TinyML meets IoBT against Sensor Hacking

Raushan Kumar Singh
Department of Computer Science Engineering
IIT Ropar, Punjab, India
raushan.21csz0020@iitrpr.ac.in

Sudeepta Mishra
Department of Computer Science Engineering
IIT Ropar, Punjab, India
sudeepta@iitrpr.ac.in

***Abstract*—Modern technology is advancing on many different levels, and the battlefield is no exception. India has 15000 km of lengthy land borders shared with many other neighboring countries, and only 5 of the 29 states in India do not have any shared international borders or coastlines. Wire fences and conventional sensor-based systems are used to protect terrestrial borders. Wire fences, being the only line of defense against intrusions at most unmanned borders, result in frequent cases of unreported incursion, smuggling, and human trafficking. Typically, intruders cut the fence to gain access to Indian land, and sensor-based systems are prone to false alarms due to animal movements. We propose combining the intelligence of Tiny Machine Learning (TinyML) with the communication capability of IoT to make borders safer and intrusion more challenging. To learn the typical fence movements from natural causes, we use TinyML. Our learning technique is created explicitly to differentiate between regular fence movement and suspicious fence disturbance. The system is efficient enough to detect metal fence cuts and trespassing carefully. With the aid of online learning environments, the sophisticated TinyML microcontroller's built-in accelerometer can differentiate between different movement patterns. To identify the most effective defense against sensor-level attacks, we conducted tests to gauge the tolerance levels of conventional microcontroller sensor systems against TinyML-powered microcontrollers when exposed to Electromagnetic Pulse (EMP) based sensor hacking attempts. To the best of our knowledge, this is the first research conducted for the Identification of the best suite sensor system for high-precision Internet of Battlefield Things (IoBT) applications. During the real-time model test, the system is found to be 95.4% accurate and readily deployable on TinyML microcontrollers.**

***Index Terms*—TinyML, IoBT, EMP, microcontroller.**

I. Introduction

IOBT is a field that utilizes massive technological innovations to procure human life and territorial protection. With the advancement of technology used on the battlefield, we can efficiently utilize the available resources to come up with an upper hand against cross-border terrorism. India has one of the longest borders worldwide, making it crucial to develop technological innovations to keep the country's border secure for such a long stretch. The development of a multi-sensing structure utilizing Wireless Sensor Network (WSN) has started in response to recent acts of terrorism and border intrusion that have become very frequent in many parts of the world. Border surveillance and human infiltration detection systems are developed using sensor fusion. The border areas of various geographical types are considered, including flat area borders, riverside borders, and places with vegetation. Multi-sensor and vision systems have been used for intruder direction recognition and surveillance for border security [5]. Some systems effectively use infrared cameras to spot suspicious activity in border regions. The invader is located using automatic spotlights and laser weapon equipment. In addition, a sound-based strategy is used to identify any unusual sounds and add an extra layer of security against unauthorized border crossings. Researchers have also devised border security systems that use thermal imaging for higher accuracy and detection. Detecting intruders beyond the range of sensors, especially in adverse weather, becomes a benefit of using thermal cameras [4].

Due to the usage of multisensor coordination for sensor fusion in various applications, sensor hacking has become a prominent issue. Sensor hacking involves external forces such as lasers, sound waves, EMPs, and noise-disrupting sensor readings. This interference leads to the processing unit receiving faulty sensor data, resulting in issues with the overall system's interpretation and decision-making capabilities. The primary challenge is identifying appropriate hardware units (microcontrollers & sensors) for security-constrained applications, such as avionics, space science, medical electronics, and other high-precision uses. Most digital setups nowadays often contain additional sensors to optimize their accuracy and durability. Sensor fusion governs the combined use of several sensors to exploit their features to solve the same issue or gather collective information. Previous studies have demonstrated that EMP attacks indeed have the capability to modify the readings obtained from sensors [7, 15, 16]. Being the first layer of the whole automation mechanism, its weakness has the potential to bring the entire system tumbling. We focus on sensor hacking against our TinyML module to check for any potential flaw in the output. We created a medium power Electromagnetic Pulse generator to show the effects of sensor hacking by wirelessly inducing a charge on the sensors and manipulating their output. TinyML and sensor hacking are both in their beginning phases presently; therefore, it will be incredibly advantageous for researchers in the future to learn more about both from our proposed

Workshop on Security and Privacy in Standardized IoT (SDIoTSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-6-9
https://dx.doi.org/10.14722/sdiotsec.2024.23010
www.ndss-symposium.org

work. Electronic circuits and sensors are known to perform differently from how they are intended to under the influence of electromagnetic interference. During our experiment, we utilize this property of Electromagnetic Interference (EMI) to mislead the microcontrollers and read the false value as the original one [16].

Our paper compares the EMI tolerance levels of traditional sensor microcontroller systems with those equipped with machine-learning capabilities. Our emphasis lies in evaluating their performance within IoBT applications, specifically in comparing the vulnerability of sensor-secured border fences to attacks on microcontrollers powered by TinyML.

We summarize our proposed experimental analysis as follows.

- We test hardware-level machine learning intelligence in the form of TinyML for Battlefield applications using Edge Impulse and test EMI attacks against that.
- We exert a mid-level EMP attack on an Arduino Nano's tethered Inertial Measurement Unit (IMU) sensor (MPU6050) without causing physical damage and observe the changes in its output during the attack.
- We test an EMI attack over Arduino Nano 33 BLE Sense and its inbuilt IMU sensor without physically damaging them and evaluate its result variation against the attack.
- We compare both the above results and identify the better suite for high precision and fault intolerant setups against EMI attacks like IoBT.
- Our envisaged work goes a little farther. We perform multiple stress tests to investigate the potential impact of Sensor Hacking over TinyML.
- We attempt to mitigate electromagnetic interference with the hardware-level intelligence and its perception layer security.

Countries that are radical in the War field have huge Research & Development (R&D) budgets to facilitate new inventions to increase their nation's security and strengthen offensive war fields. At present, the mainstream fields like robotics, biomedical, avionics, space science, etc., use hardware-level intelligence (TinyML) to incorporate Machine learning models to make smart and effective decisions, but for defensive battlefield applications, our proposed Internet of IoBT integration with TinyML approach is probably the first.

The essential observations from earlier works are covered in our next section so that we can extend our study based on earlier research footprints.

II. IoBT: Background and Observations

IoBT, or the Internet of Battlefield Things, is a specialized subset of the Internet of Things(IoT) designed to optimize and strengthen a unified military force. It achieves this by harnessing the potential of sensors and microcontrollers capable of efficiently utilizing edge or cloud computing capabilities. The fundamental components of the IoBT architecture include wireless connectivity between sensors and a central control and reporting unit. However, deploying Machine Learning (ML) models poses challenges regarding system requirements, power consumption, and intricate designs, hindering seamless field deployment. To address these obstacles, TinyML has been introduced. This technology focuses on compressing models to align with the architecture and capabilities of resource-constrained computing devices, such as Arduino Nano 33 BLE Sense modules. This adaptation enables real-time implementation in the dynamic settings of IoBT, overcoming the limitations associated with traditional ML deployment. Although there has been some research on battlefield incursion, there is still much need for improvement. The key works in the field are listed here, along with their development.

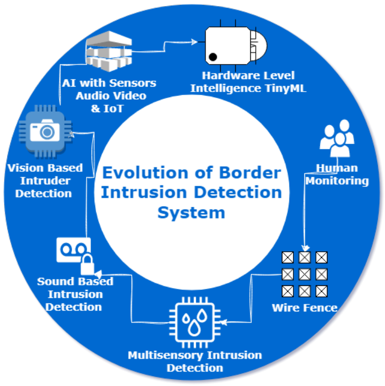

AI with Sensors Audio Video & IoT
Hardware Level Intelligence TinyML
Human Monitoring
Wire Fence
Multisensory Intrusion Detection
Sound Based Intrusion Detection
Vision Based Intruder Detection
Evolution of Border Intrusion Detection System

Fig. 1: Evolution of Border Security Systems

As depicted in Fig. 1, At the onset of border protection, the only barriers protecting the borders were armed soldiers and wire fences. Further advancements in this domain included early sensor, sound, and vision-based technologies, which laid the groundwork for more advanced sensor and audio-visual systems. Artificial intelligence-based solutions are in high demand due to their high precision and low overhead costs.

TABLE I: Different Technologies Against Battlefield Intrusion

S.No.	**Development and Approach**		
	Approach	**Year**	**Key Component**
1.	Spartan Sensor for Intrusion alert [2]	1990	Sensor
2.	Guided Radar Sensors [10]	2007	Sensor
3.	Wide Area Surveillance Radars [6]	2008	Sensor/LoS
4.	Intrusion-Detector and Firing [13]	2010	Image Processing
5.	IoT System Modeling [1]	2017	UML Modelling
6.	Automated Border Check gates [8]	2018	FMT
7.	WSN & Multi-sensing framework[5]	2019	WSN
8.	Ship Detection using CNN [3]	2021	CNN

The power plant was used first to deploy the sensor-based intrusion detection system and served as the test subject for future secure premises like military bases, precious and dangerous object storage units, national security buildings, etc. The overall development of protecting military and civilian properties with advanced sensors in the Battlefield domain started there. The real-time intrusion detection was evaluated during the trial to ensure that the sensors, known as Sparton sensors, would be suitable for more complicated deployment in the future. Magnetic, seismic, acoustic, infrared, pressure sensors, etc., were the main sensors involved [2]. A substantially improved version of earlier studies, the guided radar sensors-based security systems for battlefields took around two decades to build. Compared to its predecessors, this system proposed an ultra-wide spread spectrum range guided radar that was both highly developed and extremely affordable. It essentially served as a ground surveillance radar that could be quickly deployed in various types of soil [10]. Later, cutting-edge technologies like Wide Area Surveillance Radars, which considered extra design variables to make intrusion detection far more precise and easily deployable, improved the intrusion detection systems [6]. The enhancements in the following decade, improvements in design, accuracy, cost, and other factors led to the state it is in today [1, 3, 8, 13].

With the advancement of technologies, the precision of the intrusion detection system as a whole significantly improved. The first systems depended on people; hardware systems were later developed from those (Sensors and Actuators). With the aid of software-level intelligence in computer vision and image processing, these hardware-based systems started to have greater accuracy and cheaper implementation costs. While cameras were employed to detect unauthorized entries, they were already prone to susceptibility to dust, weather conditions, and camouflage attacks. As technology advanced, it became necessary to use machine learning for very accurate detection but at the cost of high-speed internet connectivity to run ML models remotely on distant servers. The delay in response due to network delay started pulling down the benefits of machine learning during field operations. To overcome this, we required field-level intelligence by means of TinyML-compatible microcontrollers to make IoBT devices self-sufficient standalone units. In our proposed work, we utilize ML intelligence on the battlefield to reduce the likelihood of false alarms, processing latency, deployment complexity, etc.

The next section provides the paper's system design methodology and explanation. It provides a comprehensive overview of all the proposed tasks and necessary components.

III. TinyML Design Methodology

The planned task and its constituent parts are divided into subsections for ease of comprehension. The components are linearly developed and used concerning one another. The evolution of the entire system follows the linear application of the design process shown here.

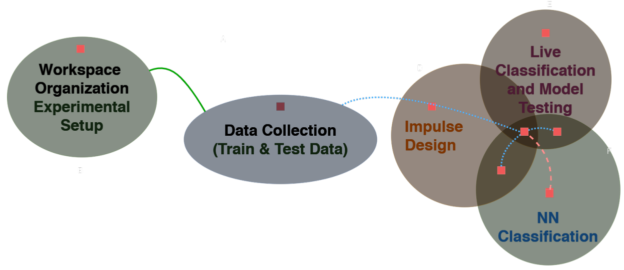

Workspace Organization Experimental Setup
Data Collection (Train & Test Data)
Impulse Design
Live Classification and Model Testing
NN Classification

Fig. 2: Design Overview

A. Workspace Organization

To train and test our model, we set up a near-realistic arrangement that should combine our suggested smart fencing with the standard cable-based wire fence found in most border zones.

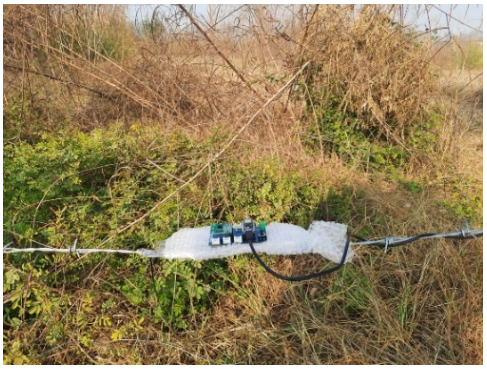


Fig. 3: Model Train and Test Environment

However, if the border area's fence design differs, our model can be retrained for different situations. The unifying denominator in all conceivable scenarios is that there should be a very low possibility of human passing without altering the fence position. Our trained model detects any deviation from normal movement. Our experimental setup consists of the fence wire connected and tied to two stationary spots in a large open area. To record natural interventions and real-world situations, we ensure that our system receives a variety of difficulties while being trained as a model. This increases the consistency and precision of our models, enabling them to operate well when applied to border regions. Finding the location on the battlefield where fencing may be most vulnerable is the first step in the implementation process, especially in areas with blind spots. In some unique circumstances, we might need to switch the traditional metal wire fences for our suggested model-friendly fences, which will increase the accuracy of spotting unauthorized access.

B. Data Collection

Edge Impulse is a world-famous platform for ML deployment in edge devices. It offers ML applications a futuristic perspective and lowers the barrier to cutting-edge ML implementations on end devices [14]. Edge Impulse offers end-to-end services for high-quality applications, starting with data collecting and ending with deployment on an Arduino Nano 33 BLE Sense module [12].

Establishing a connection between the edge impulse cloud and the data collection equipment is the first step in the data collection process [9]. A development board, a smartphone, a computer, a direct data upload, or a device that integrates the cloud can all be used as the data input device. There are numerous options for selecting an input device with a seamless connection. In our scenario, we utilize data-collecting inputs from a specific phone, namely the IQOO NEO 7 5G. This is because the data collected from this phone is promptly transmitted to the Edge Impulse web service. Addressing concerns related to fluctuations and noise in accelerometer data during readings, we implement filters to enhance stability, as mentioned in the impulse design subsection. However, any other sensor setup with a suitable communication interface can be used.

No further pairing processes or application installation are needed for this. Even sensor data gathered from a phone can be directly mapped with the specifications of a microcontroller architecture in the subsequent steps.

We gathered a dataset that captures the movement of a fence in response to wind across all possible axes. The string, left to move naturally, was also subjected to controlled vibrations to simulate strong winds. Data was collected during both the simulated and actual strong wind scenarios, carefully verifying motions along the X, Y, and Z axes to encompass all potential natural movements. This dataset is labeled as ”Wind.”

We added external factors like slight shocks and vibrations that are very common for this type of real-time setup, enabling our model to face the actual test scenario. 20% of the training data set is moved to the testing data set to create a stable knowledge base.

Sensors and microcontrollers are often deployed in remote and challenging environments, relying primarily on battery and solar power. Their operation is constrained by the need to conserve power for extended periods and withstand adverse weather conditions such as extreme temperatures, humidity, dust, and moisture. Consequently, minimizing power consumption is crucial for prolonged functionality. This includes limiting the number of samples taken to keep the machine-learning models lightweight. Heavy models impose higher computational demands, which can lead to increased power consumption and sluggish performance of the processor.

TABLE II: Data Sample Collection

Data-set Details			
Label	**Type**	**Length**	**No. of Samples**
Wind	Train	8 Min 30 Sec	51
Intrusion	Train	8 Min 30 Sec	51
Wind	Test	2 Min 10 Sec	13
Intrusion	Test	2 Min10 Sec	13
Total Initial Data Set		**21 Min 20 Sec**	**128**

C. Impulse Design

The impulse design essentially serves as our own machine learning system’s pipeline. Data intake, processing, and learning systems comprise the impulse design’s subdivisions.

The time series data block refers to the adjustments in the input data samples to be fit for further processing. When the window size is increased, it automatically raises the feature size to provide the learning subsystem with more information, whereas window size relates to the raw size of features utilized for training. Frequency is automatically determined based on the type of supplied data. It refers to how frequently the data set’s values were taken. When there is no raw data, the system automatically adds zero if we tick on Zero pad data.

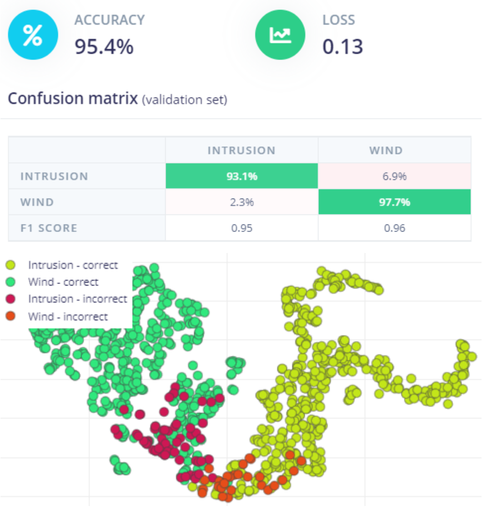
The Spectral Analysis block executes the Digital Signal Processing (DSP) operation to extract the characteristics that the learning model depends on. Edge Impulse consistently suggests the best block based on our input data. The processing block is the Neural Network(NN), which trains using our raw data. Different learning blocks are suggested based on our input, and we choose NN classification, but there are also possibilities to include Anomaly Detection using K-Means and Regression.

The extraction of the signal’s power and frequency features is the main focus of the spectral feature section. In order to make our input signals considerably more suitable for further processing, we can employ filters like low/high pass to smoothen data to the system. In our case, a low pass filter provides much smoother graphs and accurate results.

D. Neural Network Classification

Our categorization is based on Keras, which serves as an interface for the TensorFlow library and is in charge of training and interacting with Neural Network(NN) models. NN takes input data and outputs a probability score indicating which category of training dataset the test data belongs to. The neuron is the fundamental unit of each layer that makes up a NN. Training data that will be applied to predictions are delivered to the neurons. The weight of the neuron connections is then readjusted depending on the outcome after the anticipated outputs are compared to the accurate outcomes. Up till accurate predictions are made, this process is repeated.

The training cycle is the number of times the algorithm completes the learning cycle and updates its model parameters, and how fast the NN learns depends on the learning rate.



ACCURACY
95.4%

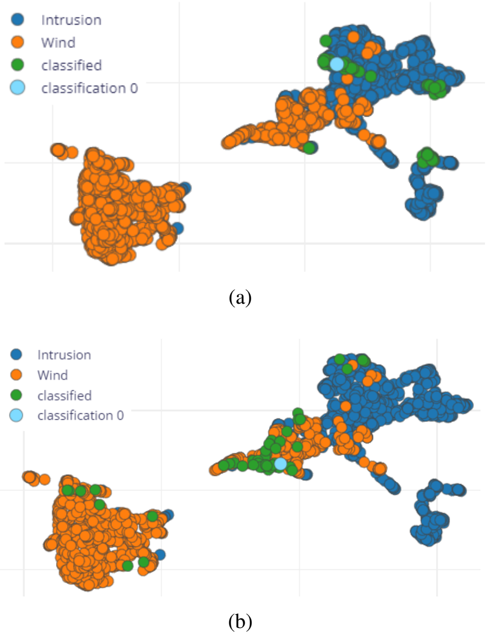
LOSS
0.13

Confusion matrix (validation set)

	INTRUSION	WIND
INTRUSION	93.1%	6.9%
WIND	2.3%	97.7%
F1 SCORE	0.95	0.96

Fig. 4: Training Performance and Confusion Matrix

As depicted in Fig.4, the confusion Matrix and training performance indicate how many correct/incorrect responses our model produces. Our result shows for 2.3% time, “Wind” is predicted as intrusion, and 6.9% time, “Intrusion” is predicted as Wind. Overall, our model predicts with 95.4% accuracy the correct samples, and that is highly acceptable



(a)

(b)

Fig. 5: Live Test Scenario : (a) Intrusion Test Result, (b) Wind Test Result

for learning models and real-time implementations. The green and yellow show Wind and Intrusion data being correctly predicted, whereas red and orange show incorrectly predicted data.

TABLE III: On Device Performance

Device Resource Consumption		
1.	Inferencing	1 ms
2.	Peak RAM Usage	1.7 KB
3.	Flash Usage	16.3 KB

The on-device performance measures how well our model will function on a real device and the resources needed to operate the model effectively. This aids in planning the hardware system that will allow us to run the model smoothly and evenly.

E. *Live Classification and Model Testing*

We can connect our device or phone to test our model against real-time data during the live classification. We can quickly validate the model and expect a similar performance during the field test. We can use existing test data or provide live data to test from the device directly. To test the efficiency of our model, we gave live accelerometer input using the phone. The phone's movement was imitated as the "Intrusion" movement input samples. This resulted in matching 47 sample counts displayed in Fig.5(a). Green color matches are for our input accelerometer behavior prediction, which is similar to Intrusion. This results in the successful detection of intrusion attempts using our model. In the second instance displayed in Fig.5(b), we provided live data from a phone with a similar motion to the "Wind" dataset to test our model, and 41 counts matched with the wind model, 4 counts with the "Intrusion" model, and 2 uncertain. Green color matches are for our input accelerometer behavior prediction like "Wind."

The following section of our study discusses integrating the NN model into the sophisticated Arduino Nano 33 BLE Sense module and stress tests its resistance against sensor hacking.

IV. Test Modules

Neural network intelligence against sensor hacking is the prime forte of our paper, in which we first train the NN model with the help of Edge impulse and deploy it into the Arduino Nano 33 BLE Sense module. We compare the efficiency of Neural Network powered microcontroller and standard sensor microcontroller duo against the sensor hacking approach exerted by EMP. In the forthcoming line of our work, we deploy the learned model as the library to Arduino IDE and later to Arduino Nano 33 BLE Sense module.

A. *Impulse Compilation and Deployment*

Our NN classifier has two optimization options, both of which offer an accuracy of 99.29%, but Quantized (int8) provides a latency of just 1 ms, which is quick for making decisions, so we choose that and download the learned model in the form of library file for Arduino for the entire model access into TinyML form factor.

This gets dumped into the microcontroller to become available for field-ready operation. We use one of today's most powerful and sophisticated microcontrollers. The sensors inside the recommended model (Arduino Nano 33 BLE Sense Module) are embedded in the microcontroller. Thus, their deployment is wire-free and easy. This makes our system compact, not as much of power-hungry, portable, and less visible from a distance.

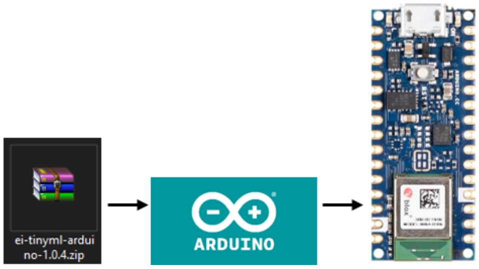


Fig. 6: Overall TinyML burn to Microcontroller

To prevent hacking and tampering assaults, these properties are desirable for the secure deployment of the proposed system.

B. *Conventional Sensor System Development*

We intend to develop another system that functions similarly to our work to compare our system tolerance to conventional sensor systems.

We connect the Arduino Nano to the MPU6050 IMU sensor. We established an inclination threshold to determine whether the wire fence had crossed the maintainable limit. Our motivation is to compare the effects of sensor hacking on

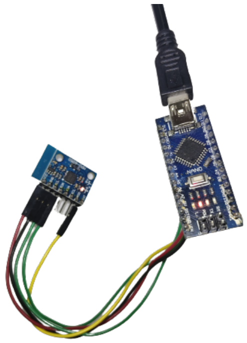


Fig. 7: Conventional Microcontroller Sensor System

traditional microcontroller sensors to those on microcontrollers that run on ML training (TinyML).

C. EMP Hacking Unit

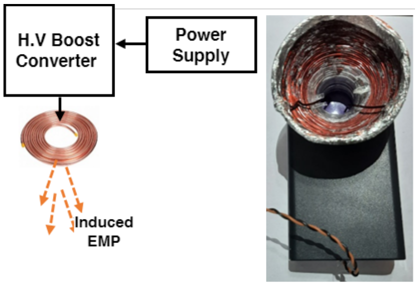


Fig. 8: EMP Generator

We develop an Electromagnetic Interference Source to target EMP to the desired device, which projects the Electromagnetic Interference on the victim device to induce a charge on metallic parts. Microcontrollers read the charge coming from sensors as the indicator of the input to analyze. Based on this analysis, they make the decisions, but the synthetic charge produced by our EMI device confuses them, and they start to make wrong decisions. This ultimately leads to the system application disorientation.

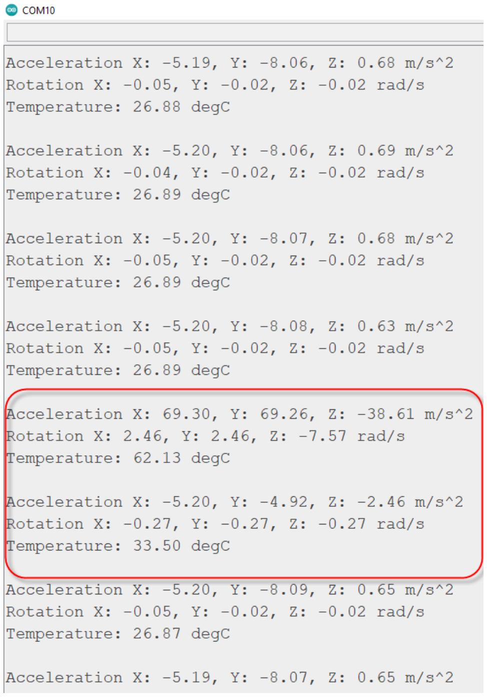
In the next section, we implement the attack over different sensor setups and analyze their outcomes.

V. Experimental Observations

In this section, we use our EMP generator and project it over the Victim sensor system, which could have been utilized in a battle (for IoBT application) to monitor suspicious fence movement for encroachment into Indian land.

A. Effect of Sensor Hacking on MPU6050 and Arduino Nano System

We determined the threshold of various MPU6050 accelerometer readings using the setup shown in Fig. 7 to discriminate between normal and suspicious fence motions. Due to its small form factor and low power characteristics, the Arduino Nano microcontroller we utilized is reasonably straightforward and used in various applications and projects [11]. An alarm is generated when a sensor determines that a fence is being placed improperly. This sensor deployment is incredibly deployable and practical. Even a net-style border incursion protection system can be used for this setup. To detect a particular movement of the fence or net in any other fashion, we only need to adjust the threshold value of the MPU6050 sensor.



```
Acceleration X: -5.19, Y: -8.06, Z: 0.68 m/s^2
Rotation X: -0.05, Y: -0.02, Z: -0.02 rad/s
Temperature: 26.88 degC

Acceleration X: -5.20, Y: -8.06, Z: 0.69 m/s^2
Rotation X: -0.04, Y: -0.02, Z: -0.02 rad/s
Temperature: 26.89 degC

Acceleration X: -5.20, Y: -8.07, Z: 0.68 m/s^2
Rotation X: -0.05, Y: -0.02, Z: -0.02 rad/s
Temperature: 26.89 degC

Acceleration X: -5.20, Y: -8.08, Z: 0.63 m/s^2
Rotation X: -0.05, Y: -0.02, Z: -0.02 rad/s
Temperature: 26.89 degC

Acceleration X: 69.30, Y: 69.26, Z: -38.61 m/s^2
Rotation X: 2.46, Y: 2.46, Z: -7.57 rad/s
Temperature: 62.13 degC

Acceleration X: -5.20, Y: -4.92, Z: -2.46 m/s^2
Rotation X: -0.27, Y: -0.27, Z: -0.27 rad/s
Temperature: 33.50 degC

Acceleration X: -5.20, Y: -8.09, Z: 0.65 m/s^2
Rotation X: -0.05, Y: -0.02, Z: -0.02 rad/s
Temperature: 26.87 degC

Acceleration X: -5.19, Y: -8.07, Z: 0.65 m/s^2
```

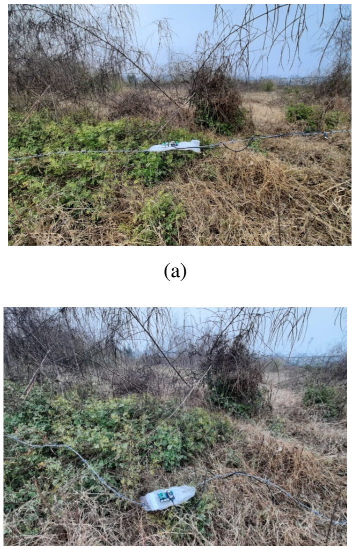
Fig. 9: Variation in MPU6050 during EMI attack

B. Effect of Sensor Hacking on Arduino Nano 33 BLE Sense

As depicted in Fig.10, we mounted our Arduino Nano 33 BLE Sense module on the fence for the real-time test and verified for Wind and Intrusion conditions. Our system is successfully able to identify natural wire movement and when someone tries to intrude by cutting or moving the fence.

The action was correctly identified as an intrusion, as shown in Fig.11. To analyze the natural condition of the fence (Wind), we gently left the fence idle and shook it to check for really strong winds. Different readings were acquired, each precise enough to pick up even the slightest variations. We attempted to cut the wire and enter from the passageway to inspect the fence's anomalous state (Intrusion). Intrusion's detection value abruptly raised enough to activate the alert system. The scenario was run numerous times, and each time, our plan was accurate enough to detect even the most minor differences and classify the occurrences appropriately.

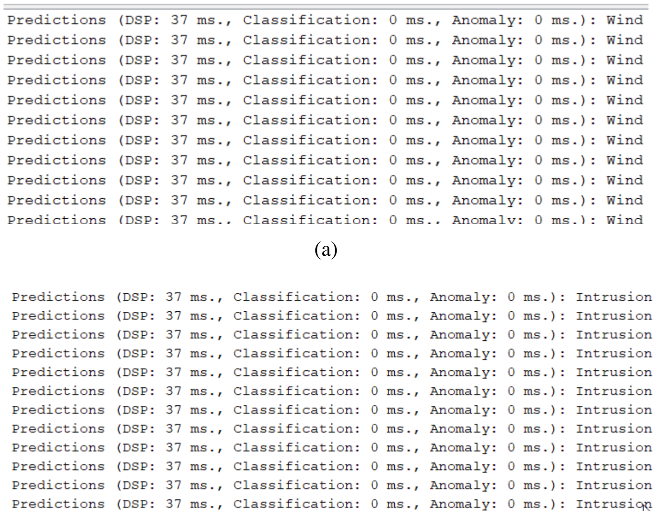
We applied EMP in a variety of circumstances, but the outcome was unchanged. The tiniest changes brought on by a sensor hacking effort were not noticed over the TinyML module.



(a)

(b)

Fig. 10: Test Setup: Arduino Nano 33 BLE Sense (a). Wind & (b). Intrusion



```
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Wind
```

(a)

```
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
Predictions (DSP: 37 ms., Classification: 0 ms., Anomaly: 0 ms.): Intrusion
```

(b)

Fig. 11: No Variation during the attack - (a) Wind (Normal Condition Detection) & (b) Intrusion Scenario (Intrusion Attempt Detection)

VI. Result and Analysis

A. MPU6050 + Arduino Nano Vs. Sensor Hacking

The sudden variations in the X, Y, and Z values of MPU6050 are due to an attack from the EMP generator exerted over the victim circuit. We compare readings obtained from various distances with and without a sensor hacking attempt and notice the differences. The fluctuation readings are regarded as the result of EMP-based sensor hacking. The MPU6050 sensor uses three separate position points to indicate X, Y, and Z.

The column is referred to as ”Before Attack” under normal circumstances when no attack is exerted on the sensor system. Values from the MPU6050 sensor that are listed in the ”After Attack” column are XYZ values. The columns labeled ”Exerted Variation” and ”Times change than expected” show the total number of times the initial value has changed and the total number of units that have changed.

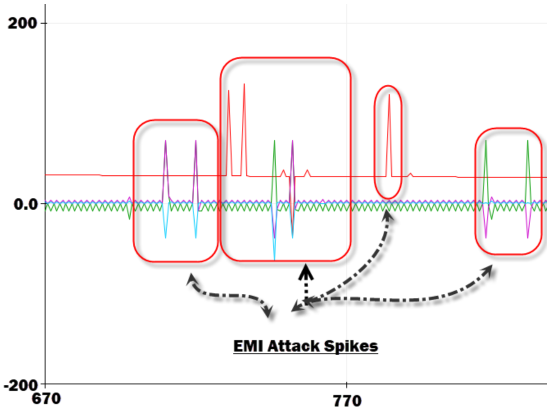


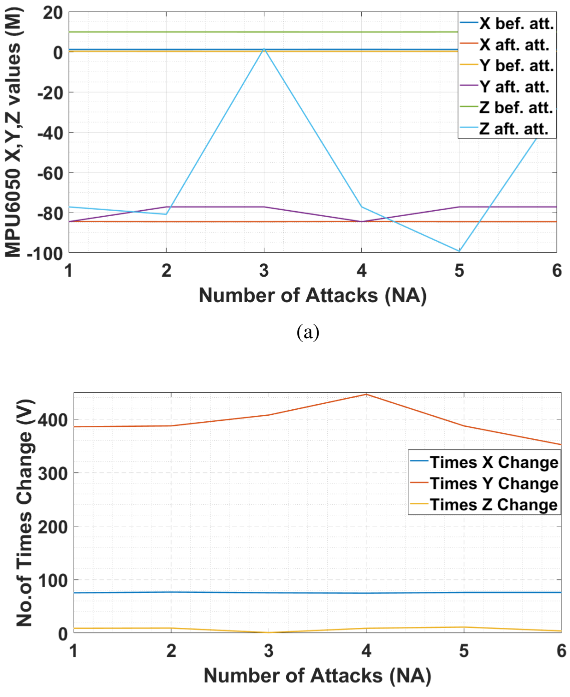
Fig. 12: Variations in MPU6050 Sensor Values due to Attack

TABLE IV: Sensor Variations - Values before and After Attack (At different attack points)

S.No.	**Value**	**Before Attack**	**After Attack**	**Exerted Variation**	**∼Times Change than expected**
1.	X	1.14	-84.51	85.65	75.131
	Y	0.22	-84.58	84.8	385.454
	Z	9.84	-77.23	87.07	8.848
2.	X	1.12	-84.58	85.7	76.517
	Y	0.20	-77.23	77.43	387.150
	Z	9.84	-80.90	90.74	9.221
3.	X	1.14	-84.58	85.72	75.192
	Y	0.19	-77.22	77.41	407.421
	Z	9.83	1.62	8.21	0.835
4.	X	1.15	-84.51	85.66	74.486
	Y	0.19	-84.58	84.77	446.157
	Z	9.81	-77.22	87.03	8.871
5.	X	1.13	-84.58	85.71	75.849
	Y	0.20	-77.22	77.42	387.100
	Z	9.84	-99.28	109.12	11.089
6.	X	1.13	-84.58	85.71	75.849
	Y	0.22	-77.22	77.44	352
	Z	9.81	-28.18	37.99	3.872

Fig. 13 (a) and Table IV show that the MPU6050 X, Y, and Z values significantly varied during six sensor hacking attempts. This demonstrates incorrect sensor positioning, which can be interpreted as sensor failure or system disconnection.

Fig. 13 (b) and Table IV reading demonstrate that the variation in the MPU6050’s various values for 6 sensor hacking attempts is up to a maximum of 446 times more than the initial value without any hacking attempt. The values of ”Y”



(a)

(b)

Fig. 13: (a) NA Vs. M Graph & (b) NA Vs. V Graph

fluctuated the most, and the values of ”Z” fluctuated the least.

This demonstrates how easily sensors can be hacked in conventional sensor technologies. Although sensor accuracy is determined to be extremely good, attempts to hack sensors render them useless. Even in the case of low-power EMP attacks, the values are promptly altered, and in the case of high-power EMP strikes, the system has no chance.

Our model prediction for accuracy during practical implementation was 99.29%. We did not come across even one instance during our live testing when the model could not anticipate the actual state of the fence. Results remain precise even after using the Sensor Hacking method (EMP). This demonstrates how the model achieves the accuracy it promises. To get accuracy, the window size description is as follows.

TABLE V: Sensor Variations - Values before and After Attack

Training Vs. Testing Model Performance- Window Size (WS)					
	Total WS	Intrusion WS	Wind WS	Testing WS	Accuracy
Training	4794	2397	2397	—	95.40 %
Testing	1222	564	564	94	99.29 %

The Arduino Nano 33 BLE Sense module’s prediction ML model requires 33ms of DSP processing. We determined the accuracy for ”Wind” for 3 random test instances to be 96.484 % and 99.609 % for the following 2 case, respectively. We also used 3 random test instances, and the accuracy for ”Intrusion” was 92.969 %, 98.828 %, and 99.609 %, respectively.

In the event of a mid-range EMP attack where the EMP does not physically damage the sensors or microcontrollers but alters their readings, machine learning-trained microcontrollers remain unaffected. This is because their functionality isn’t reliant on external sensors alone. Moreover, multiple data points make them more resilient against errors. This resilience stems from that calculations primarily occur internally rather than relying solely on data imported from external sensors. The conclusion effectively summarizes the findings of the study and highlights the potential future enhancements and applications of TinyML in the context of sensor hacking.

VII. Conclusion

The primary goal of our experimental paper was to look for any resistance from intelligent microcontroller subsystems designed with Neural Network learning models over conventional microcontroller sensor systems against sensor hacking for IoBT applications. The secondary goal was to predict the detection accuracy during battlefield operations when EMP is exerted over the system to achieve unauthorized entry. We compared the performance of results obtained after exerting EMP type Sensor Hacking approach to both MPU6050+Arduino Nano and TinyML+Arduino Nano 33 BLE Sense systems. During the field test, TinyML+Arduino Nano 33 BLE gave an accuracy performance close to its predicted performance based on the confusion matrix, which was 95.4%. The deployment test accuracy was expected to be somewhere around 99.29%, which was easily achieved during the real-time test. With the injection of sensor hacking, the traditional sensor microcontroller duo completely failed and gave unreasonable results with a variation of up to 446 times the original value. The TinyML system didn’t have any effect of sensor hacking on the system. The future enhancement of the paper lies in other performance tests of TinyML in terms of another form of sensor hacking and exploring another domain where TinyML has not yet been reached.

Acknowledgment

First and foremost, we sincerely thank our colleagues, Pooja Bhardwaj and Ruchi Bhatt, for their assistance with extensive experimentation and analysis. Additionally, we would like to thank the anonymous reviewers for their insightful feedback and suggestions, which helped improve this paper’s quality.

References

[1] H. Afzaal and N. A. Zafar. Modeling of iot-based border protection system. In *2017 First International Conference on Latest trends in Electrical Engineering and Computing Technologies (INTELLECT)*, pages 1–6, Karachi, Pakistan, 2017.

[2] P. Alexander. External intrusion detection systems for critical facilities. In *IEEE International Carnahan Conference on Security Technology, Crime Countermeasures*, pages 139–140. IEEE, 1990.

[3] J. Alghazo, A. Bashar, G. Latif, and M. Zikria. Maritime ship detection using convolutional neural networks from satellite images. In *2021 10th IEEE International Conference on Communication Systems and Network*

Technologies (CSNT), pages 432–437, Bhopal, India, 2021.

[4] D. ALshukri, V. L. R, S. E. P, and P. Krishnan. Intelligent border security intrusion detection using iot and embedded systems. In *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*, pages 1–3. IEEE, 2019.

[5] D. Arjun, P. K. Indukala, and K. A. Unnikrishna Menon. Panchendriya: A multi-sensing framework through wireless sensor networks for advanced border surveillance and human intruder detection. In *2019 International Conference on Communication and Electronics Systems (ICCES)*, pages 295–298. IEEE, 2019.

[6] W. Butler. Design considerations for intrusion detection wide area surveillance radars for perimeters and borders. In *2008 IEEE Conference on Technologies for Homeland Security*, pages 47–50, Waltham, MA, USA, 2008.

[7] Y. Cao, Y. Wang, Z. Liao, Q. Liu, C. Xu, and N. Li. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proc. 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19)*. ACM, 2019.

[8] L. R. Carlos-Roca, I. H. Torres, and C. F. Tena. Facial recognition application for border control. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Rio de Janeiro, Brazil, 2018.

[9] S. Hong, S. Lee, and J. Kim. Cos status predictive system based on machine learning. In *2022 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–4, Jeju, Korea, Republic of, 2022.

[10] G. Loney. Border intrusion detection: Thinking outside the perimeter. In *2007 41st Annual IEEE International Carnahan Conference on Security Technology*, pages 103–108, Ottawa, ON, Canada, 2007.

[11] R. K. Singh, J. Saravanan, A. C. Raj, and A. Taye. An efficient security implementation with power cane for visually challenged. In *International Conference on Information and Communication Technology for Development for Africa*, pages 270–277, 2019.

[12] K. Trivedi and H. Shroff. Identification of deadliest mosquitoes using wing beats sound classification on tiny embedded system using machine learning and edge impulse platform. In *2021 ITU Kaleidoscope: Connecting Physical and Virtual Worlds (ITU K)*, pages 1–6, Geneva, Switzerland, 2021.

[13] K. P. Vittal, A. P. P., A. S. B., and C. H. S. Rao. Computer controlled intrusion-detector and automatic firing-unit for border security. In *2010 Second International Conference on Computer and Network Technology*, pages 289–293, Bangkok, Thailand, 2010.

[14] B. Wang et al. Ecg diagnosis device based on machine learning. In *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)*, pages 383–386, Chongqing, China, 2021.

[15] D. A. Ware. Effects of intentional electromagnetic interference on analog to digital converter measurements of sensor outputs and general purpose input output pins. Master's thesis, Utah State Univ., 2017.

[16] Y. Zhang and K. Rasmussen. Detection of electromagnetic interference attacks on sensor systems. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 203–216. IEEE, 2020.

Raushan Kumar Singh is currently a research scholar in the CSE department at IIT Ropar, with expertise in Embedded Systems, Robotics, TinyML, and edge-level Intelligence. He has a successful track record of organizing academic events, having run an academic event organization for 10 years and organized over 100 educational events worldwide in different institutions. Additionally, he has industry experience as the Technical Director of Spectrum Solutions in Pondicherry, India, for nine years and has been recognized for his academic achievements with a gold medal in Embedded Systems and an excellent academic record with a CGPA of over 9 in both his undergraduate and postgraduate studies, as well as several publications, patents, and awards.

Dr. Sudeepta Mishra is an Assistant Professor in the Department of Computer Science and Engineering at IIT Ropar, joining in February 2020. He completed his Bachelor's degree in Computer Science and Engineering from B.P.U.T. (Biju Pattnaik University of Technology) in Rourkela, his Master's degree in Computer Science and Engineering from KIIT (Kalinga Institute of Industrial Technology) University in Bhubaneswar, and his Ph.D. degree in Computer Science and Engineering from the Indian Institute of Technology Madras. Prior to his current position, Dr. Mishra served as an Assistant Professor in the Department of Computer Science and Information Systems at BITS Pilani's Hyderabad campus from May 2017 to January 2020. He also worked as a Systems Engineer at TATA Consultancy Services Ltd. from June 2008 to December 2010. His research interests are focused on wireless networking, including cellular networks, ad-hoc wireless networks, and the Internet of Things.