Towards an Air-coupled Ultrasonic Sensor Network and Camera for Non-destructive Testing with Integrated Machine Learning

Stefan Bosse[1]
[1]University of Koblenz, Dept. Computer Science, RG Practical Computer Science, Koblenz, Germany

Abstract - The study introduces a low-cost air-coupled measuring technique using a 1$ MEMS microphone as a signal receiver and a surface-coupled transducer as a sender. Each sensor is integrated with data processing and communication (ICT), providing a "Smart Sensor Node" for signal processing, feature extraction, and Machine Learning for feature prediction. The approach provides higher freedom of design and scalability, especially for spatially large extended sensor networks. The typical spacing of sensor nodes is 2-5 cm, but can be reduced to 4 mm at a minimum. The aim is to provide an active measuring system that processes all signal data locally, including Machine Learning on board. The main research and engineering questions are whether it is possible to integrate a self-contained signal processing system in a resource-constrained embedded computer and if there is a benefit in improving output information quality using a distributed message-passing sensor network arranged in mesh-grid architecture. A proof-of-concept stufy shows the suitability of a distributed sensor network for the detection of hidden defects and damages by using Guided Ultrasonic Waves (GUW).

1. Introduction

There are basically three different measuring techniques with Guided Ultrasonic Waves (GUW) used for Non-destructive Testing (NDT) with respect to the coupling method: Contact-based surface-coupled transducers, immersion fluid coupled transducers, and air-coupled transducers and microphones (investigated by Asokkumar in [1] and earlier introduced by Hillger in [2]). The advantages of air-coupled over contact-based techniques are outlined and discussed by Stoessel in [3], but still using an separated signal measuring and processing architecture (off-line processing) . We introduce and demonstrate a fully low-cost air-coupled measuring technique with a 1$ MEMS microphone as a signal receiver and a surface-coupled transducer as a sender. Each sensor is fully integrated with data processing and communication (ICT) providing a "Smart Sensor Node", performing signal processing, feature extraction and Machine Learning for feature prediction. Initially, only one microphone is used in a spatial 2-dim. scanning mode. Transducer arrays are well known and established, but used normally close together to provide phase-sensitive measurements. The next step is an array of such MEMS microphones, in this work, a 3×3 matrix network, which should be later extended to a 10×10 network (or more nodes), towards an spatially extended Ultrasonic camera. To avoid a separation of the microphones from the analog-digital conversion (ADC) processing units (with a bunch of cables) we couple the microphones directly to a microcontroller placed near by the microphone with a 1:1 node mapping. Each node uses a low-cost 32-Bits STM32 ARM-Cortex microcontroller equipped with ADC and communication controllers. All nodes are connected in a mesh-like grid creating a distributed communicating network cluster computer. Additionally, signal pre-processing and application-specific distributed Machine Learning will be implemented in this tiny sensor network on node-level. We will deploy and evaluate the first mock-up prototype for fast and real-time hidden defect detection in laminate structures (e.g., Fibre-Metal Laminates).

This work addresses ultrasonic monitoring as well as distributed computing and distributed algorithms. Compared with conventional transducer arrays coupled to a single measuring and signal processing unit, our approach provides a much higher freedom of design and there is basically no limit on scalability, especially regarding spatially large extended sensor networks. The typical spacing of sensor nodes (microphones) is about 2-5 cm, but can be reduced to a spacing of 4 mm at a minimum, limited by the size of the microphone. The aim is to provide an active measuring system that processes all signal date locally including Machine Learning on board.

The two main research and engineering questions are: Is it possible to integrate an entire and fully self-contained signal processing system in such strictly resource-constrained embedded computer which is capable to provide sufficient information to detect hidden damages by using directly measured Ultrasonic signals? And is there is a benefit (improvement of output information quality) by using a distributed message-passing sensor network arranged in mesh-grid architecture?

The data processing part should include all parts of:

1. Signal acquisition (ADC and DAC control);
2. Digital signal pre-processing;
3. Data transformation, scaling, normalization;
4. Message-based communication in mesh-like networks (code and data);
5. Synchronization (clock, measurements);
6. Dynamic program-controlled processing and HAL using an advanced Virtual Machine;
7. Machine Learning (e.g., FC-ANN, decision trees, SVM) using integer arithmetic.

2. Node Architecture

Each sensor node consists of:

1. A microcontroller (32-Bits ARM Cortex M0/M4 CPU core, optionally with FPU) with RAM (about 10-20 kB) and ROM (about 64-128 kB);
2. At least one ADC (12 Bits resolution, at least 1 MS/s sampling rate) with an analog multi-channel selector (at least 4 channels), DMA capable;
3. Optionally at least one DAC (12 Bits resolution, at least 2 MS/s sampling rate), DMA capable;
4. Timer with microseconds resolution;
5. Serial communication channel devices, e.g. UART, I2C, or SPI, at least 100 kb/s bandwidth and at least two devices (channels), DMA capable;
6. Digital control bus; and
7. Analog signal amplifiers arranged in a cascade circuit, at least 300 kHz bandwidth with total amplification factors in the range from 20 - 160.

The basic node architecture is shown in Fig. 1. The pitch signal is generated by a surface coupled piezo transducer. The high voltage amplifier is actually not part of the sensor node. The sensor is a MEMS microphone (SPU0410LR5H, Knowles[6]), which is typically deployed in consumer electronics like smartphones. The datasheet proposes a nearly linear frequency response up to 10 kHz and from 40-80 kHz, with a higher sensitivity in the range of 20-30 kHz. This frequency response is sufficient for air-coupled Ultrasonic measurements with a base pitch frequency around 40 kHz. The MEMS microphone has a very small footprint of 3.7×3.0 mm, well suited for a sensor mesh-grid with sensor distances (grid spacing) down to 4 mm, as proposed in this work.

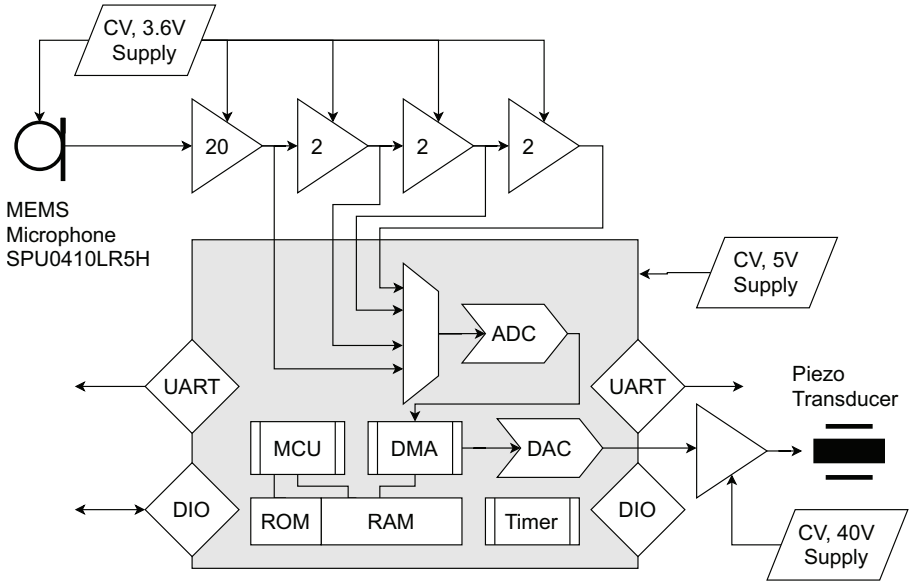

Fig. 1. Sensor node architecture

3. Network Architecture

Each sensor node provides at least two communication links consisting of:

1. A serial data communication link either consisting of two unidirectional links (RX/TX lines) or one shared bus link (Data and clock line);
2. A digital control bus consisting of a request (REQ) and a combined acknowledge-busy (ACK/BUSY) line that can be driven by both connected sides.

The sensor network architecture is shown in Fig. 2. The dotted lines are only available if a sensor node provides at least three communication channels creating a full mesh-grid network, otherwise a chained network is created. At least one sensor node is connected to an external computer.

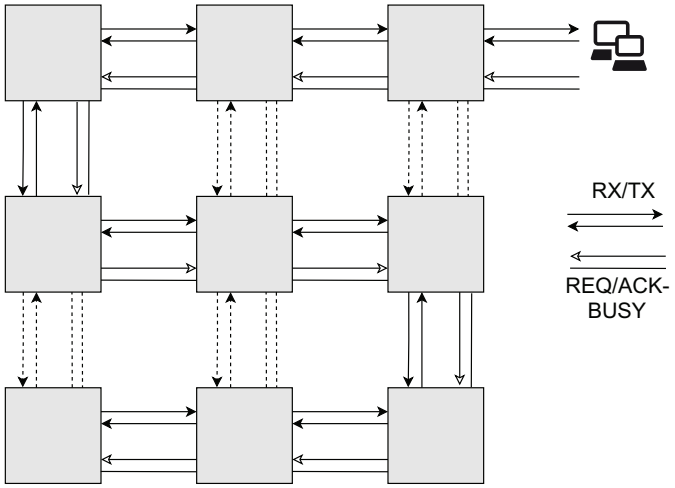

Fig. 2. Sensor network node architecture with communication links

The digital control bus using hardware ports is split into a directed and undirected connection, used by two nodes concurrently, although both lines are implemented with input pull-up ports allowing both sides to drive the line. The control bus serves different purposes:

1. Fast directed event propagation through the network, e.g., a capture start or clock synchronization events;
2. Communication control for the data links (busy signal, bi-directional).

The data communication protocol and the relation to the control bus is shown in Fig. 3. There are different text-based message formats determined by the outer delimiters. Program code `[..]` is propagated through the entire network (downwards), i.e., the network is a Single-Instruction-Multiple-Data computer (SIMD) architecture. Each node processes the same program. A received program is compiled and started. Alternatively, programs can be messaged incrementally (line by line), i.e., pre-compiled but started later if the program is complete. Data vectors (of numbers and strings) can be transferred peer-to-peer (P2P) between neighbor nodes (`$..$`) with event-based processing or propagated through the network (`!..!`) without processing (always upwards, commonly longer measuring data messages), basically delivering data to an external computer for further processing.

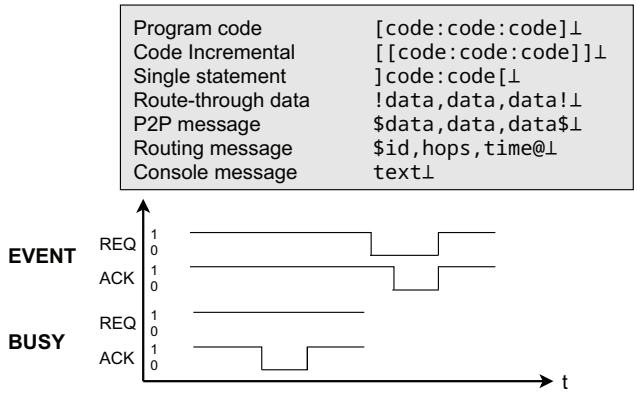

Fig. 3. Communication protocol and control bus states (bottom is a message terminator symbol, commonly a new-line character)

Using two link channels per node, i.e., creating a chain network, simplifies communication significantly (compared to a mesh network with more than two links per node), and one link is a down link, and the other link is the up link. The entire sensor network processes the same code (SIMD machine). Code is always propagated downwards, data and console messages are always propagated upwards. The disadvantage of a chain network in a mesh-grid placement of sensor nodes is a linear increasing latency of signal and code propagation. Measurements of the hardware signal latency (node to node) showed a delay of about 3 µs per node connection (using pin interrupt handler on the receiver side, STM32 F303K8 with 64MHz core frequency). Although the signal delay is high, the jitter (statistical variations) were below 100 ns.

Each sensor nodes executes the same program but requires a unique identification number (UID). The UID is passed on start-up by an injection of a control message `@uid,hopcnt,time@` (first node). If this message is received by a node it parses the content, assigns the *uid* field to its UID, and forwards the message to the next node with incremented *uid* and *hopcnt* fields. This messages is always send downwards (down link direction).

4. Virtual Machine

The VM consists of a lexer, compiler, and an execution loop. The input is always a program text, output is a pre-compiled token stream that is processed by the execution loop. The programming language is a BASIC dialect, one of the older interpreter languages. BASIC is both a sequential procedural and an event-based language well suited for IO and messaging applications. BASIC supports basic constructs for event and error handling which is required for embedded systems. Originally BASIC was interpreted, i.e., the input text was directly executed. A widely used dialect is TinyBASIC with an open-source implementation [8], which uses a token lexer, but without a real compiler, more being an interpreter. The MBBASIC VM introduced in this work is a hybrid architecture with a token-based processor and a just-in-time compiler. The original TinyBASIC token stream was nearly a one-to-one mapping of text onto tokens. The token stream here is compiled into a compacted Bytecode stream including resolved branch addresses in advance (post-token compilation). In contrast to previous work [4] using the FORTH language and a pure multi-stack processor, MBBASIC can be programmed more natural with improved event management. MBBASIC is targeting microcontrollers with lowest RAM and ROM resources of about 10 kB and 64 kB, respectively, and is optimized for STM32 microcontrollers, but not limited to. Fast Core-coupled Memory (CCMRAM) can be utilized [9].

4.1 Core Architecture

The programming language is line-orientated, consisting of data and control statements. A stand-alone program transferred via the serial link used to perform a measuring task with a single node is shown in Example 1. An in-line event handler is used to wait for the completion of the ADC task. The pitch signal is generated by an built-in Gaussian windowed sine wave function (`vgausswin`). The built-in combined `adac` function starts the DAC task to output the signal and immediately after the ADC sampling task. The ADC channel selects the gain amplification stage. Example 2 shows a typical event-driven message processing program providing clock synchronization. The main program installs event handlers by using the `event` function. The handlers are called by the VM loop signaled by the host application processing low-level communication tasks and interrupt service routines. There is a `micros(n)` and `millis(n)` function which returns the current node-local time in micro- or millisecond resolution divided by n (scaling factor to fit into 16 Bits integer values). Each time a hardware request signal is received, a timestamp (microsecond resolution) is set, which can be used to synchronize a clock.

```
ADSAMPLES=100:DACFREQ=12500
SIGSAMPLES=2000:SIGAMPLITUDE=1100
SIGFREQUENCY=40:SIGXOFFSET=50
SIGXWIDTH=40
dim tdata(SIGSAMPLES)
dim sdata(ADSAMPLES)
vgausswin($tdata,SIGSAMPLES,SIGFREQUENCY,
          2048-SIGAMPLITUDE,2048+SIGAMPLITUDE,
          SIGXOFFSET,SIGXWIDTH)
rem wait for trigger
on event REQ1,EVSIG stop
adac($sdata,ADSAMPLES,ADCHANNEL,$tdata,SIGSAMPLES,DACFREQ)
rem wait for ADC finished samplinh
on event 1,EVADC stop
rem send signal data (via up-link)
print #",","!",sdata(),"!"
```

Ex. 1. An example of a standalone measuring program computing the pitch signal, starting the DAC, starting the ADC sampling, and finally sending the results via the serial link. This program was used to record the signal data in this work.

```
clocksync=0: time0=0: dim msg(8)
CLOCKA=1: CLOCKB=2
rem install event handlers
on error goto 3000
on event REQ1,EVSIG gosub 1000
on event REQ2,EVSIG gosub 1000
on event LINK1,EVINP gosub 2000
on event LINK2,EVINP gosub 2000
rem stop idle
stop
```

```
11: 1000: rem request handler for REQ event
12: if clocksync=1 then time0=micros(100): ack(@event)
13: return
14: 2000: rem message handler
15: input cmd,arg1
16: rem start clock synchronization, wait for hardware signal
17: if cmd = CLOCKA then clocksync=1
18: rem finalize clock synchronization
19: if cmd = CLOCKB then clock(100,time0,arg1): clocksync=0
20: rem forward message to next link
21: if event = LINK1 then print &LINK2,cmd,arg1 else print &LINK1,cmd,arg1
22: return
23: 3000: rem error handling
```

Ex. 2. A typical event-based program. The program starts at the first line and installs event handlers. The program is suspended by the `stop` operation, but it is not discarded. The main IO loop will raise events finally calling the appropriate event handler subroutines.

The overall Virtual Machine architecture with control and data flows is shown in Fig. 4, consisting of the main IO loop of the host application program, the VM compiler and execution loop functions, which can be executed stepwise.

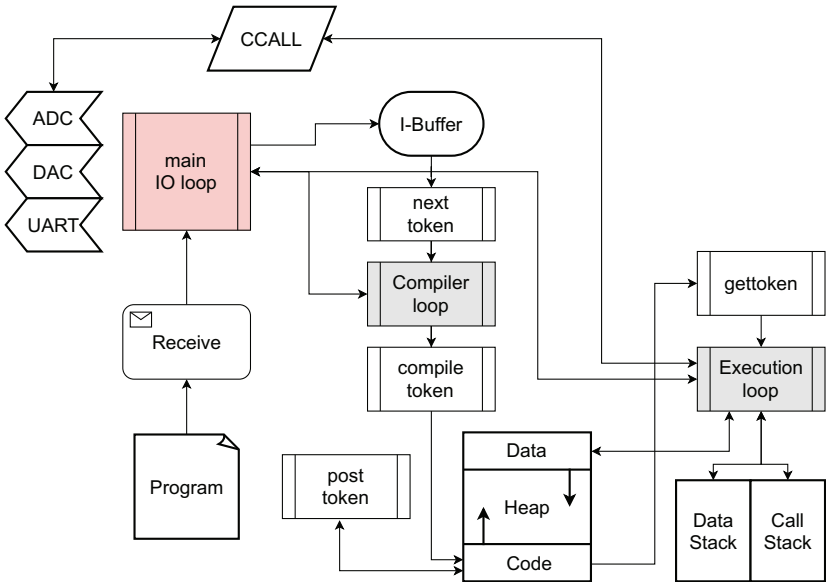

Fig. 4. Virtual Machine architecture with control and data flows. The main IO loop (host application) controls the compilation and execution of the program. The maximal number of computational steps (processed tokens) as well as interruption can be controlled by the main IO loop.

4.2 Machine Learning ISA

The VM supports the computation of ML models (inference only), mainly ANN, with a set of unified vector operations. All parameters and intermediate data is stored in (16 Bits) Integer format. Internal computations, especially calculating the non-linear functions, can be performed with approximated Integer arithmetic, but is done with Floating-point arithmetic if available (in hardware). The ML Instruction Set Architecture (ISA) is the same as in [4]. The model parameters (weights, bias, scaling factors) can be either embedded in the program code using the `DATA` and `INPUT`statements, or by reading the parameters from separate messages using the `INPUT` statement.

5. Ultrasonic Camera and Spatial Scanning Measuring Method

This work investigates the design, programming, and communication of a distributed sensor network. The distributed sensor network consists of single nodes that measure nearby Ultrasonic signals with a MEMS microphone. Node distances down to 4 mm can be achieved theoretically. The nodes should process the measure signals to predict a (hidden) nearby damage of the test material by using ML classifier models. Finally, a compact GUW camera is created that can be placed above surface (with external transducer to inject GUW). The signal data can be read out for further processing, too. A prior proof-of-concept study should evaluate the design parameters and the suitability of the proposed concept by using a spatial scanning technique, as shown in Fig. 5. The same microphone and electronics are used as in the final distributed sensor network, but the signal processing is performed off-line in a separate computer. The singe-board camera design is not discussed in this paper, only the proof-of-concept study.

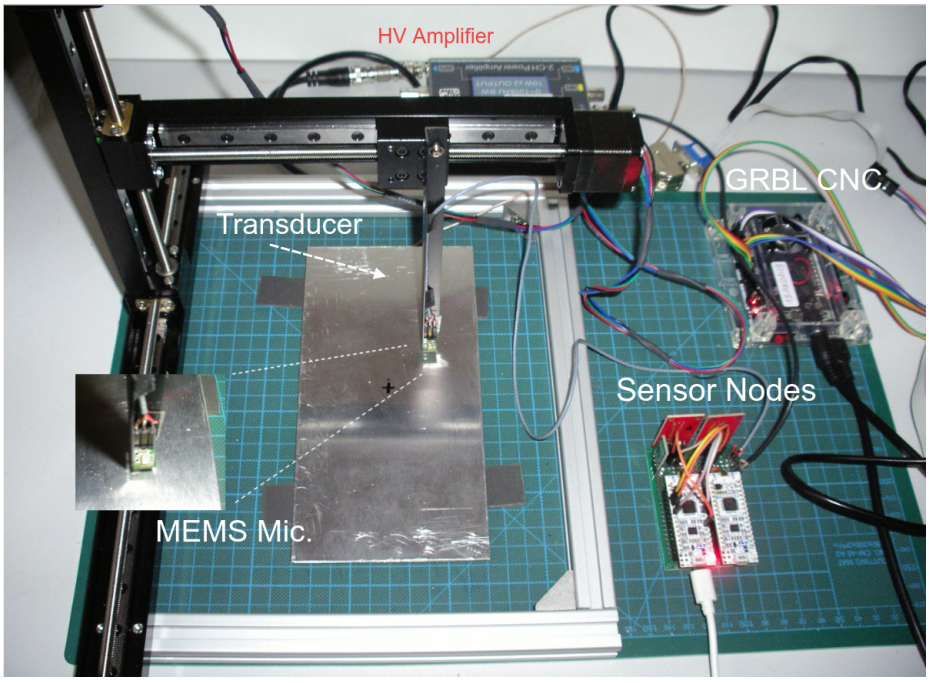

Fig. 5. The experimental prototype: The 3-dim. portal arm device with a mounted MEMS microphone and the test plate. On the lower right side there is an experimental demonstrator consisting of two sensor nodes and the amplifier cascade circuit. On the upper right side there is the GRBL CNC controller driving the portal arm machine. The transducer is mounted on the down side of the plate and driven by a high-voltage amplifier (V_{max}=40 Vpp, f_{3dB}=100kHz, G=20dB).

6. Data-driven Defect and Damage Prediction

The purpose of this work is a fully self-contained and integrated damage and defect detection system using GUW signals as an input. A data-driven classificator model is used to predict a damage or defect nearby a particular sensor. If we assume a mesh-grid sensor network, all sensors signals can be merged as a three-dimensional signal tensor $\boldsymbol{S}(x, y, t)$.

It is assumed that GUW signals interact with hidden damages or defects which can be extracted from the measured signal $s(t)$. Commonly, the raw time-dependent signal is transformed in the frequency domain. Fourier transforms are expensive and not well suited for damage feature extraction, so we use a fast discrete wavelet transform (FWT) by applying a simple Haar wavelet function, which can be finally be implemented with Integer arithmetic on microcontrollers.

The data-flow architecture implemented in each sensor node (except model training) is shown in Fig. 6. After signal pre-processing and normalization, the FWT is applied. The filtered signal is the input for a simple Fully Connected Neural Network (FCNN) with three layers and 10 Neurons in each layer, as shown in Alg. 1. An optional down-sampling of the filtered signal (by factor 4) can be applied, reducing the input vector of the FCNN and the number of parameters. The hyperbolic tangent function was chosen as an activation function. Experiments showed that a sigmoid function is not suitable for this problem (resulting in a significantly increased classification error). The experiments (off-line) were performed with the ConvNetJS software framework [7] using floating point arithmetic. After training and test the model was analyzed and transformed (scaled) into an 16 Bits signed Integer arithmetics model, which can be implemented in and processed by the sensor node microcontrollers using the proposed VM.

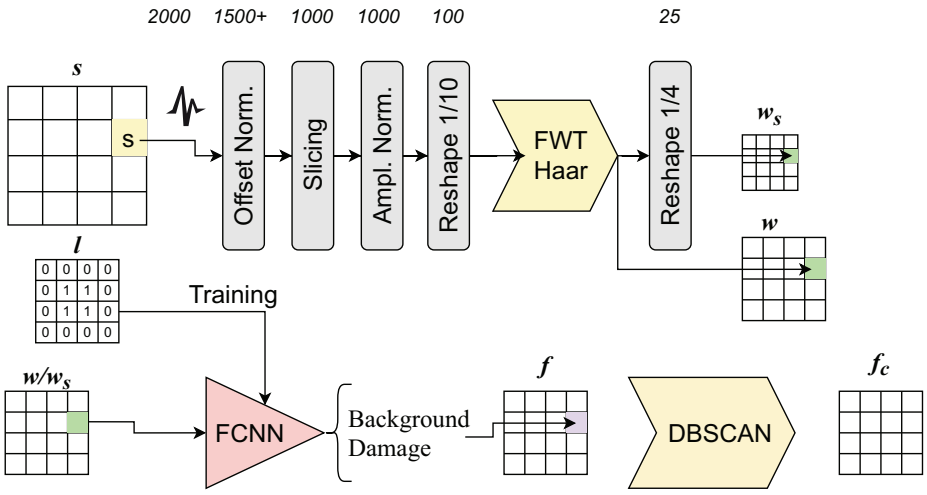

Fig. 6. Data-flow architecture implemented in each sensor node (except model training)

```
Fully Connected Neural Network
============================================================
Classes:    2 {B,D}
Input:      [N,1,1]
Output:     [2]
```

```
Layers:    [L5 P9]
[1] input :             out=[N,1,1]  params=0
[2] fc :      in=[25]   out=[10]     params=N*10+10
[3] tanh :              out=[10]     params=0
[4] fc :      in=[10]   out=[10]     params=110
[5] tanh :              out=[10]     params=0
[6] fc :      in=[10]   out=[10]     params=110
[7] tanh :              out=[10]     params=0
[8] fc :      in=[10]   out=[2]      params=22
[9] softmax : in=[2]    out=[2]      params=0
-------------                        -------------------
Predictors: N                        Parameters=N*10+252
```

Alg. 1. FCNN summary with N as the size of the input vector (output from FWT)

7. Experiments and Evaluation

The experimental work-flow was as following:

1. Preparing a simple composite test plate consisting of three aluminum layers (0.5 mm thickness, top, middle, bottom layers) and two fibre-glass FR4 layers (inner layers). A 8 mm hole was drilled into the three inner layers at the center position. The size of the plate is 150x100 mm. All layers were glued with Epoxy resin under moderate load, but without applying vacuum and heat.
2. Recording of spatially distributed sample data using the air-coupled GUW scanning method with the same microphone and electronics (STM32 F303K8 with FPU) that are used in the sensor network.
3. Training and test of the FCNN model (float32 data type) using signal data from three different data sets with different microphone-surface heights (Z=4,5,6 mm), finally applying DBSCAN to the feature images to reduce noise.
4. Transformation of the trained float32 model into a scaled int16 model suitable for microcontroller processing. Performing tests and analysis, too.

The training and test data require annotated label data, too. The binary labels (no damage, background := B=0, and damage/defect := D=1) were assigned based on the known geometric parameters of the plate defect (square of 10x10 mm at center position). In total there were 75 positive (P) and 2808 negative (N) signal samples, with a training sub-set of 300/75 N/P samples randomly selected.

The statistical quality is shown in Tab. 1, achieved after 2000 training epochs and a batch size of 1. Please note that the results only reflect a proof-of-concept evaluation missing a rigorous statistical analysis, a solid experimental plan, data augmentation by using Monte Carlo simulation methods, and data cleaning. The overall quality with respect to true-positive (damage correctly detected) and true-negative (no damage correctly detected) is satisfying, although there is an increased false-positive rate of about 10%. There is no significant difference between float and integer arithmetic models, and there is no significant lowering of the prediction quality with the reduced input feature vector (25 instead of 100 elements). This is an important result because it reduced storage requirements significantly (about 500 versa 1200 model parameters). The results are mostly independent from the microphone-surface distance, which is important for the single board GUW camera, although the signal differ significantly, as shown in Fig. 7 (Bottom).

Parameters	Error [%]	F1 Score	FP [%]	FN [%]
FWT=100 DT=Float32	8.3	0.89	10	0
FWT=25 DT=Float32	6.8	0.90	10	4
FWT=100 DT=Int16	8.3	0.90	12	2
FWT=25 DT=Int16	6.6	0.89	10	2.6

Tab. 1. Statistical evaluation of the experiments (FCNN classification results without clustering)

The feature image maps are shown in Fig. 7, Fig. 8, and Fig. 9, for the floating point and integer arithmetic models. The false-positive noise can be removed by applying a classical density-based clustering algorithm (DBSCAN).

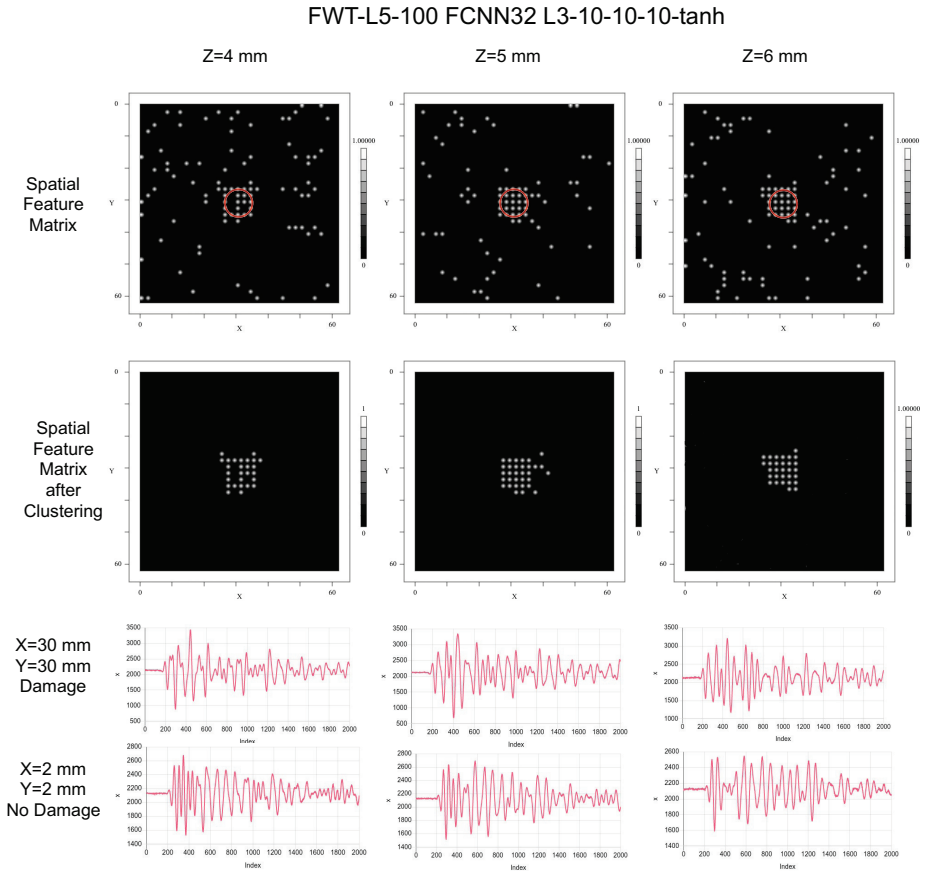

Fig. 7. Selected results for the float32 FCNN model and 100 FWT points (Top) Damage prediction results as a spatial feature matrix with signals measured at different heights above surface; mm scale (Middle) Result image after clustering (Bottom) Selected signals at different heights and positions; arbitrary scales. The red circle is the ground truth labeling.

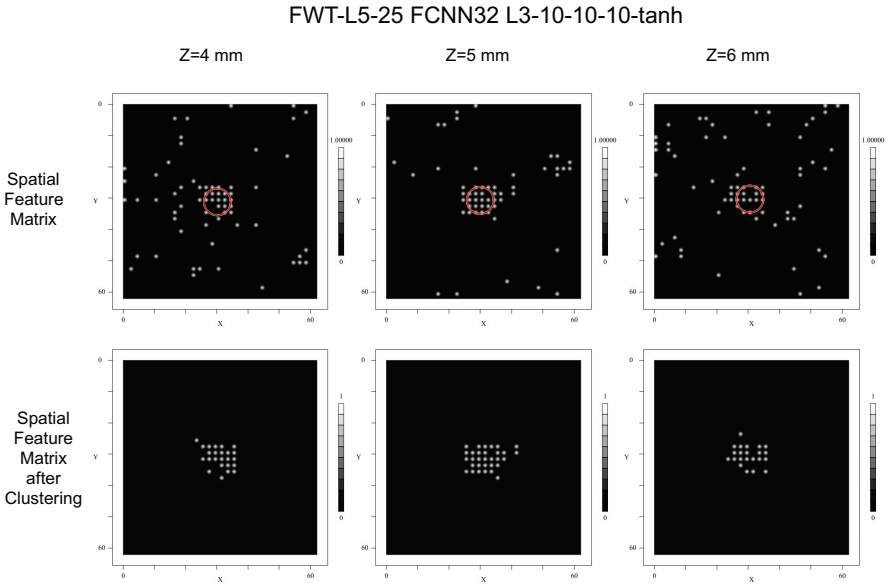

Fig. 8. Selected results for the float32 FCNN model and 25 FWT points (Top) Damage prediction results as a spatial feature matrix with signals measured at different heights above surface; mm scale (Bottom) Results after clustering. The red circle is the ground truth labeling.

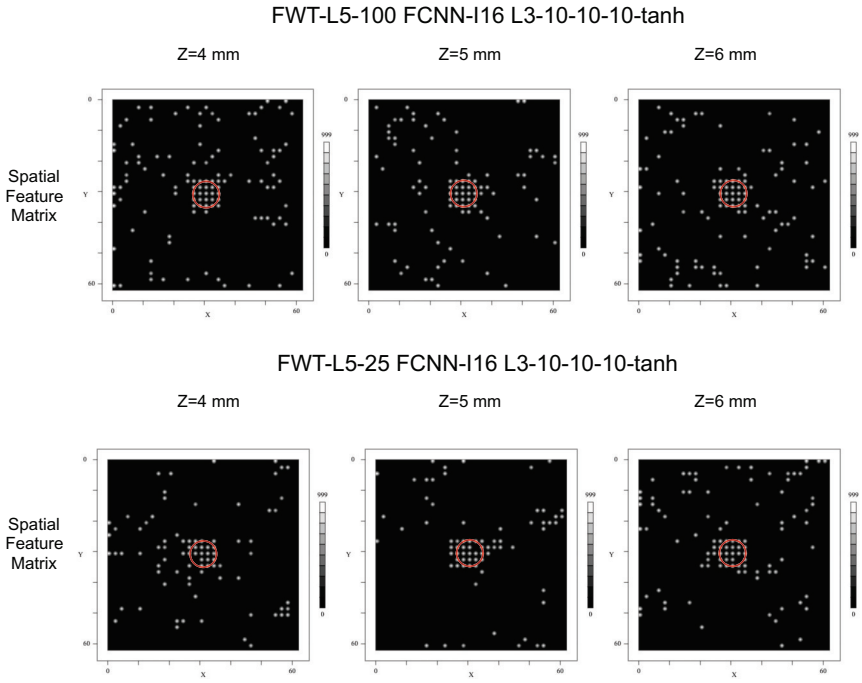

Fig. 9. Selected results for the int16 FCNN model and 100/25 FWT points (Top) 100 FWT points (Bottom) 25 FWT points (down-sampled). The red circle is the ground truth labeling.

Finally, some results measured by the sensor node and distributed computer system are shown in Tab. 2.

Parameter	**Average**	**Variance**
Hardware Signal Propagation (Node-to-Node)	3 ms	<100 ns
VM Execution Time (1 Token==VMachine Instr.)	2 ms	<100 µs
FCNN Computation + FWT (STM32)	< 100 ms	< 1 ms
Signal Recording	12 Bits, 2MS/s, 2000 points (1ms)	jitter < 100 ns

Tab. 2. Some selected measured parameters of the distributed computer system

8. Conclusions and Outlook

This work introduced a low-cost air-coupled measuring technique using a 1$ MEMS microphone as a signal receiver and a surface-coupled transducer as a sender. Each sensor is integrated with data processing and communication (ICT), providing a "Smart Sensor Node" for signal processing, feature extraction, and Machine Learning for feature prediction, finally composed to a distributed sensor network with mesh-grid communication. A simple FCNN model was able to predict a hidden damage in a Fibre-Metal laminate plate with high accuracy. If applied in a spatially distributed sensor network, neighboring node correlation and clustering can be used to reduce false-positive predictions (background noise) significantly, here demonstrated by the DBSCAN algorithm for the sake of simplicity. Each sensor node provides the full processing work-flow including the ML inference and is fully program-controlled by uisng an advanced VM architecture. All network nodes process the same program. In this work only a proof-of-concept study with spatial air-coupled GUW scanning (but with the same sensor) was shown, but the results can be transferred to a real microcontroller-based sensor network creating a GUW camera system.

9. Abbreviations

ADC: Analog-Digital Converter
CCM: Core-coupled Memory
DAC: Digital-Analog Converter
DMA: Direct Memory Access (data transfers between devices and memory without CPU involvement)
FPU: Floating-point Unit (arithmetic)
FWT: Fast Wavelet Transformation
I2C: Inter-Integrated Circuit
MCU: Microcontroller
RAM: Random Access Memory
ROM: Read Only Memory
UART: Universal Asynchronous Receiver-Transmitter (RS232)
SPI: Serial Protocol Interface

10. References

[1] A. Asokkumar, E. Jasiūnienė, R. Raišutis, and R. J. Kažys, “Comparison of Ultrasonic Non-Contact Air-Coupled Techniques for Characterization of Impact-Type Defects in Pultruded GFRP Composites,” Materials, vol. 14, no. 1058, 2021.

[2] W. HILLGER, L. BÜHLING, and D. ILSE, “Air-coupled Ultrasonic Testing-Method, System and practical Applications,” in 11th European Conference on Non-Destructive Testing (ECNDT 2014), October 6-10, 2014, Prague, Czech Republic, 2014.

[3] R. Stoessel, N. Krohn, K. Pfleiderer, and G. Busse, “Air-coupled ultrasound inspection of various materials,” Ultrasonics, vol. 40, 2002

[4] S. Bosse, A Virtual Machine Platform Providing Machine Learning as a Programmable and Distributed Service for IoT and Edge On-Device Computing: Architecture, Transformation, and Evaluation of Integer Discretization, Algorithms. 2024; 17(8):356. https://doi.org/10.3390/a17080356

[5] C. Polle, S. Bosse, D. May, Transformation of Guided Ultrasonic Wave Signals from Air Coupled to Surface Bounded Measurement Systems with Machine Learning Algorithms for Training Data Augmentation, in Proceedings of the 11th International Electronic Conference on Sensors and Applica-

tions, 26–28 November 2024, MDPI: Basel, Switzerland, doi:10.3390/ecsa-11-20448

[6] SPU0410LR5H-QB, Zero-Height SiSonic TM Microphone, Knowles Electronics, datasheet from 27.3.2013, https://www.knowles.com, accessed on-line 1.1.2025

[7] ConvNetJS, https://cs.stanford.edu/people/karpathy/convnetjs, accessed on-line, 1.2.2025

[8] Stefan's BASIC, https://github.com/slviajero/tinybasic, accessed on-line, 1.2.2025

[9] Use STM32F3/STM32G4 CCM SRAM with IAR Embedded Workbench, Keil MDK-ARM, STMicroelectronics STM32CubeIDE and other GNU-based toolchains, https://www.st.com/resource/en/application_note/an4296-use-stm32f3stm32g4-ccm-sram-with-iar-embedded-workbench-keil-mdkarm-stmicroelectronics-stm32cubeide-and-other-gnubased-toolchains-stmicroelectronics.pdf, accessed on-line 1.2.2025