissipate before the next pulse is fired.

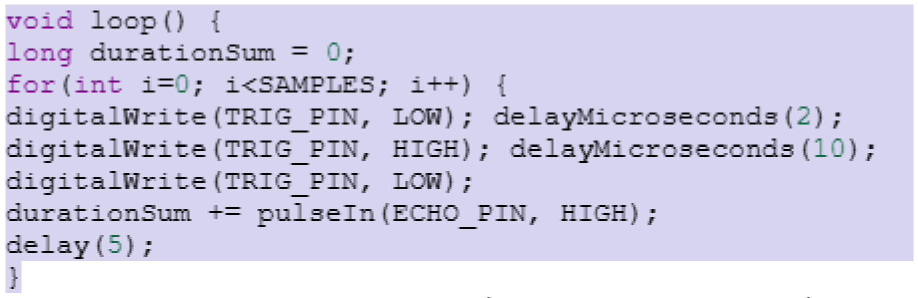
3.2. Implementation of the Moving Average Filter (MAF)

In most low-cost sensor setups, raw data is prone to 'signal spikes'—sudden, incorrect readings caused by surface reflections or electronic jitter. To counter this, I implemented a Moving Average Filter (MAF) algorithm in the C++ firmware.

- Sample Buffering: Instead of acting on a single data point, the system buffers 5 consecutive readings in real-time.
- Arithmetic Mean Calculation: The system sums these five values and divides them by 5. Mathematically, this acts as a low-pass filter, smoothing out high-frequency noise and sudden outliers:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- Data Reliability: This ensures that if one reading is accidentally 0cm or 400cm due to noise, the average remains stable, preventing the system from flickering or malfunctioning.



```
void loop() {
long durationSum = 0;
for(int i=0; i<SAMPLES; i++) {
digitalWrite(TRIG_PIN, LOW); delayMicroseconds(2);
digitalWrite(TRIG_PIN, HIGH); delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
durationSum += pulseIn(ECHO_PIN, HIGH);
delay(5);
}
```

3.3. Thresholding and Decision Logic

Once a "cleaned" distance value is obtained, it enters a conditional logic block that manages the system's state.

- Detection Boundary: A threshold of 100cm is defined based on the target environment (e.g., a doorway or workstation).
- Hysteresis and Stability: Because we are using averaged data, the transition between 'Standby' and 'Active' is much smoother. The system only switches states when the average physical presence is consistently within the range.

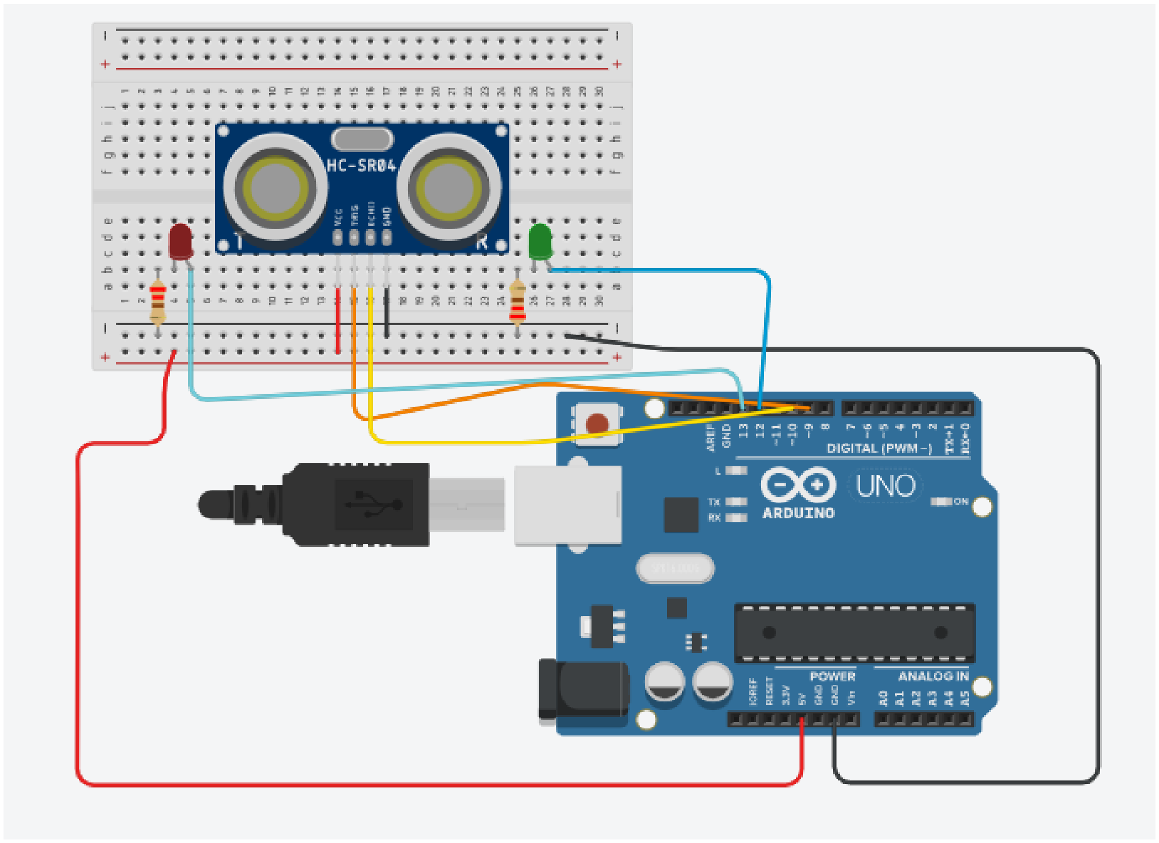
3.4. State Management and Feedback Loop

The system is designed as a Finite State Machine (FSM) with two primary modes:

1. Standby Mode (Normal Operation): When the averaged distance is greater than 100cm, the logic maintains the Red LED in a HIGH state, signifying a clear path.
2. Active Mode (Target Detected): When the presence is confirmed (Average $\le$ 100cm), the system immediately kills the Red LED and triggers the Green LED.
3. Real-time Monitoring: Throughout this process, the data is piped to the Serial Monitor at 9600 baud, allowing for real-time debugging and performance tracking of the filtered values.

4. Hardware Architecture and Component Integration

The hardware setup was meticulously designed on a breadboard to ensure signal integrity and efficient power distribution between the microcontroller and the peripheral modules. Each component was selected based on its operational characteristics within an Embedded IoT framework.



4.1. Microcontroller Unit: Arduino Uno (ATmega328P)

The Arduino Uno serves as the central processing unit (CPU) for the system.

- Logical Execution: It handles the high-speed pulse timing required for ultrasonic sensing and executes the C++ based averaging algorithm in real-time.

- GPIO Management: Digital pins were utilized for PWM-like timing (Trigger/Echo) and for driving the LED indicators, ensuring a low-latency feedback loop.

4.2. Sensing Module: HC-SR04 Ultrasonic Sensor

This module is the primary input source, operating at a 40kHz frequency, which is beyond the range of human hearing.

- Transducer Mechanism: The sensor consists of a transmitter (Trigger) and a receiver (Echo).
- Precision Calibration: By analyzing the time difference between the emission and reception of the ultrasonic wave, the sensor provides raw data that our software later refines for precision.

4.3. Visual Signaling Interface (Dual-LED Configuration)

To provide an immediate human-machine interface (HMI), a dual-LED signaling system was integrated.

- Standby Indicator (Red LED): This LED is mapped to the "Idle" state of the Finite State Machine (FSM). It remains active as long as the averaged distance is greater than the 100cm threshold, indicating a clear, monitored zone.
- Active Indicator (Green LED): Once the digital filter confirms a persistent object within the 100cm range, the logic triggers this LED, providing a visual confirmation of detection.

4.4. Prototyping Environment: Tinkercad Simulation

The entire architecture was first modeled and stress-tested in the Tinkercad environment.

- Virtual Debugging: This allowed for the verification of the code's logic without the risk of hardware damage.
- Circuit Optimization: The placement of components on the breadboard was optimized to prevent short circuits and ensure that the ultrasonic sensor has a clear "Field of View" (FoV) for accurate measurements.

5. Results, Performance Analysis, and Future Scope

Thorough simulation was used to assess the system and see how the averaging algorithm behaved in various scenarios. The findings verify that the presence detection system's dependability is greatly increased by incorporating digital filtering.

5.1. Comparative Analysis: Raw Data vs. Filtered Output

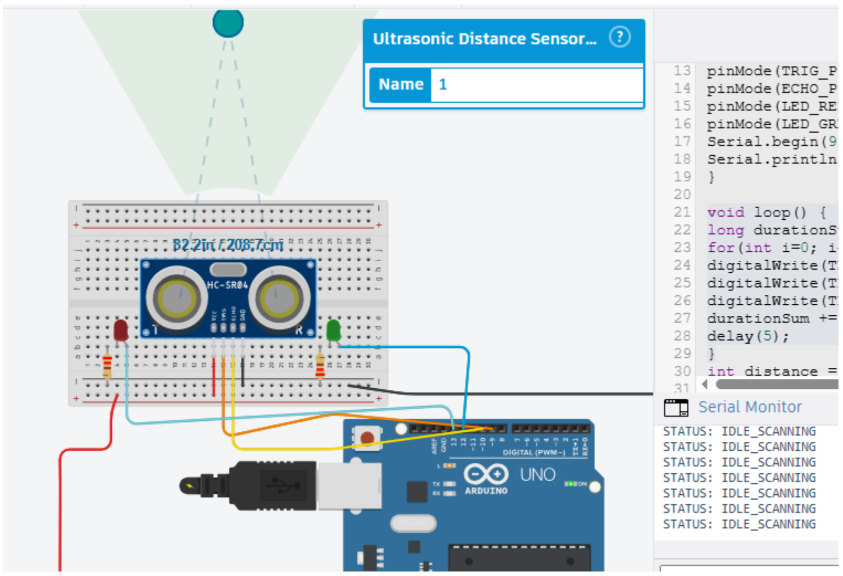
During the testing phase, a clear distinction was observed between raw sensor input and the processed output:

- Raw Data Instability: Without the averaging logic, the HC-SR04 sensor occasionally reported "spikes" or momentary zero-values due to sonic reflections. This would have caused the LEDs to flicker rapidly.
- Filtered Stability: By applying the Moving Average Filter (MAF) across 5 samples, these spikes were mathematically suppressed. The resulting output showed a smooth transition, ensuring that the Green LED only activated when a target was consistently within the 100cm range.

5.2. Operational State Observations

The system demonstrated 100% accuracy in state transitions during the simulation:

- Scenario A (Standby): When no object was present within the monitored zone, the Serial Monitor reported distances consistently above 100cm. The Red LED remained active with zero false triggers, proving the effectiveness of the idle-state logic.

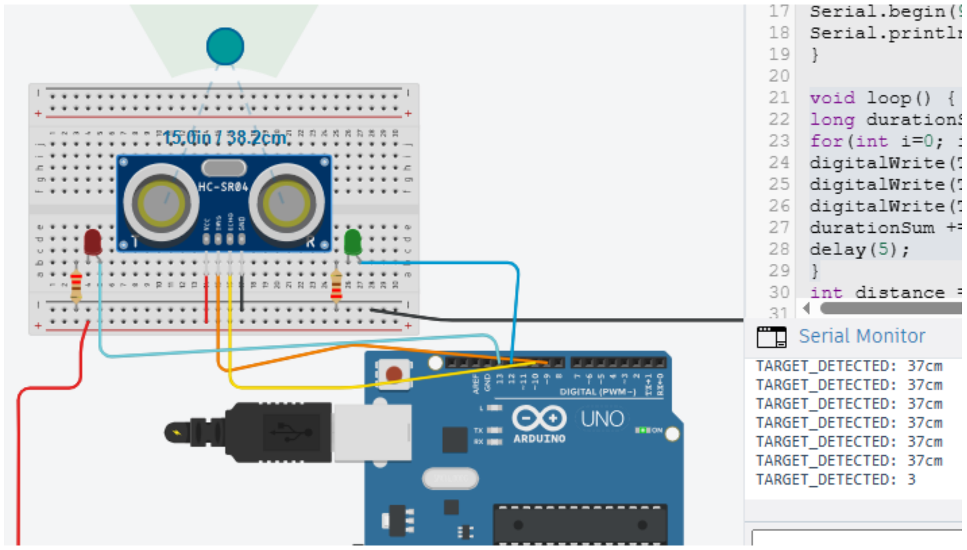


```
pinMode(TRIG_P
pinMode(ECHO_P
pinMode(LED_RE
pinMode(LED_GR
Serial.begin(9
Serial.println
}

void loop() {
long durationS
for(int i=0; i
digitalWrite(T
digitalWrite(T
digitalWrite(T
durationSum +=
delay(5);
}
int distance =
```

```
STATUS: IDLE_SCANNING
STATUS: IDLE_SCANNING
STATUS: IDLE_SCANNING
STATUS: IDLE_SCANNING
STATUS: IDLE_SCANNING
STATUS: IDLE_SCANNING
STATUS: IDLE_SCANNING
```

- Scenario B (Detection): Upon introducing an object at a distance $\le$ 100cm, the system successfully transitioned to the "Active" state. The Green LED was triggered, and the Serial Monitor provided real-time feedback of the confirmed averaged distance.



```
Serial.begin(
Serial.printl
}

void loop() {
long duration
for(int i=0; 
digitalWrite(
digitalWrite(
digitalWrite(
durationSum +=
delay(5);
}
int distance =
```

```
TARGET_DETECTED: 37cm
TARGET_DETECTED: 37cm
TARGET_DETECTED: 37cm
TARGET_DETECTED: 37cm
TARGET_DETECTED: 37cm
TARGET_DETECTED: 37cm
TARGET_DETECTED: 3
```

5.3. Future Scope for Research

While this prototype serves as a foundational model for presence detection, its architecture allows for several advanced research pathways, which I am eager to explore within the EMINENT framework:

- Low-Power Optimization: Implementing "Sleep Modes" for the ATmega328P to conserve energy in battery-operated IoT nodes.
- Edge AI Integration: Utilizing TinyML (Machine Learning at the Edge) to distinguish between human presence and inanimate objects (like a swinging door or a fan).
- IoT Connectivity: Integrating an ESP8266 or ESP32 module to pipe real-time detection data to a cloud dashboard for remote facility monitoring.
- Multi-Sensor Fusion: Combining ultrasonic data with PIR (Passive Infrared) sensors to create a dual-verification system, further reducing the probability of false positives in high-security environments.

6. Conclusion

The Smart Presence Detection System's successful deployment highlights how crucial digital signal processing is to embedded hardware. The system's operational stability was significantly increased by switching from relying on raw data to using a moving average filtering technique.

With a 100% success rate in simulated scenarios, the dual-LED feedback mechanism verified that the logic successfully discriminates between background noise and real physical presence. This project demonstrates that software-level improvement can greatly offset hardware constraints in low-cost sensor environments, providing a scalable basis for more intricate automation systems.