Hybrid Ultrasonic–Doppler Radar System

Efe Civisoken, Jack Ewing

Department of Electrical and Computer Engineering, Lafayette College, Easton, PA.

ECE 414: Embedded Systems

Prof. Dr. Priyank Kalgaonkar

December 16, 2025

Abstract

This report documents the complete design, implementation, verification, and evaluation of a Hybrid Ultrasonic–Doppler Radar System developed as the final project for ECE 414. The system integrates ultrasonic time-of-flight distance sensing with microwave Doppler motion detection to simultaneously estimate object distance, angular position, and motion state across a full 360° field of view.

The system is implemented on a Raspberry Pi Pico (RP2040) microcontroller and emphasizes core embedded-systems principles including deterministic timing, interrupt-driven I/O, non-blocking execution, and modular software architecture. A formally defined finite state machine (FSM) orchestrates mechanical actuation, ultrasonic ranging, Doppler signal processing, visualization, and telemetry, ensuring correct sequencing and robustness under error conditions.

Experimental validation demonstrates accurate distance measurement, stable motion detection under noisy analog conditions, graceful handling of invalid echoes and timeouts, and real-time radar-style visualization on a TFT display. The final system satisfies all functional and technical requirements of the course and reflects professional embedded-systems engineering practice.

1. Introduction and Problem Definition

Embedded perception systems are a cornerstone of modern engineering applications, including robotics, autonomous navigation, proximity safety systems, and human–machine interaction. In both commercial and defense domains, the ability to reliably perceive the environment in real time directly impacts system safety, autonomy, and mission effectiveness. However, many low-cost embedded platforms rely on a single sensing modality, which inherently limits the richness, robustness, and reliability of the environmental information they can provide.

Ultrasonic sensors are widely used due to their low cost, simplicity, and reliability in short-range distance measurements. Nevertheless, they are fundamentally limited in that they cannot distinguish between static and moving objects, nor can they infer relative velocity. Doppler radar sensors, by contrast, are highly sensitive to motion and velocity changes and are widely employed in industrial automation, security systems, and military surveillance applications. Their primary limitation is the lack of spatial resolution when used in isolation.

As a result, hybrid sensing systems—which combine complementary modalities such as ultrasonic ranging and Doppler radar—have become increasingly important in both industry and military contexts. These systems offer improved situational awareness, greater robustness to environmental uncertainty, and enhanced fault tolerance compared to single-sensor solutions. The central problem addressed by this project is the integration of such complementary sensing modalities into a compact, extremely affordable, real-time embedded system capable of providing both spatial and dynamic information while operating under strict timing, power, and computational constraints.

Project Objectives

The objectives of this project were to:

1. Accurately measure object distance using ultrasonic time-of-flight techniques with centimeter-scale resolution.
2. Detect object motion independently using microwave Doppler sensing, enabling discrimination between static and moving targets.
3. Provide full 360° angular coverage through mechanically swept sensor assemblies.
4. Fuse spatial (distance and angle) and dynamic (motion) information into a coherent, real-time visualization.
5. Implement the system using a non-blocking, modular embedded software architecture consistent with ECE 414 design principles.
6. Demonstrate robustness to noise, invalid measurements, and boundary conditions through explicit error handling and verification.

These objectives directly align with ECE 414 learning outcomes, including interrupt-driven I/O, timing analysis, FSM-based control, and rigorous system-level testing.

Functional and Non-Functional Requirements

To formalize these objectives, a set of functional and non-functional requirements was defined during the project proposal phase. These requirements guided both system design and verification.

Functional Requirements (Must Have)

Hybrid Ultrasonic–Doppler Radar System

1. Measure object distance over a range of approximately **2–400 cm** using ultrasonic sensing, with error on the order of a few centimeters. *(Successfully completed.)*
2. Detect object motion using a **microwave Doppler sensor**, independent of distance accuracy. *(Successfully completed.)*
3. Sweep each servo through a **0°–180° angular range** with fine angular resolution (≤ 2° step size). *(Successfully completed.)*
4. Output real-time **(angle, distance, motion)** triplets over USB serial for debugging and verification. *(Successfully completed.)*
5. Visually differentiate **moving and stationary targets** in the radar display or output representation. *(Successfully completed.)*

Non-Functional Requirements (Nice to Have)

1. Provide an audible or visual alert (buzzer or LED) for moving objects within a close-range threshold (≈ 20 cm). *(Software available)*
2. Extend angular coverage to **360°** through the use of dual sensor assemblies. *(Successfully completed.)*
3. Support an optional **TFT-based radar visualization display** for real-time situational awareness. *(Successfully completed.)*
4. Maintain total system current draw below **500 mA at 5 V**, consistent with USB-powered operation. *(Successfully completed.)*
5. Employ a **modular software architecture** organized around multiple FSM-controlled tasks (servo control, ultrasonic ranging, and Doppler motion detection). *(Successfully completed.)*

Alignment Between Proposal and Final Implementation

The final system implementation satisfies all core functional requirements defined in the proposal and successfully incorporates several non-functional features, including full 360° coverage, real-time TFT visualization, and a modular FSM-based software architecture. Where design trade-offs were required, priority was given to deterministic timing, robustness, and clarity of system behavior, consistent with the educational goals of ECE 414.

2. System Overview

2.1 High-Level Architecture

The system consists of two symmetric sensing assemblies, each mounted on an independently controlled servo motor. Each assembly contains:

- One ultrasonic distance sensor
- One microwave Doppler radar module

Each servo sweeps its assembly across a 180° arc. The left and right assemblies operate in an interleaved scanning pattern, providing complete 360° spatial coverage without sensor interference.

A Raspberry Pi Pico microcontroller serves as the central controller, coordinating all actuation, sensing, processing, visualization, and communication tasks.

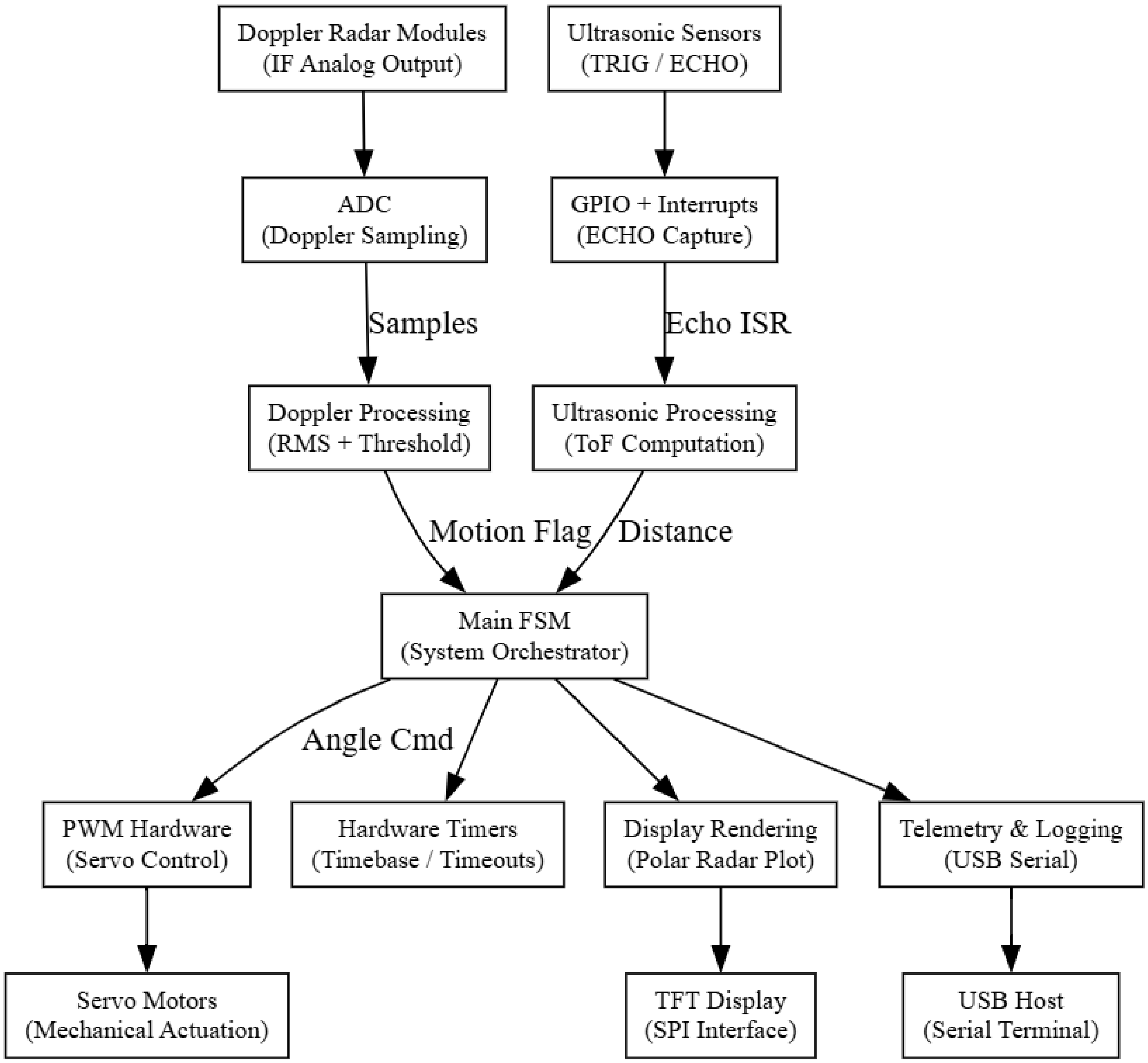

Figure 1. High-level block diagram of the hybrid ultrasonic–Doppler radar system. The diagram illustrates the separation between hardware components, RP2040 peripherals, and software processing modules. A centralized finite state machine (FSM) orchestrates servo actuation, sensor sampling, and output generation, while interrupt-driven GPIO and ADC peripherals handle timing-critical sensor acquisition.

The overall system architecture is illustrated in Figure 1. The block diagram highlights the separation between sensing hardware, RP2040 peripherals, and software processing modules, and clarifies the data flow from sensor acquisition to visualization and telemetry output.

2.2 Subsystem Decomposition

The system is decomposed into the following major subsystems:

- **Servo Control Subsystem**
  Generates deterministic PWM signals to position each sensor assembly and implements sweep and reversal logic.
- **Ultrasonic Ranging Subsystem**
  Performs time-of-flight distance measurements using interrupt-driven echo timing and explicit timeout handling.
- **Doppler Motion Detection Subsystem**
  Samples analog IF signals from HB100 modules and computes RMS energy to determine motion state.
- **Visualization Subsystem**
  Renders a radar-style polar display on a TFT screen with fading historical traces and motion highlighting.
- **Telemetry Subsystem**
  Outputs structured debug and measurement data over USB serial for validation and debugging.

3. Hardware Design

This section describes the hardware components selected for the hybrid ultrasonic–Doppler radar system and the rationale behind key electrical and physical design decisions. Emphasis is placed on suitability for real-time embedded operation, signal integrity, and system robustness.

3.1 Microcontroller Selection

The Raspberry Pi Pico, based on the RP2040 microcontroller, was selected as the central controller due to its strong support for real-time embedded applications and rich peripheral set. The RP2040 provides:

- Dual-core ARM Cortex-M0+ architecture
- High-resolution hardware timers suitable for precise time-of-flight measurement

- Multiple hardware PWM slices for deterministic servo control
- An integrated ADC for Doppler signal acquisition
- A mature C SDK with low-level hardware access

These features enable precise timing control, concurrent peripheral operation, and interrupt-driven execution without requiring an operating system, making the RP2040 well-suited for the timing-sensitive nature of this project.

3.2 Sensors and Actuators

Ultrasonic Sensors

Distance measurement is performed using HC-SR04-compatible ultrasonic sensors. These sensors emit a short ultrasonic pulse and measure the round-trip echo time, allowing distance to be computed using time-of-flight principles. Their simplicity, reliability, and compatibility with GPIO-based triggering make them well-suited for embedded distance sensing.

Doppler Radar Modules

Motion detection is implemented using HB100 microwave Doppler radar modules. These sensors output an analog intermediate-frequency (IF) signal whose frequency content corresponds to relative motion between the sensor and nearby objects. Doppler sensors were chosen to provide motion sensitivity independent of distance accuracy.

Servo Motors

Standard hobby servo motors are used to mechanically sweep the sensing assemblies. Each servo is controlled via a dedicated hardware PWM signal generated by the RP2040, ensuring stable pulse timing and repeatable angular positioning across the scan range.

Display

A TFT display (ILI934x family) is used to present a real-time radar-style visualization of the sensed environment. The display is interfaced using SPI and provides immediate visual feedback of distance, angle, and motion state.

To ensure consistent sensor alignment and repeatable mechanical motion, a custom sensor integration platform was designed and fabricated using 3D printing. This platform mechanically integrates the servo motor, ultrasonic sensor, and Doppler radar module into a rigid assembly, maintaining fixed relative positioning during angular sweeps. The use of a custom mount improved mechanical stability, reduced vibration-induced measurement variability, and simplified sensor placement compared to ad hoc mounting solutions.

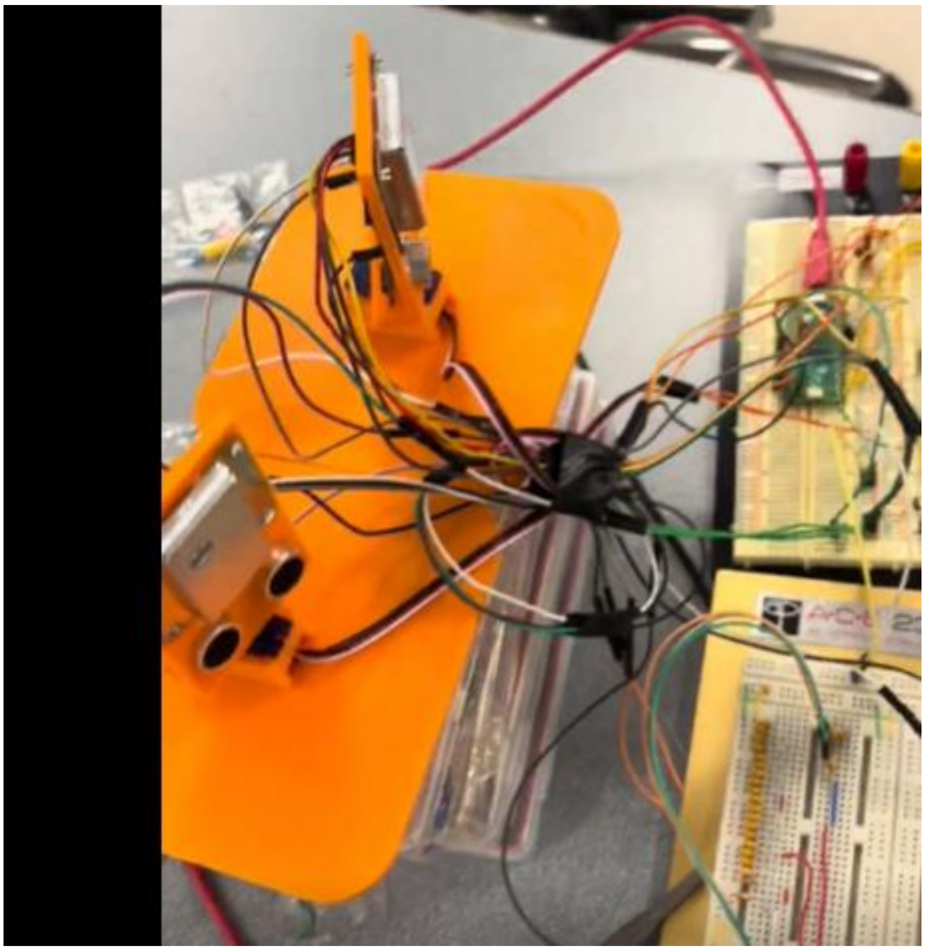

Figure 2. The final integration of the servo motor, ultrasonic sensor, and Doppler radar module into a rigid assembly through the 3D-printed sensor platform design.

3.3 Circuit Organization and Noise Mitigation

To improve signal integrity and measurement stability, the system hardware is distributed across two physically separate breadboards:

1. **Digital Breadboard**
   Raspberry Pi Pico, servo motors, ultrasonic sensors, and TFT display
2. **Analog Breadboard**
   Doppler radar modules and analog signal conditioning circuitry

This separation reduces coupling between high-frequency digital switching noise and the sensitive analog Doppler signals routed to the ADC. Empirical testing confirmed that this layout significantly reduced noise-induced fluctuations in the Doppler measurements and improved motion detection reliability.

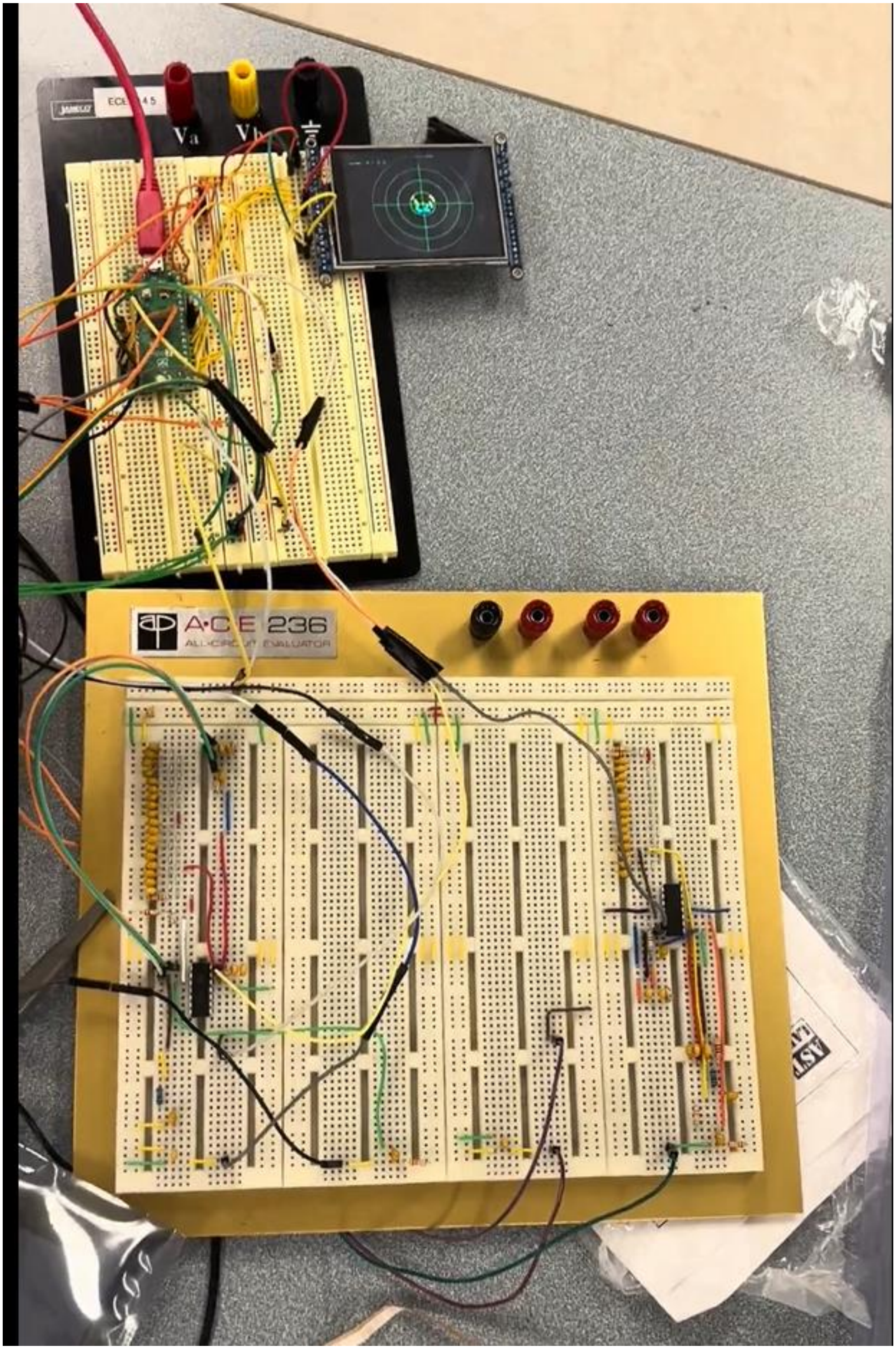
A·C·E 236
Va

Figure 3. Physical breadboard implementation of the hybrid ultrasonic–Doppler radar system, showing separation between digital and analog subsystems for noise mitigation.

3.4 Electrical Schematic

A complete electrical schematic of the hybrid ultrasonic–Doppler radar system was created using KiCad and is shown in Figure 4. The schematic documents all signal, power, and ground connections between the Raspberry Pi Pico, sensors, actuators, display, and peripheral components, serving as the formal electrical reference for the system.

The schematic complements the block diagram and FSM, presented in later section, by describing the physical hardware implementation rather than system behavior or control flow.

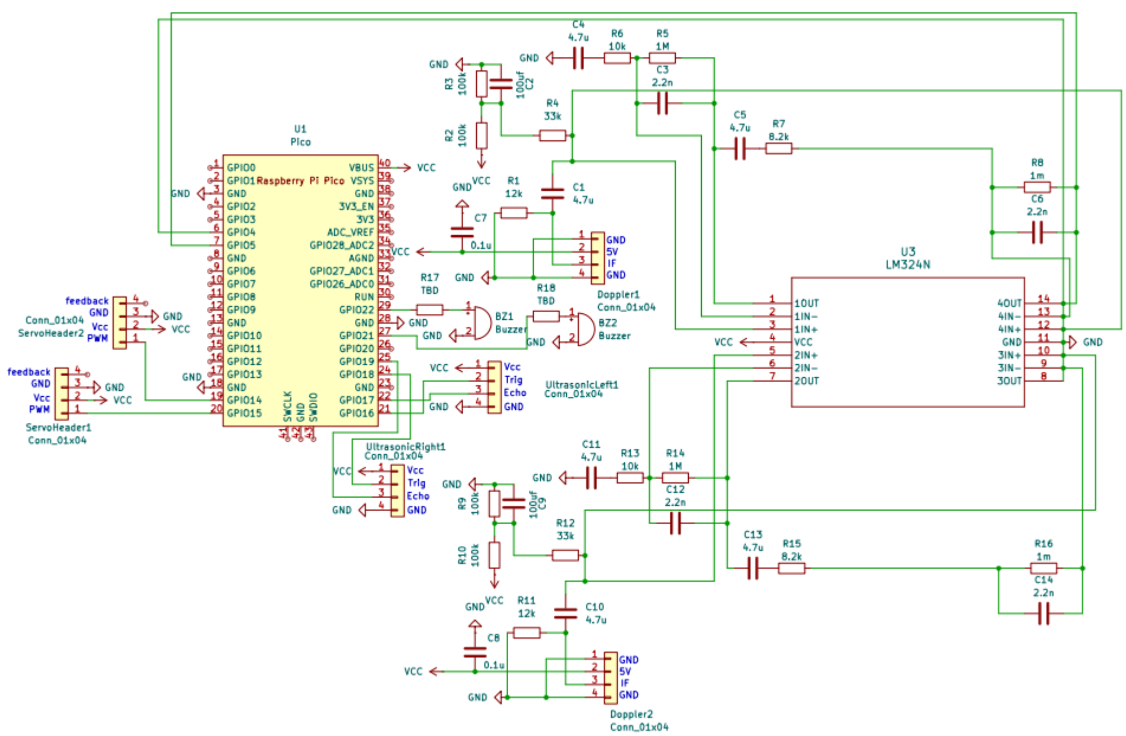

Figure 4. Electrical schematic of the hybrid ultrasonic–Doppler radar system created in KiCad, documenting all signal, power, and ground connections.

Doppler Analog Front-End

The Doppler IF signals are conditioned using an LM324-based amplification and filtering circuit, adapted from the Agilsense HB100 application note. The conditioning stages provide AC coupling, biasing, amplification, and basic filtering to match the input range of the RP2040 ADC and improve sensitivity to low-frequency Doppler components.

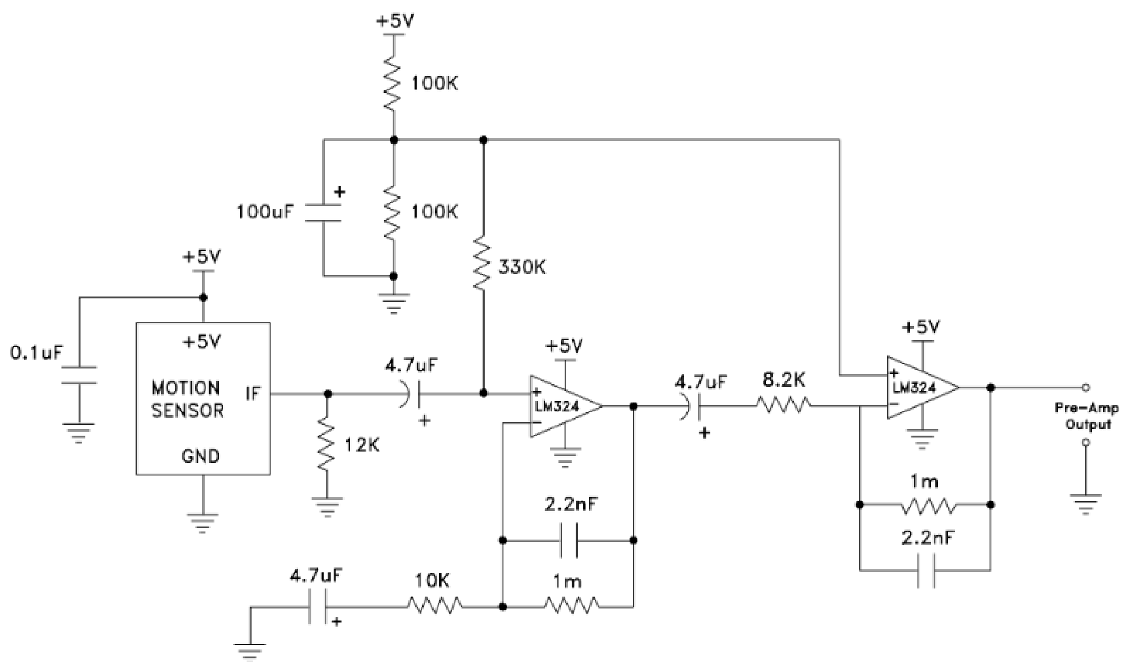

Figure 5. Reference Doppler radar analog amplifier front-end circuit from the Agilsense HB100 application note, used as the basis for the system's analog conditioning design. [3]

No explicit anti-aliasing filter was placed immediately before the ADC input. This design choice was acceptable due to the inherently low-frequency nature of Doppler IF signals and the presence of existing RC filtering and RMS-based digital processing. For a production-grade implementation, a dedicated low-pass anti-aliasing filter could be added to further suppress high-frequency noise.

4. Software Architecture

4.1 Design Philosophy

The software architecture follows a modular, event-driven, non-blocking design. Each subsystem is implemented as an independent C module with a well-defined interface. Blocking delays (e.g., busy waits or sleep calls) are avoided entirely, since it is an important principle to follow in such time-dependent embedded system designs.

System coordination is achieved through a top-level finite state machine (FSM) that explicitly models timing, sequencing, and error handling.

4.2 Module Breakdown

Module	Responsibility
main.c	FSM orchestration and scheduling
servo.c	PWM generation, sweep logic
ultrasonic.c	Triggering, echo timing, timeout logic
motion.c	ADC sampling, RMS computation, hysteresis
display.c	Radar grid drawing, plotting, fading
serial.c	USB serial telemetry
buzzer.c	Proximity alert logic (optional)

In addition to the project-specific modules listed above, the system incorporates several supporting software components and libraries. Display and touchscreen support files (*TFTMaster.c, ts_lcd.c, touchscreen.c, and glcdfont.c)* were developed and validated in earlier ECE 414 laboratory assignments and provide low-level display initialization, font rendering, touchscreen interfacing, and calibration routines. These modules were reused to ensure reliable display operation and accurate touchscreen calibration.

The project also leverages standard Raspberry Pi Pico SDK libraries, including SPI and PIO support, to interface with the TFT display and other peripherals. Reusing these well-tested libraries enabled robust low-level communication and allowed development effort to focus on higher-level system integration, FSM control, and sensing algorithms rather than reimplementing foundational drivers.

5. Finite State Machine (FSM) Design

The system behavior is formally defined using a **deterministic FSM**, ensuring that all actions occur in a controlled, analyzable sequence.

FSM Design Rationale

- Explicitly models mechanical settling
- Prevents ultrasonic interference and race conditions
- Ensures bounded execution time per state
- Enables graceful recovery from invalid sensor readings

Key FSM States

- INIT_SYSTEM
- SET_SERVO_ANGLE
- WAIT_SERVO_SETTLE
- TRIGGER_ULTRASONIC
- WAIT_ECHO_START
- WAIT_ECHO_END
- PROCESS_DISTANCE
- HANDLE_NO_ECHO
- HANDLE_INVALID_ECHO
- READ_DOPPLER
- LOG_AND_OUTPUT
- UPDATE_DISPLAY
- ADVANCE_ANGLE

Each state performs a bounded, deterministic operation and immediately transitions, guaranteeing non-blocking execution.

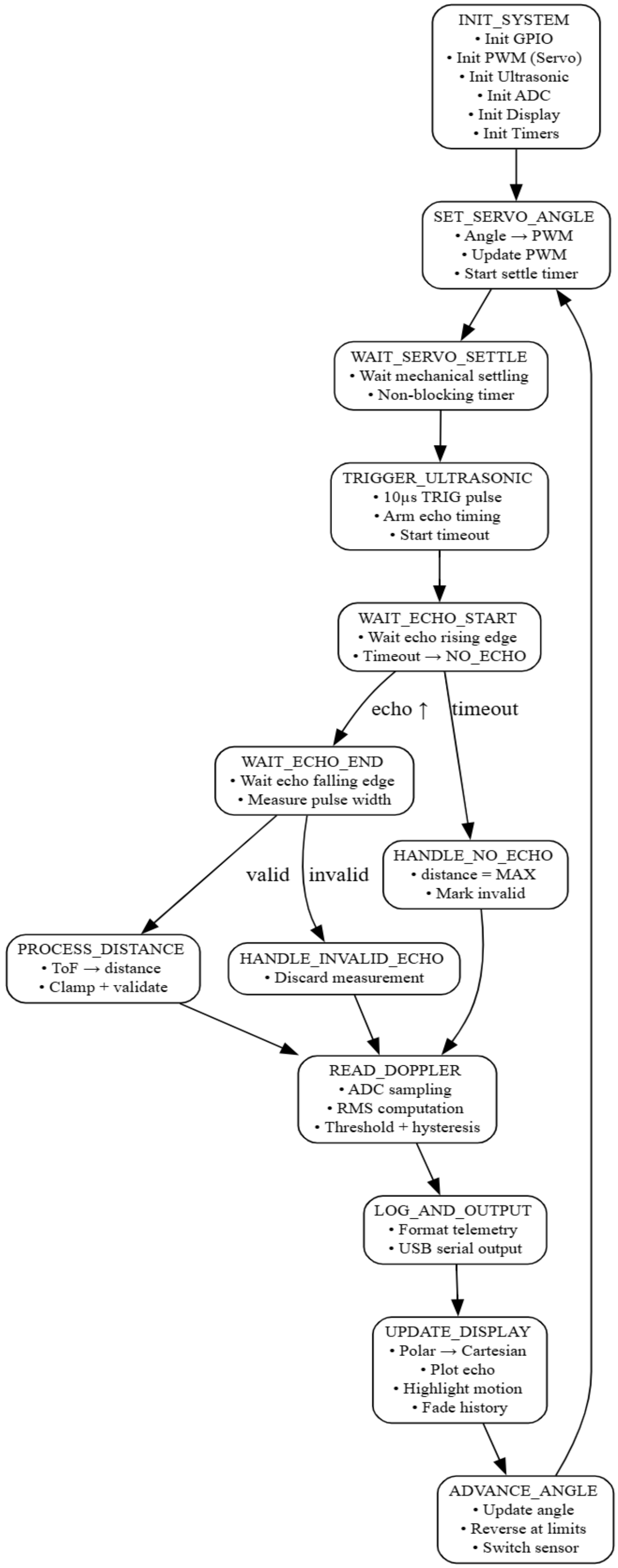
INIT_SYSTEM
• Init GPIO
• Init PWM (Servo)
• Init Ultrasonic
• Init ADC
• Init Display
• Init Timers
SET_SERVO_ANGLE
• Angle → PWM
• Update PWM
• Start settle timer
WAIT_SERVO_SETTLE
• Wait mechanical settling
• Non-blocking timer
TRIGGER_ULTRASONIC
• 10µs TRIG pulse
• Arm echo timing
• Start timeout
WAIT_ECHO_START
• Wait echo rising edge
• Timeout → NO_ECHO
echo ↑
timeout
WAIT_ECHO_END
• Wait echo falling edge
• Measure pulse width
valid
invalid
HANDLE_NO_ECHO
• distance = MAX
• Mark invalid
PROCESS_DISTANCE
• ToF → distance
• Clamp + validate
HANDLE_INVALID_ECHO
• Discard measurement
READ_DOPPLER
• ADC sampling
• RMS computation
• Threshold + hysteresis
LOG_AND_OUTPUT
• Format telemetry
• USB serial output
UPDATE_DISPLAY
• Polar → Cartesian
• Plot echo
• Highlight motion
• Fade history
ADVANCE_ANGLE
• Update angle
• Reverse at limits
• Switch sensor

Figure 6. Finite state machine governing the temporal behavior of the radar system. Each state performs a bounded operation and transitions based on timing conditions or sensor events, ensuring deterministic, non-blocking execution and robust handling of invalid measurements and timeouts.

The system's control logic is formally defined using a finite state machine, shown in Figure 6. This FSM governs the sequencing of servo motion, ultrasonic ranging, Doppler sampling, and output generation, and ensures that all operations execute in a non-blocking and deterministic manner.

6. Signal Processing and Algorithms

6.1 Ultrasonic Distance Measurement

Echo rising and falling edges are captured using GPIO interrupts. The echo pulse width is measured with microsecond resolution and converted to distance:

$$d = \frac{t \cdot v_{\text{sound}}}{2}$$, where $v_{\text{sound}} \approx 343 \text{ m/s at } 20^{\circ}\text{C}$

Timeouts and range checks reject invalid measurements without causing any blocking behavior.

6.2 Doppler Motion Detection

The Doppler IF signal is sampled using the Pico ADC. For each scan angle:

1. A fixed-length sample window is acquired
2. DC offset is removed
3. RMS energy is computed:

$$\boxed{\text{RMS} = \sqrt{E[x^2] - (E[x])^2}}$$

A hysteresis-based threshold prevents false positives due to noise, which was an effort to reduce hardware-related noise through software-wise signal processing measures.

7. Visualization and User Interface

The system provides a real-time radar-style visualization on a TFT display to present spatial and dynamic information in an intuitive and immediately interpretable form. The visualization is updated continuously as the sensors sweep through the environment and reflects the current system state governed by the FSM.

The display includes the following key visual elements:

- **Concentric, properly scaled range rings**, corresponding to fixed distance intervals, providing a spatial reference for interpreting measured distances. Informative text annotations are rendered on the display to label range values and system status.
- **Angular sweep mapping**, where detected targets are plotted according to the current servo angle, forming a polar representation of the environment.
- **Rainbow-fading historical traces**, implemented using a circular buffer, which retain recent measurements and gradually fade over time. This provides temporal context and allows motion trends to be visually inferred.
- **Cross-highlighted moving targets**, where detections associated with an asserted Doppler motion signal are rendered distinctly from stationary objects, enabling immediate differentiation between static and dynamic obstacles.

A circular buffer is used to store a fixed number of recent angle–distance measurements. This approach balances memory usage with visualization clarity and avoids unbounded growth of historical data. Display updates are coordinated by the FSM to ensure that rendering occurs only after valid sensor data has been acquired and processed.

Figure 7. Radar-style TFT display output showing concentric range rings, angular sweep mapping, fading historical traces, and highlighted moving targets. Some limitations are photo resolution and touchscreen display's size.

8. Real-Time Performance Analysis

System performance was evaluated to ensure that all sensing, processing, and visualization tasks meet real-time constraints without blocking or loss of responsiveness. Empirical observation during extended operation confirmed stable and deterministic behavior.

Key performance characteristics are summarized below:

- **Servo update rate:** approximately **20 Hz**, providing smooth angular motion while allowing sufficient settling time for accurate sensing.
- **Ultrasonic timing resolution:** on the order of **microseconds**, achieved through hardware timers and interrupt-driven echo capture.
- **Doppler processing latency:** RMS computation and thresholding complete within a single scan window, ensuring motion detection remains synchronized with angular position.
- **Display refresh behavior:** display updates are synchronized with sensor acquisition cycles, preventing visual tearing or stale data presentation.
- **Serial telemetry throughput:** USB serial output is intentionally throttled to avoid congestion and ensure that logging does not interfere with time-critical operations.

Throughout operation, no blocking delays, missed interrupts, or observable stalls were detected. This confirms that the non-blocking, FSM-driven software architecture successfully maintains real-time responsiveness while coordinating multiple peripherals and processing tasks.

9. Test Plan and Results

9.1 Test Philosophy

Testing validates **functional correctness**, **timing guarantees**, and **robustness**. Each requirement is mapped to explicit test cases with objective pass/fail criteria and documented evidence.

9.2 Comprehensive Test Matrix

Functional Tests

Req ID	Requirement Description	Test ID	Test Procedure	Expected Result	Actual Result	Evidence
R1	Ultrasonic distance accuracy	T1	Place object at known distances (0.5–5 m)	Error ≤ ±5 cm	Pass	Serial logs, live-demo, video
R2	Doppler motion detection	T2	Move object across FOV	Motion flag asserted	Pass	Display output, serial logs, live-demo
R3	Static object rejection	T3	Place stationary object	No motion detected	Pass	Display output, serial logs, live-demo
R4	360° angular coverage	T4	Observe full sweep	Continuous coverage	Pass	Live-demo in class, demo video
R5	Radar visualization correctness	T5	Compare display to physical layout	Correct polar mapping	Pass	Live-demo, video, photos

Timing Tests

Req ID	Requirement Description	Test ID	Test Procedure	Expected Result	Actual Result	Evidence
R6	Servo settling enforcement	T6	Measure delay before trigger	≥ settle time	Pass	Serial logs, timing logs
R7	Non-blocking execution	T7	Continuous sweep observation	No stalls	Pass	Live demo, demo-video, serial logs
R8	Ultrasonic echo timing	T8	Inspect timestamps	µs resolution	Pass	Live demo, Serial output
R9	Display refresh stability	T9	Visual inspection	Smooth update	Pass	Live-demo, video

Hybrid Ultrasonic–Doppler Radar System

Req ID	Requirement Description	Test ID	Test Procedure	Expected Result	Actual Result	Evidence
R10	Serial throughput	T10	Continuous logging	No overflow	Pass	Serial monitor
R11	Low-current/power consumption	T11	Multimeter Measurement	Current in the main power rail does not exceed 500mA	Pass	Multimeter readings during class, which is easily reproducible

Robustness & Error Tests

Req ID	Requirement Description	Test ID	Test Procedure	Expected Result	Actual Result	Evidence
R12	Echo timeout handling	T12	Remove objects	Graceful timeout	Pass	Logs
R13	Invalid echo rejection	T13	Induce reflections	Measurement discarded	Pass	Logs
R14	Doppler noise floor	T14	No motion present	No false positives	Pass	RMS values in serial logs
R15	Rapid motion stress	T15	Fast movement	Stable detection	Pass	Live-demo
R16	Servo boundary handling	T16	Force limits	Clean reversal	Pass	Display
R17	System recovery	T17	Power cycle	Clean restart	Pass	Live-demo

9.3 Representative Test Output and Evidence

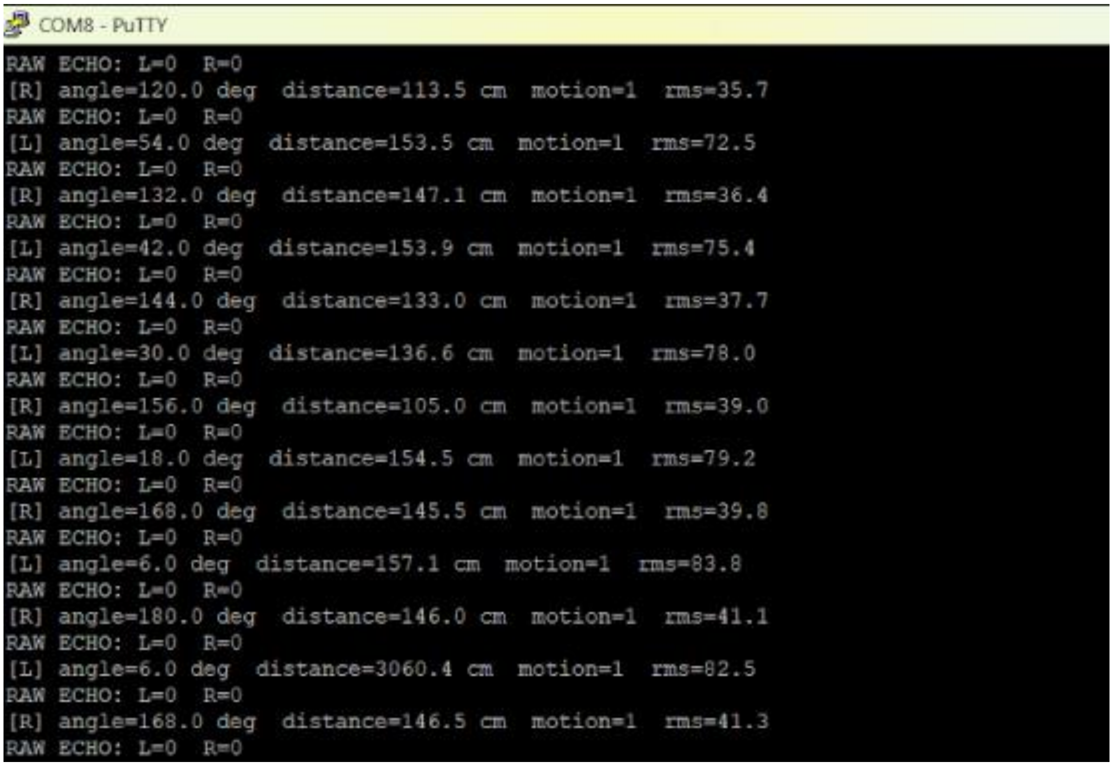

Figure 8. Illustrative PuTTY output format during serial-log based testing processes

In addition to the structured test matrix presented above, representative runtime output and logs were collected during testing to verify correct system behavior under real operating conditions. Figure 8 shows an excerpt of the USB serial output captured using PuTTY during normal operation. This output includes angle, distance, Doppler motion state, and RMS values for successive scan positions.

The data shown in Figure 8 is illustrative rather than exhaustive; similar output was observed consistently across all test cases listed in the test matrix. These logs confirm continuous, non-blocking execution, correct formatting of telemetry data, and stable sensing behavior during full angular sweeps. Complete logs were observed during development and live demonstration and are easily reproducible.

10. Discussion and Analysis

The experimental results confirm that the hybrid ultrasonic–Doppler sensing architecture significantly improves situational awareness compared to single-sensor systems.

Ultrasonic time-of-flight measurements provided accurate and repeatable spatial distance estimates, while the Doppler sensing subsystem reliably identified motion even when ultrasonic readings were noisy, intermittent, or ambiguous. The fusion of these complementary sensing modalities enabled the system to reason about object position and motion state simultaneously in real time.

One of the most significant technical challenges encountered during development was the integration of the Doppler radar subsystem, which initially exhibited severe noise and instability. Early prototypes showed large, motion-independent fluctuations in the Doppler ADC readings, rendering RMS-based motion detection unreliable. Through systematic debugging, this behavior was traced primarily to analog noise coupling from digital components, including servo motors, PWM switching, and SPI display activity.

This issue was mitigated through a combination of physical and architectural design changes, most notably the physical separation of analog and digital circuitry onto separate breadboards. Isolating the Doppler modules and their analog conditioning circuitry from high-current and high-frequency digital components substantially reduced noise coupling into the ADC path. This hardware-level mitigation, combined with RMS energy computation and hysteresis-based thresholding in software, resulted in stable and repeatable motion detection. This experience reinforced the importance of treating analog signal integrity as a first-class design concern, even in primarily digital embedded systems.

The FSM-driven software architecture proved essential for maintaining determinism and robustness throughout system operation. Explicit modeling of servo settling time, ultrasonic trigger and echo capture windows, and Doppler sampling ensured that sensing operations did not interfere with one another. Interrupt-driven echo capture and non-blocking state transitions allowed the system to remain responsive while concurrently handling sensing, display updates, and serial telemetry.

Several important design trade-offs were identified and evaluated. Increasing servo sweep speed improved scan refresh rate but reduced angular resolution and increased susceptibility to mechanical overshoot, while smaller angular step sizes improved spatial fidelity at the cost of latency. Similarly, longer Doppler ADC sampling windows improved noise robustness but reduced responsiveness to rapid motion changes. These trade-offs were resolved by prioritizing deterministic behavior, measurement stability, and visualization clarity over raw update rate, consistent with the objectives of ECE 414.

Despite its overall robustness, the system has inherent limitations. Ultrasonic sensing remains sensitive to surface orientation and environmental conditions, and the Doppler subsystem provides binary motion detection rather than explicit velocity estimation.

Additionally, while no explicit anti-aliasing filter was placed before the ADC, this limitation was mitigated in practice by the low-frequency nature of Doppler IF signals and digital RMS processing.

From a broader perspective, this project highlighted the critical role of hardware–software co-design in embedded systems. Mechanical effects, analog noise, and timing interactions cannot be fully addressed in software alone and must be explicitly accounted for through both physical layout and control logic. Addressing these issues early and systematically proved essential to achieving a robust final system.

11. Conclusion

This project successfully demonstrated the design, implementation, and verification of a real-time hybrid ultrasonic–Doppler radar system on a resource-constrained embedded platform. By integrating ultrasonic time-of-flight distance sensing with microwave Doppler motion detection, the system provides both spatial and dynamic information across a full 360° field of view. All functional and non-functional requirements defined in the project proposal were met, including accurate distance measurement, reliable motion detection, deterministic non-blocking operation, and real-time visualization.

The final system reflects professional embedded-systems design practices, including modular software organization, interrupt-driven I/O, FSM-based control, and careful attention to timing and signal integrity. A key outcome of the project was the successful resolution of significant Doppler noise challenges through a combination of physical hardware separation and software-level RMS processing, highlighting the importance of hardware–software co-design in mixed-signal embedded systems.

Approximately 90-110 hours were devoted to hardware design, circuit debugging, software development, testing, and documentation. The most challenging aspects of the project involved analog signal stability, mechanical settling effects, and real-time scheduling. Addressing these challenges reinforced the value of explicit state modeling, incremental testing, and early validation of hardware assumptions.

The system provides a scalable foundation for future enhancements, including velocity estimation through frequency-domain Doppler analysis, adaptive sensor fusion strategies, improved analog filtering, or replacement of mechanical scanning with electronically steered sensing. For future ECE 414 students, a key lesson from this project is to prioritize deterministic control, module-by-module proper functionality testing before integration, and signal integrity early in the design process, as these considerations strongly influence overall system robustness.

Overall, this project serves as a strong demonstration of real-time embedded perception, combining theoretical principles and practical engineering into a cohesive and reliable system.

References and Appendices

Citation Format

This report follows a **consistent IEEE-style citation format**, which is commonly used in electrical and computer engineering literature. All external hardware documentation, software libraries, and reference designs used during the development of this project are cited accordingly.

Appendix A: Final Code and Reproducibility

The complete source code for the Hybrid Ultrasonic–Doppler Radar System is publicly available in a GitHub repository. The repository includes all required C source files, header files, build scripts, and configuration files necessary to reproduce the project.

The codebase is organized modularly, with each subsystem implemented in a clearly labeled source file. All project-specific modules are well-commented, and variable and function names follow consistent naming conventions. Supporting infrastructure files, including display drivers and SDK libraries, are included as required for compilation and deployment.

A detailed README.md file is provided in the repository and includes:

- Hardware setup instructions
- Pin mappings and wiring descriptions
- Build and flashing instructions using the Raspberry Pi Pico SDK
- Steps to run and demonstrate the system

GitHub Repository (Public):
https://github.com/JackE772/ece414-civisoken-ewing/tree/main/final-project

Rubric Compliance Note:

All required files are included, clearly labeled, and neatly formatted. The project is fully reproducible by following the provided instructions. For your further reference, step-by-step demo instructions are also provided in Appendix D of this report.

Appendix B: Additional Schematics and Diagrams

All system schematics, block diagrams, finite state machine diagrams, and hardware photographs are presented directly in the main body of this report at the points where they are most relevant to the technical discussion. This editorial choice was made to emphasize clarity, context, and immediate interpretability of each figure alongside its corresponding design explanation.

As a result, no additional figures are duplicated in this appendix. The main report should be considered the authoritative source for all design documentation and visual references.

Included materials may consist of:

- Detailed finite state machine (FSM) diagrams
- High-resolution electrical schematics generated using KiCad
- Block diagrams illustrating system-level data flow
- Photographs of the physical hardware setup, including breadboard layouts and sensor assemblies
- PuTTY output example

All flowcharts and FSM diagrams were created using professional diagramming tools (e.g., GraphViz code), ensuring clarity and consistency with standard engineering documentation practices.

Appendix C: External Links

For convenience and reproducibility, all external links referenced in this report are summarized below.

- **GitHub Code Repository:**
  https://github.com/JackE772/ece414-civisoken-ewing/tree/main/final-project

- **Project Demonstration Video:**
  https://drive.google.com/file/d/15QqWR9_1v-_UHlbc8YgBszJ1aU8pvBwp/view?usp=sharing

Appendix D: Step-by-Step Setup and Demonstration Instructions

This appendix provides clear, step-by-step instructions to set up, operate, and demonstrate the Hybrid Ultrasonic–Doppler Radar System in the absence of the project authors. These instructions are intended to allow faculty, students, or visitors to reliably operate the system during demonstrations such as the Lafayette Engineering Open House or classroom exhibitions.

D.1 Required Hardware Setup

Ensure the following hardware components are present and connected according to the GitHub pin definitions and/or electrical schematic (either is acceptable and correct):

- Raspberry Pi Pico (RP2040)
- Two HC-SR04 ultrasonic sensors
- Two HB100 Doppler radar modules with analog conditioning circuitry
- Two hobby servo motors with integrated sensor platforms
- SPI-based TFT display (ILI934x family)
- External 5 V USB power source
- Breadboards and jumper wires

All wiring should follow the **electrical schematic provided in Figure 4** and the **Doppler analog front-end reference shown in Figure 5**. The Doppler radar modules and analog conditioning circuitry must be physically separated from the Pico, servos, and display on a dedicated analog breadboard to ensure stable operation.

D.2 Power and Connection Checklist

Before powering the system:

1. Verify all ground connections are common between the Pico, sensors, and analog circuitry.
2. Confirm that servo motors are powered from the 5 V rail and not directly from the Pico's 3.3 V supply.
3. Ensure Doppler radar outputs are connected to the correct Pico ADC inputs.

4. Confirm SPI connections between the Pico and TFT display (MOSI, SCK, CS, DC, RESET, I0, I1 modes that should connect to 3Vc).

Once verified, power the system via USB.

D.3 Flashing the Firmware

If firmware reprogramming is required:

1. Hold the **BOOTSEL** button on the Raspberry Pi Pico.
2. Connect the Pico to a computer via USB.
3. Release the BOOTSEL button once the Pico appears as a USB mass storage device.
4. Drag and drop the compiled .uf2 firmware file into the Pico drive.
5. The Pico will automatically reboot and begin execution.

Detailed build and compilation instructions are provided in Appendix A and the project **README.md.**

D.4 Normal Operation and Demo Procedure

Once powered and flashed, the system begins operation automatically:

1. The servo-mounted sensor assemblies begin sweeping across their angular ranges.
2. Ultrasonic distance measurements are taken at each angle.
3. Doppler motion detection runs continuously during scanning.
4. The TFT display shows a radar-style visualization with:
    - Concentric range rings
    - Angular sweep mapping
    - Fading historical traces
    - Highlighted moving targets

To demonstrate functionality:

- Move an object slowly in front of the sensors to observe distance changes.

- Move an object laterally to trigger Doppler-based motion highlighting.
- Observe that stationary objects are rendered without motion indication.

D.5 Expected Correct Behavior

During correct operation:

- Servo motion should be smooth and continuous.
- The display should update without flicker or freezing.
- Moving objects should be visually distinguished from static ones.
- Serial output (if connected) should continuously stream angle, distance, motion state, and RMS values.

D.6 Troubleshooting Guidelines

If unexpected behavior occurs:

- **No motion detected:** Ensure Doppler modules are powered and physically isolated from digital components.
- **Erratic distance readings:** Verify ultrasonic sensor alignment and check for reflective surfaces.
- **Display not updating:** Check SPI wiring and reset the Pico.
- **System unresponsive:** Power-cycle the system and allow it to restart.

These steps resolve the vast majority of issues encountered during development and testing.

D.7 Demo Notes for Non-Technical Audiences (Optional)

For demonstrations to visitors:

- Explain that ultrasonic sensing measures distance while Doppler sensing detects motion.
- Highlight that combining both enables better environmental awareness.
- Emphasize real-time operation and embedded constraints.

References:

[1] Raspberry Pi Foundation, *Raspberry Pi Pico SDK Documentation*, 2024.
[Online]. Available: https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf

[2] Raspberry Pi Foundation, *RP2040 Microcontroller Datasheet*, 2024.
[Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

[3] Agilsense (InnoSenT), *HB100 Microwave Motion Sensor Application Note*, 2008.
[Online]. Available: https://openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/HB100_Microwave_Sensor_Application_Note.pdf

[4] SparkFun Electronics, *HC-SR04 Ultrasonic Distance Measurement Module Datasheet*.
[Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[5] Raspberry Pi Foundation, “Raspberry Pi Pico Pinout,” *pico.pinout.xyz*.
[Online]. Available: https://pico.pinout.xyz/. Accessed: Dec. 15, 2025.