Wood moisture identification for Internet of Things devices

Moritz Waldleben, June 2021
ESL - Embedded Systems Laboratory, EPFL

I. Introduction

Before processing or cutting wood, the degree of dryness must be thoroughly checked through the entire thickness of the piece, making moisture detection of great interest for the wood industry. However, the available devices are costly, and the non-invasive ones can only detect the moisture content of the surface. To solve this problem, new approaches have been explored that analyze the wood's response to acoustic stimuli for in-depth material analysis.

The change in acoustic response depends on several factors, such as density, temperature, thickness, and moisture content of the given wood. To detect only the changes due to moisture, one possibility is to use machine learning (ML) techniques to relate the moisture content of the wood to the acoustic transmission in the material. This could be used to predict how to further process a piece of wood. This project aims to develop this approach further, taking advantage of new Internet of Things (IoT) devices equipped with multiple sensors and advanced signal processing techniques. Therefore a final product could be a compact measuring device to process and classify real-time data on the acoustic response of wood.

The study will start with a suitable hardware implementation for an embedded platform, followed by analyzing and processing the acoustic data. The data obtained will be used to build an ML algorithm. Finally, the implementation focuses on the optimizations needed to reach a real-time execution on the selected embedded platform.

II. Procedure

To approach this task, the project was divided into several steps:

1) Integration of an embedded platform with an acoustic emission source that would act as excitation for the material (Section III and IV)
2) Offline analysis of the data collected with the built-in microphone to identify the possible preprocessing and a suitable ML algorithm for moisture content quantification. (Section VII-D and VIII)
3) Porting the ML algorithm to C for its execution on an embedded platform. (Section IV)
4) Evaluation and comparison of real-time models based on different preprocessing of the collected data. (Section IX and X)

III. Hardware and setup

The selected embedded platform is the STEVAL-MKSBOX1V1 [1] (named SensoreTile.box) evaluation board from ST Microelectronics. The board is equipped with an Ultra-low-power ARM Cortex-M4 32-bit microcontroller (STM32L4R9) containing 2048kBytes of Flash memory and 640kBytes of RAM. The microcontroller supports digital signal processing (DSP) and floating-point unit (FPU) at 120MHz. The board also contains multiple sensors and a MEMS microphone (MP23ABS1). Additionally, USB or battery powering, a Bluetooth application processor and an SD-card slot for data acquisition are available.

The SensorTile.box is not equipped with an internal loudspeaker. The possible connection from the board is a 14 pin JTAG and SWD interface purposed for debugging. For simplicity reasons, an Arduino Uno was added to the setup to power the acoustic emission source. On the other hand, the Arduino itself has to be triggered. This is done by reprogramming one of the SWDIO pins from the SWD of the board. The chosen loudspeaker uses 1 Watt, 8 Ohms, and has sensitivity of 80dB.

A. Experimental setup

The experimental setup as followed:
The loudspeaker generates acoustic waves on one side of the wood. At the same time, the acoustic impact in the wood is to be measured on the other side with the microphone. To achieve this, a fork-like mount was built, which fixes the loudspeaker and the STM board at a certain distance. The Arduino is triggered by the SensorTile.box, which in turn triggers the speaker. In Figures 1, 2 and 3, views from the top and sideshow the setup.

Components:

1) Loudspeaker
2) Wood sample
3) SensorTile.box
4) Arduino Uno
5) JTAG/SWD connection
6) Built-in Microphone

IV. Implementation

Two main programs have been developed, one to acquire data to train a model and one for real-time acquisition and predictions with the ML model in C. There exist several

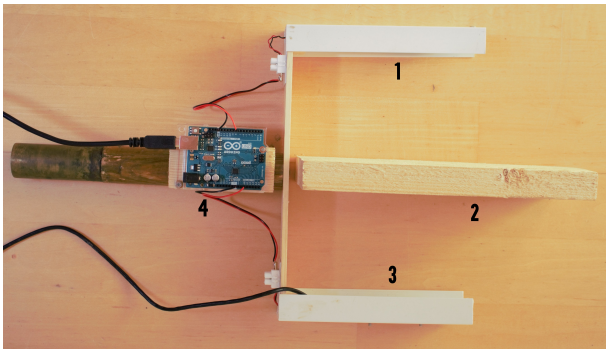

Fig. 1: Experimental setup for moisture identification

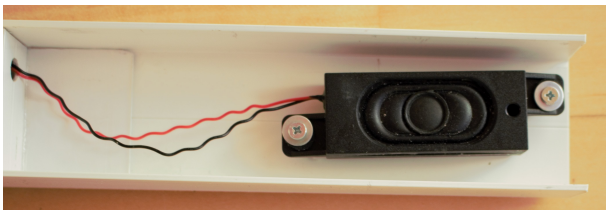

Fig. 2: Loudspeaker attached to retainer

software packs for the board from ST Microelectronics. Two of them were adapted to the current problem. Here, only a short overview will be given on the key aspects of the implementation. The STM32CubeIDE was used as a toolchain.

A. Audio-logging application

The standard firmware [2] of the SensorTile.box includes a sensor data-log application. This software was adapted for higher sampling rates (48kHz, 96kHz). The code involves the real-time-operating system FreeRTOS and the main program runs with one thread. The board support package (BSP) provides a way to receive the analog audio as a preprocessed array. The event to store the digital audio values is triggered with a semaphore release, and a callback function will store the digital values in an intermediate ring buffer. When half full or full, this larger ring buffer will store the values to the SD card.

B. ML-prediction application

The SENSING1 [3] function pack gives a sample application to do audio scene classification with a sampling rate of 16kHz. Classification can be triggered and tracked via Bluetooth by a smartphone. This program was also modified to handle higher sampling rates. Similar to the audio logging application, an intermediate buffer of size 1024 is filled with the digital audio values. These values are then preprocessed with a short-time Fourier transform (STFT) operation and scaled according to a Log-Mel scale resulting in a column with 32 bins. The details of the preprocessing are described in Section VII-D.

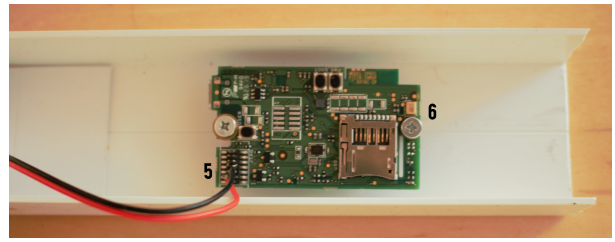

Fig. 3: SensorTile.box attached to retainer

After having processed 30 columns, the neural network will run. Finally, the classification result is displayed as a blinking LED. The Bluetooth tracking is omitted in this case.

V. Preliminary results

To characterize the emission and acquisition system, a frequency swipe from 500Hz up to 48kHz was performed. The swipe consisted of 500Hz steps and one-second emission on each tone, giving a total recording time of 100 seconds. The microphone was sampling at 96kHz, which allows frequency reconstruction up to 48kHz. The bit-depth for the audio was set to 16bit. The resulting frequency spectra are shown in Fig. 4 together with the ambient noise detected by the microphone. The plot shows the absolute values of the FFT amplitudes relative to the maximum captured by the device.

Although power densities measured in the hearing spectrum are dominant, it can be stated that frequencies above 20kHz can be recorded by the loudspeaker, as proved in Fig. 5.

It is thus possible to emit and measure ultrasound waves with appropriate scaling of the measured amplitudes allowing frequencies above the audible range, making the system less susceptible to background noise. Furthermore, it makes it possible to perform a silent acquisition. Here, frequencies between 4-24kHz are considered for a first investigation to cover a large spectrum (see Section VI). It is important to note that some more tests showed that the length of the emission time can be decreased down to 5ms and still provide accurate peaks in the FFT plot.

VI. Sampling and dataset

To construct the dataset, physical samples were selected, classified, and irradiated with an appropriate emission frequency swipe. Finally, a real-time available preprocessing was applied to the sample windows to get the frequency spectrum. The result was fed to the ML algorithm.

A. Physical samples

Fir has been the type of wood selected since it is commonly used as building and construction timber. Wood samples were cut into rectangular pieces (25x10x2.5cm) from a wooden batten. The wood was split into three classes: *dry*, *semi-wet*, and *wet*.

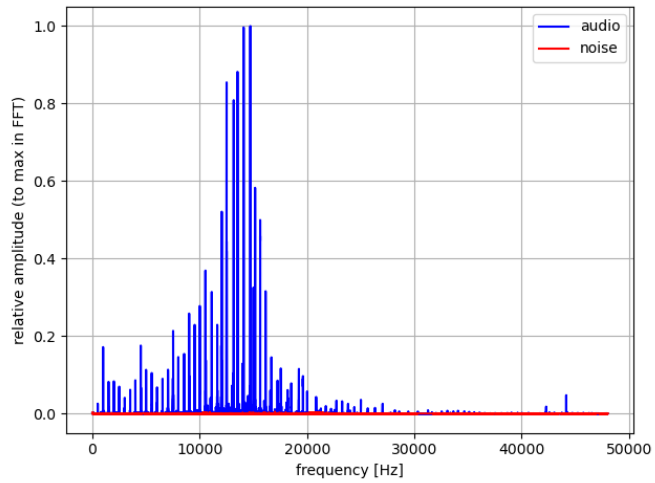

Fig. 4: FFT of recorded step wise frequency swipe from 500Hz to 48KHz

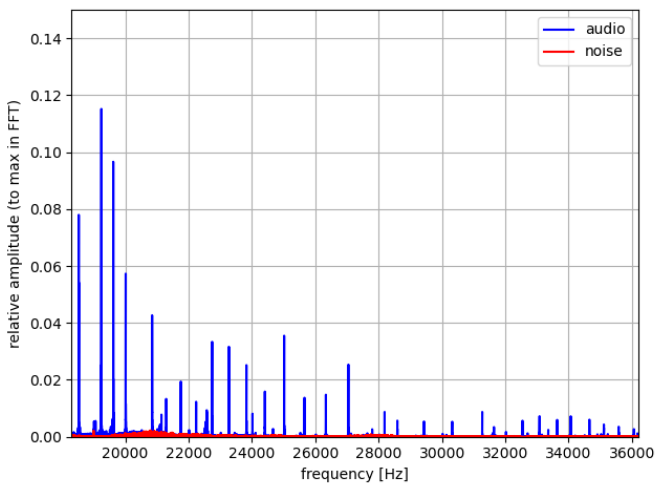

Fig. 5: Zoom of Fig. 4 on a subset of ultra sonic frequencies

- The *dry* class defines two pieces (placed one next to another) of wood having adopted the humidity of a living room. The moisture content usually lies between 8-10%.
- The *wet* class defines two pieces of wood (placed one next to another), that had been immersed in water for at least 10 minutes and therefore had a moisture content of 100%.
- For a semi-wet sample, a dry and a wet piece of wood were placed next to each other (Fig. 6). This does not correspond perfectly to a 50% sample as usually the moisture content is spread throughout the material. For simplicity reasons, this approximation helps to explore if the ML algorithm can detect a change in moisture.

The dataset includes samples acquired with blocks of wood of various thickness to make the model thickness invariant, see Fig. 7. On the other hand, switching between the wet and dry samples for the semi-wet class should as well give no change in classification. These aspects will be considered through the creation of different models.

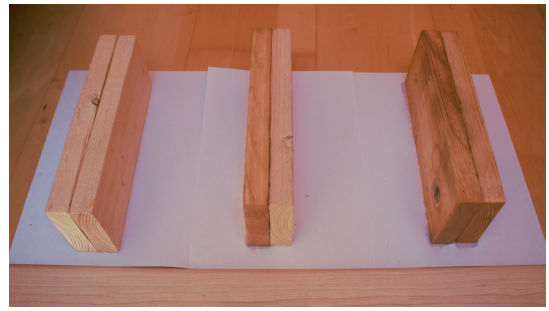

Fig. 6: Three moisture levels. From left to right: *dry, semi-wet* and *wet*

Fig. 7: Half the thickness samples for the dry and wet class

VII. Sample creation

A. Emission loudspeaker

As described in the preliminary results section, a step-wise frequency swipe will be tested. For a first comparison, a sampling frequency of 48kHz was used. The swipe produced by the Arduino was defined as follows:

for *i = 4000Hz to 24000Hz* **do**
  emit tone i for 6ms
  i = i + 400Hz
end

Resulting in a total emission time of 300ms.

B. Microphone acquisition

To subsequently work with an STFT frequency spectrum, 16986 (roughly 353ms at 48kHz) digital audio data points are required.

The emission window is 52ms shorter than the recording window. This is intentional to ensure that a frequency swipe is included in each sample window. And a delay of 53ms is added to the emission speaker to ensure exactly one recording window per swipe.

C. Dataset creation

The dataset construction and model training is implemented in Python. Specifically, the audio preprocessing is done with the librosa library.

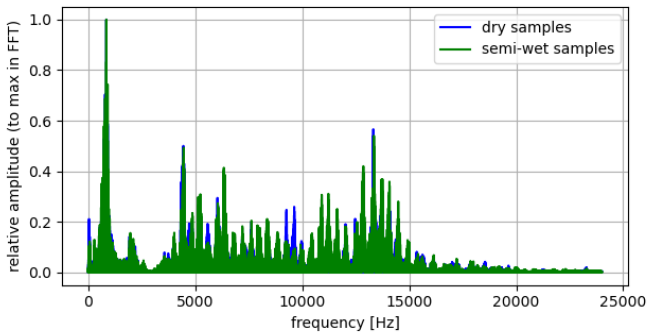

Fig. 8: FFT of 100 frequency swipes, dry vs. semi-wet, one full recording

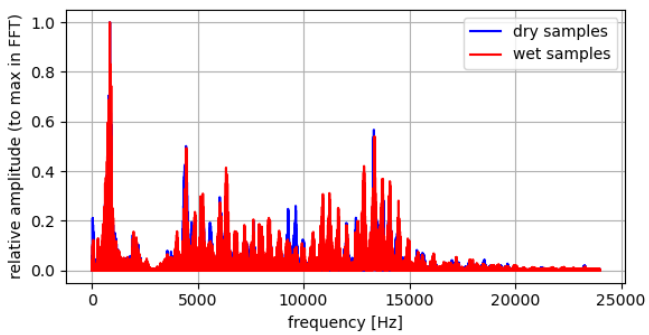

Fig. 9: FFT of 100 frequency swipes, dry vs. wet, one full recording

A complete recording consists of 100 swipes, thus 100 recording windows. The FFT of a complete recording is shown in Fig. 8 and 9. 3 complete recordings were acquired for each sample configuration.

To increase the number of samples in training, 1000 sections were randomly generated from the acquisition with 100 windows. This resulted in a dataset of 18000 samples, including the thickness invariances.

This data set was divided into a training set (75%) and a test set. Further, 25% of the training set was used for validation, meaning in total 10125 samples for training, 5625 for validation, and 7500 for testing.

Overall, this is limited data set and just for evaluation.

D. Preprocessing

Instead of feeding raw audio data to the neural network model, the data is first decomposed into smaller sub-frames to create a set of spectrogram 'features'. These features are then used for model training. Each frame contains 16896 samples (32 * 512 + 512). They create a 32 column spectrogram with an STFT operation. The STFT windows have a size of 1024 and are stored with 50% overlap every 512 samples.

The output is a spectrogram of size 32x513. To reduce the size, a Log-Mel scaling with 30 bins was applied. This converts the amplitudes to Decibels and scales the frequencies according to the Log-Mel formula. The range for the scale can be set and will be considered when creating a model. The final output is a 32x30 spectrogram (Fig. 10). These are essentially 960 features, which are normalized before being fed to the ML algorithm. All in all these many features are still feasible in the real-time C implementation.

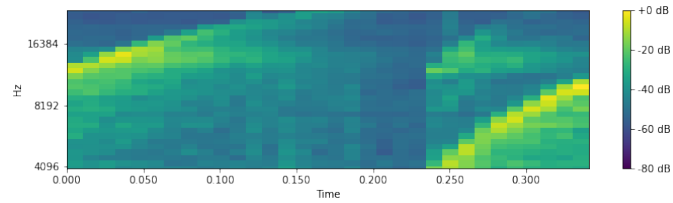

Fig. 10: Log-Mel-spectrogram, 32x30 features of a dry sample

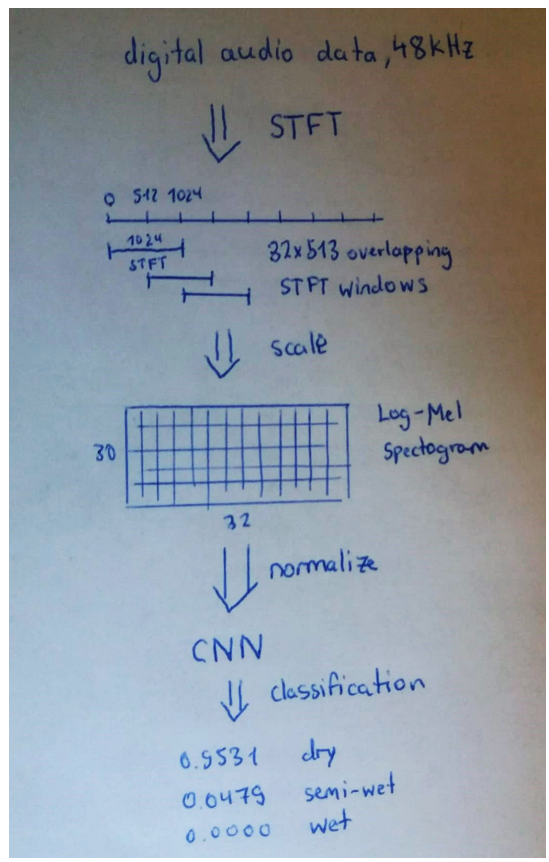

Fig. 11: ML pipeline

Real-time preprocessing:
In real-time, the preprocessing is essentially the same as explained above. Due to limited resources, the STFT and the Log-Mel scaling are applied every 512 samples (around 11ms at 48kHz). This creates one column for the final spectrogram. After 32 executions, enough samples are acquired and the trained model will evaluate the moisture content.

It should be noted that the transformation to a Log-Mel spectrogram is not necessary since the algorithm does not require a scale used for the human audio range. Non the

less the available functions in the ST AI audio preprocessing library provided an easy way to match the 513 STFT bins to the 30 bins of the Log-Mel. In a feature work, this should be optimized and CPU and memory usage have to be analyzed further.

VIII. Neural Network

The chosen ML algorithm consists of a simple convolutional neural network (CNN). This is efficiently implemented by the X-Cube AI tool of ST Microelectronics. The neural network was constructed with two convolutional layers and ReLu activation followed each by a Max-Pooling operation. This is connected in a full layer of 576 features followed by two dense layers. The resulting input feature size after preprocessing is 960 and the output size is a vector of length 3, corresponding to the three classes.

A Tensor Flow lite model was created and ported to C by the X-Cube AI tool. In C, the model has 7836 8-bit unsigned integer values. In addition, 32-bit floating-point values must be stored for normalization before applying the model (2x960 = 1920 parameters).

TABLE I: Architecture of neural network

Layers	**Parameters**
input	30×32 (960) features
Conv2d ReLU MaxPool	$1 \rightarrow 16$ channels, kernel = $(3, 3)$ kernel = $(2, 2)$
Conv2d ReLU MaxPool	$16 \rightarrow 16$ channels, kernel = $(3, 3)$ kernel = $(2, 2)$
Fully connected layer	$576 \rightarrow 9$ features
Final layer	$9 \rightarrow 3$ features

IX. Models

Based on the considerations and arguments of the previous sections, four different models were trained:

TABLE II: Models

Model	With different thickness	Log-Mel scale range
1	No	0-48kHz
2	No	4-48kHz
3	Yes	0-48kHz
4	Yes	4-48KHz

X. Results

The training was standardized with mini-batches of size 500 and running 20 epochs. All models achieve over 99% train, test, and validation accuracy. Overall the real-time models worked well. A slight decrease in inaccuracy was experienced. This phenomenon came from the fact that the models overfit on the complete recording windows used in training. To tackle this problem an independent test set was built where no sections of the recording were used for training. The accuracies for the different sample types are tabulated below.

TABLE III: Models and sample type accuracies of full recordings

Model	dry	semi-wet	semi-wet 2	wet	dry thin	wet thin
1	100	98.8	94.1	100	10.4	96.2
2	99.3	91.3	89.9	100	8.2	100
3	100	98.8	93.8	100	100	99.7
4	91.1	98	98.1	100	85.4	100

XI. Discussion

On the independent test set, all models perform well for the dry, semi-wet, and wet double pieces. Model 1 and 2 have not seen one-piece samples and are unable to predict one-piece dry wood. They are classified mistakenly as semi-wet. On the other hand, even if unseen by the algorithm the thin wet samples are classified perfectly.

The results show that it is possible to determine the moisture content in the wood. Of course, this is only tested within a set of three classes. These high accuracies (95-100%) are attainable because there is a physical relation between acoustic excitation and wood moisture content. But other factors, such as thickness, also influence the transmission in the wood. The distinction in thickness from one to two pieces was introduced in the model. However, when testing a three block wide sample, it is misclassified. This shows the importance of the thickness. It has to be mentioned that the position and length of the samples in the experimental setup were always the same. Additionally, it would be interesting to test different kinds of woods and measure the effect of density change.

There are two ranges for frequency scaling, one from 0 to 48kHz and the other from 4 to 48kHz. The swipe starts at 4kHz. Including lower frequencies, on the cost of having fewer bins at higher frequencies, still gives better overall performance. The sample windows were recorded without background noise. More tests have shown that background noise can be introduced in the model giving better results for the range from 4-48kHz.

Based on the measurement results, applications in industry are imaginable. For example, it would be conceivable to track the moisture content of wood slices on a conveyor belt continuously and noninvasive with the condition that the thickness is predetermined.

A. Further improvements

As shown, this project provides further potential. A shortlist of enhancement and ideas is provided below.

Improve implementation:

- Adjust frequency range, possibly by using a higher sampling rate (noiseless predictions)
- Test linear scaling of the STFT bins that precisely match the emitted frequencies
- Sample window size, number of bins/columns, and the architecture of the CNN are adjustable. This should be further optimized for an efficient implementation.

Improve setup:

- Use another loudspeaker that has a better response at high frequencies
- To have less interference from the sample shape, one could reduce the measurement area and hold the microphone and speaker directly against both sides of the wood surface.

Design a new model:
Train an extended model for example with exact values of moistened pieces of wood from a humidity chamber. Depending on the application, logistic regression can be trained with specific moisture steps.

XII. Conclusion

Moisture content measurement of wood with acoustic stimuli proved to be an efficient and accurate procedure. In this project, an experimental setup was built and the relevant software was adapted to train a moisture detection model and classify it in real-time. The dummy set woods dry, semi-wet and were classified with accuracy's over 95% in real-time on an IoT device. The use of a neural network is justified through the complexity of the problem. However, the thickness of the wood seems to be limiting and needs to be investigated further.

References

[1] Sensortile.box product. https://www.st.com/en/evaluation-tools/steval-mksbox1v1.html. Accessed: 2021-06-12.

[2] Default firmware of the sensortile.box. https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32-ode-function-pack-sw/fp-sns-stbox1.html. Accessed: 2021-06-12.

[3] Sensing1 function pack. https://www.st.com/en/embedded-software/fp-ai-sensing1.html. Accessed: 2021-06-12.