Radar Guard

Introduction

This project is a radar-like security system that detects objects using an ultrasonic sensor. Its main purpose is to monitor an area and alert when an object is detected nearby. The idea started from the concept of creating a simple motion detection system using basic electronic components and Arduino. I believe it is useful for others and for myself because it teaches how to integrate sensors, LEDs, a buzzer, a servomotor, and an LCD display into one functional project, while also being a practical solution for learning about automation and basic security systems.

General Description

This project is a radar-based security system using an ultrasonic sensor and servomotor to detect nearby objects and give visual and audio alerts. When no object is detected, two green LEDs are lit. When an object is detected within a certain range, the system lights two red LEDs, sounds a buzzer, and displays “Object Detected” on the LCD screen. The sensor continuously scans the area by rotating with the help of a servomotor.

Ultrasonic Sensor

- Detects the distance of objects by emitting sound waves and measuring the time it takes for the echo to return. Mounted on a servomotor to sweep the detection area.
- Function:
  - The Arduino sends a trigger pulse through pin 11.
  - The sensor sends out a sound wave, and when it bounces back, the echo is received on pin 10.
  - The time taken for the echo determines the distance to an object.

Arduino Board (UNO)

- Acts as the main controller, processing data and driving outputs like LEDs, buzzer, LCD, and servo.
- Function:
  - Acts as the central controller for the entire system.
  - Reads distance data from the ultrasonic sensor.
  - Processes logic to decide if an object is detected or not.
  - Controls outputs: turns LEDs on/off, activates the buzzer, moves the servomotor.
  - Communicates with the LCD to display messages like “Object Detected”.
  - Ensures timing and coordination of all components in the system.

LCD Display (I2C)

- Shows status messages like “Object Detected” or “Area is Empty”.
- Function:
  - Communicates with the Arduino via I2C.
  - Displays messages based on sensor input.

LEDs (Red and Green)

- Red indicates detection; green indicates clear area.
- Function:
  - Red LEDs are turned on when an object is detected within a certain distance.
  - Green LEDs are turned on when no object is detected.
  - Visual alert for danger/obstacle detection/area is clear.

Buzzer

- Emits sound when object is detected.
- Function:
  - Works as an audio alert alongside red LEDs.

Servomotor

- Rotates the ultrasonic sensor to scan.
- Function:
  - Controlled by the Arduino to sweep the ultrasonic sensor left and right.
  - The sweep allows scanning across a wide area, making detection more dynamic.

1k Resistors

- 1kΩ resistors are used to limit current flowing through the LEDs.
- They protect the LEDs from burning out due to excess current.
- They prevent the Arduino pins from delivering too much current (which could damage them).
- Based on Ohm’s Law, 1kΩ keeps the current around 3 mA, which is safe.

Hardware Design

Name	Connection (pins)	Link
Ultrasonic Sensor HC-SR04	TRIG → 11, ECHO→ 10	Ultrasonic Sensor
Servomotor Metalic Digital MG996	Signal → 12	Servomotor
LCD Display LCD 1602 with I2C	SDA → A4, SCL → A5	LCD Display
Red LEDs	Anodes connected together → 3	LEDs
Green LEDs	Anodes connected together → 4	LEDs
Buzzer	Positive → 2	Pasive Buzzer
Arduino UNO	-	Kit Plusivo Microcontroller Starter

Software Design

Development Environment

The application was developed using the Arduino IDE.

Libraries and 3rd-Party Sources

1. Servo.h: library for controlling servo motors. It handles the PWM signal generation required for precise positioning of the servo.

2. Wire.h: I2C communication library. This provides the low-level functions needed for communicating with I2C devices like the LCD display.

3. LiquidCrystal_I2C.h: A third-party library that builds upon Wire.h to provide high-level functions for controlling I2C LCD displays. It simplifies the process of initializing, writing to, and controlling the backlight of the LCD.

4. avr/interrupt.h: Part of the AVR-libc, this header provides interrupt-related functions and definitions for the AVR microcontroller, enabling the implementation of interrupt service routines.

Algorithms and Structures Implemented

1. *Interrupt-Driven Sensor Reading*

I implemented a timer-based interrupt system to handle the ultrasonic sensor readings at regular intervals without blocking the main program flow:

- Timer2 Configuration: Timer2 is configured in CTC (Clear Timer on Compare Match) mode to generate interrupts at approximately 100Hz.
- Sensor Reading Timing: A counter within the ISR ensures that the ultrasonic sensor is only read approximately once per second, reducing unnecessary processing.
- Distance Calculation: The time duration between sending and receiving the ultrasonic pulse is converted to distance using the speed of sound formula.

2. *Non-Blocking State Machine*

Rather than using blocking delay functions, I implemented a non-blocking state machine using the millis() function:

- Component-Specific Timers: Separate timing variables and intervals for each component (LCD, servo, LED flashing)
- Flag-Based Updates: The ISR sets a flag when new sensor data is available, which the main loop checks to determine if display updates are needed
- Time-Based Actions: Each component is updated only when its specific time interval has elapsed

3. *Object Detection Algorithm*

A simple threshold-based detection algorithm determines the presence of objects:

- Distance Threshold: Objects closer than 15cm are considered “detected”
- State Management: The system maintains and responds to an “objectDetected” state variable

- Validation Check: Only positive, non-zero distances are considered valid to filter out sensor errors

4. *Servo Scanning Algorithm*

I implemented a continuous scanning motion for the servo motor in “safe” mode:

- Position Tracking: Variables maintain the current position and direction of the servo
- Boundary Detection: Direction is reversed when the servo reaches its minimum (0°) or maximum (180°) position
- Speed Control: Movement speed is controlled by both the step size and update interval

5. *Alarm System*

When an object is detected, a multi-component alarm activates:

- Visual Alarm: Red LEDs flash at regular intervals
- Auditory Alarm: A high-frequency (approximately 500Hz) tone is generated on the buzzer
- Information Display: The LCD shows the exact distance of the detected object

Implemented Functions

- Setup Function: configures all hardware components and initializes the timer interrupts
- Interrupt Service Routine: handles the periodic sensor reading based on Timer2 interrupts
- Main Loop Function: implements the core logic of the system based on the object detection state
- Timekeeping System: multiple timing variables track when different components need updates

Project Innovation

The novelty of this project lies in the integration of interrupt-driven sensor processing with real-time multi-component coordination. Unlike basic proximity detection systems, this project combines continuous servo scanning with non-blocking alarm systems, creating a dynamic radar-like surveillance system. The use of Timer2 interrupts for sensor reading while maintaining smooth servo operation and responsive LCD updates represents a different approach to embedded system design.

Project Architecture and Validation

The system follows a modular architecture where the ISR handles time-critical sensor operations while the main loop manages user interface components. Validation was performed through systematic testing:

- Distance accuracy: Tested with objects at known distances (5cm, 10cm, 15cm, 20cm) showing ±0.5cm precision
- Response time: Verified 1-second detection latency meets system requirements
- Scanning consistency: Confirmed smooth 180° servo sweep without stuttering
- Component coordination: Verified simultaneous operation of LEDs, buzzer, LCD, and servo without interference

Sensor Calibration Process

The HC-SR04 ultrasonic sensor calibration involved:

- Distance Formula Validation: Verified the 0.034 cm/µs speed of sound constant through measurements against known distances

- Threshold Optimization: Through testing various objects and distances, established 15cm as optimal detection threshold balancing sensitivity and false positives

Performance Optimizations

- LCD Update Optimization: Implemented 250ms update intervals to eliminate flickering while maintaining responsiveness
- ISR Efficiency: Minimized ISR execution time by moving complex operations to main loop using flag-based communication
- Timer Configuration: Selected Timer2 to avoid conflicts with Servo library's Timer1 usage

The demonstration video showcases (the video is in the archive)

- Initial startup with “Safe area” display and green LEDs
- Continuous servo scanning motion (0-180°)
- Object detection triggering red LED flashing and buzzer alarm
- LCD displaying exact distance measurements
- Return to safe mode when object is removed

Results Obtained

The project yielded a fully operational proximity detection system that successfully integrates the core concepts learned in laboratory sessions (I2C communication, timers, interrupts, and buzzer control) while incorporating additional components like the ultrasonic sensor and servomotor. The implemented system accurately detects objects within 15cm range, provides clear visual feedback through an LCD display and LED indicators, generates audible alerts via the buzzer when objects are detected, and maintains a continuous scanning motion with the servomotor during safe conditions. Testing confirmed reliable performance with consistent object detection, accurate distance measurements, responsive alarm activation, and stable operation across extended usage periods.

Conclusion

My proximity detection system successfully demonstrates the application of multiple technologies learned throughout laboratory sessions. I effectively implemented I2C communication for the LCD display, configured hardware timers and interrupts for sensor readings, and utilized a buzzer for audio alerts. Building upon this foundation, I successfully integrated new components not covered in labs: an ultrasonic distance sensor for proximity detection and a servomotor for continuous scanning motion.

Download

radar_guard.zip

Journal

- [✓] 15/05/2025 - Hardware Design
- [✓] 20/05/2025 - Software Design

Bibliography/Resources

Software Resources

- Arduino IDE - “Arduino Software (IDE)” - https://www.arduino.cc/en/software
- LiquidCrystal_I2C Library - https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library
- Arduino Timer Interrupts - https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
- AVR Libc Reference Manual - https://www.nongnu.org/avr-libc/user-manual/
- PM laboratories

Hardware Resources

- Arduino UNO - https://docs.arduino.cc/hardware/uno-rev3/
- HC-SR04 Ultrasonic Sensor Datasheet - https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
- LCD 1602 I2C Display - https://www.vishay.com/docs/37299/37299.pdf

Export to PDF