Lightweight Anomaly Detection Framework for IoT

Bianca Tagliaro Beasley
Electrical and Electronic Engineering
UCC

George O'Mahony
Electrical and Electronic Engineering
UCC

Sergi Gómez Quintana
Electrical and Electronic Engineering
UCC

Andriy Temko
Electrical and Electronic Engineering
UCC

Emanuel Popovici
Electrical and Electronic Engineering
UCC

***Abstract*— Internet of Things (IoT) security is growing in importance in many applications ranging from biomedical to environmental to industrial applications. Access to data is the primary target for many of these applications. Often IoT devices are an essential part of critical control systems that could affect well-being, safety, or inflict severe financial damage. No current solution addresses all security aspects. This is mainly due to the resource-constrained nature of IoT, cost, and power consumption. In this paper, we propose and analyse a framework for detecting anomalies on a low power IoT platform. By monitoring power consumption and by using machine learning techniques, we show that we can detect a large number and types of anomalies during the execution phase of an application running on the IoT. The proposed methodology is generic in nature, hence allowing for deployment in a myriad of scenarios.**

Keywords*— *IoT, security, embedded systems, low power, ARIMA, SARIMA, Machine Learning, anomaly detection

I. Introduction

Security is a growing concern as the world becomes more connected. The Internet of Things (IoT) market is growing at a rapid pace, and security becomes a deployment bottleneck [1]. A large amount of data will be transmitted wirelessly, sometimes in public spaces. They are exposed to malicious attackers with a wide variety of aims, which is worrisome in a time where we expect 24/7 IoT monitoring through smartphones and wearable devices [2]. One of the greatest challenges in IoT security is the limited resources available, be it the physical hardware (small area/memory), power consumption (linked to battery lifetime), or limited bandwidth for data transmission.

In this paper, an intrusion detection system (IDS) to detect anomalies for IoT applications was built and evaluated. This development is for general purposes by comparing the expected behavior of the device to the real-time data readings. A set of power data was collected from the application running on the IoT device and used to train a machine learning (ML) algorithm to generate an accurate model. Currently, a lot of anomaly detection research is using ML as it offers high accuracy. For example, convolutional neural network (CNN) algorithms show some of the best results, with accuracies as high as 97%-98% [3] [4]. In [3], one to three layers were tested as well as the hybrid algorithm (combining CNN to the recurrent neural network or long short-term memory). These features use a lot of resources, making CNN not suitable for resource-constrained IoT. A study comparing some classical ML algorithms (not including CNN) concluded that the best option for the detection of common IoT attacks is by using random forest algorithms [5]. But it should be noted that these were only simulated, and none of the algorithms were specifically developed for IoT, such that other algorithms might perform better in real-life situations.

In [6], a lightweight anomaly detection was created using recursive least squares but showed poor performance in real-world tests, with the best true positive rate (TPR) being 53.6%. In [7], an IDS was made for IoT communication protocols using open-source software, but it relied on transmitting data to a central base station to perform anomaly detection analysis. Sending data to a central base station for anomaly detection is a common way to circumvent the issue of limited resources, at the cost of speed. However, it is known that wireless communication is very costly compared to computing when it comes to power consumption. Some research was also done on focusing on building hardware. In [8], an IDS was built using an FPGA, where an FSM would monitor the expected state that the device should go next. If it entered an unexpected state, then this would indicate an anomaly and could be detected within three clock cycles. In [9], an anomaly detection system was implemented using an autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) forecast for smart electricity meters. The goal was to detect electricity theft by checking the mean and variance. Applying this to the test data showed a reduction in 99.4% of stolen electricity, but it only functions for a limited time period (seven days) since seasonality was not implemented.

II. Methodology

In this paper, an IDS was implemented for anomaly detection, with the aim of requiring minimal resources and low power consumption. For the detection system, an ML algorithm was used to generate a low weight model.

A. Framework

The general framework for this IDS (Figure 1) has three main components: a power monitor, an IoT device, and a security watchdog. Both the IoT and the power monitor were implemented using existing boards that match the requirements set in this paper (low power, low resource usage). The watchdog checks for anomalies and is the focus of the research completed for this paper.

- The PSoC has a set of components that are found in most IoT applications, such as sensors, actuators, and configurable analog parts. For this paper, an application using LEDs and Linear Feedback Shift Registers (LFSRs) was implemented with a clock running at 24 MHz.
- The power monitor is connected to the IoT device, and it measures the power consumption. The role of the monitor is two-folds. Firstly, it is used in the application characterization/modelling process, and

secondly, it sends real-time power data to the watchdog/IDS.

- In the watchdog, a previously trained machine learning application model will analyse the newly acquired data during application runtime. If an anomaly is detected, it will send a signal to the IoT device to either reset to factory settings or turn it off completely (using power gating). The main constraint is that the power consumption of the energy monitor and the watchdog to be smaller than that of the IoT device.

The reason power data was chosen is because it reflects the instruction sequence performed in the core processor. If the same set of instructions is repeated (which is the case of a duty-cycled application), then the normal power characteristics of a cycle should be repeatable. But if any modification is made in the code, then the instruction sequence changes, and therefore the power consumption will be different than expected. It can be observed in Figure 2 (using a low-cost, low power power-monitor) and Figure 3 (using an expensive high-end oscilloscope), where each curve shows the data after a reset, that it is replicable. To observe this, the data was input through erosion, a dilation, a low pass, and a high pass filter. The goal of this process is to eliminate any noise and offset that might be present, such that only the clean signal is present. The data shown in Figure 2 was collected using the STMicroelectronics power monitor samples current at a rate of 3.2 MSamples/s, using three 12-bit ADC for different current ranges. This is then averaged every 32 points, resulting in an output of 100 kSamples/s. The data in Figure 3 was collected using a Tektronix DPO4054 oscilloscope, which samples the data at 2.5 GSamples/s.

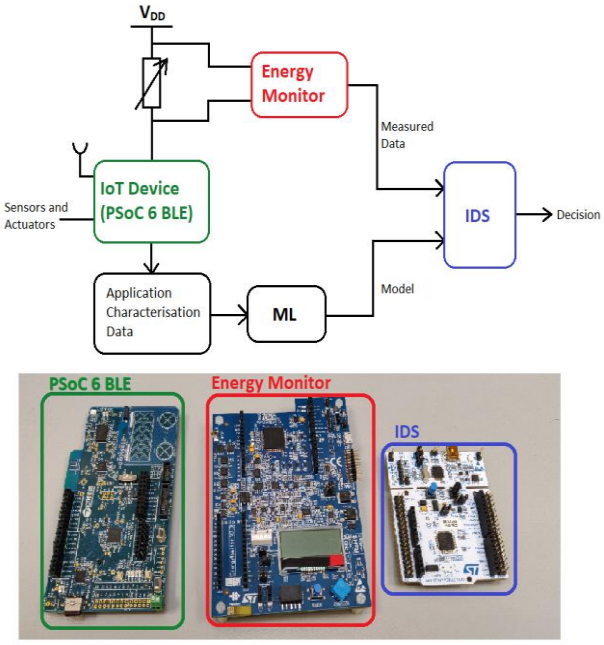

Figure 1 - Framework of IDS (with photos).

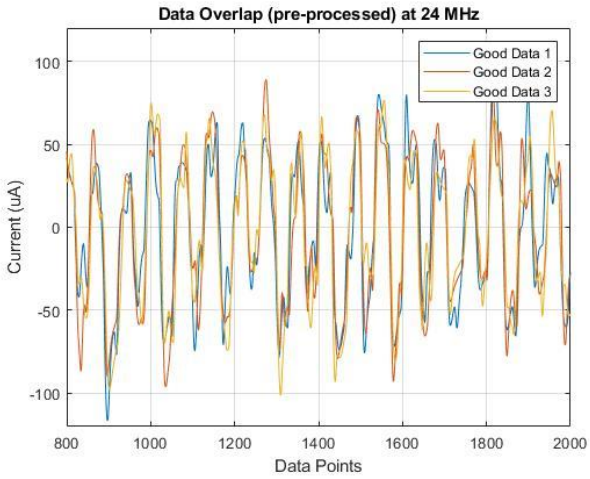

Figure 2 - Application Characterisation: check for replicability, with three reruns (power monitor).

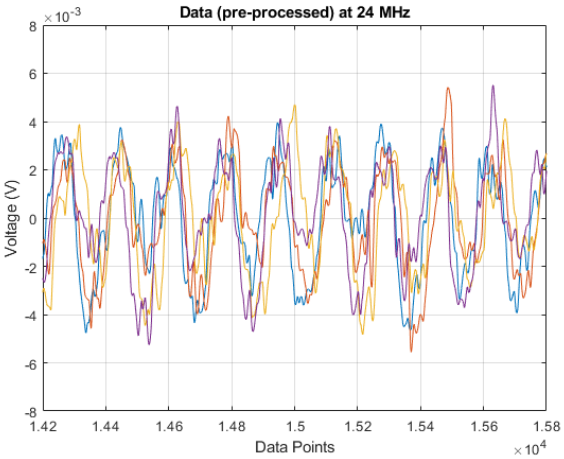

Figure 3 - Application Characterisation: check for replicability, with four reruns (using Tektronix DPO4054 oscilloscope, at 2.5 GSamples/s).

B. *Machine Learning Algorithm*

Many types of ML algorithms were studied for this IDS, but ultimately the ARIMA model was chosen [10] due to its simplicity and suitability for implementation on a microcontroller. This is a statistical method that creates a model that approximates the training data. It consists of a function, as shown in (1), with one part representing the moving average (MA) and the other part being the autoregressive (AR) component. Y_t is the lag of the series, β and ϕ are the coefficients for the AR and MA components respectively, ε represents the errors of the lag, and α is a constant.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p}\varepsilon_t + \phi_1\varepsilon_{t-1} + \phi_2\varepsilon_{t-2} + \ldots + \phi_q\varepsilon_{t-q} \quad (1)$$

Neural networks perform better in terms of detection rate [3], but they demand a lot of resources that might not be available in IoT devices. The ARIMA is lightweight and easy to implement in the embedded system. This is because, for the inference, the number of model parameters is much smaller than for other ML models. This leads to reduced memory usage as well as a reduced number of operations while providing high accuracy predictions.

An important characteristic is that this data presents seasonality; in other words, it shows a repeating pattern at regular intervals. Figure 4 shows the data used for training the ML algorithm after pre-processing for improved data stationarity. The data approximates a square waveform which repeats itself every 40 points. In Figure 4, it can be observed that the rolling mean of this data is not constant, and therefore the data is not stationary. As a result, the ARIMA model does not follow the trends of the training data (Figure 5). This is corrected by using seasonal ARIMA (SARIMA), which allows to model seasonality in the data.

The SARIMA model developed in this paper was made using Python libraries. This allows the user to select the order of *p* (AR), *q* (MA), and *d* (differencing components). For seasonal models, it is also necessary to select $p_{seasonal}$, $q_{seasonal}$, $d_{seasonal}$ and the period which the season repeats, due the need to represent the seasonality in the data. The parameters chosen for this paper are (p, d, q)x($p_{seasonal}$, $d_{seasonal}$, $q_{seasonal}$) = (0, 1, 2)x(0, 1, 0), with a seasonal frequency of 40. The chosen values provide an accurate enough model within a reasonable time since each coefficient is iterated multiple times. The differencing components were chosen to improve the stationarity of the data. Although some pre-processing is already in place for this purpose, the additional differencing (and differencing for the seasonal component) significantly improves the resulting model. Sometimes, in low order models, the AR and MA parts cancel each other's effect. For this reason, it was decided to have no AR components.

C. Validation and Performance Metrics

To compare different SARIMA models, the Akaike Information Criterion (AIC) is used [11]. It is an estimate of how much information was lost between the model and the training data. This value is automatically calculated by the algorithm used in this paper. For data prediction, AIC should be as close to zero as possible such that it follows the training data closely. For example, the AIC value for the ARIMA model in Figure 5 is -1959.917, while the AIC for a similar order SARIMA is -1678.8095. This shows that the SARIMA model is more suitable in this case. However, in anomaly detection, the lowest AIC value is not always the best option. A good model for IDS should be able to follow the trends accurately without having the exact same points. The reason for this is that two separate data sets will have some differences, which are not anomalies, for example, due to process, voltage, or temperature (PVT). If the model is too close to the training data, then it may detect a false anomaly.

Therefore, the goal is to find a model with small AIC, but that still follows the trends. For example, in Table 1 the AIC for model (2,1,0)x(0,1,0,40) was lower than model (0,1,2)x(0,1,0,40). But in Figure 6 and Figure 7, it can be seen that model (2,1,0)x(0,1,0,40) has more noise that might affect the performance of the anomaly detection, while model (0,1,2)x(0,1,0,40) has a smoother plot that follows the trends.

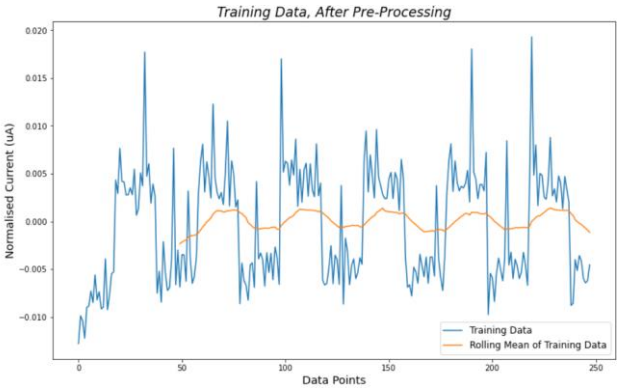

Figure 4 - Training data after pre-processing.

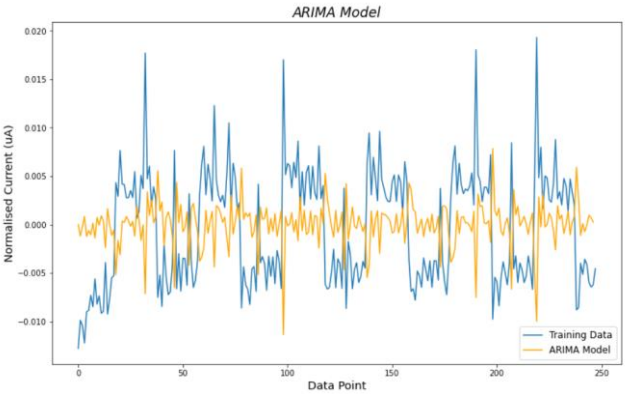

Figure 5 - ARIMA Model for Training Data.

ARIMA Model AIC Comparison	
Model Parameters $(p,d,q)x(p_{seasonal},d_{seasonal},q_{seasonal},40)$	***AIC***
(0,1,1)x(0,1,0,40)	-1681.3801
(1,1,0)x(0,1,0,40)	-1610.4896
(0,1,1)x(0,1,1,40)	-1671.9873
(1,1,0)x(0,1,1,40)	-1671.9873
(2,1,0)x(0,1,0,40)	-1651.2217
(0,1,2)x(0,1,0,40)	-1678.8095
(0,1,2)x(0,1,1,40)	-1739.4169
(2,1,0)x(0,1,1,40)	-1703.4882

Table 1 - AIC values for each ARIMA model.

Figure 6 - SARIMA model with parameters (0,1,2)x(0,1,0,40).

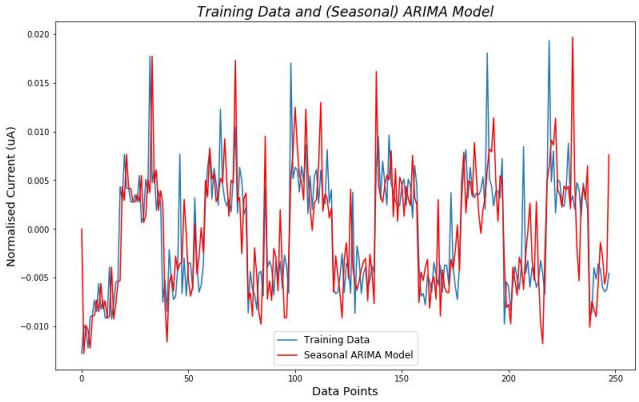

Figure 7 - SARIMA model with parameters (2,1,0)x(0,1,0,40).

To measure the performance of the anomaly detection, the true positive rate (TPR) and false-positive rate (FPR) were calculated [4], as shown in (2) and (3). The figures required for these calculations are the number of true positives (TP), the number of false negatives (FN), the number of false positives (FP), and the number of true negatives (TN).

TPR is also known as detection rate or recall and is a measure of how many of the anomalies tested were actually detected by the IDS. FPR is sometimes called a false alarm rate and indicates how often will be IDS falsely show an anomaly. FPR is an important measure as too many false positives can impact the performance of the device, for example, if a reset occurs every time. In general, there is a trade-off between TPR and FPR. Relaxing the features used to characterise an anomaly can improve TPR, but it will also make the IDS more susceptible to false positives.

$$TPR = TP / (TP + FN) \quad (2)$$

$$FPR = FP / (TN + FP) \quad (3)$$

The relationship between TPR and FPR is very important, so a common way to evaluate an IDS is to use the receiver operating characteristic (ROC) curve. This is useful for comparing different IDS. A high accuracy IDS will have a steep ROC curve, i.e., TPR grows faster than FPR.

D. Types of Inserted Anomaly

A Python-based algorithm was used to generate random anomalies with a variable length between 2 to 5 data points. Many of these were based on the anomalies described in [6].

- Addition or subtraction: these generate peaks and troughs types of anomalies, by adding (or subtracting) to the signal.
- Multiplication: similar to addition and subtraction, but the anomaly is proportional to the signal.
- Constant: the signal plateaus as a certain value.
- Noise: generates random noise that is within three standard deviations of the error of the measured data. This is not exactly an anomaly but rather used to test for false positives.

E. Central ARIMA Algorithm

Another potential algorithm that was tested for this paper was Central ARIMA. For this, the SARIMA models of the original training data (original model) and the inverted form (inverted model) are made. Then a window of length $2N + 1$ is taken where i (the point under analysis) is in the middle of this window. All points between i-N and i-1 are taken from the original model, while points i+1 to i+N are taken from the inverted model (where they are equivalent to points i-N to i-1, in this case). These points are summed, and the average will be the estimate for point i. The new value for point i is then used for anomaly detection analysis.

III. Results

A few anomaly detection techniques were tested. Initially, the system checked if the error from the measured data was normally distributed (i.e., within three standard deviations from the training data error). To improve it, analysis of a window average was applied as this means the anomaly would have an impact on the points surrounding it, facilitating detection. Another way of improving was to implement an integration of the errors, such that a large difference between the integrals at that point (determined by a threshold) would indicate an anomaly, even if the data is still within the bounds of three standard deviations.

A. Comparison Between Detection Methods

Table 2 shows the results when only the three standard deviations are checked. The TPR is low compared to current state-of-the-art IDS. One of the main issues was that this IDS had difficulties detecting constants since they sometimes are well within the threshold. For this reason, the window averaging was implemented, as it means that the IDS will detect the average approximating that value. The TPR significantly improved with windowing and integration, reaching over 80%. However, the trade-off for this is an increase in FPR and a larger memory requirement to store the points. In Figure 8, the ROC curves between three window lengths were compared, and it was found that the optimal window length is eight points. It showed similar TPR to the 10-point window but without any false-positive results. Although a window of six points also has FPR of zero, the TPR is much worse, at only 55%.

In Figure 9, a graph of error with an anomaly after 100 points is shown. The detection system, in this case, uses a 10-point window. It is clear that the anomaly was detected (indicated by the blue crosses). Figure 10 shows the integrals of the errors from Figure 9, and there is a difference of approximately 0.046 where the anomaly occurs. Figure 11 shows the same anomaly but this time using the 6-point window. Although the anomaly is still detected, only one point of it was identified, meaning that this IDS is not as robust. Note that the upper and lower bounds are slightly different between Figure 9 and Figure 11, due to the window average used (similarly, the integration is different between Figure 10 and Figure 12).

Three Standard Deviations					
FN	***TP***	***TN***	***FP***	***TPR***	***FPR***
27	53	12	7	66.25%	36.84%
Window of 10 points (+ Integration)					
8	34	2	6	80.95%	75%
Window of 8 points (+ Integration)					
9	36	5	0	80%	0%
Window of 6 points (+ Integration)					
18	22	10	0	55%	0%

Table 2 - Performance metrics for each IDS scheme tested.

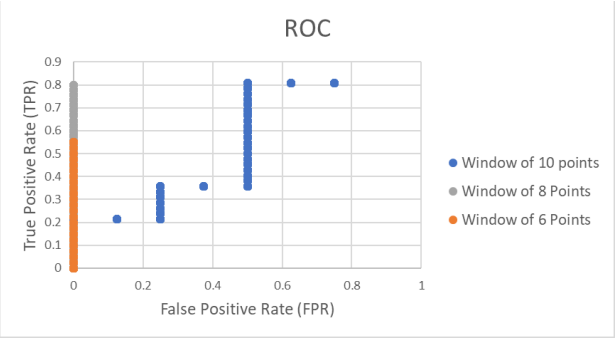

Figure 8 - ROC curve comparing the three window average lengths.

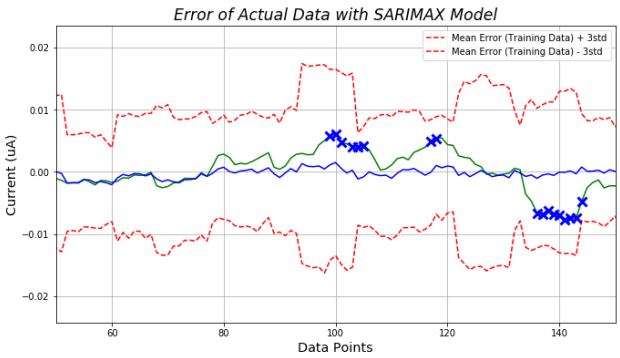

Figure 9 - Anomaly detection for a 10-point window.

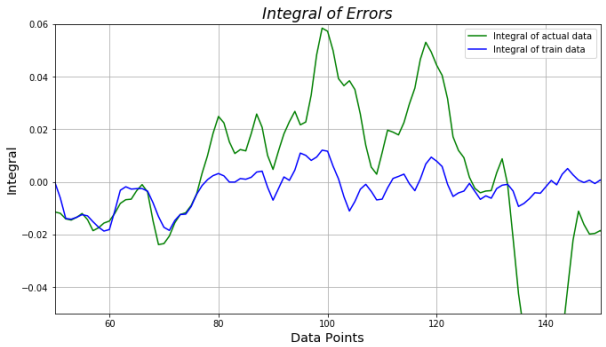

Figure 10 - Integrals of the error of the training data and the test data for 10-point window.

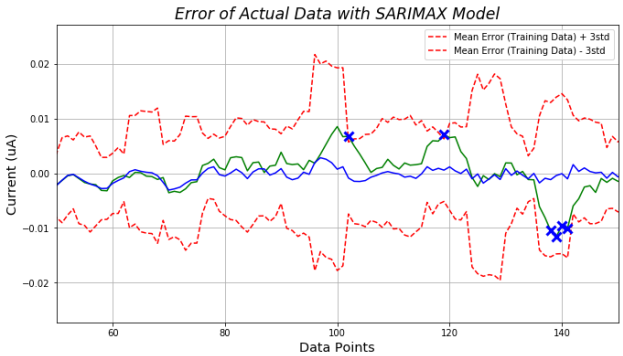

Figure 11 - Anomaly detection for a 6-point window.

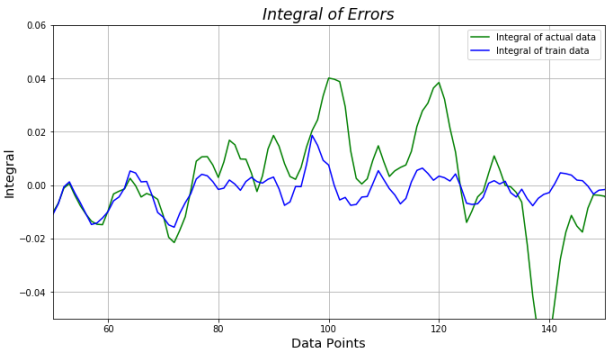

Figure 12 - Integrals of the error of the training data and the test data, for 6-point window.

B. *Comparison Between Types of Anomalies*

Table 3 shows the TPR for each anomaly type that was described in section II.D. It shows very clearly that this IDS can easily detect values added or subtracted from the signal. However, it has difficulties detecting constants and multiplications. The TPR for constants increased by over 10% for window averages of eight and ten points, which is a significant improvement. For multiplication, it is possible that some of the undetected anomalies are occurring in points where the signal is close to zero, resulting in a very small anomaly. The length of the anomaly occurrence also has an impact on the detection rate. For multiplication anomalies, 40% of undetected anomalies were three points long and 30% were two points long. This might indicate that the IDS is not suitable for short anomalies.

Three Standard Deviations			
	FN	***TP***	***TPR***
Multiplication	11	14	56%
Addition	2	13	86.67%
Subtraction	2	13	86.67%
Constant	12	13	52%
Window of 10 points (+ Integration)			
Multiplication	2	6	75%
Addition	1	8	88.89%
Subtraction	0	10	100%
Constant	5	8	61.54%
Window of 8 points (+ Integration)			
Multiplication	3	3	50%
Addition	1	13	92.86%
Subtraction	1	13	92.86%
Constant	4	7	63.64%
Window of 6 points (+ Integration)			
Multiplication	5	6	54.55%
Addition	5	6	54.55%
Subtraction	0	8	100%
Constant	8	2	20%

Table 3 - Total number of false negatives and true positives per anomaly for each IDS scheme tested.

C. Central ARIMA Results

For the Central ARIMA algorithm, the 4-point window averaging with integral analysis IDS scheme was used as this provided the best ROC curve for the SARIMA model. In Figure 13, it can be seen that Central ARIMA can accurately follow the trends of the data, even in new data that was not used during the training process.

Table 4 summarises the results for Central ARIMA. The TPR is better than all the previously tested detection methods for SARIMA. Compared to SARIMA with 10-point window, which had the best TPR but also had a very high FPR, the Central shows improvements for both TPR and FPR.

Total Results					
FN	***TP***	***TN***	***FP***	***TPR***	***FPR***
11	55	12	4	83.33%	25%
Per Anomaly Type					
		FN	***TP***	***TPR***	
Multiplication		2	13	86.67%	
Addition		3	10	76.92%	
Subtraction		2	24	92.31%	
Constant		4	8	66.67%	

Table 4 - Performance metrics for Central ARIMA.

D. Implementation

For the SARIMA model inference, it requires two coefficients to be stored, ten addition operations and eight constant multiplication operations. For the decision process, it requires two subtractions, some memory operations and integral of eight points. This shows that the implementation of this IDS is lightweight.

IV. Conclusion

The IDS proposed in this paper consists of an ARIMA model, which is used to find if the measured data is within the normal distribution and has a small difference in the integral for the window of points being analysed. After optimising the detection method, it was found that TPR can be as high as 80%. This IDS shows better or similar performance compared to other non-neural network algorithms, which have TPR ranging from 45.74% to 87.65% [4]. It also performed much better when compared to another lightweight algorithm in [6], which achieved 53.6% TPR. Our framework provides the potential of implementation on a low power device with a high detection rate.

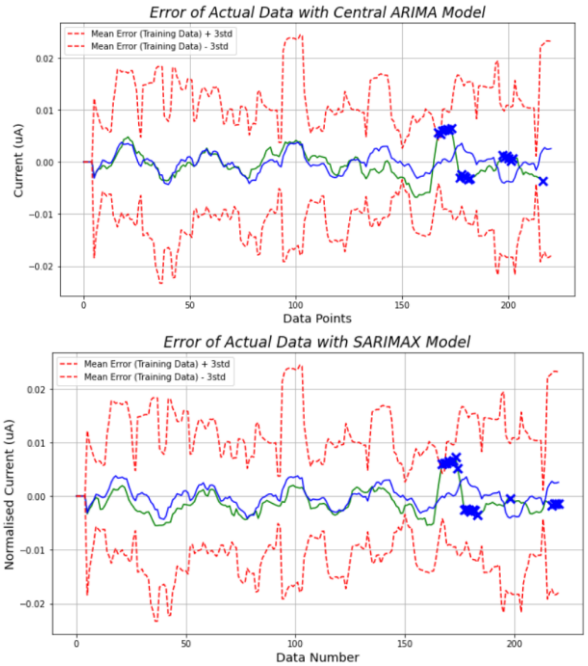

Figure 13 - Plot of errors using Central ARIMA for two different test data sets, with an anomaly at points 168-171. A blue x indicates a detected anomaly.

Further work will be done on implementing this IDS into hardware for demonstration and testing another algorithm (Central ARIMA). Our algorithms focus on anomaly detection within the computing unit. In the future, this IDS will be implemented into a device (PSoC 6 BLE) with RF and wireless communications with a focus on both computing and communication.

Acknowledgments

We would like to acknowledge the support of SFI INSIGHT Centre.

We would like to express our gratitude to Cypress Semiconductor for their continuous support and aid throughout this project.

References

[1] J. A. Stankovic, 'Research Directions for the Internet of Things', *IEEE Internet of Things Journal*, Vol. 1, pp. 3-9, 2014.

[2] A. AboBakr and M. A. Azer, "IoT Ethics Challenges and Legal Issues", *12th International Conference on Computer Engineering and Systems (ICCES)*, Cairo, Egypt, 1st Feb 2017, 2018

[3] R. Vinayakumar, K. P. Soman and P. Poonachandran, "Applying Convolutional Neural Network for Network Intrusion Detection", *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 13th – 16th September 2017, 2017

[4] Y. Liu, S. Liu and X. Zhao, "Intrusion Detection Algorithm Based on Convolutional Neural Network", *4th International Conference on Engineering Technology and Application (ICETA 2017)*, Nagoya, Japan, 29th – 30th June 2017, 2017

[5] M. Hasan, M. M. Islam, M I. I. Zarif and M. M. A. Hashem, 'Attack and Anomaly Detection in IoT Sensors in IoT Sites Using Machine Learning Approaches', *Internet of Things*, Vol. 7, 2019

[6] H. H. W. J. Bosman, A. Liotta, G. Iacca and H. J. Wörtche, "Anomaly Detection in Sensor Systems Using Lightweight Machine Learning", *2013 IEEE International Conference on Systems, Man and Cybernetics*, Manchester, UK, 13th - 16th October 2013, pp. 7-13, 2014

[7] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone and M. A. Spirito, "DEMO: An IDS Framework for Internet of Things Empowered by 6LoWPAN", *2013 ACM SIGSAC Conference on Computer & Communications Security*, Berlin, Germany, 4th - 8th November 2013, pp. 1337-1339, 2013

[8] M. Rahmatian, H Kooti, I. G. Harris and E. Bozorgzadeh, 'Hardware-Assisted Detection of Malicious Software in Embedded Systems', *IEEE Embedded Systems Letters*, Vol. 4, pp. 94-97, 2012

[9] V. B. Krishna, R. K. Iyer, and W. H. Sanders, "ARIMA-Based Modeling and Validation of Consumption Readings in Power Grids", 10th International Conference on Critical Information Infrastructures Security, Berlin, Germany, 5th – 7th October 2015, pp. 199-210, 2015

[10] W. R Kinney Jr., 'ARIMA and Regression in Analytical Review: An Empirical Test', *The Accounting Review*, Vol. 53, pp. 48-60, 1978

[11] H. Akaike, 'Fitting Autoregressive Models for Prediction', *Annals of the Institute of Statistical Mathematics*, Vol. 21, pp. 243-247, 1969