IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of completion 04, 2024.

Implementing Kalman Filter on ESP32 for DHT11 Sensor Accuracy

Yuanpeng Wang [1]
[1]Department of Electrical and Computer Engineering, The University of Western Ontario, London, Ontario, N6A 5B9, Canada
E-mail: ywan5646@uwo.ca

ABSTRACT This paper presents a practical implementation of the Kalman filter algorithm on the ESP32 microcontroller to enhance the accuracy of noisy real-time temperature and humidity measurements in a wireless sensor network. This research demonstrates that the Kalman algorithms significantly reduce noise and the filtered data closely match the true values for these measurements in real-time.

INDEX TERMS Adaptive Filter, Kalman Filter, Real-Time Noisy Measurements, Linear Kalman Filters, ESP32, DHT11 Sensor, Wireless Sensor Network, Raspberry Pi 4.

I. INTRODUCTION

The Kalman Filter (KF) is a well-known recursive algorithm used to estimate the state of a process by minimizing the system's error. KF relies on the previously estimated state and new data input for each updated estimate. Each result in the KF process informs the prediction or estimation for the next iteration. It is computationally more efficient to store only the previous estimate instead of the entire history of past data, making it well-suited for data processing on ARM devices, which are not equipped with powerful processors. The filter calculates the approximate states of linear time-varying systems, taking into account the influence of stochastic disturbances such as white noise. The Kalman Filter is an indispensable tool for target-tracking problems in various industries. The Kalman Filter is also adaptable to nonlinear systems through variations like the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF), extending its applicability. This versatility enables its use in a range of applications beyond target tracking, such as navigation and tracking systems, economic forecasting, and more. Its ability to provide real-time updates and handle uncertainty makes it invaluable in fields requiring precise and reliable data analysis[1]-[2].

The ESP32 microcontroller has become a cornerstone in the rapidly expanding Internet of Things (IoT) market. As a powerful tool in both industrial and domestic applications, the ESP32 excels in providing reliable, wireless communication and data transfer capabilities, aligning with the 6A framework—Anything, Anytime, Anyone, Anyplace, Any service, and Any network. ESP32 not only facilitates robust networking and versatile application development but also drives significant changes in behavior and lifestyle, impacting areas such as intelligent transportation, home automation, and e-health. Through its extensive features and flexible usability, the ESP32 is advancing the development of IoT technologies in multiple sectors[3].

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi obtains its power from a micro USB adapter, with a minimum range of 2.5 watts (500MA), making it ideal for low-power consumption project design[4].

DHT11 sensor is a compact digital device designed to measure both temperature and humidity. DHT11 sensor consumes a minimal 0.3mA, making it an efficient choice for a variety of applications. In projects like automated room temperature control systems, the DHT11 can control devices such as heaters or fans based on the environmental data it collects, with real-time results displayed on an LCD. This functionality demonstrates its utility in creating responsive and automated environments[5].

II. SYSTEM ANALYSIS

A. Theoretical analysis: Kalman filter algorithm analysis

The Kalman Filter operates in two main steps: prediction and correction. Initially, the filter predicts the state of the dynamic system based on a predefined model. These predicted states are then adjusted in the correction step based on new sensor measurements. This process not only refines the state estimates but also minimizes the estimator’s error covariance. Repeated across each time step, this method considers the state from the previous time step as the initial state for the current calculation, making

the Kalman Filter inherently recursive. The graphical description of the predictor-corrector is shown in Fig. 1[1].

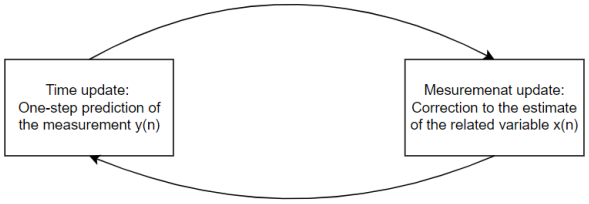


FIGURE 1. The graphical description of the predictor-corrector.

The signal-flow graph of a Kalman filtering model is shown in Fig. 2[1].

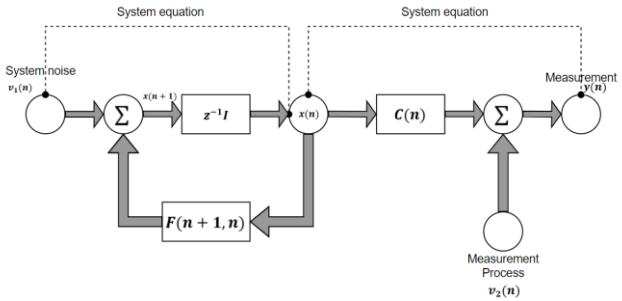


FIGURE 2. A linear, discrete-time dynamic model.

Where the system equation can be represented as:

$$\boldsymbol{x(n+1) = F(n+1,n)x(n) + v_1(n)} \quad \textbf{(1)}$$

The measurement equation can be represented as:

$$\boldsymbol{y(n) = C(n)x(n) + v_2(n)} \quad \textbf{(2)}$$

We can summarize the Kalman filter algorithm based on one-step prediction in the following table[1]:

Input vector process :

Measurements = {y(1), y(2), …, y(3)}

Know parameters:

Transition matrix = F(n+1, n)

Measurement matrix = C (n)

Correlation matrix of system noise = $\boldsymbol{Q_1(n)}$

Correlation matrix of measurement noise = $\boldsymbol{Q_2(n)}$

Computation: n = 1, 2, 3, …

$\boldsymbol{G(n) = F(n+1,n)K(n,n-1)C^H(n)[C(n)K(n,n-1)C^H(n) + Q_2(n)]^{-1}}$

$\boldsymbol{\alpha(n) = y(n) - C(n)\hat{x}(n|y_{n-1})}$

$\boldsymbol{\hat{x}(n|y_n) = F(n+1,n)\hat{x}(n|y_{n-1}) + G(n)\alpha(n)}$

$\boldsymbol{K(n) = K(n,n-1) - F(n,n+1)G(n)C(n)K(n,n-1)}$

$\boldsymbol{K(n+1,n) = F(n+1,n)K(n)F^H(n+1,n) + Q_1(n)}$

Initial conditions:

$\boldsymbol{\hat{x}(1|y_0) = E[x(1)]}$

$\boldsymbol{K(1,0) = E[(x(1) - E[x(1)])(x(1) - E[x(1)]^H) = \prod_0}$

Table 1. The summary of the Kalman filter algorithm.

B. Practical analysis: Kalman filter algorithm code

The DHT11 sensor detects temperature and humidity, which are considered the true values. Additional white noise is added to account for potential sensor malfunction. By applying the Kalman filter to the noisy values, we can obtain filtered data, demonstrating the effectiveness of the implemented Kalman algorithms.

The function for generating white noise is shown in Fig. 3.

```
// Function to get noise
float getNoise(float noiseLevel) {
    return (random(1000) / 500.0 - 1) * noiseLevel;
}
```

FIGURE 3. The function for generating white noise.

The parameters for the Kalman filter algorithms are shown in Fig. 4 and main the Kalman filter algorithms for filtering the noisy temperature and humidity values are shown in Fig. 5.

```
// Kalman filter variables for temperature
float varMeasure_temp = 1e-5; // Measurement variance for temperature
float varProcess_temp = 1e-8; // Process variance for temperature
float k_temp = 1.0; // Estimate correlation matrix of error for temperature
float X_est_temp = 0.0; // Estimated temperature
float G_temp = 0.0 ; // Kalman gain for temperature
float k_temp_pred = 0.0;  /// Update correlation matrix of error
float sigma_temp = 5;

// Kalman filter variables for humidity
float varMeasure_humidity = 1e-5;  // Measurement variance for humidity
float varProcess_humidity = 1e-8; // Process variance for humidity
float k_humidity = 1.0; // Estimate correlation matrix of error for humidity
float X_est_humidity = 0.0; // Estimated humidity
float G_humidity = 0.0 ; // Kalman gain for humidity
float k_humidity_pred = 0.0;  // Update correlation matrix of error
float sigma_humidity = 5;
```

FIGURE 5. The Kalman filter algorithm parameters.

```
//Sensors operation
 float temp = dht.readTemperature();  // Read temperature
 float humidity = dht.readHumidity(); // Read humidity

 // Kalman filter process for temperature
 // (F=1). Temperature does vary drastically; No control input to the system.
 float temp_noisy = temp + getNoise(sigma_temp); // Add noise to temperature measurement
 k_temp_pred = k_temp + varProcess_temp; // Predicted correlation matrix of error
 G_temp = k_temp_pred / (k_temp_pred + varMeasure_temp); // Kalman gain
 k_temp = k_temp_pred *(1 - G_temp) ; // Update correlation matrix of error
 X_est_temp = X_est_temp + G_temp * (temp_noisy - X_est_temp); // Update estimate with measurement
float EK_temp = abs(temp - X_est_temp); //Error of Kalman Filtered Temperature Value
float EN_temp = abs(temp - temp_noisy); //Error of Noisy Measured Temperature Value

 // Kalman filter process for humidity
 //(F=1). Humidity does vary drastically; No control input to the system.
 float humidity_noisy = humidity + getNoise(sigma_humidity); // Add noise to humidity measurement
 k_humidity_pred = k_humidity + varProcess_humidity; // Predicted correlation matrix of error
 G_humidity = k_humidity_pred / (k_humidity_pred + varMeasure_humidity); // Kalman gain
 k_humidity = k_humidity_pred * (1 - G_humidity); // Update correlation matrix of error
 X_est_humidity = X_est_humidity + G_humidity * (humidity_noisy - X_est_humidity); // Update estimate with measurement
 float EK_humidity = abs(humidity - X_est_humidity);  //Error of Kalman Filtered Humidity Value
 float EN_humidity = abs(humidity - humidity_noisy);  // Error of Noisy Measured Humidity Value
```

FIGURE 6. The Kalman filter algorithm.

We can write the Kalman filter algorithms code with the following equations for processing both temperature and humidity data affected by noise:

Adding white noise to the true value:

$$\boldsymbol{X_{noisy} = X + Noise} \quad \textbf{(3)}$$

The predicted correlation matrix of error:

$$\boldsymbol{K\_pred = K + Q_1(n)} \quad \textbf{(4)}$$

The Kalman gain:

$$\boldsymbol{G = \frac{K\_pred}{K\_pred + Q_2(n)}} \quad \textbf{(5)}$$

Updating the correlation matrix of error:

$$K = K\_pred * (1 - G) \tag{6}$$

The filtered value:

$$X_{est} = X_{est} + G * (X_{noisy} - X_{est}) \tag{7}$$

III. SYSTEM DESIGN & IMPLEMENTATION

The system comprises an ESP32 microcontroller connected to a DHT11 sensor on a breadboard, with a Raspberry Pi 4 serving as a server for data storage and display on a Grafana dashboard. The devices are shown in Fig. 5. The ESP32 reads temperature and humidity values from the DHT11 sensor, adds white noise to these values, and applies Kalman filter algorithms. Communication between the ESP32 and Raspberry Pi 4 is facilitated through the MQTT protocol, with JSON files being transmitted from the microcontroller to the Raspberry Pi. The system architecture is orchestrated using the Node-RED flow editor, as depicted in Fig. 6. Additionally, Figure 7 shows the database table containing the relevant values stored on the Raspberry Pi 4. Data visualization on the Grafana dashboard enables real-time monitoring and analysis, enhancing the utility of the collected environmental data. The integration of these components allows for scalable and efficient IoT solutions, crucial for dynamic data management and sophisticated environmental control.

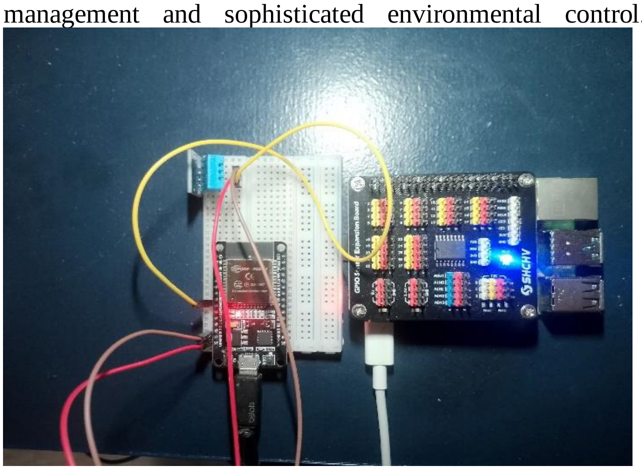


FIGURE 5. The devices of the system.

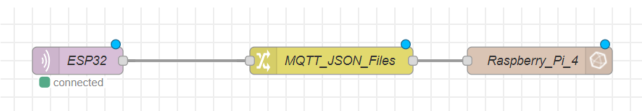
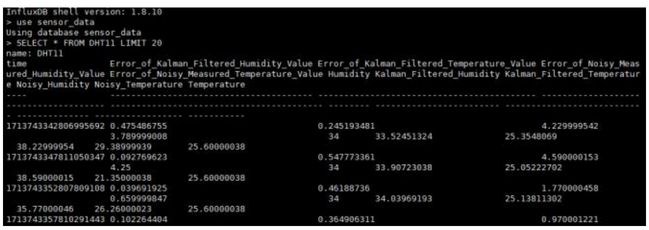


FIGURE 6. The flowchart of the simulation design and implementation.



```
InfluxDB shell version: 1.8.10
> use sensor_data
Using database sensor_data
> SELECT * FROM DHT11 LIMIT 20
name: DHT11
```

time	Error_of_Kalman_Filtered_Humidity_Value	Error_of_Kalman_Filtered_Temperature_Value	Error_of_Noisy_Measured_Humidity_Value	Error_of_Noisy_Measured_Temperature_Value	Humidity	Kalman_Filtered_Humidity	Kalman_Filtered_Temperature	Noisy_Humidity	Noisy_Temperature	Temperature
1713743342806995692	0.475486755	0.245193481	4.229999542	3.789999008	34	33.52451324	25.3548069	38.22999954	29.38999939	25.60000038
1713743347811050347	0.092769623	0.547773361	4.590000153	4.25	34	33.90723038	25.05222702	38.59000015	21.35000038	25.60000038
1713743352807809108	0.039691925	0.46188736	1.770000458	0.659999847	34	34.03969193	25.13811302	35.77000046	26.26000023	25.60000038
1713743357810291443	0.102264404	0.364906311	0.970001221							

FIGURE 7. The influx database table.

IV. IMPLEMENTATION RESULTS FOR VERIFICATION

A. Temperature data

The true temperature data is shown in Fig. 8. Observing the plot, it becomes apparent that the DHT11 sensor operates under relatively stable conditions, displaying minimal fluctuations. Since ESP32 reads the sensor's digital output with 1°C resolution. The ESP32 reads this data at intervals, resulting in stepped changes rather than a smooth curve, as temperature variations are captured in discrete increments. The time range for these data is set from 12:20 to 19:10.

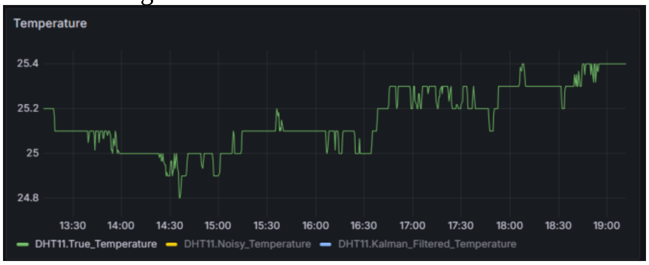


FIGURE 8. The true temperature data.

The temperature data with noise is shown in Fig. 9. Due to the noise evident in the plot, it's challenging to obtain accurate temperature values at any given time.

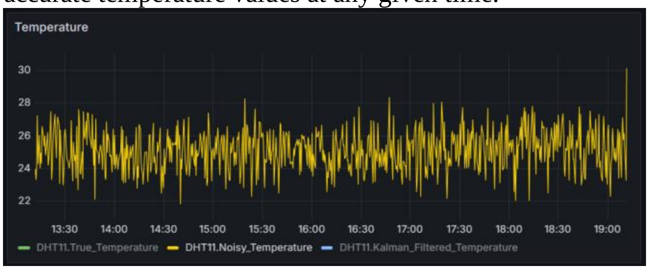


FIGURE 9. The noisy temperature data.

The temperature data filtered using the Kalman algorithm is shown in Fig. 10. Notably, the filtered values exhibit a considerable noise reduction, aligning closely with the true temperature values. This serves as a testament to the efficient denoising capabilities of the Kalman algorithm in real-time scenarios, while also maintaining low power consumption, owing to its computational efficiency across various ARM devices. Fig. 11 presents a comprehensive comparison of the true temperature data, the noisy temperature data, and the Kalman algorithm-filtered temperature data, thereby highlighting the effectiveness of the Kalman algorithm.

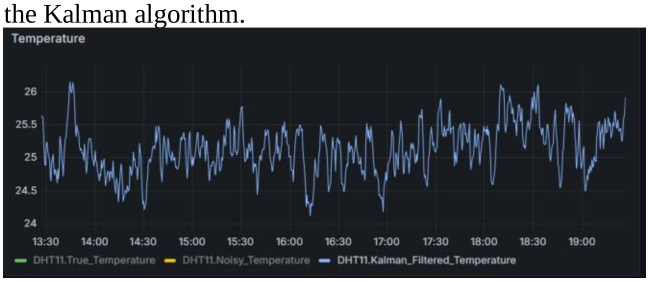


FIGURE 10. The Kalman algorithm filtered temperature data.

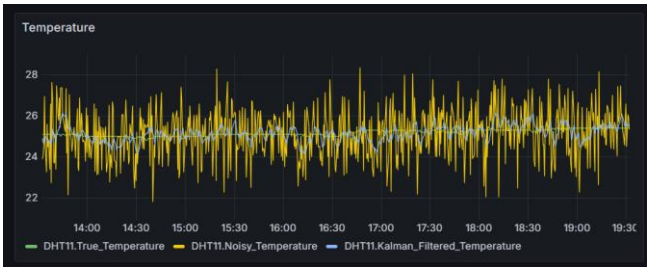


FIGURE 11. **Comparison between the true temperature data, the noisy temperature data, and the Kalman algorithm filtered temperature data.**

The comparison between the error of the noisy measured temperature and the error of the temperature filtered by the Kalman algorithm is shown in Fig. 12. This plot serves as additional evidence of the excellent performance of the Kalman algorithm for real-time filtering and the error control capability.

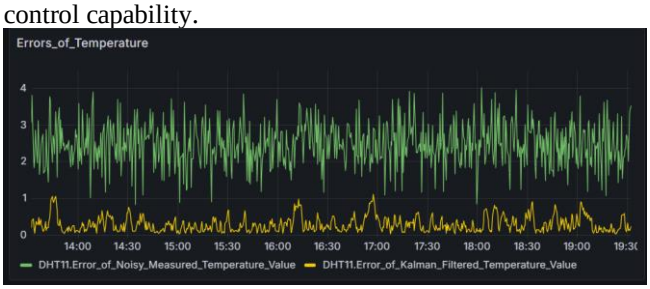


FIGURE 12. **The comparison between the error of noisy measured temperature and Kalman algorithm filtered temperature.**

B. Humidity data

The true humidity data is shown in Fig. 13. We can observe DHT11 sensor also provides relatively stable humidity data with minimal fluctuations. The time range for these data is also set from 12:20 to 19:10.

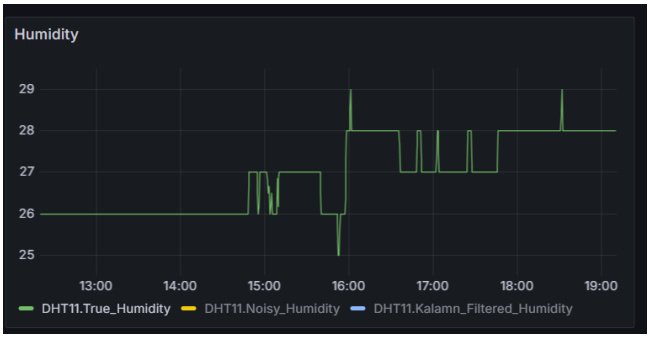


FIGURE 13. **The true humidity data.**

The humidity data with noise is shown in Fig. 14. We cannot obtain accurate humidity values at any given time due to significant fluctuations that obscure the true readings. This variability introduces substantial uncertainty, complicating any reliable analysis of environmental conditions.

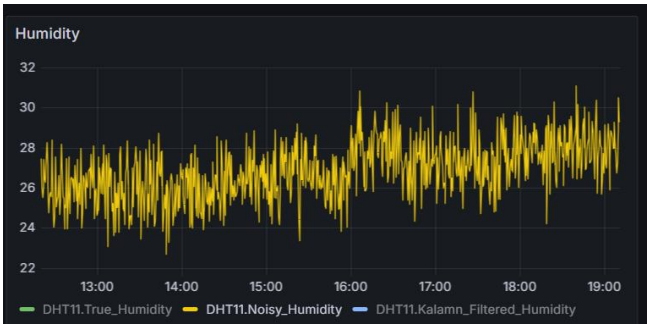


FIGURE 14. **The noisy humidity data.**

Fig. 15 displays the humidity data after filtering. Fig. 16 provides a comprehensive comparison of the actual humidity values, the values with noise, and the values filtered using the Kalman algorithm. This demonstrates the Kalman algorithm's versatility in various fields for enhancing data processing accuracy.

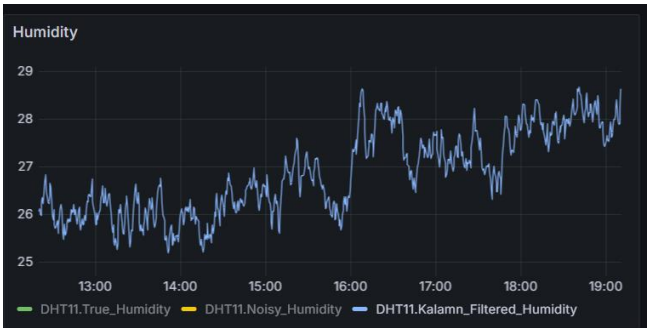


FIGURE 15. **The Kalman algorithm filtered humidity data.**

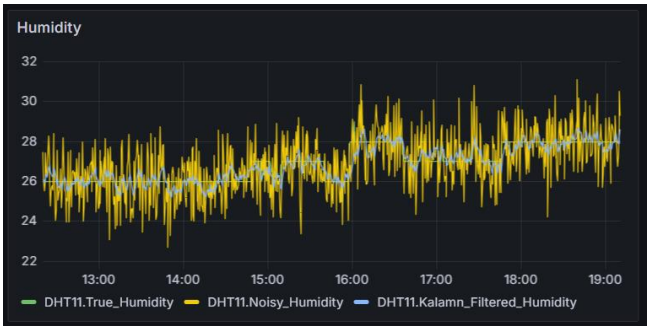


FIGURE 16. **Comparison between the true humidity data, the noisy humidity, and the Kalman algorithm filtered humidity data.**

The comparison between the error in the noisy measured humidity and the error in the humidity filtered by the Kalman algorithm is shown in Fig. 17. This visualization clearly illustrates the effectiveness of the Kalman filter in reducing measurement noise. It highlights the substantial improvement in data accuracy, demonstrating the filter's capability to enhance sensor reliability and precision in real-time environmental monitoring scenarios.

FIGURE 18. The comparison between the error of noisy measured humidity data and Kalman algorithm filtered humidity data.

V. CONCLUSION

This work has successfully demonstrated the feasibility of utilizing the Kalman filter on the ESP32 platform to enhance the accuracy of the DHT11 sensor, especially in dynamic and real-time environmental conditions. The results definitively show improved sensor readings, with significant enhancements in precision and stability across a range of temperatures and humidity levels. These improvements are clearly evidenced through statistical analysis and real-time performance metrics.

The integration of the Kalman filter with the ESP32 and DHT11 sensor presents a cost-effective and accessible solution that can be significantly beneficial in various applications requiring reliable environmental data. This includes home automation systems, weather monitoring stations, and IoT applications where environmental monitoring is crucial. Furthermore, the use of the ESP32 module adds flexibility and scalability to the design, making it suitable for educational purposes and DIY projects.

Future work will focus on refining the algorithmic approach and exploring the integration of additional sensor types to further expand the system's capabilities. Moreover, the implementation of real-time data analytics and machine learning, coupled with cloud integration, could significantly enhance the system's utility and functionality within complex IoT networks, thereby widening its application spectrum in smart device technology.

Acknowledgment

The author expresses deep gratitude to Dr. Parsa for providing extensive personal and professional guidance.

REFERENCES

[1] Simon Haykin, Adaptive Filter Theory, Prentice-Hall, Inc., 1996.

[2] C. T. Ginalih, A. S. Jatmiko and R. Darmakusuma, "Simple Application of Kalman Filter On a Moving Object in Unity3D," 2020 6th International Conference on Interactive Digital Media (ICIDM), Bandung, Indonesia, 2020, pp. 1-3, doi: 10.1109/ICIDM51048.2020.9339662.

[3] A. Maier, A. Sharp and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," 2017 Internet Technologies and Applications (ITA), Wrexham, UK, 2017, pp. 143-148, doi: 10.1109/ITECHA.2017.8101926.

[4] A. M. Abd-Elrahim, A. Abu-Assal, A. A. A. -A. Mohammad, A. -I. M. Al-Imam, A. -H. A. Hassan and M. -A. M. Muhi-Aldeen, "Design and Implementation of Raspberry Pi based Cell phone," 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), Khartoum, Sudan, 2021, pp. 1-6, doi: 10.1109/ICCCEEE49695.2021.9429601.

[5] G. M. Debele and X. Qian, "Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020, pp. 428-432, doi: 10.1109/ICCWAMTIP51612.2020.9317307.