A Real-time Machine Learning-based GPS Spoofing Solution for Location-dependent UAV Applications

M. Nayfeh(1), J. Price(1), M. Alkhatib(1), K. Al Shamaileh(1), N. Kaabouch(2), and V. Devabhaktuni(3)

(1) Electrical and Computer Engineering Department, Purdue University Northwest, Hammond 46323, IN, USA
(2) School of Electrical Engineering and Computer Science, University of North Dakota, Grand Forks 58202, ND, USA
(3) Electrical and Computer Engineering Department, The University of Maine, Orono 04469, ME, USA
E-mail: kalshama@pnw.edu

2023 IEEE International Conference on Electro Information Technology (eIT) | 978-1-6654-9376-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/EIT57321.2023.10187344

***Abstract*—In this paper, a three-class machine learning (ML) model is implemented on an unmanned aerial vehicle (UAV) with a Raspberry Pi processor for classifying two global positioning system (GPS) spoofing attacks (i.e., static, dynamic) in real-time. First, several models are developed and tested utilizing a dataset collected in a previous work. This dataset conveys GPS-specific features, including location information. Models evaluations are carried out using the detection rate, F-score, false alarm rate, and misdetection rate, which all showed an acceptable performance. Then, the optimum model is loaded to the processor and tested for real-time detection and classification. Location-dependent applications, such as fixed-route public transportations are expected to benefit from the methodology presented herein as the longitude, latitude, and altitude features are characterized in the developed model.**

***Index Terms*— Global positioning system (GPS), machine learning (ML), spoofing classification, unmanned aerial vehicles (UAVs).**

I. Introduction

THE contributions of unmanned arial vehicles (UAVs) have exponentially increased in many applications, such as infrastructure inspection, traffic surveillance, landslide monitoring, 3D modeling of historic sites, urban planning, disaster management, communications for hard-to-reach areas, modern agriculture, transportation, and mailing delivery [1–3]. These critical applications leverage the grave importance of maintaining the cybersecurity and trustworthiness of UAVs against threats that may lead to irreversible consequences.

A wide range of cyberattacks endangers the UAVs integrity by targeting different components, such as the onboard sensors and modules [4]. One instrumental yet vulnerable component is the global positioning system (GPS) receiver, which uses unencrypted communications to obtain the UAV location. Due to this unencrypted nature, attacks including GPS spoofing can be launched to falsify positioning awareness, resulting in damaging private properties, public infrastructure, or compromising both the payload and technology [5–10]. Thus, different spoofing detection solutions were studied, such as autocorrelating received signals, exploiting spatial processing, comparing GPS data with that from other measurement units (e.g., inertia unit), using vision-based methods, and featuring artificial intelligence [11–19]. Although such solutions were successfully validated, they require hardware modifications or expensive computations, and face practical limitations, especially those developed in simulated environments.

In a previous work, multiple scenarios were established and demonstrated for two UAV-tailored spoofing attack types: static and dynamic [20]. These scenarios allowed for collecting a dataset comprising several GPS features and developing machine learning (ML) models for detecting and classifying GPS spoofing. The developed models were trained, validated, and tested and their performance was evaluated. The work presented herein differs from [20] in the following aspects:

1. Several ML classifiers are trained and tested with a subset of the features collected in [20]. These features are readily available, in real-time, without modifying the existing messaging protocol that facilitates the communications between the onboard GPS receiver, flight controller, and the processor (i.e., Raspberry Pi).
2. A spoofing detection and classification solution is hosted in the aforementioned processor via a routine that characterizes the optimum resulting ML classifier and is validated with received GPS data in real-time.

The remaining of this article is organized as follows: Section II details the development and evaluation of different ML models for detecting and classifying GPS spoofing. Section III elaborates on implementing the optimum model, enabling the UAV to classify, in real-time, two types of spoofing attacks. Finally, conclusion and future work are provided in Section IV.

II. Classifiers Development and Testing

In [20], three distinctive scenarios were established to collect a dataset capturing three cases of a UAV: 1) under no attack, 2) experiencing a static attack, and 3) experiencing a dynamic attack. The static attack forces the target to lock on a fake GPS signal with a location defined by the adversary that is different from the actual target location. On the other hand, in a dynamic attack, a GPS signal with moving coordinates along a predefined path is launched by the adversary to spoof the target's flightpath with a fake one, even though it may be stationary or moving along a different path. The resulting dataset conveyed twenty-seven GPS-specific signal features for developing location-dependent and location-independent spoofing classification schemes. The former scheme contains location-specific features (e.g., latitude, longitude, altitude); whereas the latter scheme does not use such features to develop

TABLE I
GPS FEATURES DESCRIPTION

Extracted Feature	Short Description	Unit
lat	Latitude in 1E–7	Degrees
lon	Longitude in 1E–7	Degrees
alt	Altitude in 1E–3 above sea level	Millimeters
hdop	Horizontal dilution of precision	–
vdop	Vertical dilution of precision	–
vel	GPS ground speed	Meters/second
COG	Course over ground	Radians
satellites	Number of satellites used	–

the ML classifiers. In this work, the location-dependent scheme is brought into practice as it does not require modifications to the messaging protocol (more details are given in Section III). In other words, the existing protocol forwards the location-dependent features together with the horizontal/vertical dilution of precision, velocity, course over ground, and number of seen satellites to the processor from the GPS receiver through the flight controller without modifying the protocol in the flight controller firmware. Table I summarizes the features used to implement the real-time detection and classification solution. It is noteworthy to point out that further details about the attack setups and the overall set of features can be found in [21].

The features in Table I are analyzed for correlation using the Spearman algorithm, which assumes nonlinearity among features. The resulting correlation depicted in Fig. 1 suggests weakly correlated pairs; thus, validating their use for model development without the need for feature elimination. Also, a standard scaling is applied such that $x_{ij}' = (x_{ij} - \mu_j)/\sigma_j$, where x_{ij}' is the scaled ith sample of the jth feature, and μ_j and σ_j are the mean and standard deviation of the sample values within the jth feature, respectively.

A total of 37,506 samples per feature are used to develop seven models; particularly, random forest (RF), k-nearest neighbor (KNN), multi-layer perceptron (MLP), logistic regression (LR), decision tree (DT), support vector machine (SVM), and naïve Bayes (NB). Model development entailed optimizing the associated hyperparameters using random search; whereas training and testing are carried out following a 10-fold cross-validation process. Training, validating, and testing the ML models are performed on a PC with 64-bit Windows 10, AMD® Ryzen™ 7 3700X CPU @ 3.6 GHz processor and 32 GB of DDR4-3600 MHz memory. Table II shows the optimized parameters for all the models; whereas Table III presents their evaluation scores, characterized by the detection rate (DR), Precision, Recall, F-Score (FS), false alarm rate (FAR), and miss detection rate (MDR). The DR is a measure of the percentage of correctly classified samples. The precision measures the classifier performance in classing negative samples as negatives and positive samples as positives. The recall measures the ability of the classifier to correctly predict all positive samples. The FS calculates the harmonic mean of the precision and recall. The FAR measures the probability of false detection. Finally, the MDR measures the probability of not detecting an attack. It is paramount to point out that the aforementioned evaluation metrics are obtained with the use of the following equations:

TABLE II
OPTIMIZED HYPERPARAMETERS

Category	Classifier	Parameter
Ensemble	**RF**	Quality of split criterion: Log loss Max. tree depth: 21 Min. number of samples at a leaf node: 33 Min. number of samples to split a node: 183 Number of trees: 129 Cost-complexity pruning parameter: 9.3E–3
Instance	**KNN**	Leaf size: 48 Number of neighbors: 12 Weight function: Distance Nearest neighbor comp. algorithm: Brute Distance metric: Euclidean Power parameter for distance metric: 4
	SVM	Norm used in penalty: L2 Loss function: Squared Hinge Dual optimization algorithm: False Max. number of iterations: 701 Regularization parameter: 9.479
Regularization	**LR**	Optimization: Stochastic AVG gradient Norm used in penalty: None Regularization parameter: 2.079 Max. number of iterations: 834
Tree	**DT**	Quality of split criterion: Log loss Max. tree depth: 21 Min. number of samples at a leaf node: 33 Min. number of samples to split a node: 183 Node split strategy: Best Cost-complexity pruning parameter: 9.3E–3
Neural network	**MLP**	Optimization: Limited-memory Broyden–Fletcher–Goldfarb–Shanno Hidden layers and neurons: 1 and 232 Activation function: Logistic Max. number of iterations: 866 L2 regularization term strength: 93.7E–3 Early stopping: True
Bayesian	**Gaussian NB**	Smoothing stability parameter: 23.96E–3

$$\text{Detection Rate (DR)} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.a}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1.b}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1.c}$$

$$\text{F1-score (FS)} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.d}$$

$$\text{False Alarm Rate (FAR)} = \frac{FP}{FP + TN} \tag{1.e}$$

$$\text{Misdetection Rate (MDR)} = \frac{FN}{TP + FN} \tag{1.f}$$

where *TP*, *TN*, *FP*, and *FN* are the positive samples predicted as positive (i.e., true positive), negative samples predicted as negative (i.e., true negative), negative samples predicted as positive (i.e., false positive), and positive samples predicted as negative (i.e., false negative), respectively. Table III illustrates that RF has the optimum performance with a DR of 91.78%, FAR of 5.50%, and MDR of 9.66%, these scores are associated with a reasonable prediction time of 43.36 ms. On the other hand, LR, DT, and SVM have demonstrated the lowest prediction times among all the developed models at the expense of either lower DRs and/or higher MDRs.

TABLE III
ML MODELS EVALUATION SCORES (PT: PREDICTION TIME)

Model	DR (%)	Precision (%)	FS	FAR (%)	MDR (%)	PT (ms)
RF	**91.78**	**92.44**	**0.91**	**5.50**	**9.66**	**43.36**
KNN	85.11	86.33	0.84	9.01	22.96	109.28
MLP	88.70	89.35	0.88	7.42	13.24	22.36
LR	66.06	73.28	0.67	19.02	27.14	0.29
DT	89.26	89.64	0.89	6.64	16.68	0.29
SVM	70.47	79.77	0.72	15.48	24.03	0.29
NB	84.95	86.74	0.84	8.38	23.41	1.37

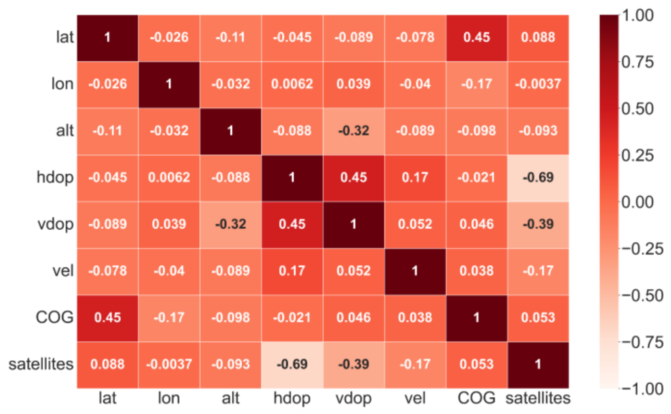


Fig. 1. Spearman correlation of the features described in Table I.

The resulting optimum model (i.e., RF) is also tested on the Raspberry Pi processor to compare its training and prediction times with those achieved with the PC. Such a processor is equipped with 64-bit ARM Cortex-A72 quad-core CPU @ 1.5 GHz and 1 GB of memory. Table IV shows that the Raspberry processor has a prediction time of 671.74 ms, which is significantly higher than its counterpart obtained with the PC. Nevertheless, the reported times are from predicting the entire testing dataset, which consists of 9,600 samples. Hence, the resulting prediction time per sample with the Raspberry Pi is around 70 µs, facilitating real-time detection and classification.

III. IMPLEMENTATION AND EVALUATION

In this section, the RF model developed in Section II is implemented on an open-source Clover drone from COEX, which is equipped with a Pixracer flight controller that uses PX4 autopilot and a u-blox M8 GPS receiver [22–24]. The RF model is loaded to the onboard processor, which is interfaced with the flight controller via a USB connection. The micro air vehicle link (MAVLink) messaging protocol is used to live stream the data collected by the GPS receiver module through the flight controller to the processor. In particular, the *GPS_RAW_INT* message is utilized to stream the eight features detailed in Table I. Fig. 2 shows the configuration of the hardware tools and their functions in validating the real-time classification procedure, which can be summarized as follows:

1. The values of μ_j and σ_j for each feature are stored in the classification model (hosted at the processor) to scale the received live samples. Such values are obtained from the training dataset as detailed in Section II.
2. The MAVLink *GPS_RAW_INT* messages are streamed to a preprocessing routine for the extraction of features. Here, feature samples are scaled with μ_j and σ_j and are fed to the hosted RF classification model.
3. The predicted class is appended to an array to calculate live accuracy rates. Classes 0, 1, and 2 are assigned for no attack, static attack, and dynamic attack, respectively. Such accuracies are calculated as follows:

$$\text{Accuracy (\%)} = \frac{\text{No. of correctly predicted samples}}{\text{Total No. of predicted samples}} \quad (2)$$

4. The actual attack, its classification, and the live accuracy are printed and saved for further processing.

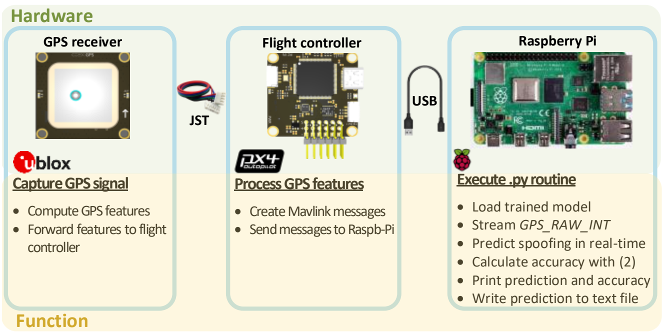


Fig. 2. The hardware devices and their functions in detecting and classifying the GPS spoofing attacks.

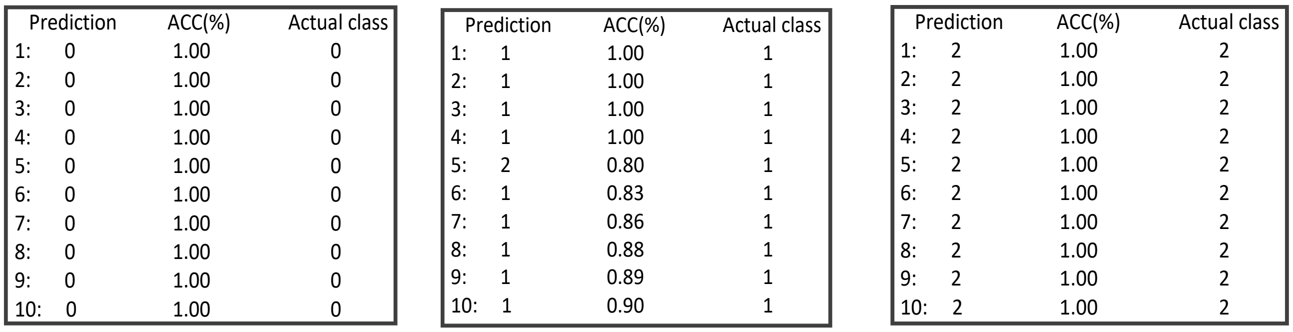
```
Algorithm: Real-Time GPS Spoofing Classification
Define Class_Actual = 0, 1, or 2

1: Procedure: Real_Time_GPS_Spoofing_Classification
2: Load the values of $\mu_j$ and $\sigma_j$ for each feature
3: Load trained MLmodel
4: File              =   open(Output_File.txt)
5: Start live-streaming GPS_features
6: Standardize the features using $\mu_j$ and $\sigma_j$
7: Prediction        =   MLmodel.predict(GPS_Features)
8: Class.append(Prediction)
9: DR                =   DetectionRate(Class_Actual, Class)
10: Loop: for each Prediction do
11:          print Prediction to console window
12:          print DR to console window
13:          File.write(Prediction, DR, Class_Actual)
14: end for
15: end Procedure
```

Fig. 3. Pseudocode of the GPS spoofing classification procedure.

TABLE IV
PC AND RASPBERRY COMPARISON (TT: TRAINING TIME, PT: PREDICTION TIME)

Device	TT (ms)	PT (ms)
PC	1515	43
Raspberry Pi	18602	671



Prediction		ACC(%)	Actual class
1:	0	1.00	0
2:	0	1.00	0
3:	0	1.00	0
4:	0	1.00	0
5:	0	1.00	0
6:	0	1.00	0
7:	0	1.00	0
8:	0	1.00	0
9:	0	1.00	0
10:	0	1.00	0

(*a*)

Prediction		ACC(%)	Actual class
1:	1	1.00	1
2:	1	1.00	1
3:	1	1.00	1
4:	1	1.00	1
5:	2	0.80	1
6:	1	0.83	1
7:	1	0.86	1
8:	1	0.88	1
9:	1	0.89	1
10:	1	0.90	1

(*b*)

Prediction		ACC(%)	Actual class
1:	2	1.00	2
2:	2	1.00	2
3:	2	1.00	2
4:	2	1.00	2
5:	2	1.00	2
6:	2	1.00	2
7:	2	1.00	2
8:	2	1.00	2
9:	2	1.00	2
10:	2	1.00	2

(*c*)

Fig. 4. Model output file in the case of exposing it to live data samples of (*a*) clean (i.e., no attack), (*b*) static attack, and (*c*) dynamic attack.

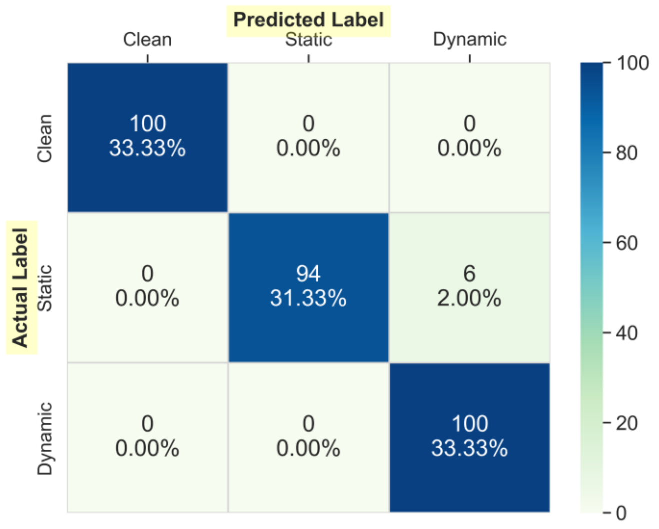


Fig. 5. Confusion matrix of the real-time classification test.

Fig. 3 depicts a pseudo code for the real-time classification procedure, which is tested with live data presenting classes 0, 1, and 2. The attacks are prepared using the open-source library GPS-SDR-SIM and launched with an Ettus B210 software defined radio [25, 26]. Static spoofing is created by specifying the fake coordinates, which are fed to GPS-SDR-SIM along with the ephemeris data file that conveys precise information about the satellites orbits. Then, GPS-SDR-SIM generates the attack file, which consists of the bit stream of the spoofing signal. This generated file is loaded into GNURadio, which is the companion software for the B210 that is used to launch the attack. On the other hand, dynamic spoofing is created by generating a user motion file using SatGen3 [27]. This file contains the moving coordinates of the spoofed flightpath along with the altitudes and velocities. This motion file is used in GPS-SDR-SIM in conjunction with the ephemeris data to generate the attack file, which is then launched via the radio. Elaborations on both attacks with greater details can be found in [20, 21].

Sample output files of the classification model are shown in Fig. 4. Each class is executed individually for a length of 100 samples. With these samples, the model yielded a DR of 98.67%. Also, this live real-time testing achieved an excellent overall FAR of 1.00% and MDR of 2.00% as depicted from the confusion matrix shown in Fig. 5. The values of such metrics will be in proximity to those reported in Table III as more samples are fed to the model.

IV. Conclusion

A real-time ML model for classifying static and dynamic GPS spoofing attacks is implemented on an open-source UAV with an onboard Raspberry Pi processor. The training data is obtained from a previous work. Only the features that do not impose modifications on the messaging protocol are used to develop the model. These features include the 3D coordinates; therefore, this study is best suited for fixed route applications. The resulting model is evaluated and tested, and the associated DR, FS, FAR, and MDR are 91.78%, 0.91, 5.50%, and 9.66%, respectively. The open-source nature of the hardware and software used herein justify the adoption of the proposed model in a wide variety of readily available UAVs. Future work entails exploring GPS spoofing attack mitigation techniques.

Acknowledgment

This research is funded by the National Science Foundation, Secure and Trustworthy Cyberspace under Award no. 2006662.

References

[1] G. Albeaino, M. Gheisari, and B. Franz, "A systematic review of unmanned aerial vehicle application areas and technologies in the AEC domain," *Journal of Information Technology in Construction*, vol. 24, pp.381, 2019.

[2] C. Norasma, M. Fadzilah, N. Roslin, Z. Zanariah, Z. Tarmidi, and F. Candra, "Unmanned aerial vehicle applications in agriculture," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 506, 2019.

[3] N. Mohamed, J. Al-Jaroodi, I. Jawhar, A. Idries, F. Mohammed, "Unmanned aerial vehicles applications in future smart cities," *Technological Forecasting and Social Change*, vol. 153, 2020.

[4] A. Rugo, C. Ardagna, N. El Ioini, "A security review in the UAVNet era: Threats, countermeasures, and gap analysis," *ACM Computing Surveys*, vol. 55, no. 21, pp. 1-35, 2023.

[5] Electromagnetic Interference Behind Darling Harbour Drone Crash. [Online]. Available: https://australianaviation.com.au/2022/06/electromagnetic-interference-behind-darling-harbour-drone-crash/

[6] SkyJack Software Finds and Hijacks Drones. [Online]. Available: https://www.pcmag.com/news/skyjack-software-finds-and-hijacks-drones

[7] HK$1 million in damage caused by GPS jamming that caused 46 drones to plummet during Hong Kong show. [Online]. Available: https://www.scmp.com/news/hong-kong/law-and-crime/article/2170669/hk13-million-damage-caused-gps-jamming-caused-46-drones

[8] Drone Crash Due To GPS Interference in U.K. Raises Safety Questions. [Online]. Available: https://www.forbes.com/sites/davidhambling/2020/08/10/investigation-finds-gps-interference-caused-uk-survey-drone-crash/?sh=57b389bd534a

[9] Drones crash during light display at lantern festival. [Online]. Available: https://www.taipeitimes.com/News/taiwan/archives/2020/02/24/2003731529

[10] D. da Silva, "GPS jamming and spoofing using software defined radio," Department Of ISTA, University Institute of Lisbon, 2017.

[11] F. Rothmaier, Y. Chen, S. Lo, and T. Walter, "GNSS spoofing detection through spatial processing," *Navigation*, vol. 68, no. 2, pp. 243–258, 2021.

[12] A. Khan, N. Iqbal, A. Khan, M. Khan, and A. Ahmad, "Detection of intermediate spoofing attack on global navigation satellite system receiver through slope based metrics," *The Journal of Navigation,* vol. 73, no. 5, pp. 1052–1068, 2020.

[13] S. Wang, J. Wang, C. Su and X. Ma, "Intelligent detection algorithm against UAVs' GPS spoofing attack," *IEEE 26th International Conference on Parallel and Distributed Systems* (*ICPADS*), Hong Kong, China, pp. 382–389, 2020.

[14] D. Akos, "Who's afraid of the spoofer? GPS/GNSS spoofing detection via automatic gain control (AGC)," *Navigation*, vol. 59, no. 4, pp. 281–290, 2012.

[15] Y. Qiao, Y. Zhang and X. Du, "A vision-based GPS-spoofing detection method for Small UAVs," 13th *International Conference on Computational Intelligence and Security* (*CIS*), Hong Kong, China, pp. 312–316, 2017.

[16] N. Xue, L. Niu, X. Hong, Z. Li, L. Hoffaeller, and C. Pöpper, "DeepSIM: GPS spoofing detection on UAVs using satellite imagery matching," *Annual Computer Security Applications Conference* (*ACSAC* '20), NY, USA, pp. 304–319, 2020.

[17] P. Jiang, H. Wu, C. Xin, "DeepPOSE: Detecting GPS spoofing attack via deep recurrent neural network," *Digital Communications and Networks*, 2021.

[18] Y. Dang, C. Benzaïd, T. Taleb, B. Yang and Y. Shen, "Transfer learning based GPS spoofing detection for cellular-connected UAVs," *2022 International Wireless Communications and Mobile Computing* (*IWCMC*), Dubrovnik, Croatia, pp. 629–634, 2022.

[19] S. Semanjski, A. Muls, I. Semanjski, and W. De Wilde, "Use and validation of supervised machine learning approach for detection of GNSS signal spoofing," *2019 International Conference on Localization and GNSS (ICL-GNSS)*, Nuremberg, Germany, pp. 1–6, 2019.

[20] M. Nayfeh, Y. Li, K. Al Shamaileh, V. Devabhaktuni, and N. Kaabouch, "Machine Learning Modeling of GPS Features with Applications to UAV Location Spoofing Detection and Classification," *Computers & Security*, vol. 126, 2023.

[21] Resources. [Online]. Available: https://github.com/mnayfeh/gps_spoofing_detection

[22] COEX. [Online]. Available: https://coex.tech/clover.

[23] COEX Pix. [Online]. Available: https://clover.coex.tech/en/coex_pix.html

[24] UBX-M8030 series. [Online]. Available: https://www.u-blox.com/en/product/ubx-m8030-series

[25] GPS-SDR-SIM. [Online]. Available: https://github.com/osqzss/gps-sdr-sim

[26] USRP B210. [Online]. Available: https://www.ettus.com/all-products/ub210-kit/

[27] SatGen Software. [Online]. Available: https://www.labsat.co.uk/index.php/en/products/satgen-simulator-software