Artifact: WirelessEye – Seeing over WiFi Made Accessible

Philipp H. Kindt[†], Cristian Turetta[¶], Florenc Demrozi[§],
Alejandro Masrur[†], Graziano Pravadelli[*], and Samarjit Chakraborty[‡]
[*]*Dept. of Engineering for Innovation Medicine, University of Verona*, Italy, name.surname@univr.it
[¶]*Dept. of Computer Science, University of Verona*, Italy, name.surname@univr.it
[†]*Faculty of Computer Science, TU Chemnitz*, Germany, name.surname@informatik.tu-chemnitz.de
[‡]*Dept. of Computer Science, University of North Carolina at Chapel Hill*, USA, samarjit@cs.unc.edu
[§]*Dept. of Electrical Engineering and Computer Science, University of Stavanger*, Norway, name.surname@uis.no

2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 979-8-3503-0436-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/PerComWorkshops59983.2024.10502539

I. Scope

WirelessEye is a framework for developing WiFi-based sensing systems [1]. It serves as a link between a (patched) radio firmware and a classifier for detecting, e.g., human activity. In brief, WirelessEye supports the following functionality related to a WiFi's channel state information (CSI) in an "out-of-the-box" manner:

- Visualizing CSI data in real-time
- Recording CSI data using different file formats
- Customizable preprocessing of CSI data
- Streaming of the preprocessed data to a classifier
- Visualizing machine and deep learning classification results in real-time

WirelessEye can be extended and enhanced using plugins. Additional documentation, including additional details on setting up and running a WiFi sensing system, is available in the WirelessEye repository [2].

II. Hardware and Software Prerequisites

To perform WiFi-based sensing using WirelessEye, the following hardware or software is needed.

- A Raspberry PI 4B
- A Nexmon-patched firmware (see Section III-B)
- A Linux PC or Laptop (e.g., Linux Mint)
- The GNU C Compiler (GCC) on a PC and Raspberry Pi
- GNU Make on a PC and Raspberry Pi
- QT library version 5 on a PC or Raspberry Pi
- A WiFi AP to create WiFi signals to capture

III. Getting Started

This section describes how WirelessEye is installed and run. All commands mentioned in this description are to be entered into a Linux terminal. A more detailed explanation can be found in the WirelessEye Repository [2].

A. Software Components

WirelessEye consists of the following two components:

1) CSIServer_ng: A simple TCP server to be run on the Raspberry PI. It will receive local UDP broadcasts from Nexmon and make them available over the network via TCP Port 5501.

2) WirelessEye Studio: A GUI to display, record, and export CSI data in real-time. To be run on any Linux PC or laptop from which the Raspberry Pi that runs *CSIServer_ng* is reachable over the network.

B. Preparing the Raspberry Pi

Before WirelessEye can be used, the Raspberry Pi needs to be prepared. We next describe this preparation and then present the steps needed to install WirelessEye.

WirelessEye receives CSI data from a patched radio firmware. This firmware patch is provided by the Nexmon framework [3] [4]. Hence, before using WirelessEye, the Raspberry Pi has to be prepared by installing the Nexmon firmware patches. For this purpose, Nexmon CSI needs to be installed and configured as described in the WirelessEye repository [2], or, in a more generic fashion, in the Nexmon CSI Repository [5].

Using some firmware variants, no received signal strength indicator (RSSI) is obtained. WirelessEye expects RSSI data by default and will fail if none is received. Therefore, one option is to deactivate RSSI processing in `networkThread.h` by setting the `CSI_CONTAINS_RSSI` marco to `false`. To choose this option, the corresponding line must be modified as follows: `#define CSI_CONTAINS_RSSI false`. Alternatively, a modified version of Nexmon that supports RSSI can be installed. A tutorial on how this is done can be found in the WirelessEye repository [2].

C. Compiling and Running CSIServer_ng

WirelessEye repository contains a TCP server to access the CSI data from a different computer, which is called `CSIServer_ng`. It needs to be compiled and run on the Raspberry Pi. For this purpose, the following steps need to be carried out on the Raspberry Pi:

1) Copy the *CSIServer_ng* folder from the downloaded code to the Raspberry Pi.
2) Type `cd CSIServer_ng` and `make`.
3) Run the server with `./CSIServer`.
4) It is recommended to configure Nexmon and run *CSIServer_ng* automatically at every startup of the Raspberry Pi. Additional details on how this is done can be found in the WirelessEye repository.

D. Compiling and Running WirelessEye Studio

Next, *WirelessEye Studio*, the software running on a PC or laptop, needs to be compiled. This works as follows.

1) In the `WirelessEye` folder, type `make`.
2) Run *WirelessEye Studio* by typing `./WirelessEye`.

IV. Using WirelessEye Studio

We next briefly describe how WirelessEye can be used. Our related paper [1] provides a good overview of WirelessEye's functionality beyond this description.

WirelessEye Studio provides a graphical user interface designed to be intuitive and self-explanatory. To support the user, it offers interactive help: When the mouse hovers over an object or option, a quick description of this is shown in the status bar.

A. WiFi Sensing using WirelessEye

Here is a quick how-to on using WirelessEye:

1) In the *settings*-tab, under *connection*, the IP address or the hostname of the Raspberry Pi running *CSIServer_ng* needs to be entered. Also the remaining settings provided in this tab (e.g., channel bandwidth, etc.) should be adjusted to match the settings on the Raspberry Pi.
2) In the *visualization* tab, a click on the *connect* button establishes a connection to the Raspberry Pi. Upon success, the text in the button will change to *connected*, and streaming data from the Raspberry Pi starts. The CSI data of every received WiFi frame is now plotted in real-time.
3) In the *visualization* tab, the range of CSI amplitudes that are displayed can be adjusted, such that events of interest can be identified in the CSI data.
4) A click on the *record* button initiates storing all further received CSI data into a file. The filename can either be selected in the *settings* tab or is automatically assigned based on the current time and date. The actual filename is shown in the console when the recording starts. WirelessEye supports three different formats for recording, which can be chosen in the *settings* tab. These file formats are documented in the document `doc/fileFormats.pdf`, which is available along with the downloaded code.
5) In the *Preprocessing* tab, all options related to the preprocessing of the CSI data (e.g., gain compensation, etc.) can be controlled.
6) In the *Display Signals* menu, the live visualization of other data, e.g., the RSSI, can be activated.
7) A real-time export of the CSI data, e.g., to a classifier, can be initiated in the *Real-Time Classification* tab. More on this is described below.

V. Real-Time Export

WirelessEye can stream the preprocessed CSI data to any external program, e.g., a classifier that uses machine learning techniques. This is controlled using the *Real-Time Classification* tab. Here, any external program can be executed from *WirelessEye Studio*. The command to be executed and its command line parameters can be specified by the options provided.

The preprocessed CSI data is written to the standard input of the launched executable. The *Simple CSV* data format is used for this, which is documented in `doc/fileFormats.pdf`. The classifier can write its classification results to its standard output, which are read back by WirelessEye. WirelessEye can annotate these results synchronously to the real-time visualization of the CSI data. It also supports displaying the results of multiple classifiers. Since WirelessEye can only launch one executable at a time, the launched executable itself needs to launch additional classifiers if needed, or needs to implement multiple of them in a single executable. The data format for importing results back into WirelessEye is also documented in `doc/fileFormats.pdf`.

A pair of scripts for accessing Tensorflow to 1) train a classifier using previously recorded data, and 2) perform a live classification using the real-time export mechanism are distributed along with WirelessEye. They are outlined below.

VI. Using Real-Time Classification

WirelessEye comes with a pair of scripts to access TensorFlow [6] for generating a machine-learning model. Based on a recorded and labeled set of CSI data, a model can be trained, which can be used both for real-time and offline classification. In order to generate a model, a custom solution to access a machine learning framework needs to be developed. To make this task more accessible, the script `scripts/model_generation.py` can be used as a template or guideline for generating a Tensorflow-based model. This script needs to be adjusted to set the proper parameterization (e.g., sampling frequency of the CSI data, size of a time-window for classification, etc.). After the model has been generated, it is ready to be used for real-time classification. For querying this model, WirelessEye comes with the script `scripts/realtime_classification.py`. This script needs to be modified, e.g., by adding the filename of the generated model (e.g., *model.h5*) and matching the settings used in the learning phase. To execute the model from WirelessEye in real-time, the *Real-Time Classification* tab can be used to launch this script.

References

[1] P. Kindt, C. Turetta, F. Demrozi, A. Masrur, G. Pravadell, and S. Chakraborty, "Wirelesseye: - seeing over wifi made accessible," in *2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2024.

[2] "Wirelesseye repository," https://github.com/pkindt/WirelessEye, accessed October 2021.

[3] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free your CSI: A channel state information extraction platform for modern Wi-Fi chipsets," in *International Workshop on Wireless Network Testbeds, Experimental Evaluation& Characterization (WiNTECH)*, 2019, p. 21–28.

[4] M. Schulz, D. Wegemer, and M. Hollick. (2017) Nexmon: The C-based firmware patching framework. [Online]. Available: https://nexmon.org

[5] "Nexmon csi repository," https://github.com/seemoo-lab/nexmon_csi, accessed October 2021.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/