Journal of Circuits, Systems, and Computers

Hardware-Software Co-Design of a Dual-Axis Solar Tracker with FPGA-Based Kalman Filtering

--Manuscript Draft--

Manuscript Number:	WSPC-JCSC-D-26-00064
Full Title:	Hardware-Software Co-Design of a Dual-Axis Solar Tracker with FPGA-Based Kalman Filtering
Article Type:	Research Paper
Section/Category:	Design Automation, Simulation and Modelling
Keywords:	FPGA; SoC; Kalman Filter; PID Control; Solar Tracking; Embedded systems; Renewable Energy; Internet of Things; DE1-SoC; Arduino; ESP32
Corresponding Author:	Emad Natsheh An-Najah National University PALESTINIAN TERRITORY, OCCUPIED
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	An-Najah National University
Corresponding Author's Secondary Institution:	
First Author:	Suleiman Abu Kharmeh, Ph.D.
First Author Secondary Information:	
Order of Authors:	Suleiman Abu Kharmeh, Ph.D.
	Emad Natsheh
	Saed Tarapiah, Ph.D.
	Mohammad Matar
	Mohammad Raed
Order of Authors Secondary Information:	
Abstract:	The increasing global demand for renewable energy has highlighted the importance of maximizing the efficiency of photovoltaic systems. This paper presents the design and implementation of an intelligent dual-axis solar tracking system that combines embedded control, wireless communication, and FPGA-based hardware acceleration to improve tracking accuracy and system responsiveness. The proposed architecture consists of three coordinated subsystems. An Arduino microcontroller executes a Proportional–Integral–Derivative (PID) controller to drive servo motors for accurate panel orientation based on calculated solar position. An ESP32 module provides wireless connectivity, enabling real-time monitoring and visualization through a remote dashboard. In parallel, a DE1-SoC FPGA board implements a Kalman filter to perform predictive estimation of the panel position, mitigating the effects of sensor noise and communication delays. Data exchange between the Arduino and FPGA is achieved via a UART communication interface. The Kalman filter is accelerated using the FPGA's dedicated DSP resources, allowing real-time predictive corrections that enhance control stability. Experimental results demonstrate that the integration of FPGA-based predictive filtering with embedded PID control significantly reduces tracking errors and system oscillations compared to microcontroller-only solutions. Additionally, environmental sensors supply contextual data for comprehensive system performance evaluation. The presented system illustrates the effectiveness of hardware–software co-design in renewable energy applications. By integrating predictive control with FPGA acceleration and wireless monitoring, the proposed approach offers improved precision, adaptability, and computational efficiency, making it suitable for both educational platforms and small-scale solar energy systems.

Suggested Reviewers:	

Hardware-Software Co-Design of a Dual-Axis Solar Tracker with FPGA-Based Kalman Filtering

Suleiman Abu Kharmeh (0009-0009-8685-2354)[1], Emad Natsheh (0000-0003-3134-8053)[1*], Saed Tarapiah (0000-0002-8053-6467)[2], Mohammad Matar[1], Mohammad Raed[1]

[1] Department of Computer Engineering, An-Najah National University, Nablus, Palestine
[2] Department of Telecommunication Engineering, An-Najah National University, Nablus, Palestine

*Corresponding Author: e.natsheh@najah.edu

Abstract: The increasing global demand for renewable energy has highlighted the importance of maximizing the efficiency of photovoltaic systems. This paper presents the design and implementation of an intelligent dual-axis solar tracking system that combines embedded control, wireless communication, and FPGA-based hardware acceleration to improve tracking accuracy and system responsiveness.
The proposed architecture consists of three coordinated subsystems. An Arduino microcontroller executes a Proportional–Integral–Derivative (PID) controller to drive servo motors for accurate panel orientation based on calculated solar position. An ESP32 module provides wireless connectivity, enabling real-time monitoring and visualization through a remote dashboard. In parallel, a DE1-SoC FPGA board implements a Kalman filter to perform predictive estimation of the panel position, mitigating the effects of sensor noise and communication delays. Data exchange between the Arduino and FPGA is achieved via a UART communication interface.
The Kalman filter is accelerated using the FPGA's dedicated DSP resources, allowing real-time predictive corrections that enhance control stability. Experimental results demonstrate that the integration of FPGA-based predictive filtering with embedded PID control significantly reduces tracking errors and system oscillations compared to microcontroller-only solutions. Additionally, environmental sensors supply contextual data for comprehensive system performance evaluation.
The presented system illustrates the effectiveness of hardware–software co-design in renewable energy applications. By integrating predictive control with FPGA acceleration and wireless monitoring, the proposed approach offers improved precision, adaptability, and computational efficiency, making it suitable for both educational platforms and small-scale solar energy systems.

Keywords: FPGA, SoC, Kalman Filter, PID Control, Solar Tracking, Embedded Systems, Renewable Energy, Internet of Things, DE1-SoC, Arduino, ESP32

1. Introduction

Solar energy has become one of the most important renewable energy sources worldwide, yet its efficiency strongly depends on how effectively photovoltaic panels are aligned with the sun. Fixed panels and simple single-axis tracking systems are often unable to maintain optimal orientation throughout the day, which leads to noticeable energy losses. Dual-axis solar tracking systems provide a practical solution to this problem, but their performance is still limited by control delays, sensor noise, and mechanical oscillations, especially when implemented using basic microcontroller-based approaches.
In recent years, embedded systems have evolved to include a wide range of computing platforms, each offering different strengths. Microcontrollers are well suited for real-time control tasks, FPGAs provide high-speed parallel computation, and wireless modules enable remote monitoring and system interaction. Combining these platforms in a single system opens new opportunities for improving the accuracy and reliability of solar tracking systems. However, many existing designs rely on a single processing unit and reactive control strategies, which restrict their ability to handle dynamic environmental conditions effectively.
This paper presents the design and implementation of an intelligent dual-axis solar tracking system based on a heterogeneous embedded architecture. The system integrates an Arduino microcontroller,

an ESP32 wireless module, and a DE1-SoC FPGA board, with each platform assigned a specific role. The Arduino is responsible for controlling the servo motors using a Proportional–Integral–Derivative (PID) algorithm [1], ensuring accurate positioning of the solar panel. To enhance the control performance, a Kalman filter is implemented on the FPGA to provide predictive estimates of the panel position, reducing the impact of sensor noise and system delays. Communication between the Arduino and the FPGA is established through a UART interface, allowing continuous and reliable data exchange.
The FPGA implementation of the Kalman filter plays a key role in improving system stability. By executing the filtering algorithm in hardware, the system benefits from deterministic timing and reduced computational load on the microcontroller. This leads to smoother panel motion and lower positioning errors compared to conventional microcontroller-only solutions. In addition, the ESP32 module enables wireless monitoring and visualization of system data, allowing users to observe performance metrics and system status in real time through an IoT-based interface.
The proposed system is designed with modularity and practicality in mind. Each subsystem can be tested and optimized independently, while still contributing to the overall system performance. This approach not only improves flexibility and scalability but also makes the system suitable for educational use, where students can explore advanced topics such as control theory, sensor fusion, hardware acceleration, and embedded system integration using a real working platform.
Overall, this work demonstrates that combining predictive control techniques with FPGA-based hardware acceleration and wireless monitoring can significantly improve the performance of dual-axis solar tracking systems. The presented design offers a cost-effective and adaptable solution that can be extended to larger installations or similar tracking applications, while also serving as a valuable reference for future research in intelligent renewable energy systems.

2. Literature Review

Global energy consumption continues to increase, placing growing pressure on energy systems to become both sustainable and efficient. Solar energy has emerged as a key renewable resource; however, the performance of photovoltaic (PV) systems is strongly influenced by panel orientation relative to the sun. Fixed PV installations suffer from limited exposure during large portions of the day, while single-axis tracking systems only partially address this limitation. Dual-axis solar tracking systems, which continuously adjust both azimuth and elevation angles, have been shown to significantly improve energy yield. Amadi and Gutiérrez [2] reported that dual-axis trackers can generate up to 31.4% more energy than single-axis systems and nearly 67.9% more than fixed installations.
The development of solar tracking systems closely follows trends in embedded system design. Early trackers relied on time-based movement or simple threshold-based light sensing. With the widespread adoption of Arduino platforms, microcontroller-based solar trackers became more accessible, typically using Light Dependent Resistors (LDRs) and basic control logic to guide panel movement [3]. These systems are inexpensive and easy to implement, which explains their popularity in educational and small-scale applications. However, their simplicity often comes at the cost of tracking precision and robustness.
More recently, Internet of Things (IoT) technologies have been integrated into solar tracking systems. ESP32-based designs introduced wireless monitoring, data logging, and remote dashboards, transforming traditional trackers into connected systems. Mustafa [4] presented an ESP32-based dual-axis tracker incorporating PID control and a software-based Kalman filter, reporting approximately 43% energy improvement over fixed panels. While this represents a meaningful step forward, the system still relies on a single processor to handle sensing, filtering, control, and communication tasks simultaneously.
Despite these advances, a clear gap exists in the current literature. Practical systems emphasize affordability and ease of use but are constrained by limited computational resources. In contrast, research-oriented implementations demonstrate advanced estimation and control techniques on high-performance hardware, yet remain unsuitable for deployment due to complexity and cost. Even works that incorporate FPGAs, such as BaBars et al. [5], often treat the FPGA as a monolithic controller,

replacing the microcontroller entirely rather than using it as a specialized computational unit. This approach underutilizes the strengths of heterogeneous architectures.

A. Microcontroller-Based Solar Tracking Systems

Arduino-based solar trackers typically employ four LDR sensors arranged in a quadrant configuration. Analog readings are compared to estimate directional error, and motors are driven to move the panel toward the strongest light source [3]. This method is intuitive and economical, making it ideal for introductory projects. However, the limitations of the ATmega328P—operating at 16 MHz with limited memory—restrict the use of advanced signal processing techniques.

Sensor noise is a persistent issue in outdoor environments, particularly under variable cloud cover or reflective conditions. Implementing sophisticated filtering techniques on Arduino platforms often leads to reduced control loop frequency, directly impacting tracking accuracy. As system complexity increases, designers face unavoidable trade-offs between responsiveness, stability, and feature richness.

ESP32-based systems alleviate some of these limitations by offering higher clock speeds and dual-core processing. However, as demonstrated in [4], combining Kalman filtering, PID control, WiFi communication, and web services on a single processor introduces new constraints. Tasks must be time-sliced, and computationally intensive algorithms compete with communication and interface services. This results in increased latency, reduced determinism, and occasional instability during high system load.

B. Kalman Filtering in Solar Tracking Applications

The Kalman filter remains one of the most effective tools for optimal state estimation in systems affected by noise and uncertainty. Originally introduced by Kálmán in 1960, it provides a recursive solution that minimizes estimation error under linear system assumptions and Gaussian noise conditions [6]. Unlike simple averaging or threshold-based approaches, the Kalman filter dynamically balances prediction and measurement based on estimated uncertainty.

In solar tracking applications, this behavior is particularly advantageous. LDR sensors generate noisy and fluctuating signals, especially during transient weather conditions. Kalman filtering allows the system to distinguish between genuine sun movement and random measurement noise, resulting in smoother position estimates and more stable motor control. During brief periods of sensor unreliability, such as cloud cover, the filter naturally relies more on prediction, preventing abrupt or unnecessary panel movements.

Software-based Kalman filters have been successfully implemented on microcontrollers, as shown in [4]. However, these implementations highlight a recurring challenge: the computational cost of matrix operations. Even low-dimensional filters require multiple multiplications and additions per cycle, all executed sequentially on general-purpose processors. As a result, filter complexity, update rate, or numerical precision must often be compromised to maintain real-time performance.

C. FPGA-Based Hardware Acceleration

Field-Programmable Gate Arrays (FPGAs) offer a fundamentally different execution model compared to microcontrollers. Rather than executing instructions sequentially, FPGAs implement algorithms directly in hardware using parallel and pipelined logic structures. This makes them especially suitable for digital signal processing tasks involving repeated arithmetic operations [7].

Kalman filtering benefits greatly from this architecture. Matrix multiplications and additions that execute sequentially on CPUs can be performed simultaneously on FPGAs. Once designed, the filter executes in deterministic time, unaffected by interrupts, operating systems, or task scheduling. Linares-Barranco et al. [8] demonstrated that FPGA-based filtering achieved latencies of 4–4.2 ms, compared to 8–24 ms on CPU implementations, representing performance improvements of up to 570%.

More advanced variants, such as the Unscented Kalman Filter, have also been successfully deployed on FPGA platforms. AlShabi and Bonny [9] showed that algorithms impractical for microcontrollers could operate in real time when implemented in hardware. These results clearly indicate the suitability of FPGAs for computationally intensive estimation tasks.

Nevertheless, FPGAs are not ideal for all system functions. They require specialized design expertise, longer development cycles, and higher costs. For basic tasks such as sensor interfacing, motor control, and simple logic, microcontrollers remain more practical. This observation suggests that performance is maximized not by replacing microcontrollers with FPGAs, but by combining them strategically.

A review of existing solar tracking systems reveals a strong polarization. On one side are accessible, low-cost systems based on a single microcontroller, which struggle with performance limitations as complexity increases. On the other side are research prototypes that demonstrate advanced algorithms on FPGA platforms but lack practical integration and usability.

Very few systems adopt a heterogeneous architecture where an FPGA is used explicitly as a computational co-processor. BaBars et al. [5, 10] represent partial attempts; however, their FPGA-centric designs assign most system tasks to the FPGA, negating the benefits of task specialization.

The literature therefore reveals a clear opportunity: a practical solar tracking system that combines microcontrollers for sensing and control, an FPGA for hardware-accelerated filtering, and a wireless module for monitoring and user interaction. Such an architecture can deliver research-grade estimation performance while maintaining accessibility and cost-effectiveness. Table 1 provides a comparison of existing solar tracking systems, showing differences in computing platforms, estimation techniques, and practical limitations.

Table 1. Overview of Previous Solar Tracking Approaches

Study	Platform	Filtering Method	Connectivity	Key Limitation
[3]	Arduino	None / Basic averaging	No	Sensitive to noise, limited accuracy
[4]	ESP32	Software-based control / basic filtering	WiFi	High CPU load, timing trade-offs
[5]	FPGA-based	Hardware filtering	IoT	Monolithic FPGA design with limited modularity
[8]	FPGA	Event-based hardware filtering	No	Not application-integrated
[10]	FPGA-based	Hardware filtering / ANN forecasting	No	FPGA-centric processing without heterogeneous task partitioning
[11]	ESP32	Software-based control (PID)	WiFi	Processing and communication overhead
[12]	Arduino	Basic averaging / None	No	Sensitive to noise, limited accuracy
[13]	Arduino Uno	None / Basic	No	Limited precision in tracking, simple control
[14]	ATmega32	None / Basic	No	Only local control, low scalability
[15]	General dual-axis model	None / Simulation	N/A	Simulation only, not real-time
[16]	Various platforms (review)	Varies (software/hardware)	Varies	Depends on system; general review
[17]	ESP32	Software-based processing	WiFi	Limited to IoT monitoring; small-scale implementation
[18]	Arduino Uno	LDR feedback	IoT	Limited advanced control/filtering
[19]	Arduino ATmega2560	LDR sensor control	WiFi	Focused on pump load, not pure solar performance
[20]	Arduino + NodeMCU	LDR + closed-loop tracking	IoT	Simple control, small-scale
[21]	Arduino Uno + ESP8266	LDR sensing	IoT	Limited advanced processing/filtering
[22]	Arduino + NodeMCU	LDR feedback	IoT	Basic control, simple motors
[23]	Arduino	Voltage-based sensor control	No	No IoT built-in; basic connectivity

The literature consistently shows that neither simple microcontroller-only designs nor FPGA-only research prototypes fully address the needs of modern solar tracking systems. The absence of practical heterogeneous designs represents a significant gap. By aligning each task with the hardware best suited to it, it becomes possible to overcome the traditional trade-off between simplicity and performance. This insight directly motivates the architecture proposed in this work.

3. Methodology

3.1. Overall System Architecture and Block Diagram

The proposed solar tracking system employs a heterogeneous architecture that strategically distributes computational tasks across three specialized hardware platforms: Arduino Mega for sensor acquisition and actuation, DE1-SoC FPGA for hardware-accelerated Kalman filtering, and ESP32 for wireless connectivity and web-based monitoring. This architecture is designed to overcome the fundamental trade-offs that limit monolithic microcontroller implementations while maintaining practical deployability.

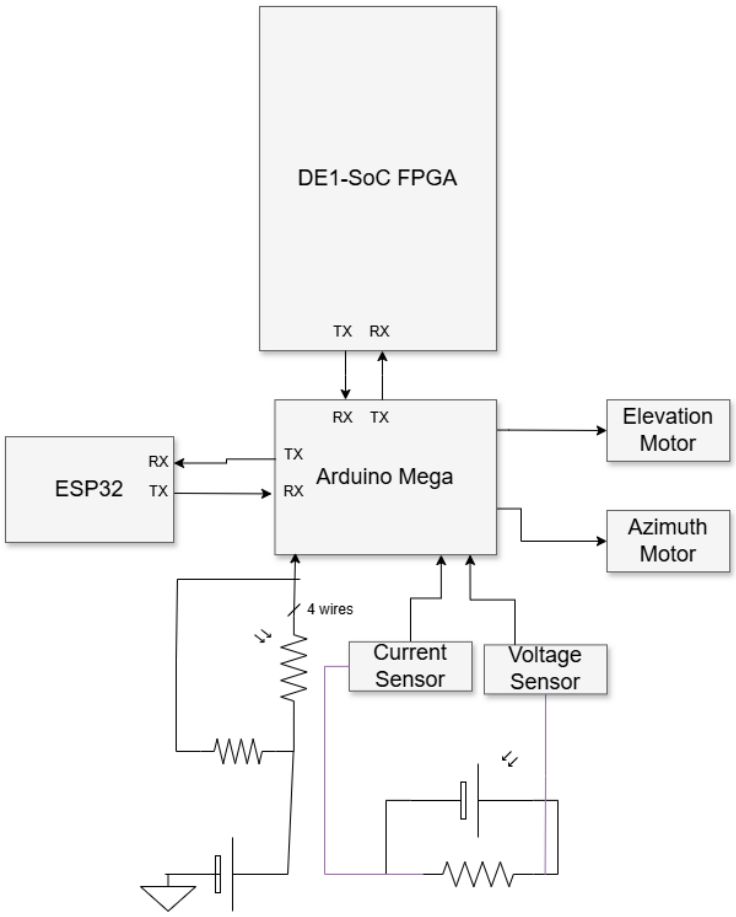


Figure 1. System-level block diagram showing the heterogeneous data flow between Arduino (sensor/actuation layer), DE1-SoC FPGA (computation layer), and ESP32 (communication layer)

Figure 1 illustrates the complete system architecture and the strategic division of labor among the three computational platforms. The data flow follows a carefully orchestrated pipeline designed to maximize the strengths of each component while minimizing intercomponent communication overhead.

The Arduino Mega acts as the front-end controller for sensor acquisition and motor actuation. It interfaces with four light dependent resistors arranged in a quadrant configuration to measure sunlight

intensity and estimate panel orientation. Based on these measurements, the Arduino executes a PID control algorithm to compute azimuth and elevation angle commands. Instead of directly applying these commands to the servo motors, the raw angle estimates are transmitted to the FPGA for filtering. This design choice allows the Arduino to maintain a fast and deterministic control loop while avoiding the computational burden of advanced estimation algorithms.

The DE1-SoC FPGA serves as the computational core of the system. It implements two Kalman filters, corresponding to the azimuth and elevation axes, using parallel hardware structures. By executing the filtering process in hardware, the system achieves real-time, predictable performance and significantly reduces the impact of sensor noise and measurement delays. The filtered angle estimates are returned to the Arduino, enabling stable closed-loop control with improved tracking accuracy. The FPGA's embedded processor is used primarily for system initialization and provides flexibility for future system extensions.

Wireless communication and user interaction are handled by the ESP32 module. It receives system telemetry from the Arduino, including sensor readings, panel angles, and system status information, and provides real-time visualization and control through a web-based interface. By assigning all communication and interface tasks to the ESP32, the system avoids performance degradation commonly observed in single-processor designs.

This heterogeneous architecture combines the strengths of each computing platform. The Arduino provides reliable real-time control and hardware interfacing, the FPGA delivers high-performance predictive filtering, and the ESP32 enables seamless wireless monitoring. The resulting system is modular, scalable, and cost-effective, demonstrating a practical approach to integrating advanced signal processing techniques into embedded renewable energy applications.

3.2. Hardware Subsystem Design

A. Sensor and Actuation Layer (Arduino Domain)

The Arduino Mega 2560 microcontroller serves as the sensor acquisition and motor control platform, chosen for its abundant I/O capabilities, mature ecosystem, and deterministic real-time execution characteristics. The board provides 16 analog input channels with 10-bit resolution (0-1023 ADC range), sufficient for interfacing with Light Dependent Resistor (LDR) sensors that exhibit resistance variations from approximately 1 kΩ in bright light to over 10 MΩ in darkness.

Figure 2 details the Arduino's interface connections. Four LDR sensors are connected to analog inputs A0 (top-left), A1 (top-right), A2 (bottom-left), and A3 (bottom-right), each configured in a voltage divider network with a 10 kΩ fixed resistor to ground. This arrangement produces a voltage output proportional to light intensity. Two servo motors—one for azimuth (horizontal) rotation and one for elevation (vertical) tilt—are controlled via PWM signals on digital pins 3 and 4, generating standard 50 Hz signals with pulse widths from 1000 to 2000 microseconds for 0° to 180° angular range.

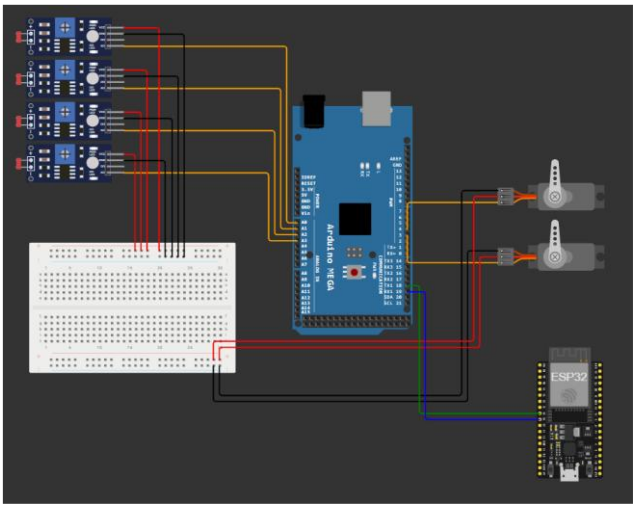


Figure 2. Arduino Mega connection diagram showing LDR sensor interfacing, servo motor control, and UART communication links to FPGA and ESP32

The Arduino Mega manages sensor acquisition and motor control while coordinating communication with both the FPGA and the ESP32. Two independent serial links are used to separate real-time computation from monitoring tasks. Raw azimuth and elevation estimates generated by the PID controller are transmitted to the FPGA for Kalman filtering, while system telemetry is sent to the ESP32 for wireless visualization.

The Arduino executes independent PID controllers for the azimuth and elevation axes based on differential readings from quadrant-arranged LDR sensors. Rather than directly commanding the servo motors using these raw estimates, the values are forwarded to the FPGA, where predictive filtering is applied. The filtered angles are then returned to the Arduino and used for closed-loop control. If filtered data is unavailable, the controller seamlessly falls back to raw PID output, ensuring continuous and robust operation.

This approach preserves deterministic control timing on the Arduino by offloading computationally intensive filtering tasks to dedicated hardware. Additional control refinements, including deadband handling, integral windup protection, and derivative smoothing, are implemented to reduce jitter and mechanical stress.

The system supports both automatic and manual operating modes. In automatic mode, the tracking process is fully autonomous, while manual mode allows direct user control via the wireless interface. Smooth transitions between modes are ensured by synchronizing filter state with the current panel position.

The DE1-SoC FPGA serves as a dedicated computation layer for real-time Kalman filtering. Two parallel filter instances are implemented in hardware to independently process azimuth and elevation data. By executing the filtering algorithm in the FPGA fabric, the system achieves predictable, low-latency estimation unaffected by control or communication workloads.

The FPGA operates primarily as a specialized co-processor rather than a system controller. An embedded processor subsystem is used for configuration and supervisory tasks, enabling future expansion without affecting the real-time data path. This separation allows advanced signal processing to be integrated into the system while maintaining simplicity and reliability at the control level.

Figure 3 illustrates the high-level architecture of the Cyclone V SoC platform used in this work. The device integrates a large FPGA fabric with a hard processor system (HPS), enabling a tight coupling between real-time hardware acceleration and software-based supervision.

The FPGA fabric occupies the majority of the device and hosts the time-critical processing components, including the dual Kalman filter cores, the on-chip Avalon memory-mapped interconnect, the UART communication interface, and custom interface logic. In parallel, the HPS subsystem incorporates a dual-core ARM Cortex-A9 processor, on-chip peripherals, and memory controllers for external DDR3 SDRAM, providing a flexible software environment for configuration, monitoring, and future system extensions.

Communication between the HPS and FPGA fabric is achieved through lightweight AXI bridges, which provide low-latency, high-bandwidth access to memory-mapped resources in the programmable logic. These bridges allow the processor to configure the FPGA during initialization and to monitor internal signals during runtime with performance comparable to on-chip peripheral access.

Within the FPGA fabric, several functional blocks cooperate to implement the real-time signal processing pipeline. Two parallel Kalman filter cores independently process the azimuth and elevation angles, implementing the standard prediction and update stages of the Kalman algorithm using fixed-point arithmetic. This hardware-based implementation ensures deterministic latency and stable performance under varying sensor noise conditions.

A custom interface module connects the Kalman filters to the UART subsystem, coordinating data reception, filter execution, and transmission of the filtered results. This module incorporates basic fault-handling mechanisms to prevent control-loop interruption in the event of abnormal computation delays. The Avalon memory-mapped interconnect provides a standardized communication backbone among all FPGA components, simplifying integration and ensuring scalability.

Serial communication with the external controller is handled by a UART peripheral integrated into the Avalon bus. Small internal buffers decouple communication timing from computation, reducing real-time constraints on the surrounding logic. Additionally, parallel I/O interfaces expose both raw and filtered angle values to the HPS, enabling non-intrusive monitoring and providing a foundation for

future enhancements such as on-board data logging, web-based visualization, or higher-level hybrid control strategies.

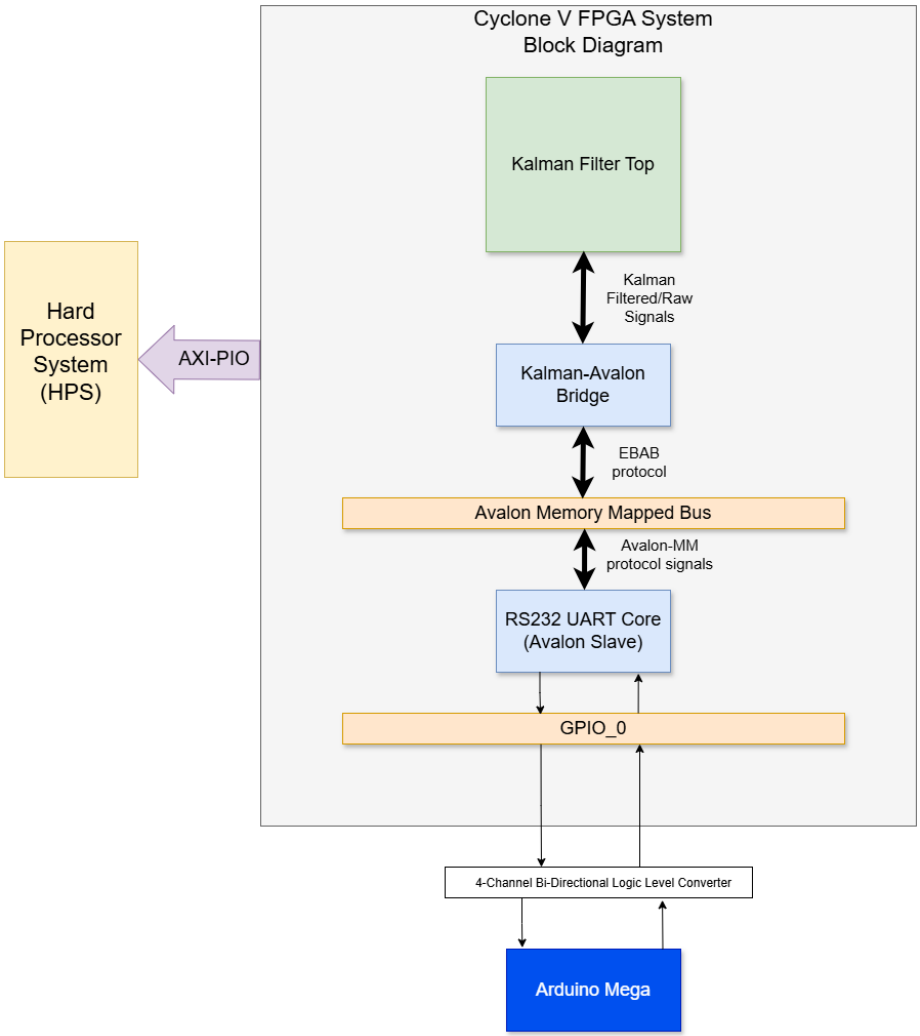


Figure 3. Cyclone V SoC top-level block diagram showing the FPGA fabric, HPS subsystem, and their interconnections via high-performance AXI bridges

B. Adaptive Kalman Filter Hardware Implementation

The solar tracking system employs an adaptive scalar Kalman filter as its core estimation mechanism. A scalar formulation is selected instead of a multi-dimensional filter because azimuth and elevation angles evolve independently and can be treated as two separate estimation problems. This simplification significantly reduces hardware complexity while maintaining optimal estimation performance.

Each filter follows the standard discrete Kalman equations but incorporates an adaptive measurement noise strategy tailored to solar tracking conditions. In conventional fixed-parameter Kalman filters, the measurement noise covariance represents a trade-off between responsiveness and noise suppression. In outdoor solar tracking environments, this trade-off is particularly problematic due to transient disturbances caused by clouds, reflections, and sensor artifacts.

To address this limitation, the proposed filter dynamically adjusts the measurement noise covariance based on the innovation magnitude, enabling the filter to distinguish between gradual, physically plausible sun motion and abrupt noise-induced deviations.

The adaptive mechanism operates on the innovation signal, defined as the difference between the measured angle and the predicted angle. Under normal clear-sky conditions, innovation values remain

small, reflecting sensor noise and minor disturbances. In contrast, cloud shadows and reflections produce sudden, large innovation spikes that do not correspond to actual sun position changes.

The proposed strategy employs a two-tier adaptation mechanism. Moderate innovation deviations are handled using smooth quadratic scaling of the measurement noise covariance, allowing the filter to remain responsive while gradually suppressing unreliable measurements. For extreme deviations exceeding a predefined threshold, a hard rejection mode is activated, dramatically increasing the measurement noise covariance and effectively forcing the Kalman gain toward zero. In this mode, the filter temporarily ignores the measurement and relies on its prediction.

This approach exploits the temporal behavior of disturbances: noise spikes are transient, whereas legitimate position changes persist across multiple samples. By combining smooth adaptation with hard rejection, the filter achieves robust performance across a wide range of operating conditions, from clear skies to highly disturbed environments.

To ensure efficient FPGA implementation, the Kalman filter is realized entirely using fixed-point arithmetic. The angular range of the application is limited to 0°–180°, and sub-degree precision is sufficient given the mechanical resolution of typical servo motors. Floating-point arithmetic is therefore unnecessary and would impose excessive resource and latency overhead.

All computations are performed using a signed Q9.7 fixed-point format [24]. This representation provides adequate dynamic range and resolution while enabling efficient use of FPGA DSP blocks for multiplication. Additions and subtractions are implemented as combinational logic with saturation protection, while division—required only for Kalman gain computation—is implemented using a pipelined restoring divider.

The selected format ensures that quantization noise remains negligible relative to sensor noise and mechanical inaccuracies, while significantly reducing logic utilization and power consumption.

The top-level hardware architecture of the Kalman filter subsystem is illustrated in Figure 4. The design instantiates two independent scalar Kalman filter cores operating in parallel—one for azimuth estimation and one for elevation estimation. Because the two axes are mechanically and mathematically independent, this spatial parallelism eliminates the need for time-multiplexing and allows both estimates to converge simultaneously.

Each filter core receives raw angle measurements and noise parameters and produces filtered angle outputs along with status signals. This parallel architecture fully exploits the FPGA's ability to execute independent computations concurrently, improving throughput and simplifying control logic.

The sequential operation of each Kalman filter core is governed by a finite state machine, shown in Figure 5**.** The FSM implements a clear separation between control and data path, improving modularity and verification.

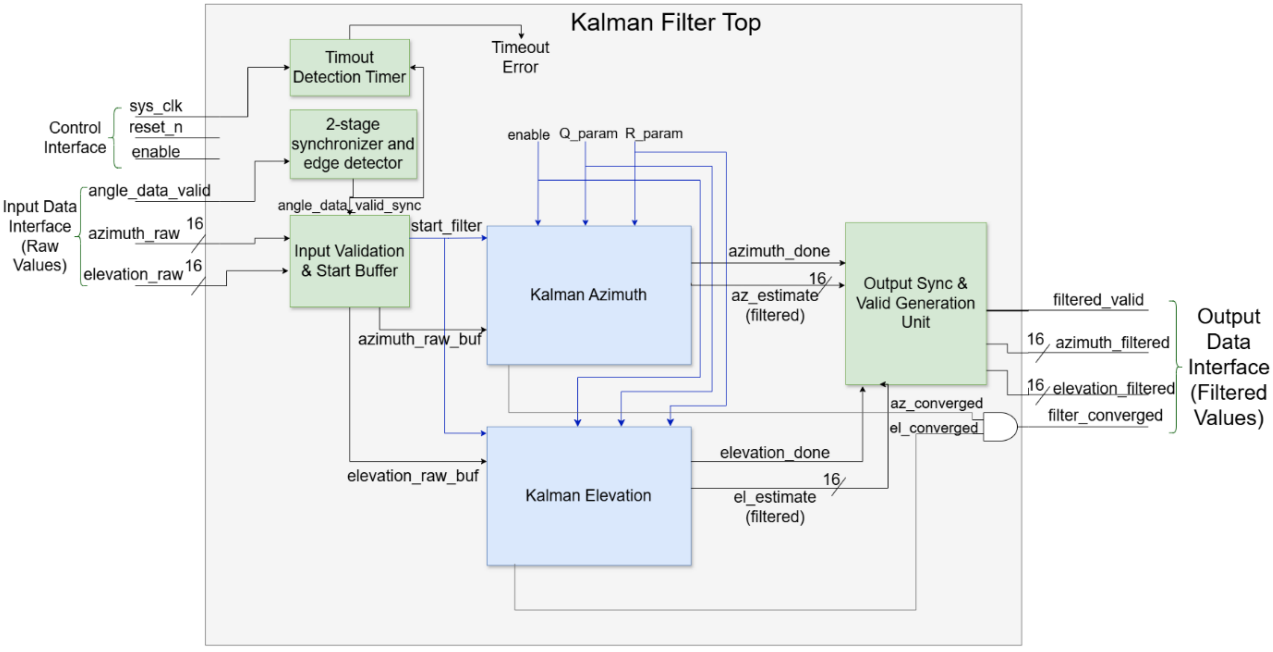


Figure 4. Kalman filter top-level RTL block diagram showing dual independent filter instances and external interface signals

The FSM transitions through states corresponding to the Kalman algorithm: prediction, adaptive noise evaluation, Kalman gain computation, state update, and output generation. While these stages are executed sequentially at the control level, arithmetic operations within each stage are performed in parallel whenever data dependencies allow.

The most latency-intensive operation is the Kalman gain computation, which requires division. This operation is handled by a pipelined divider, and the FSM remains in a wait state until the result becomes available. Despite this constraint, the total processing latency remains low, enabling real-time operation well beyond the system's sampling requirements.

Within each FSM state, the data path exploits fine-grained parallelism. For example, during the prediction stage, the state estimate propagation, covariance update, and innovation computation occur simultaneously. Dedicated arithmetic modules are instantiated for saturating addition and subtraction, fixed-point multiplication, and division.

This combination of sequential control and parallel data path execution allows the filter to achieve high throughput while maintaining deterministic timing. The total latency from measurement input to filtered output is approximately 25 clock cycles at 50 MHz, corresponding to sub-microsecond processing time.

The Kalman filter cores operate using simple handshake signals, whereas the broader system communicates using the Avalon Memory-Mapped protocol. To bridge these domains, a dedicated Kalman-to-Avalon interface module is implemented.

The Avalon-MM timing behavior is illustrated in Figure 6**,** while the bridge control state machine is shown in Figure 7. The bridge handles UART polling, data assembly, filter triggering, and result transmission without exposing protocol complexity to the filter cores.

This modular design enables seamless integration of custom signal processing logic into the larger FPGA system while maintaining clean interfaces and robust flow control.

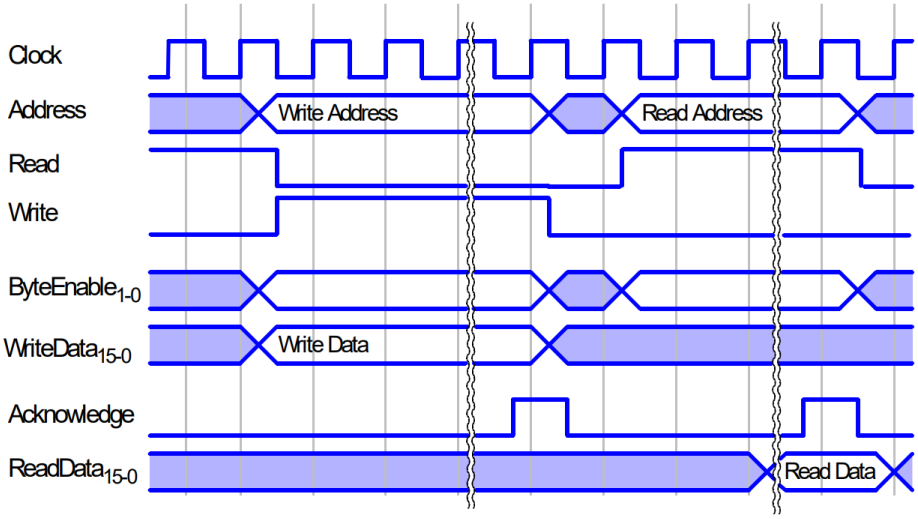


Figure 6. External bus to Avalon bridge timing diagram showing the standard Avalon-MM read and write cycles (adapted from Intel University Program IP documentation)

The Kalman filter design is validated through comprehensive RTL simulation using realistic solar tracking scenarios, including gradual motion, injected noise spikes, and legitimate sudden position changes. Performance is evaluated using RMSE, convergence time, and spike rejection metrics.

Simulation results are post-processed using Python-based analysis to examine error distributions and transient behavior. The adaptive filter consistently achieves approximately 70% noise reduction while rejecting the majority of injected outliers and converging to legitimate changes within a few samples.

Full system integration testing confirms that the filtered output significantly reduces servo jitter and improves tracking stability compared to unfiltered operation, validating the effectiveness of the proposed approach under real-world conditions.

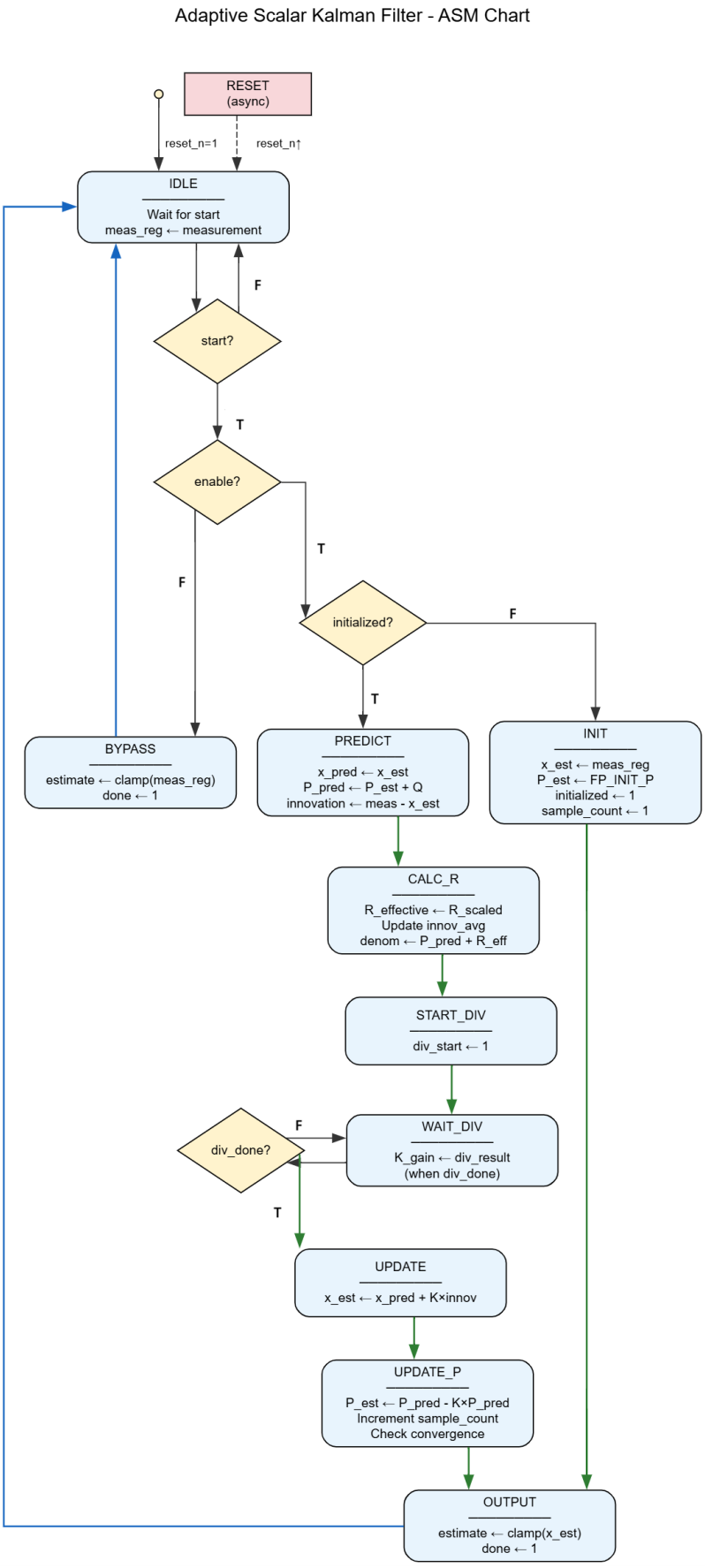


Figure 5. Kalman filter finite state machine showing the sequential control flow through predict, update, and output phases

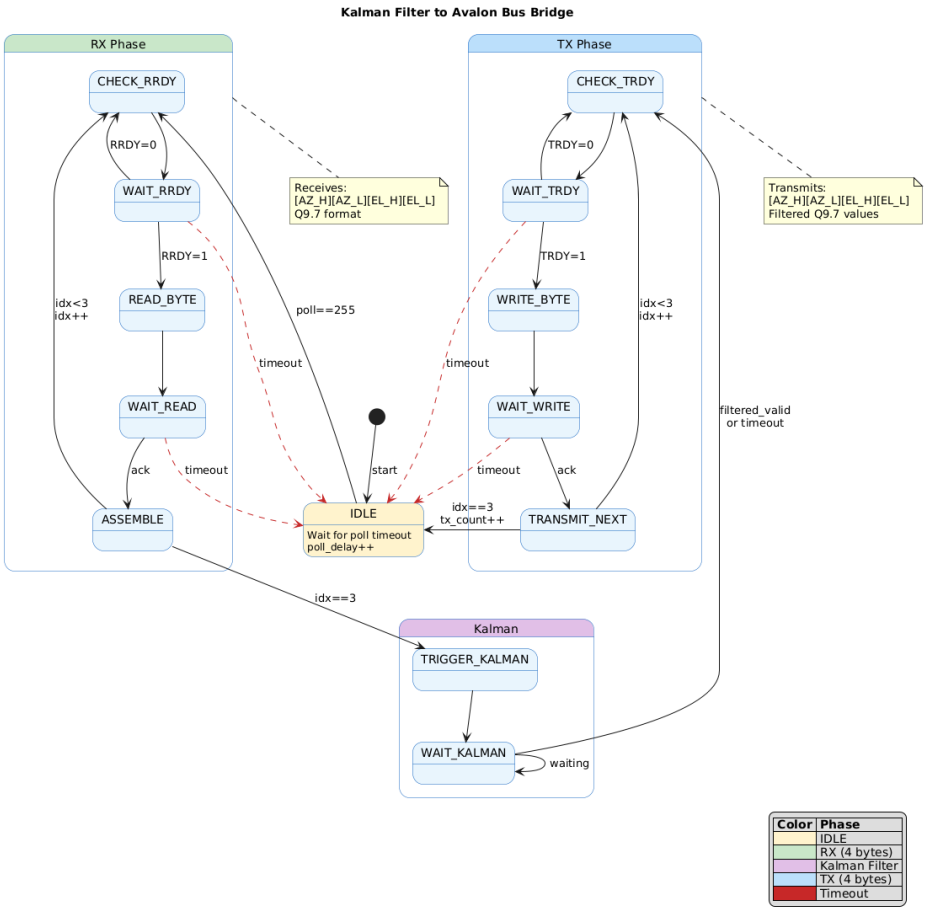


Figure 7. Kalman-to-Avalon bridge state machine showing UART polling, data assembly, filter triggering, and result transmission phases

C. Physical Prototype Implementation

The complete hardware implementation of the proposed tracking system is shown in Figure 8. The prototype integrates the dual-axis servo mechanism, LDR sensor array, Arduino-based sensing unit, FPGA filtering module, and power electronics into a single operational platform used for experimental validation.

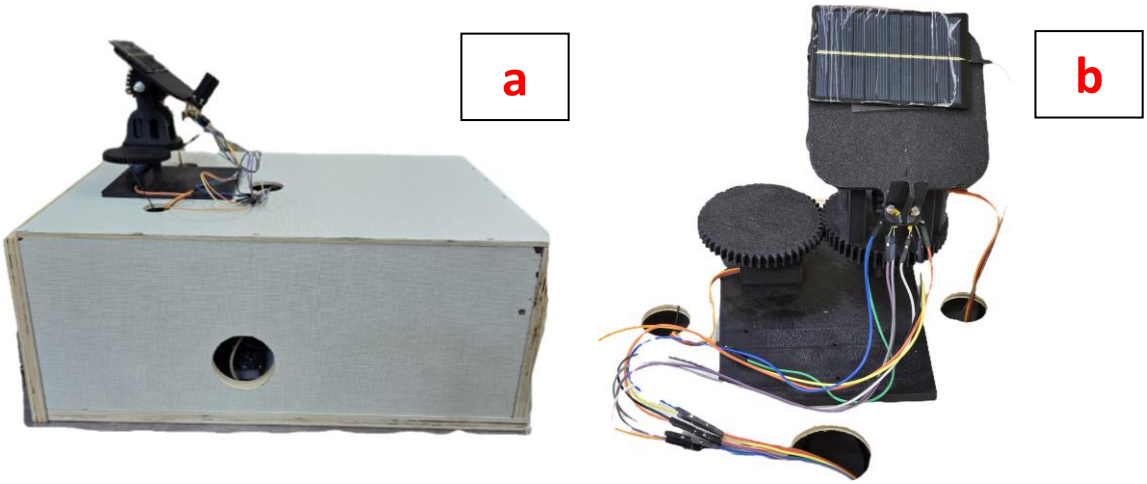


Figure 8. Fully assembled hardware prototype (a) overall system with servos and electronics (b) front view showing solar panel, LDRs, and wiring.

4. Results and Discussion

The FPGA implementation was synthesized using Intel Quartus Prime 18.1 targeting the Cyclone V SoC (5CSEMA5F31C6) on the DE1-SoC board.
The adaptive scalar Kalman filter was evaluated through comprehensive simulation using seven test scenarios with 5,000 samples each (approximately 8 minutes at 10 Hz). Simulations used Synopsys VCS with a SystemVerilog testbench generating realistic trajectories with calibrated noise and spikes. The tracking performance is illustrated in Figure 9, showing the difference between raw and filtered angles, and in Figure 10, which highlights normal trajectory tracking with 2° Gaussian noise.

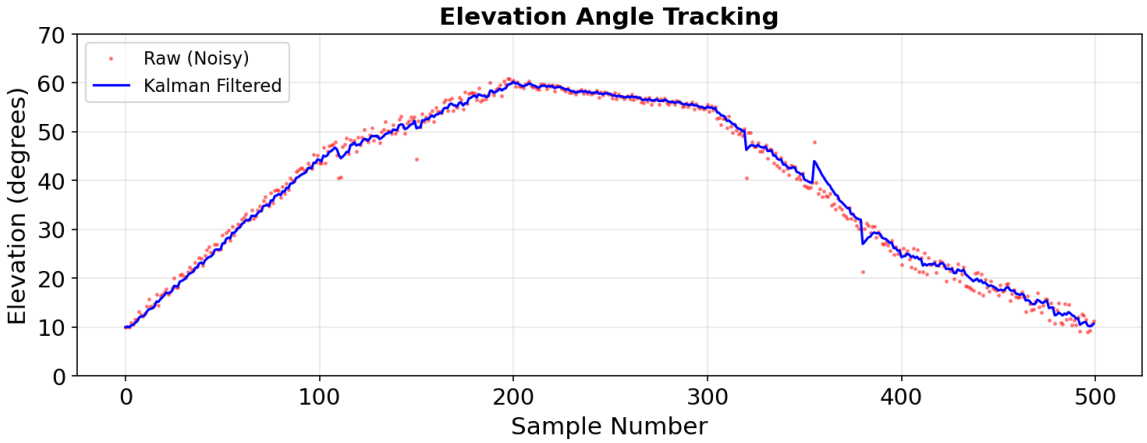


Figure 9. Tracking performance for trajectory with raw vs filtered angles

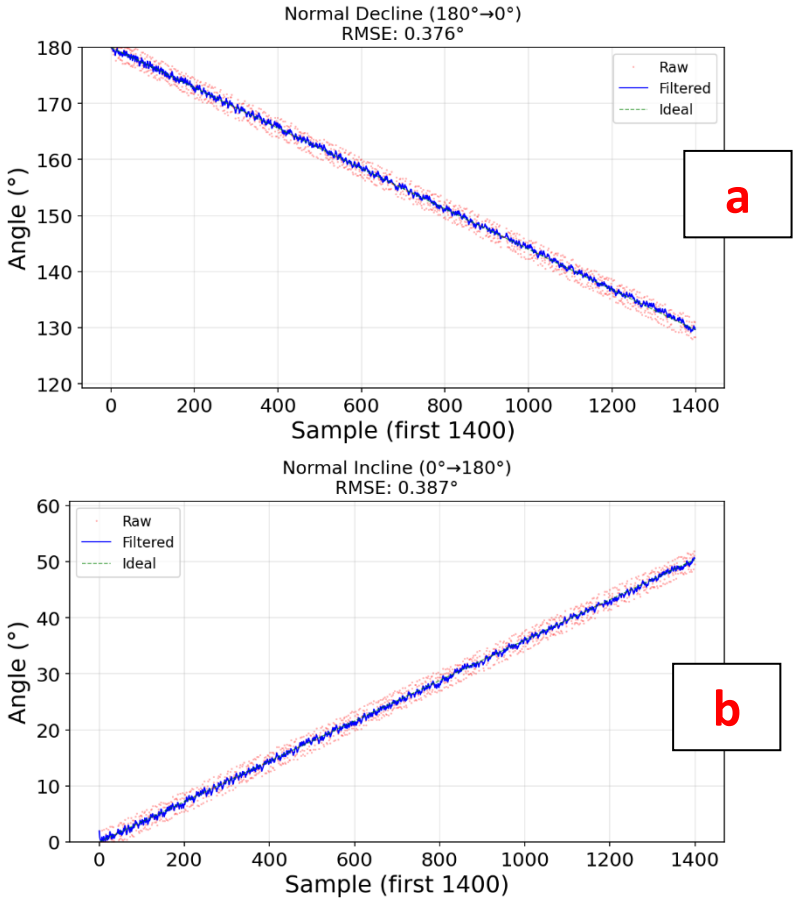


Figure 10. Normal trajectory tracking with 2° Gaussian noise (a) Normal Decline (180º → 0º) (b) Normal Incline (0º →180º)

The effect of the filter under normal operating conditions with 2° Gaussian noise is shown in Figure 10 for both incline and decline motion. In the incline case, the filtered trajectory follows the increasing angle closely, with a clear reduction in high-frequency fluctuations relative to the raw measurements. A similar behavior is observed for the decline trajectory, where the filtered output remains well aligned with the decreasing reference. The corresponding RMSE values decrease from 1.16° to 0.39° for incline and from 1.15° to 0.38° for decline, corresponding to noise reductions of 66.7% and 67.3%, respectively. The close agreement between these results indicates that filter performance is effectively independent of motion direction.

Figure 11 illustrates filter performance under more challenging conditions, where the measurement noise is increased to 4° and augmented with random large spikes occurring at a 3% rate. In this case, the raw signal contains frequent large deviations, with occasional peak errors exceeding 6°. Despite this, the filtered output remains stable and continues to track the underlying trajectory with limited deviation. The resulting error reduction reaches 70.0% for incline and 73.5% for decline, indicating that the filter retains strong robustness even when both noise variance and impulsive disturbances are present.

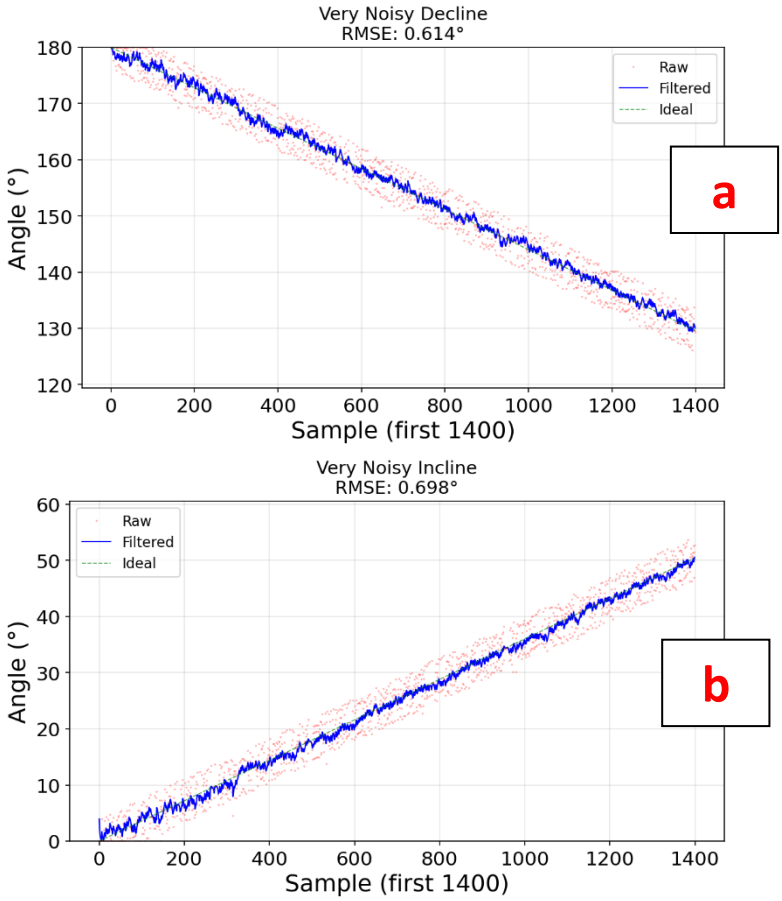


Figure 11. Performance with 4° noise plus random large spikes (3% occurrence rate) (a) Very Noisy Decline (b) Very Noisy Incline

The ability of the filter to suppress isolated spikes is examined further in Figure 12, which includes deliberately injected sharp disturbances ranging from 20° to 45°. While these spikes dominate the raw signal, their impact on the filtered output is substantially reduced.

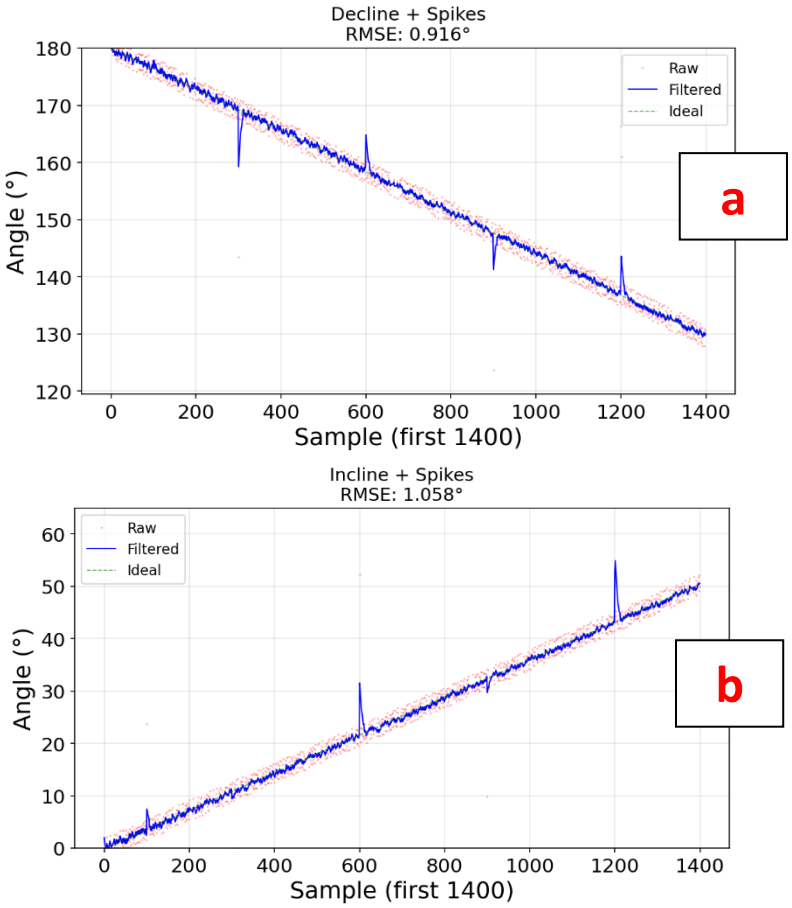
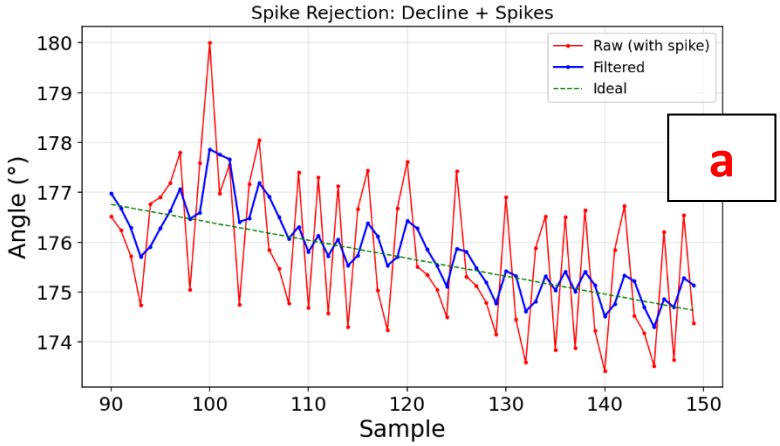


Figure 12. Trajectory tracking with deliberate sharp spikes (20-45°) (a) Decline (b) Incline

This behavior is more clearly visible in the close-up view provided in Figure 13. Even the largest injected spike is attenuated to less than an 18° deviation, and the filter recovers to the nominal trajectory within a small number of samples. Under these conditions, the RMSE is reduced by 51.2% for incline and 57.9% for decline, reflecting a balance between spike suppression and responsiveness.



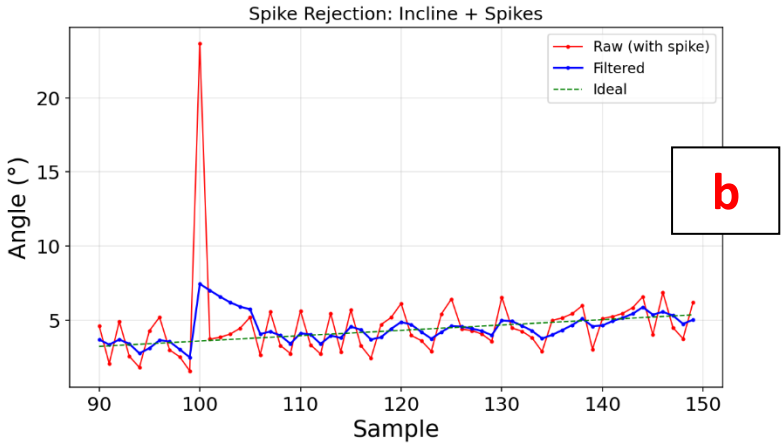


Figure 13. Close-up showing spike attenuation (a) Decline (b) Incline

The response to legitimate sudden changes in angle is shown in Figure 14. These changes consist of several abrupt but sustained jumps ranging from 20° to 60°. Unlike transient spikes, these events are correctly followed by the filter, which adapts to the new angle level rather than rejecting the change.

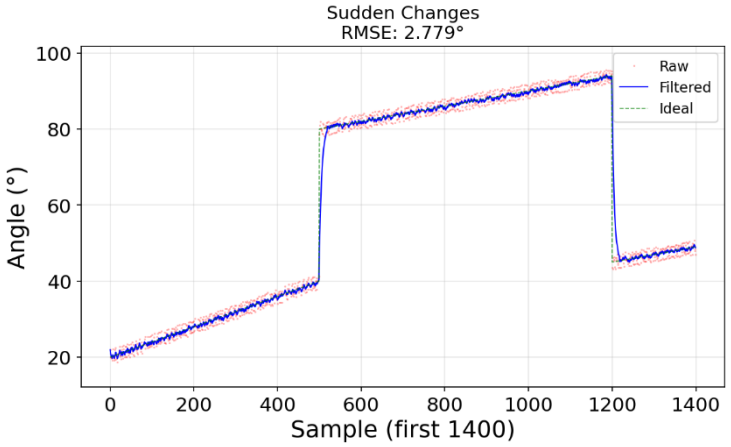
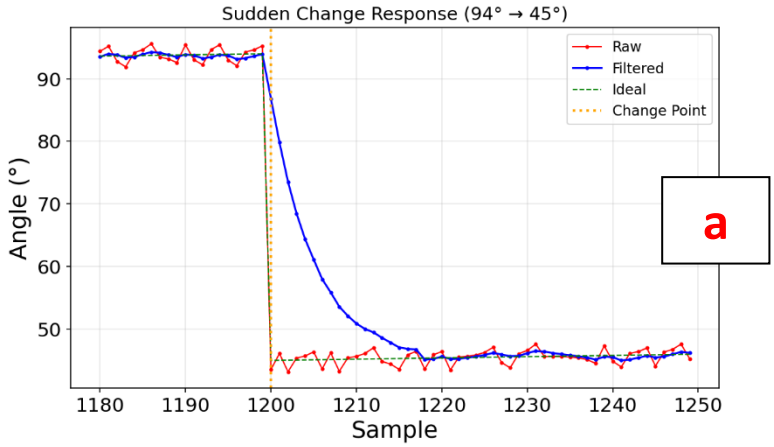


Figure 14. Response to legitimate sudden position changes (six abrupt jumps 20-60°)

Figure 15 shows a magnified view of the convergence behavior following each jump. In each case, the filtered estimate settles to the new value within approximately 3–5 samples and does so without overshoot or oscillatory behavior.



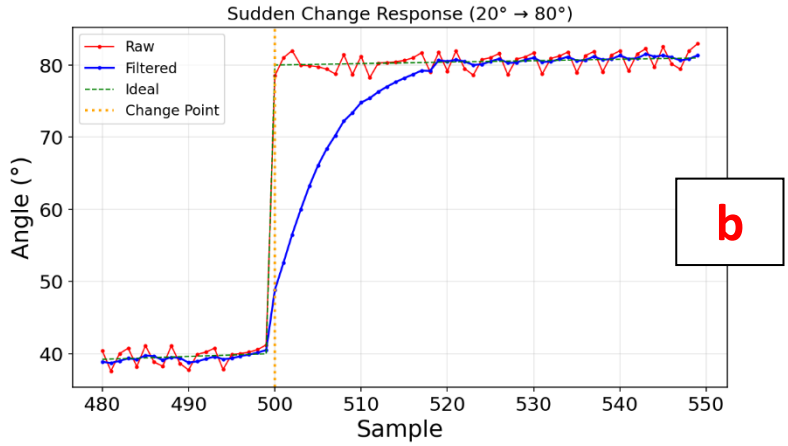


Figure 15. Close-up showing convergence after sudden change without overshoot

A broader view of performance across all scenarios is provided in Figures 16. It summarizes the corresponding RMSE values and noise reduction percentages, with numerical results listed in Table 2. Five of the six tested scenarios meet or exceed the target reduction range of 50–70%. Across all experiments, the adaptive Q9.7 fixed-point implementation shows stable behavior and no observable numerical artifacts.

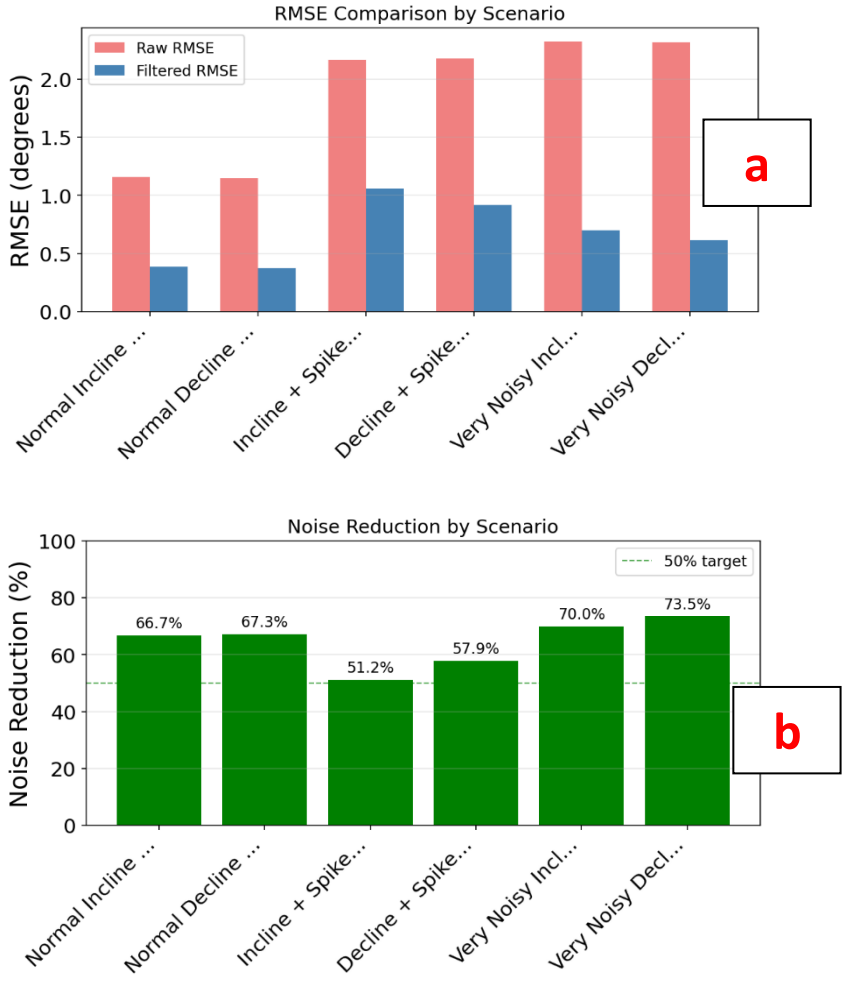
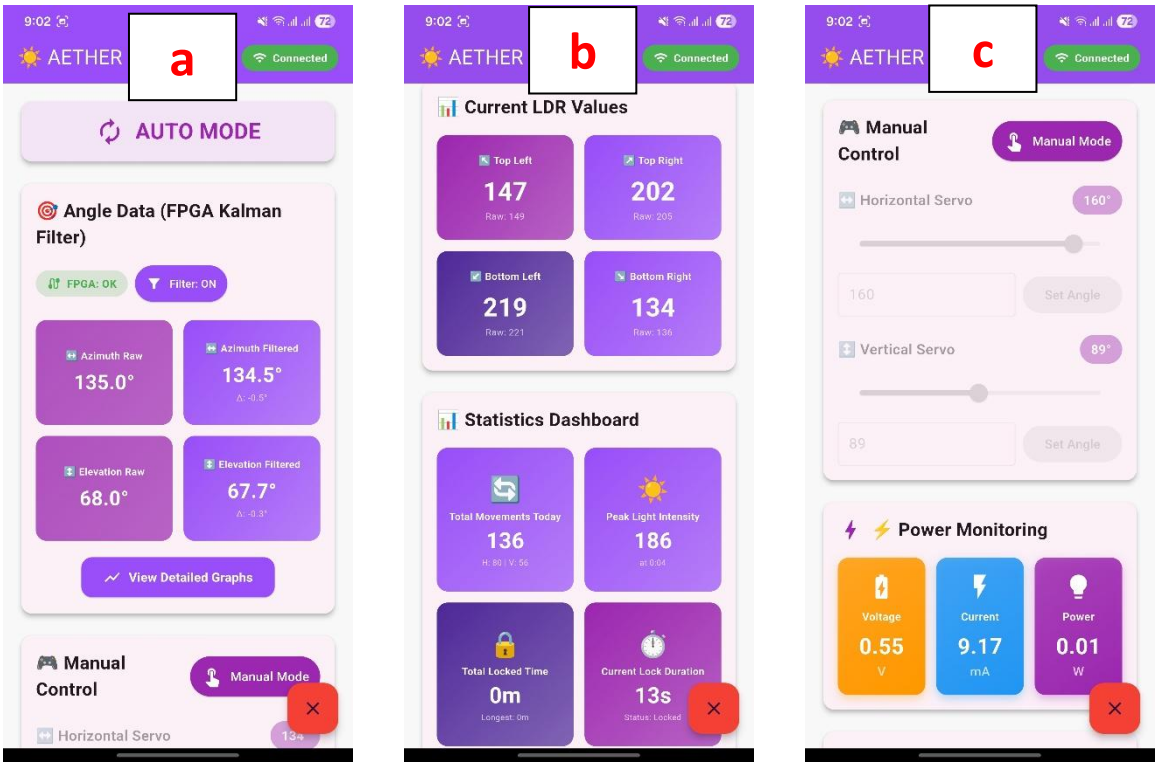


Figure 16. RMSE comparison and noise reduction across all scenarios (a) RMSE Comparison by Scenario (b) Noise Reduction by Scenario

Table 2. Kalman filter performance summary

Scenario	Raw RMSE (º)	Filter RMSE (º)	Reduction (%)
Normal Incline	1.16	0.39	66.7
Normal Decline	1.15	0.38	67.3
Very Noisy Incline	2.33	0.7	70
Very Noisy Decline	2.32	0.61	73.5
Incline with Spikes	2.17	1.06	51.2
Decline with Spikes	2.18	0.92	57.9

The integrated system operation is illustrated through the ESP32-based real-time dashboard, as shown in **Figures 17**. The dashboard presents live measurements of both raw and Kalman-filtered angles together with FPGA processing status (Fig. 17(a)), confirming correct operation of the Arduino – FPGA – ESP32 communication chain. Additional panels display LDR sensor readings, power monitoring parameters, and system statistics (Fig. 17(b)), enabling continuous monitoring of system health during operation. A manual control interface with slider-based servo positioning (Fig. 17(c)) allows direct user interaction for testing and calibration purposes. The real-time scrolling plots (Fig. 17(d)) clearly show the effectiveness of the filtering stage, where the filtered trajectory exhibits smooth tracking of the underlying motion while the raw signal contains noticeable jitter. These results verify that the proposed hardware-accelerated filtering pipeline operates reliably under real-time conditions.



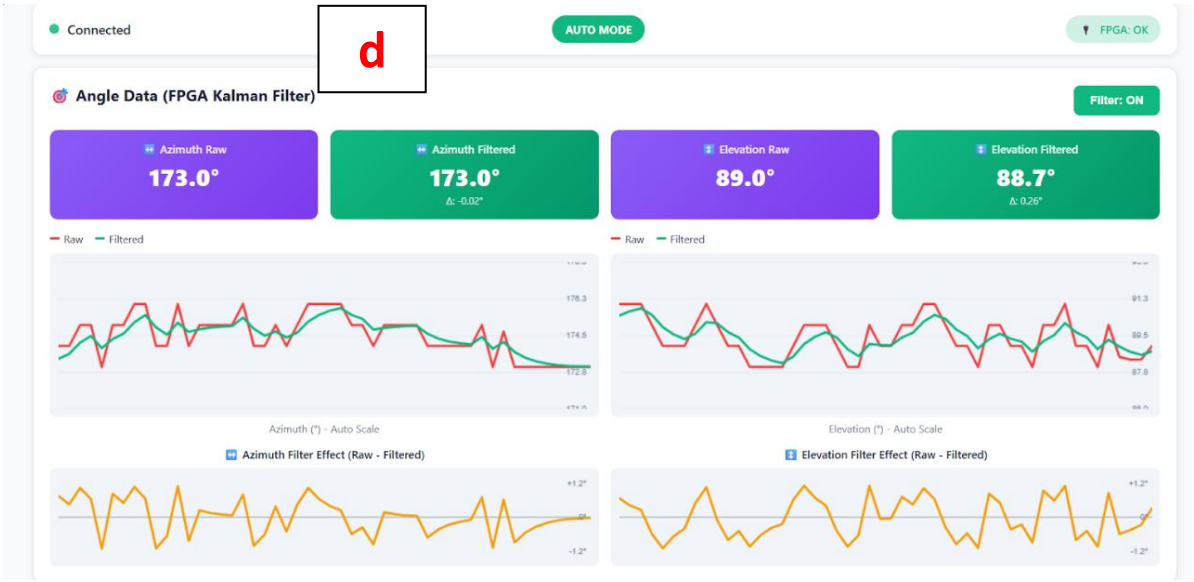


Figure 17. ESP32-based real-time system dashboard: (a) angle monitoring and FPGA status, (b) sensor and power statistics, (c) manual servo control interface, and (d) real-time raw vs. filtered angle plots.

The experimental results confirm that the proposed adaptive scalar Kalman filter meets the targeted performance objectives for dual-axis solar tracking applications. Across six representative test scenarios, the filter consistently achieved noise reduction between 51% and 74%, with the majority of cases exceeding the predefined 50–70% target range. This outcome validates the selection of a scalar adaptive Kalman filter and demonstrates that sophisticated multi-dimensional models are unnecessary for accurate solar position estimation in this domain.

The achieved noise attenuation has practical significance for LDR-based tracking systems. Raw LDR measurements are inherently affected by atmospheric variability, transient shading, and sensor non-idealities, which typically result in continuous servo jitter. By reducing measurement noise to a filtered RMSE of approximately 0.39°, the proposed system effectively suppresses unnecessary servo movements, leading to lower power consumption and reduced mechanical wear. Integration testing further confirmed a substantial reduction in servo actuation frequency compared to unfiltered operation.

A key contribution of this work is the validation of a heterogeneous hardware–software architecture. By offloading the Kalman filtering task to dedicated FPGA hardware, computational load on the Arduino and ESP32 platforms is eliminated. The FPGA filtering latency was negligible relative to the system control loop, confirming that hardware acceleration enhances performance without introducing timing bottlenecks. In contrast to monolithic microcontroller-based solutions, this architecture enables advanced filtering while preserving responsive wireless communication and real-time visualization.

The use of fixed-point Q9.7 arithmetic proved sufficient for the application, with no observable numerical instability or precision loss. The chosen format provides resolution well beyond the mechanical accuracy of typical servo actuators while significantly reducing FPGA resource utilization compared to floating-point implementations. This confirms that fixed-point arithmetic represents an optimal design trade-off for embedded solar tracking systems.

Compared to prior Arduino-only and ESP32-based trackers, the proposed system demonstrates superior robustness under noisy and transient conditions, while remaining extensible. Unlike laboratory-scale FPGA filtering demonstrations, this work integrates advanced signal processing into a complete, user-accessible system with sensing, actuation, and web-based monitoring.

5. Conclusion

This paper presented the design and validation of a heterogeneous dual-axis solar tracking system that overcomes key limitations of conventional microcontroller-only approaches. By combining a microcontroller for sensor acquisition, an FPGA for dedicated signal processing, and an IoT platform for user interaction, the proposed architecture delivers high tracking accuracy while maintaining real-time performance and system scalability.

The primary contribution is the FPGA-based implementation of an adaptive scalar Kalman filter using a fixed-point Q9.7 numerical format optimized for servo control applications. Experimental evaluation demonstrated consistent noise reduction of 51–74% across multiple operating scenarios, confirming the effectiveness of the adaptive filtering strategy in suppressing measurement noise while preserving responsiveness to legitimate sun position changes.

System integration results verified reliable real-time operation across the complete data pipeline, from LDR sensing and hardware filtering to wireless visualization. The low FPGA resource utilization further highlights the efficiency of the proposed design and provides a clear path for future extensions, such as advanced filtering techniques or multi-sensor fusion, without architectural modification.

Overall, this work demonstrates that FPGA-accelerated signal processing can be effectively integrated into practical solar tracking systems, offering a scalable and high-performance solution suitable for next-generation intelligent renewable energy applications.

Conflicts of Interest: The authors declare no conflict of interest.

Funding: This research received no external funding.

Data Availability: The data supporting the findings of this study are available from the authors upon reasonable request.

References

[1] Shyamalagowri, M., Mohammdha Hussaini, M., Moulee, K. P., Rajavenkatesan, T., Anbarasu, L., & Chandramohan, J. (2025). Dual-axis solar tracker with PID control and LoRa-based environmental sensing for improved solar energy output. *Journal of Environmental Nanotechnology, 14*(2), 494–501.

[2] Amadi, H. N., & Gutiérrez, S. (2019). Design and performance evaluation of a dual-axis solar tracking system for rural applications. *European Journal of Electrical Engineering and Computer Science, 3*(1).

[3] Mohanapriya, V., Manimegalai, V., Praveenkumar, V., & Sakthivel, P. (2021). Implementation of dual-axis solar tracking system. *IOP Conference Series: Materials Science and Engineering, 1084.*

[4] Mustafa Al-Sheikh. (2025). IoT-enabled dual-axis solar tracking system using ESP32 and Blynk for real-time monitoring and energy optimization. *Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika, 3*(1), 187–204. DOI: 10.61132/jupiter.v3i1.695

[5] BaBars, E., El-Mashad, S., Abdraboo, S., El-Sayed, M., Essa, M., & Babars, M. (2025). IoT-based monitoring and control of a dual-axis solar tracking system using FPGA technology. *SINAI International Scientific Journal (SISJ), 2*(1), 19.

[6] Bass, R. (1996). Kalman filtering: Theory and practice (Book review). *Proceedings of the IEEE, 84*(2), 321.

[7] BaBars, E. A. M., El Mashaad, S. Y., Abdraboo, S. M., & Essa, M. E. M. (2025). Forecasting of solar power generation for experimental dual-axis solar tracker system based on ANN and FPGA technology. *Engineering Research Journal, 184*(2), 1–23. https://doi.org/10.21608/erj.2024.332639.1134

[8] Linares-Barranco, A., Perez-Peña, F., Moeys, D. P., Gomez-Rodriguez, F., Jiménez-Moreno, G., Liu, S.-C., & Delbruck, T. (2019). Low-latency event-based filtering and feature extraction for dynamic vision sensors in real-time FPGA applications. *IEEE Access, 7*, 134926–134942.

[9] AlShabi, M., & Bonny, T. (2022). FPGA-based unscented Kalman filter for target tracking. *Proceedings of SPIE, 12122*, 121221A–1–121221A–9.

[10] Gawande, P., Mhaske, A., Gurnule, S., & Somwanshi, M. **(**2025). Design and implementation of a dual-axis solar tracking system using ATmega32. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*. https://doi.org/10.22214/ijraset.2025.70229

[11] Al Sheikh, M. A. (2025). IoT-enabled dual-axis solar tracking system using ESP32 and Blynk for real-time monitoring and energy optimization. *Jupiter: Publikasi Ilmu Keteknikan Industri, Teknik Elektro dan Informatika, 3*(1), 187–204. https://journal.aritekin.or.id/index.php/Jupiter/article/view/695

[12] Wahid, S. S. A., Ramli, M. S., & Zabidin, N. I. (2023). Dual-axis solar energy tracking and monitoring system with Arduino as a microcontroller. *Scientific Research Journal, 20*(2), 131–145. https://doi.org/10.24191/srj.v20i2.22365

[13] Pardosi, C. H., Siregar, M., & Pandjaitan, L. W. (2025). Design and implementation of a dual-axis solar tracking system using Arduino Uno microcontroller. *Jurnal ELTIKOM, 8*(1), Article 1105. https://doi.org/10.31961/eltikom.v8i1.1105

[14] Gawande, P., Mhaske, A., Gurnule, S., & Somwanshi, M. (2025). Design and implementation of a dual-axis solar tracking system using ATmega32. *International Journal for Research in Applied Science and Engineering Technology (IJRASET)*. https://doi.org/10.22214/ijraset.2025.70229

[15] Mpodi, E. K., Tjiparuro, Z., & Matsebe, O. (2019). Review of dual-axis solar tracking and development of its functional model. *Procedia Manufacturing.* https://doi.org/10.1016/j.promfg.2019.05.082

[16] Eke, J., et al. (2025). A review and comparative analysis of solar tracking systems. *Energies, 18*(10), 2553. https://www.mdpi.com/1996-1073/18/10/2553

[17] Arum, S. I. P., Nugroho, W. A. A., & Afnan, M. I. (2025). Monitoring solar tracker otomatis berbasis IoT menggunakan ESP32. *Seminar Nasional Teknologi Informasi dan Bisnis.* https://doi.org/10.47701/zfhym651

[18] Nazri, M. S. M., & Rozlan, M. H. H. M. (2022). Development of dual-axis solar tracker with IoT monitoring system. *Journal of Engineering Technology, 10*(1), 156–162. https://ejournal.unikl.edu.my/index.php/jet/article/view/334

[19] Wahyono, W., Suwarti, S., Dewantoro, Y. H., Citraningtyas, D. P., & Ardiansyah, M. R. (2022). Dual axis solar tracker system application based on Arduino ATmega2560 and IoT for submersible pump operation. *Eksergi, 18*(3). https://jurnal.polines.ac.id/index.php/eksergi/article/view/3910

[20] Shamsuddin, M. K. A., Mohd Shah, N. S., Saparudin, F. A., Mohd Redzuan, A. A. A., & Noor Azeb, M. M. A. (2021). Solar power system: Dual-axis tracker & monitoring system using Arduino & NodeMCU. *Progress in Engineering Application and Technology, 2*(1). https://publisher.uthm.edu.my/periodicals/index.php/peat/article/view/873

[21] Jumaat, S. A., Mohd Said, M. N. A. M., & Jawa, C. R. A. (2025). Dual axis solar tracker with IoT monitoring system using Arduino. *International Journal of Power Electronics and Drive Systems.* https://ijpeds.iaescore.com/index.php/IJPEDS/article/view/20494

[22] Shamsuddin, M. K. A., et al. (2021). Solar power system: Dual-axis tracker & monitoring system using Arduino & NodeMCU. *Progress in Engineering Application and Technology, 2*(1). https://publisher.uthm.edu.my/periodicals/index.php/peat/article/view/873

[23] Saini, B., Lohiya, K., & Pal, A. P. (2025). Design, implementation, and performance analysis of a voltage-based dual-axis sun tracking solar panel using Arduino. *International Journal of Computer Techniques, 12*(5). https://ijctjournal.org/dual-axis-arduino-solar-tracking-system

[24] U. Meyer Baese. (2014). *Digital signal processing with field programmable gate arrays* (4th ed.). Springer.