2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #33203

Implement Your DSP Algorithm on Android Tablet: Real-time DSP Laboratory Course

Prof. Thomas Moon, University of Illinois at Champaign Urbana

Thomas Moon received the B.S. degree in electrical electronic engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 2008, and the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA, USA in 2015. Between 2015 and 2017, he worked at IBM in Burlington, Vermont where he developed mmWave test equipment as a principle development engineer. He joined Coordinated Science Lab, University of Illinois at Urbana-Champaign (UIUC), IL, USA in 2017 as a post-doctoral researcher. He has been a Teaching Assistant Professor at Department of Electrical and Computer Engineering at UIUC. His current research interests include wireless sensing and communication in mmWave.

Prof. Minh N. Do, University of Illinois at Champaign Urbana

Work-In-Progress: Implement Your DSP Algorithm on Android Tablet: Real-time DSP Laboratory Course

Abstract

The rapid development of embedded systems brings new opportunities for modernized real-time digital signal processing (DSP) education. This paper introduces a novel real-time DSP laboratory course that aims to give students hands-on experience with real-time embedded systems using Android tablets at an early stage of their careers. The students broaden and deepen their understanding of basic DSP theory and techniques and learn to relate this understanding to real-world observations and applications. The students learn industrially relevant skills such as rapid design prototyping in Python and Android development of DSP applications in C++/Java for computationally constrained mobile devices. The course advances in two phases: structured labs and team projects. In the first half of the course, a series of structured labs are provided to implement and analyze real-time DSP systems that utilize fundamental DSP concepts acquired in the introductory signal processing course. The fundamental concepts include topics such as FIR and IIR filtering, multi-rate processing, sampling, windowing, and spectral analysis. The remaining weeks in the course are about implementing and simulating a DSP algorithm of a student's choice from a set of seminal DSP papers such as adaptive filtering, pitch detection, edge-aware filtering, motion tracking, pattern recognition, etc. The team project revolves around the development, testing, presentation, and documentation to help the students defend their proposed design and receive feedback from the teaching staff. We have offered this course for four years, and the student's feedback in the form of survey questionnaires has confirmed that this course has been successful.

1 Introduction

The growing popularity and trend of mobile devices have impacted our lives in a wide range. In particular, mobile devices such as smartphones and tablets have become ubiquitous. Such mobile devices with higher computing power now integrate various I/O modalities, including high-resolution cameras, microphones, speakers, and IMU sensors. The hardware improvement has inspired new mobile applications in signal processing areas such as human voice recognition, gesture tracking, music discovery, face recognition, etc. Consequently, it is important to reflect these trends in the embedded DSP education.

However, conventional courses tend to persist in the traditional application whose modality has been one-dimensional audio or communication signal. Most embedded DSP courses take a bottom-up approach, starting from the architecture, data types, and I/Os of the hardware platform, then reaching the DSP topics such as digital filters and spectral analysis. Such limitations come from their hardware platforms, typically a development kit for digital signal processors. For example, the DSP development kits from Texas Instruments are the most popular choices in the embedded DSP courses [1, 2, 3]. With the bottom-up approach, the students often find it challenging to absorb all the hardware details and are likely to lose interest before reaching the main DSP topics. Moreover, the development kits are typically limited by their computing power, memory, and modality of I/O's. Such limitation hinders the scope of courses from more advanced applications in image and video processing.

In this paper, we introduce a novel senior-level real-time DSP laboratory course. The course targets senior Electrical and Computer Engineering (ECE) students who already took their first DSP course and introductory programming courses of Python and C/C++. However, the programming skills used in the course do not require advanced concepts. The course aims to introduce the fundamentals of real-time DSP by purposefully hiding the complexities associated with the hardware. This enables students to stay focused on learning new signal processing topics and progress toward the non-traditional DSP topics. We adopted Android tablets for the hardware platform. Unlike the conventional DSP development kits, the Android tablet supports higher computing power, more memory space, various I/O modality, a user-friendly interface, and high mobility. With less burden and hassle from the hardware, the course can provide a group project helping students explore research-level topics in the signal processing field and implement one of them. Although the course targets senior students, this aspect also attracts some graduate students (10% of the students registered in Spring and Fall 2020). This paper is an extension of work that studied a hybrid real-time DSP course using DSP processors and Android tablets together [4].

The rest of this paper is organized as follows. In Section 2, we present the components of the course and how they are structured. Then, we provide the modifications made in Fall 2020 from in-person to online due to the pandemic in Section 3. In Section 4, we summarize the assessment results, then conclude the paper in Section 5.

2 Course Components

This section describes the course components, outlines how they are structured, and provides some examples.

2.1 Embedded DSP Platform - Android Tablet

A critical component in this course is the hardware platform. A large variety of devices are available, from conventional DSP development kits to more modern mobile devices. We chose a commercial Android tablet as opposed to a traditional DSP development kit or an open-source platform for the following reasons:

1. *Out-of-box experience*: Students spend zero setup time for its peripherals such as a touchscreen, speaker, microphone, camera, IMU sensors, accelerated graphics, etc. It

minimizes the distractions for the students who are not familiar with configuring the hardware setup and environment.

2. *Developer friendly*: Android Studio is the official Integrated Development Environment (IDE) for Android app development. It offers many developer-friendly features such as a flexible Gradle-based build system, a fast and feature-rich emulator, a unified environment for all Android devices, C++ support, etc. It is available on most operating systems (Windows, macOS, and Linux). All the developing tools work out-of-the-box with no need to build the tools from the source. It minimizes students' time to set up the tools and provide a consistent development environment.

3. *Mobility and wireless connectivity*: Android tablets provide integrated I/Os into a single hardware platform. Many such devices can also connect to the Internet via Wi-Fi. The mobility and wireless connectivity allow students to run more experiments with fewer efforts and easily demonstrate their works by a video conference app.

The tablets are managed by the course staff. A group of the students (two or three) shares one tablet. During the pandemic, we distribute one tablet for each student to minimize unnecessary inconvenience.

From Spring 2017 to Spring 2020, we chose Nvidia SHIELD tablet for its powerful graphic performance and low-price. As the tablet was discontinued, we adopted Lenovo Tab M10 from Fall 2020 for its comparable performance and price. Figure 1 compares their hardware specifications and price.

	Nvidia SHIELD Tablet	**Lenovo TAB M10**
Price	$200 (discontinued)	$200
CPU	2.2 GHz ARM Cortex A15	MediaTek® P22 Tab, Octa Core 4 x 2.3 GHz + 4 x 1.6GHz
GPU	192 core Kepler	Integrated IMG GE8320
RAM	2 GB	4 GB
Display	8-inch 1920x1200	10-inch 1920x1200
Android version	Android 7.0 Nougat	Android 9.0 Pie
Sensors	IMU, MIC, Speaker, Camera	

Figure 1: Android tablets used in the course

2.2 Structured Labs

The first seven weeks of the course are structured labs based on the fundamental DSP concepts learned from ECE 310. By completing the weekly labs, students should be familiar with the Android tablet as a real-time DSP hardware and be able to implement, debug, and test the real-time DSP algorithm. The attainment of these learning outcomes is assessed through quizzes, prelab assignments, and lab demonstrations. Table 1 shows the topics and desired learning outcomes of the structured labs covered in Spring and Fall 2020.

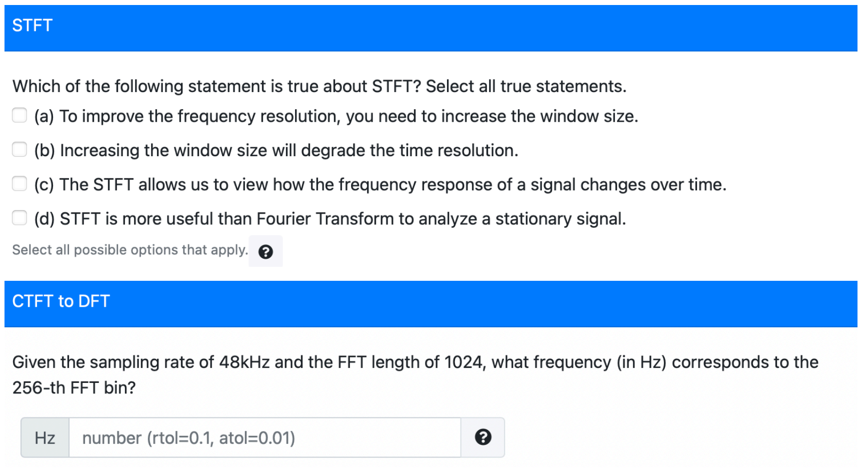
Lab number	**Topic**	**Desired Learning Outcomes**
Lab1	IMU pedometer, peak detection	Be familiar with Python and Android Studio development environment.
Lab2	Real-time audio filtering	Learn how to design FIR band-stopped filter in Python and Android. Understand real-time audio processing mechanism in Android.
Lab3	Spectrogram	Understand the effects of windowing, zero-padding, and batch-processing in the frequency domain by the Short-Time Fourier Transform (STFT).
Lab4	Pitch detection, autocorrelation	Learn how to detect the pitch of a speech signal in real time via autocorrelation.
Lab5	Pitch synthesis, TD-PSOLA	Understand a simplified model for human speech and how to use this model to shift the frequency of an incoming speech signal. Learn how to produce a continuous speech signal within a batch processing framework.
Lab6	Image processing, 2D convolution, Histogram equalizer	Learn how to process an image using histogram equalization and 2-D convolution.
Lab7	Video processing, Kalman filter, Tracking by KCF	Learn how to process video with the OpenCV library and track objects using KCF (Kernalized Correlation Filters) tracker.

Table 1: Topics and desired learning outcomes of structured labs.

1. Lab quizzes: At the beginning of each lab, a 15 minutes in-lab quiz asks the students the basic concepts learned in the previous lab. The quiz is a mix of True/False, multiple-choice, and short-answer questions. From Fall 2020, the in-lab quizzes were replaced by computer-based-testing (CBT) format due to COVID-19.

2. Prelabs: Prelab aims to reinforce the materials and knowledge covered in the lectures and the prerequisite courses. Students develop and test DSP algorithms in a high-level language such as Python before working on the corresponding lab.

3. Labs: The labs' main task is to port the algorithm tested in the prelab into Android tablets using C++/Java. We design the labs to showcase the practical challenges of implementing a real-time algorithm on embedded platforms and further introduce how to solve the challenges. For example, students work on a scenario that discontinuity and distortion arise



STFT

Which of the following statement is true about STFT? Select all true statements.

- ☐ (a) To improve the frequency resolution, you need to increase the window size.
- ☐ (b) Increasing the window size will degrade the time resolution.
- ☐ (c) The STFT allows us to view how the frequency response of a signal changes over time.
- ☐ (d) STFT is more useful than Fourier Transform to analyze a stationary signal.

Select all possible options that apply.

CTFT to DFT

Given the sampling rate of 48kHz and the FFT length of 1024, what frequency (in Hz) corresponds to the 256-th FFT bin?

Hz | number (rtol=0.1, atol=0.01)

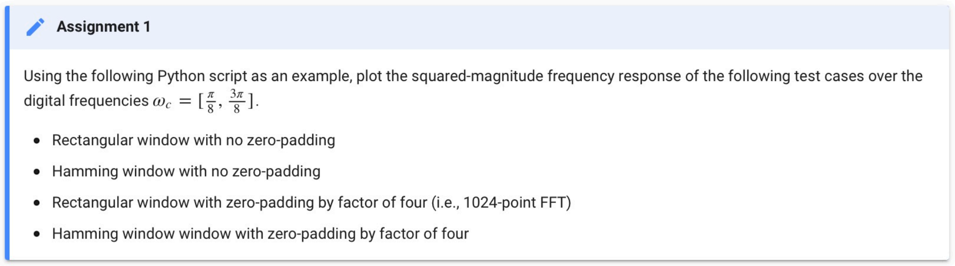
Figure 2: Examples of quizzes.

while synthesizing a speech signal. The discontinuity is caused by segmenting and processing the data in real-time, which is not a common issue in simulation. We introduce them to a block processing algorithm (PSOLA in Lab5) that addresses the discontinuity issues.

2.2.1 Example: Spectrogram lab

In this lab, students learn the effects of windowing and zero-padding on the Discrete Fourier Transform (DFF) of a signal and the effects of data-set quantities and weighting windows used in power spectral density estimation. We also introduce the Short-Time Fourier Transform (STFT) as a tool to visualize the time evolution of the frequency content of non-stationary signals, such as speech.

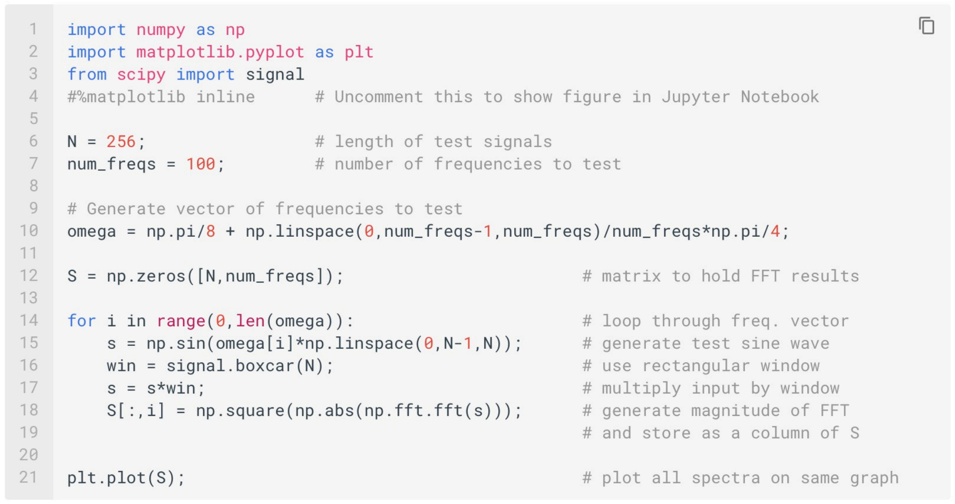
1. Lab Quiz: A set of multiple choice and short answer questions tests students' understanding of the topics. Two examples are shown in Figure 2.
2. Prelab: The prelab is composed of three parts about zero-padding and windowing in DFT, frequency resolution, and STFT. For example, students are asked to explore the effect of multiplying the time signal by different window functions and using zero-padding to increase the length (and thus the number of sample points) of the DFT. Figure 3 shows the Python code and assignment provided to students.
3. Lab: In the previous lab (Lab 2), OpenSL, a royalty-free audio API tuned for embedded system, is introduced to students. This lab aims to use OpenSL and implement a real-time STFT in Android. First, students receive a buffer from the microphone and convert the samples from byte-packed PCM-16 to an integer-type variable. Second, students use the *Kiss FFT* library for computing the FFT in Android. Importing the library has already been done in the provided code, and students need to figure out how to use it. To provide more insight into the real-time process, we emphasize the processing time when handling audio data. For example, if it takes too long to process a buffer, the next buffer may be missed. Timing is not a problem when dealing with offline data, but it is a critical factor while


Assignment 1

Using the following Python script as an example, plot the squared-magnitude frequency response of the following test cases over the digital frequencies $\omega_c = [\frac{\pi}{8}, \frac{3\pi}{8}]$.

- Rectangular window with no zero-padding
- Hamming window with no zero-padding
- Rectangular window with zero-padding by factor of four (i.e., 1024-point FFT)
- Hamming window window with zero-padding by factor of four

(a)



```
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
#%matplotlib inline      # Uncomment this to show figure in Jupyter Notebook

N = 256;                 # length of test signals
num_freqs = 100;         # number of frequencies to test

# Generate vector of frequencies to test
omega = np.pi/8 + np.linspace(0,num_freqs-1,num_freqs)/num_freqs*np.pi/4;

S = np.zeros([N,num_freqs]);                         # matrix to hold FFT results

for i in range(0,len(omega)):                        # loop through freq. vector
    s = np.sin(omega[i]*np.linspace(0,N-1,N));       # generate test sine wave
    win = signal.boxcar(N);                          # use rectangular window
    s = s*win;                                       # multiply input by window
    S[:,i] = np.square(np.abs(np.fft.fft(s)));       # generate magnitude of FFT
                                                     # and store as a column of S

plt.plot(S);                                         # plot all spectra on same graph
```

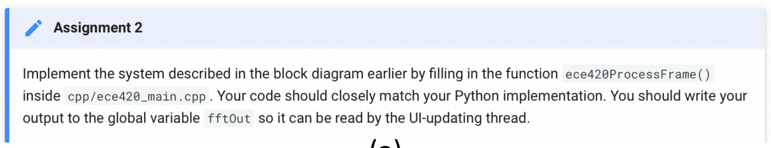
(b)

Figure 3: Example of prelab: (a) assignment and (b) Python code provided to students.

developing the algorithm on a real-time embedded system. Figure 4 (a) and (b) shows the C++ code and assignment provided to students, and Figure 4 (c) shows the final implementation of the real-time STFT.

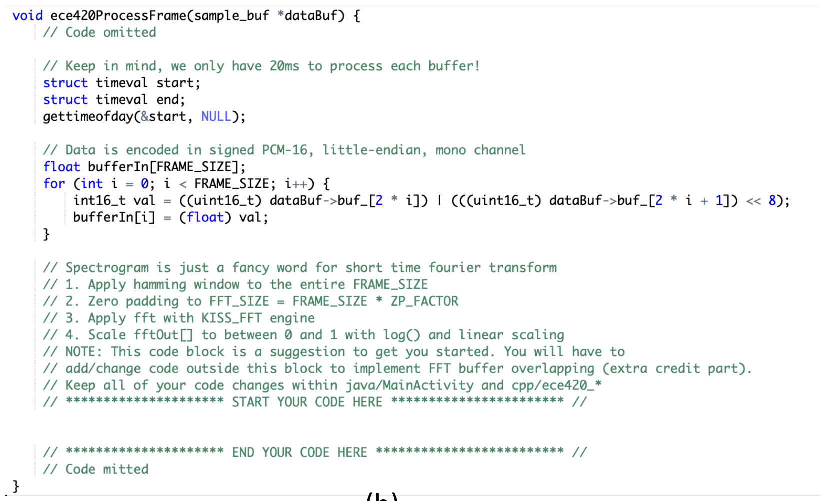
The grading is based on the completion of the laboratory assignment and oral examination during the demonstration. TA's ask questions on the underlying theory, details of the implementation and code, and the system's observed behavior. For example, in Lab3, the following questions are asked by TA's.

- Why would you use a STFT over a single-snapshot FFT?
- What allows us to ignore the second half of our FFT output? Recall the Conjugate Symmetry property of the Fourier Transform.
- Given a buffer size N and a sampling rate F_s, how much time do you have to complete your processing before the next buffer comes in?

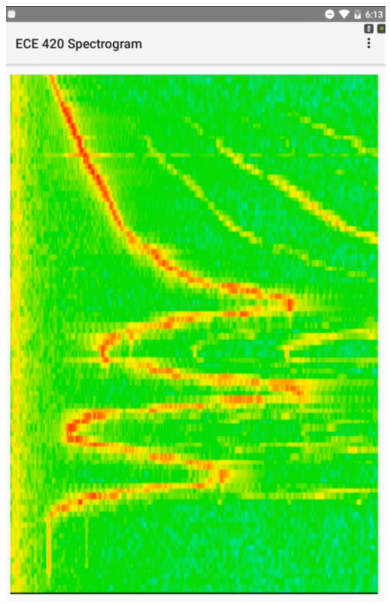

Assignment 2

Implement the system described in the block diagram earlier by filling in the function `ece420ProcessFrame()` inside `cpp/ece420_main.cpp`. Your code should closely match your Python implementation. You should write your output to the global variable `fftOut` so it can be read by the UI-updating thread.

(a)


```
void ece420ProcessFrame(sample_buf *dataBuf) {
    // Code omitted

    // Keep in mind, we only have 20ms to process each buffer!
    struct timeval start;
    struct timeval end;
    gettimeofday(&start, NULL);

    // Data is encoded in signed PCM-16, little-endian, mono channel
    float bufferIn[FRAME_SIZE];
    for (int i = 0; i < FRAME_SIZE; i++) {
        int16_t val = ((uint16_t) dataBuf->buf_[2 * i]) | (((uint16_t) dataBuf->buf_[2 * i + 1]) << 8);
        bufferIn[i] = (float) val;
    }

    // Spectrogram is just a fancy word for short time fourier transform
    // 1. Apply hamming window to the entire FRAME_SIZE
    // 2. Zero padding to FFT_SIZE = FRAME_SIZE * ZP_FACTOR
    // 3. Apply fft with KISS_FFT engine
    // 4. Scale fftOut[] to between 0 and 1 with log() and linear scaling
    // NOTE: This code block is a suggestion to get you started. You will have to
    // add/change code outside this block to implement FFT buffer overlapping (extra credit part).
    // Keep all of your code changes within java/MainActivity and cpp/ece420_*
    // ******************** START YOUR CODE HERE ********************** //


    // ******************** END YOUR CODE HERE ************************ //
    // Code mitted
}
```

(b)



(c)

Figure 4: Example lab: (a) assignment and (b) C++ code provided to students, and (c) sample result of the implementation.

2.3 Independent Group Projects

After the 7 structured labs, students explore more advanced DSP topics by choosing a research paper. Groups of two are typically recommended, but three or one is allowed on rare occasions. We provide a list of recommended papers on the course website. Students can consult with the instructor and TA's in picking a paper. Table 2 shows the count of topics chosen by the groups in Spring 2020 and Fall 2020. Note that two or more topics can be counted for some groups based on their final implementation.

The project follows the basic procedure similar to the structures labs: 1) students simulate the algorithm using a high-level language (Python) for 2 weeks (called “Assigned Project lab”), then 2) build the final implementation on the tablet (called “Final project”).

Category	Topic	SP 2020	FA 2020	Total	By Category
Audio processing	Audio classification [5]	3	0	3	10
	Speech synthesis [6]	2	1	3	
	Speech detection [7]	1	2	2	
	Music synthesis [8]	1	1	2	
Image/Video processing	Shape detection [9]	2	2	4	15
	Face recognition [10]	2	3	5	
	Text detection [11]	2	1	3	
	Object tracking [12, 13]	3	0	3	

Table 2: Count of groups by project topics.

2.3.1 Assigned Project Lab

Before starting the assigned project, students submit a proposal outlining the work to perform. In the proposals, students should 1) review the chosen papers and summarize the algorithms to be implemented, 2) develop a testing and validation plan for Python simulation, and 3) propose rough ideas for final project applications of the algorithm. By completing the 2-week-long assigned project lab, students better understand the algorithm and collect a preliminary result against embedded implementation.

2.3.2 Final Project

Similar to the assigned project lab, students start the final project by submitting a final project proposal. The proposal includes 1) a description of the intended final "product" (e.g., DSP-based program for setting calendar by text recognition), 2) a literature review on the specific algorithm or approach for the implementation, 3) the results from the assigned project lab, 4) two milestones to be achieved during the course of project development, 5) testing and validation plan, and 6) the minimum viable product (MVP) and stretch goals.

At the beginning of the final project stage (after 2-week assigned project lab), students demonstrate their assigned project work together with an oral presentation on the final project proposal. The oral presentation includes the project goals, the proposed project design, the background research findings or other arguments leading to selecting this approach as the preferred solution, and a plan for completion, including milestones, timetables, and individual responsibilities or work packages. A test and verification plan must also be included. Results and observations from the Assigned Lab are included as part of this presentation.

The final project proposal and oral presentation summarize the background research and the preliminary decisions as to the exact scope of the project, the algorithms and approach to be used, and the expected final product. The resulting revised project plan serves as a "contract" with measurable milestones to specify the technical accomplishments required for full credit. The project's final evaluation is based on completing the proposed work as agreed in the revised project plan after the oral presentation.

During the project, students demonstrate 2 milestones toward the completion of their projects.

The revised project plan must clearly state the deliverables at each of these project milestones as "contract" between the student and the teaching staff. These project milestones are evaluated based on this contract.

In the final project demonstration, the group presentation provides each group with the opportunity to show the final project deliverable and the implemented algorithm. The groups demonstrate the system or all functional components if incomplete, perform verification experiments, and show test results to the course staff.

After the final project demonstration, each group submits a final report summarizing literature reviews, technical descriptions of the implemented algorithms, final results including the testing done on the project and problems encountered, suggestions for extensions and modifications, and software/hardware documentation.

2.4 Lectures

There are one-hour lectures every week. During the structured-labs period, the lecture discusses the DSP algorithm and theoretical background of the upcoming lab. For the remaining weeks, the lecture covers a wide variety of embedded DSP topics that were not necessarily discussed in the structure labs. Special guests, including faculties, graduate students, or industry engineers, are often invited to give a seminar.

3 Modifications for Online Delivery

In Fall 2020, the course was switched from in-person to fully online due to the pandemic. Thanks to the Android tablet's mobility and wireless connectivity, we only need a few minor modifications.

- Lab and project demonstration: Using Zoom video conferencing application, students can share the tablet screen and demonstrate their work in real-time. However, as the Android version Zoom does not support in-app sound sharing, Lab 2 and 5, which grade the output sound, needed to be modified. We suggested the students use an aux cable to re-route the sound to a laptop or a desktop (the Window and Mac version Zoom supports sound sharing). For the students who did not have the second device, we provided modified labs that combine Lab 3 to display the spectrogram of the output sound.
- The in-lab quizzes were converted to computer-based testing (CBT).
- The live lectures were delivered via Zoom, and the recorded lectures were provided in the course website.

4 Results

Each student who took the course in Spring 2020 and Fall 2020 was asked to respond to five questions. Each answer was mapped to a numerical value. The survey was anonymous. The five survey questions are listed below.

- Q1: Rate the overall quality of this course. [1-Exceptionally Low, ..., 5-Exceptionally High]
- Q2: How much have you learned in this course? [1-Very Little, ..., 5-A Great Deal]
- Q3: Statement of objectives and purposes throughout course. [1-Never Clear, ..., 5-Consistently Clear]
- Q4: Quizzes - fairness. [1-Unfair Content, ..., 5-Very Fair Content]
- Q5: Quizzes - grading. [1-Unfairly Graded, ..., 5-Fairly Graded]

There were 8 responses out of 31 students in Spring 2020 and 9 out of 21 students in Fall 2020. The survey results are summarized in Table 3 with each question's mean and standard deviation (STD).

Questions	**Spring 2020**		**Fall 2020**	
	Mean	STD	Mean	STD
Q1	4.25	1.16	4.00	1.22
Q2	3.88	1.13	4.22	0.97
Q3	4.00	1.07	4.11	1.27
Q4	4.00	0.76	4.00	1.12
Q5	3.88	1.13	4.00	1.12

Table 3: Survey results from Spring and Fall 2020 courses.

Some students provided encouraging written remarks, such as

- *"The course gives us an opportunity to learn how to design an app, which involves multiple programming languages."*
- *"Very clearly laid out requirements for what was required."*
- *"I loved how hands on the course was. Being able to directly apply what was learned in lectures to python and C++/Java was one of my favorite things about the course. That leads me to the labs which I thought were another strength of the course. I liked the diversity of content and learned a lot working on them. I also liked the flexibility given with the assigned and final labs. It was fun being able to have the agency to pick what we worked on and figure out what steps we needed to take to make something functional."*
- *"Interesting topics covered, useful skills like Android taught."*

Some also offered constructive suggestions for improvement, such as

- *"I think additions to the lectures can be made to add more practice and to address real life practices. Most of the learning is done in the lab and that is most likely by design, but if there is a lecture then there might as well be a deeper dive into the material."*

Table 4 shows the number of groups who fully implemented or partially implemented or failed to implement the final project. A fully implemented project means that a group completed all the proposed features and demonstrated and presented them in time. A partially implemented project is that a group completed a part of the proposed features or did a late demo.

	Spring 2020	**Fall 2020**
Fully implemented	9	6
Partially implemented	3	3
Failed to implement	0	1

Table 4: Number of groups completed Final Project.

Based on the survey results in Table 3, the written responses, and the final project completion rate, we have made a few key observations:

- Most students felt that the course was a valuable experience. For Q1 in the survey, 82.4% of students rated the course's quality as high or exceptionally high (4 or 5 out of 5). For Q2, 76.5% of students felt they learned a good or great amount in the course (4 or 5 out of 5).
- The majority of groups were able to complete the final project. 21 out of 22 groups demonstrated their works in the final demo, although 6 of them were a partial implementation.
- The conversion for online delivery was successful. With some modifications described in Section 3, the online demos for lab sections and final project were smooth. Comparing Q4 and Q5 between Spring and Fall 2020, it suggests that the conversion to CBT quizzes was also successful.

5 Conclusion

In this paper, we present a new senior-level embedded DSP laboratory course driven by Android tablets. The Android tablet proved to be a powerful and flexible platform allowing students to explore more advanced topics in the embedded DSP field. The qualitative survey data and students' feedback indicates that the combination of the structured labs and group project succeeded in engaging students' interest. In the future, we plan to run more detailed surveys to collect students' feedback on the lab components and whether the course has impacted their career in the embedded DSP area.

References

[1] Arizona State University. Eee 404/591 real-time digital signal processing. http://lina.faculty.asu.edu/realdsp/.

[2] The University of Texas at Austin. Ee 445s real-time digital signal processing laboratory. http://users.ece.utexas.edu/ bevans/courses/realtime/.

[3] University of London. Real-time digital signal processing with tms320c6000. http://www.ee.ic.ac.uk/pcheung/teaching/ee3_Study_Project/index.html.

[4] David Jun, Douglas L Jones, and Minh N Do. From fixed-point processors to android: A hybrid course for real-time dsp laboratory. In *2013 IEEE Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, pages 279–283. IEEE, 2013.

[5] Avery Wang et al. An industrial strength audio search algorithm. In *Ismir*, volume 2003, pages 7–13. Citeseer, 2003.

[6] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467, 1990.

[7] Ronald W Schafer and Lawrence R Rabiner. System for automatic formant analysis of voiced speech. *The Journal of the Acoustical Society of America*, 47(2B):634–648, 1970.

[8] John M Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the audio engineering society*, 21(7):526–534, 1973.

[9] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[10] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[11] Hojin Cho, Myungchul Sung, and Bongjin Jun. Canny text detector: Fast and robust scene text localization algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2016.

[12] Md Zahidul Islam, Chi-min Oh, and Chil-Woo Lee. Real time moving object tracking by particle filter. In *International Symposium on Computer Science and its Applications*, pages 347–352. IEEE, 2008.

[13] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.