Fall detector by STwinbox

Alessandro Gianluca Cazzaniga, Massimiliano Michelini, Riccardo Sibilia
alessandro.cazzaniga@studenti.unitn.it-m.michelini-1@studenti.unitn.it-riccardo.sibilia@studenti.unitn.it
University of Trento
Povo, Trento, Italy

ABSTRACT

This document summarizes the operation and implementation path of the STwinbox low-power microcontroller. We have programmed this controller to collect data from various sensors, using them in an energy-efficient way to detect possible collisions involving, for example, elderly people falling, motor vehicle crush, or other objects. It also illustrates how these data were subsequently used to train a small AI capable of detecting impacts, allowing a better interpretation of the data and its future use.

1 MOTIVATION

Thanks to Professor Kasim Sinan Yildirim, we received a STEVAL-STWINBX1 [1], a development kit and low-power reference design (based on the ultra-low power Arm Cortex-M33) that simplifies the prototyping and testing of advanced industrial sensing applications in IoT contexts such as condition monitoring.

After an initial period of research into its components and functionality, we began testing its actual operation [2]. Being an ST system, and therefore not open source, it proved hostile to the introduction of non-ST signed programs. So we decided that in order to carry out a worthwhile project, we would need a simple but functional application, so that we could better control the system even with our own personal features and not those of ST. After much consideration, we decided to use the system's various accelerometers to create a device capable of detecting falls by elderly people, also relying on the ability of AI to better detect the event for greater certainty.

This project is significant because it not only explores the intelligent use of a low-consumption system, through various considerations such as which sensors to use and at what sampling frequencies; but also introduces itself into a field where devices are few, expensive, and do not provide for AI integration. Furthermore, through various experiments, we have explored the world of ST systems, understanding how to use and exploit them to the fullest without being forced to rely solely on their systems, but making the project as open source as possible.

2 LIMITATIONS OF THE STATE OF THE ART

There are currently several devices on the market that can help us detect accidents involving elderly or frail people. Unfortunately, however, these devices have high initial costs and also require monthly subscriptions to ensure their operation, which makes this field still open to exploration and fertile from a competitive point of view.

The devices already on the market are certainly well built and well designed, but they focus more on their customers and not so much on the technologies that can be implemented. This makes our small AI integration an innovation, capable of helping the device provide more accurate data. In addition, our particular focus on the low-power use of accelerometers, combined with the possibility of using a microphone as a black box in the event of a fall, can make the device more durable, reliable, intelligent, and even convenient for resolving any doubts about the event.

The platform we used for data collection, STwinbox, is an excellent starting point given its low-power nature, high number of sensors, high level of customization, and much more. It also has some flaws, such as the aforementioned difficulty in communicating with external programs and data collectors. However, through several steps and reverse engineering, we were able to collect the necessary data, convert them into a useful format, clean them, and then use them through Edge Impulse [3]. Through this platform, we created a simple AI capable of interpreting the final data in the best possible way, thus strengthening the system, which would otherwise have to use a simple threshold value instead of a sophisticated system like the one we developed.

3 KEY INSIGHTS

Referring to everything that has been written so far, the main innovation of the proposed system is the intelligent use of a low-consumption platform already in the experimental phase (i.e., in the prototype) combined with minimal use of resources and the processing capacity of a neural network trained specifically for this project with data collected specifically for this purpose.

The state of the art therefore finds innovation in the use of AI in a niche sector, where the few who offer solutions promote systems with wired logic that are not very innovative

and very expensive. Of course, other fields have not been explored, such as emergency calls for elderly rescue, but those are not part of the prototyping objective, as it would require the use of a cellular data network and a whole other study that is secondary to the results sought by this project. Having focused on other aspects, we have managed to obtain important data regarding the fall detection system, which is shown and explained in paragraph 5, thus improving our personal state of the art.

4 MAIN ARTIFACTS

The primary artifact of our work is the STEVAL-STWINBX1 board itself [Fig. 1], which was programmed using ST's own firmware, Datalog2 [4] and IDE, STM32 CubeProgrammer [5]. In particular, we decided to focus on the IIS2DLPC accelerometer [6], chosen mainly for its excellent balance between power consumption and data acquisition rate. The sensor is able to consume only 50 nA in power-down mode, and less than 1 μA in active low-power mode, while still supporting an acquisition frequency of 50 Hz. We selected this sampling rate after carrying out several tests and subsequent data analyses, as it proved to be the best compromise: high enough to obtain sufficiently precise results for an artificial intelligence model to recognize and classify, yet low enough to avoid excessive power consumption.

Figure 1: STEVAL-STWINBX1 main board and case

The second fundamental artifact of this work is the STDATALOG-PYSDK framework [7], a Python-based toolkit developed by STMicroelectronics to facilitate the acquisition, processing, and visualization of sensor data. This framework played a central role in the project, because it provided the interface through which the STEVAL-STWINBX1 board [1] could be configured and controlled during the data acquisition phase. In particular, it allows to select which onboard sensors to activate and to set their main operational parameters, such as the output data rate, the full scale for the accelerometer, and the number of samples per timestamp. Another key feature of this framework [7] is the ability of monitoring the signals acquired in real time through a graphical visualization interface. This functionality was extremely useful during preliminary tests, because it allows to check the quality of the acquired signals and to adjust the acquisition parameters before starting longer recording sessions. The only drawback of this framework was that it did not allow to analyze measurements collected while the board was not directly connected to the PC. This was a significant limitation for our work, since we needed to perform several fall and movement tests in a standalone setup. We managed to work around this issue through some reverse engineering: we found out that the framework assigned a specific name to each measurement folder, and by applying that same naming convention to the folder containing our recordings, we were able to trick the tool into processing our data as if it had been collected in the intended way.

The third key artifact of this project is the Python code we developed for data analysis through the spyder ide [8]. This code is organized into two main modules. The first module reads the sensor data which is saved by the sensor onto the SD Card in .dat format (in principle, we could access all the onboard sensors, but for our purposes we concentrated on this accelerometer [6]) and converts it into a .csv file, while also performing a data-cleaning step to remove anomalous spikes caused by noise. The second module processes these .csv files and generates plots, for example in the case of the accelerometer it generates 3 graphs, one for each dimension. During this phase, however, we discovered that, most likely for privacy reasons, ST had deliberately introduced artificial distortions into the recorded signals, such as offsets or false sawtooth-like patterns[Fig. 2]. To address this, we extended the functionality of the first module by adding a correction routine that subtracts such offsets from the collected data, thereby restoring a clean and reliable version of the original measurements[Fig. 3]. In this final figure, it can be observed that, despite the data-cleaning process, some spurious peaks are still present. These peaks are clearly generated by measure errors, yet we were unable to remove them completely. The reason is that during simulated fall events we recorded accelerations of up to 16g; therefore, as long as these glitches remain below this threshold, they cannot be automatically discarded and must be retained in the dataset.

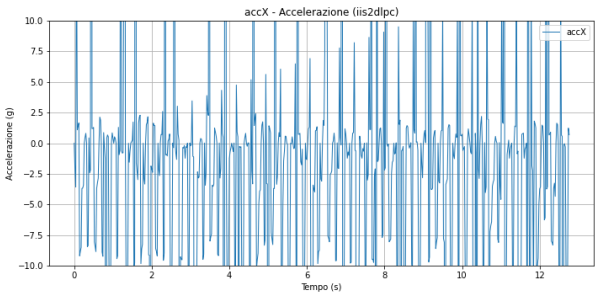

Figure 2: Original sample collected

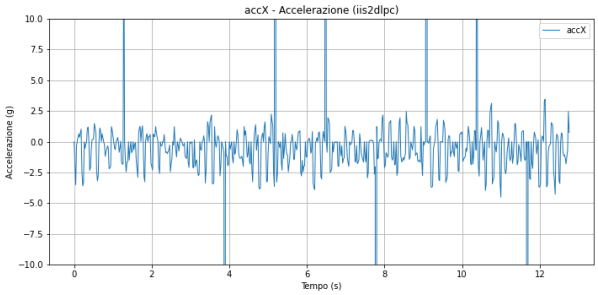

Figure 3: Sample after polishing

In the process of converting raw data from the accelerometers, the measured values are initially expressed in units of gravitational acceleration (g), where 1 g corresponds to the average gravitational acceleration at sea level on Earth. In order to physically interpret the measurements and perform analyses consistent with the International System of Units (SI), these values were converted into meters per second squared (m/s^2). The conversion was carried out by applying the scaling factor defined as 1 g = 9.80665 m/s^2, where the factor "9.80665" represents the standard definition of gravitational acceleration.

The fourth fundamental artifact of our work is the use of Edge Impulse [3], a development platform for building, training, and deploying machine learning models optimized for embedded devices with limited resources. Edge Impulse offers a graphical web-based interface which allows for rapid prototyping and testing of different pre-processing pipelines and classification algorithms without requiring extensive manual coding. The main key advantage of this artifact is its optimization for tinyML applications, allowing the deployment of models that are accurate and lightweight; this makes it perfectly suitable for the micro-controller integrated onto our board. This factor allows this last component to be particularly well-suited for low-power embedded systems, where energy efficiency is a critical requirement; this also enables inference to be performed directly on the device, without requiring a constant connection to external machines.

Using this artifact [3], a machine learning model was designed to perform fall detection [9], complementing both the hardware board and the Python processing pipeline. Its primary purpose is to develop a machine learning model capable of recognizing events such as falls, walking, and periods of no movement, based on accelerometer data acquired from the STEVAL-STWINBX1 development board [1]. The data set used in the project was obtained using our Python modules, where all 190 samples were aligned to a uniform sampling rate of 50 Hz with a duration of 13 seconds each. This step ensured that data were made directly compatible with the input requirements of Edge Impulse.

The accelerometer signals [9] were treated as time series data with three input axes, corresponding to the accelerations along the X, Y, and Z directions. Each sample was divided into segments of 1,500 ms, with a stride of 200 ms between consecutive segments. Keeping the same acquisition frequency of 50 Hz, each segment contains 75 samples per axis, providing a sufficiently detailed temporal context for activity recognition.

For feature extraction [9], the signal is analyzed in the frequency domain in order to highlight the characteristics of the accelerometer signals. Since human movements such as walking, standing, or falling typically produce accelerations with relatively low frequency content, it is adopted a 6th order low-pass filter with a cutoff frequency of 8 Hz. This ensures that only the most relevant components of the signal are preserved, while higher-frequency noise is attenuated. This setup provides a distribution where the feature map[Fig. 4] reflects the intrinsic nature of the movements studies in this work.

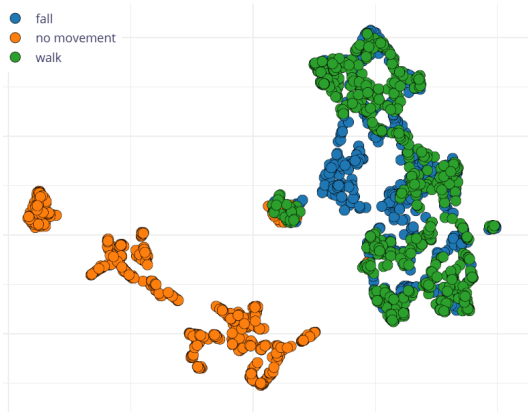

Figure 4: Feature distribution map

Periods of no movement are characterized by very low spectral content, which leads to their aggregation into compact and well-isolated clusters. Walking sequences, instead, generate more complex oscillatory patterns, which give rise to a different and equally well-separated cluster. Falls, however, represent transient and heterogeneous events: their spectral content often includes both high-amplitude spikes and low-frequency components, which explains why fall-related features appear scattered and sometimes overlap with walking or resting samples.

For the classification task [9], a fully connected feedforward neural network (Multilayer Perceptron, MLP) is used; this network was designed to process the 45 spectral features extracted from the accelerometer signals. The network architecture consists of an input layer with 45 nodes, followed by two dense hidden layers: the first with 20 neurons and the second with 10 neurons. These characteristics are selected to ensure the structure is compact as much as possible and to balance expressive power with computational efficiency, ensuring suitability for deployment on a low-power embedded system. Each hidden layer used the ReLU activation function, it is adopted because of its ability to mitigate vanishing gradient issues. For the output layer, a Softmax activation is applied, producing normalized probabilities over the three target classes (fall, walking, no movement).

The training process was carried out with 60 training cycles and a learning rate of 0.0005, which provided a good trade-off between convergence speed and stability. Given the relatively small size of the dataset and the lightweight nature of the network, GPU acceleration was not required for this process and the training phase was therefore carried out on a CPU.

For the optimization part, the Adam optimizer was adopted to achieve fast and stable convergence. The model was trained by minimizing the categorical cross-entropy loss function, which is particularly suitable for multi-class classification problems such as ours. This setup allows to obtain a model capable of distinguishing different movements between falls, walking, and no movement, while remaining lightweight enough to have a real time on board applications.

5 KEY RESULTS AND CONTRIBUTIONS

The primary outcome of this work is the successful implementation of a fall-detection pipeline on a low-power embedded system that combines custom data acquisition, signal processing, and machine learning. A crucial aspect of the final deployment was to choose the version of the neural network model. This decision is fundamental to reduce the memory footprint and computational cost of the model itself, making it suitable for real-time inference on the microcontroller; for this reason, the "quantized (int8)" version was chosen. In fact, to enable embedded deployment, quantization also contributed to lowering power consumption, which is a fundamental requirement in continuous monitoring applications where energy efficiency is critical.

Figure 5: Training performance (validation set)

The experimental evaluation highlights the effectiveness of the proposed work. In the validation set, the model achieved an overall accuracy of 87.3% with a low loss of 0.27, as it can be seen in[Fig. 5], confirming both good predictive performance and stable training. The confusion matrix further showed that the network was particularly strong in distinguishing the no movement class, reaching an F1-score of 0.99, while still maintaining satisfactory performance on the fall (F1 = 0.84) and walk (F1 = 0.81) classes.

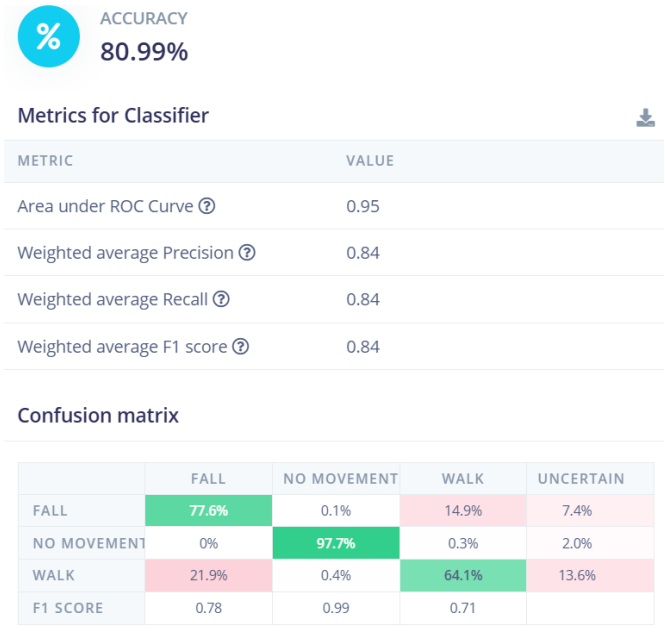

Figure 6: Results (test set)

The evaluation of the test set[Fig. 6] confirmed the robustness of the model, which achieved an overall accuracy of 80.99% with the "no movement" class recognized also this time with near-perfect reliability. Although distinguishing between walk and fall events was more challenging, the results remain competitive given the constraints of a low-power embedded device. An equally important aspect of our work concerns the on-device performances: the quantized model, which has been compiled using an Embedded Optimized Neural Network Compiler, has demonstrated highly efficient resource utilization, achieving an inference time of only 1 ms, with a peak RAM consumption of 1.5 KB and a Flash memory footprint of 15.3 KB. These results confirm that the model is lightweight enough to run in real time on the board, making it suitable for low-power continuous monitoring applications.

Another significant advantage of our implementation lies in its independence from proprietary ST platforms. Thanks to the integration of the Python-based data processing pipeline (third artifact) and the Edge Impulse framework (fourth artifact), the system can perform data acquisition, pre-processing, and model deployment without relying on ST's dedicated software tools. This modular design improves flexibility and portability, allowing the same workflow to be easily adapted to different embedded boards or sensor configurations, thus broadening the applicability of the proposed solution beyond the STEVAL-STWINBX1 platform. The complete Python code developed for data acquisition and processing (third artifact) has been made available in a dedicated GitHub repository [10], ensuring transparency and facilitating reproducibility of the results.

In conclusion, this work demonstrates the feasibility of implementing an effective fall detection pipeline in a low-power embedded system, achieving real-time performance and energy efficiency. Future improvements may focus on optimizing signal acquisition and pre-processing, as well as refining the movement recognition model to enhance accuracy in complex scenarios. Overall, the results confirm the robustness and reliability of the proposed approach, which successfully balances performance and low-power operation in embedded intelligent systems.

REFERENCES

[1] STMicroelectronics. STWIN.box - SensorTile Wireless Industrial Node Development Kit. https://www.st.com/en/evaluation-tools/steval-stwinbx1.html.

[2] STMicroelectronics. Getting started with the STEVAL-STWINBX1 SensorTile wireless industrial node development kit. https://www.st.com/resource/en/user_manual/um2965-getting-started-with-the-stevalstwinbx1-sensortile-wireless-industrial-node-development-kit-stmicroelectronics.pdf.

[3] Edgeimpulse. The Edge AI Platform. https://edgeimpulse.com.

[4] STMicroelectronics. ST High Speed Datalog: a comprehensive multi-sensor data capture and visualization toolkit. https://www.st.com/en/embedded-software/fp-sns-datalog2.html.

[5] STMicroelectronics. STM32CubeProgrammer software for all STM32. https://www.st.com/en/development-tools/stm32cubeprog.html.

[6] STMicroelectronics. IIS2DLPC - MEMS digital output motion sensor: high-performance ultra-low-power 3-axis accelerometer for industrial applications. https://www.st.com/resource/en/datasheet/iis2dlpc.pdf.

[7] STMicroelectronics. STDATALOG-PYSDK Python software development kit. https://github.com/STMicroelectronics/stdatalog-pysdk.

[8] Spyder. he Python IDE that scientists and data analysts deserve. https://www.spyder-ide.org.

[9] Alessandro Gianluca Cazzaniga, Massimiliano Michelini, and Riccardo Sibilia. Fall Detection with Embedded Sensors. https://studio.edgeimpulse.com/public/776299/live.

[10] Alessandro Gianluca Cazzaniga, Massimiliano Michelini, and Riccardo Sibilia. Data conversion and correction. https://github.com/Maxmichelini/Data_Conversion_and_Correction_Script.git.