รายงาน Notes Detection from musical instrument by using MATLAB

จัดทำโดย

นายเสฏฐภูมิ ตุลยสุข 6301012630192 Cpr.E

นายสุรชัย สันติภาพ 6401012620145 Cpr.E

นายมัครพงษ์ สมบูรณ์ 6401012620102 Cpr.E

เสนอ

อาจารย์ ดร. เรวัต ศิริโภคาภิรมย์

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา

Introduction to signals and system รหัสวิชา 010123106 ตอนเรียนที่ 1

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ.2565

คำนำ

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา Introduction to signals and system (010123106) ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ของภาคเรียนที่ 1 ปีการศึกษา 2565 มีจุดประสงค์เพื่อศึกษาการใช้งานของโปรแกรม Matlab และมีการนำมาประยุกต์ใช้ ที่สามารถทำฟังก์ชั่นจากความรู้ที่ได้ศึกษาและเรียนมาจากวิชานี้ รวมถึงได้เข้าใจถึงความสามารถและข้อจำกัดของ Matlab และ ทฤษฎีต่างๆได้ โดยรายงานฉบับนี้มีเนื้อหาเกี่ยวกับการใช้งาน Matlab และการใช้ Fast Fourier Transform มาประยุกต์ใช้ เพื่อที่ผู้อ่านจะได้ความรู้มาปรับใช้ในอนาคต ข้าพเจ้า หวังว่ารายงานเล่มนี้จะเป็นประโยชน์ต่อผู้อ่าน ถ้าหากรายงานเล่มนี้ผิดพลาดประการใดต้องขออภัยและน้อมรับคำติชมเพื่อปรับปรุง

ขอขอบคุณ ดร.เรวัต ศิริโภคาภิรมย์ ที่ให้คำแนะนำในการทำรายงานให้สำเร็จไปด้วยดี

นายเสฏฐภูมิ ตุลยสุข

นาย สุรชัย สันติภาพ

นาย มัครพงษ์ สมบูรณ์

ผู้จัดทำ

สารบัญ

หลักการและทฤษฎี

-Fourier Transform

-Fast Fourier Transform (FFT)

-Digital Signal Processing

-การประยุกต์ใช้งาน Fast Fourier Transform สำหรับใช้การวิเคราะห์ความถี่คลื่นเสียง

-การไล่ระดับความถี่ของโน้ตดนตรี

Fourier Transform

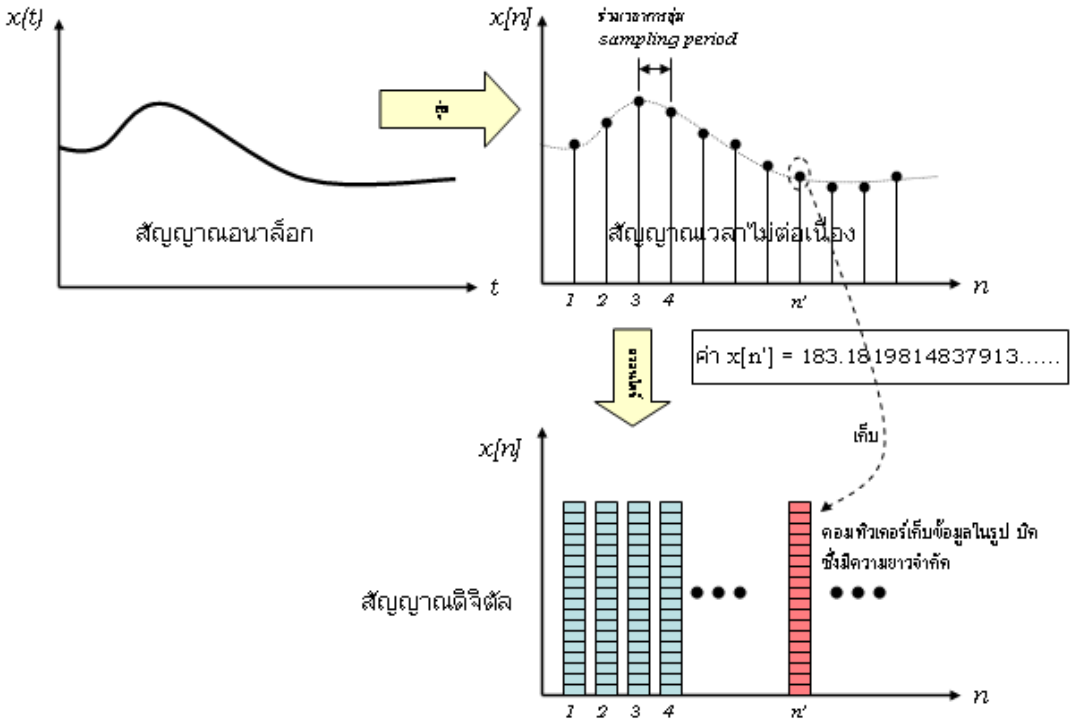
คือ การแปลงเชิงปริพันธ์ โดยเป็นการเขียนแทนฟังก์ชันใดๆ ในรูปผลบวก หรือปริพันธ์ของฐาน ที่เป็นฟังก์ชันรูปคลื่นไซน์ หรือโคไซน์ โดยปกติแล้ว Fourier Transform จะใช้หมายถึง Continuous Fourier Transform

Fast Fourier Transform (FFT)

เป็นวิธีการทางคณิตศาสตร์ที่ใช้ในการแปลง Discrete Fourier Time โดยจะแปลงสัญญาณจากโดเมนดั้งเดิม เป็นการแสดงโดเมนความถี่ ในทางกับกลับ Discrete Fourier Time มีการใช้ระยะเวลาในการคำนวณที่ช้าเกินไปที่จะใช้งานได้จริง แต่ Fast Fourier Transform สามารถคำนวณได้อย่างรวดเร็ว จนทำให้ลดความซับซ้อนของการคำนวณ Discrete Fourier Transform ลงจาก $\mathrm{O(N^2)}$ เป็น $\mathrm{O(Nlog2N)}$ หรือลดได้อีกถึง $\mathrm{O}(\frac{N}{2}\mathrm{log2N})$ ซึ่งในกรณีที่ข้อมูลมีจำนวนมากความแตกต่างของความเร็วอาจมีค่ามหาศาล โดยเฉพาะอย่างยิ่งสำหรับชุดข้อมูลที่อยู่ในหลักพันหรือล้าน Fast Fourier Transform นั้นมีความแม่นยำมากกว่า Discrete Fourier Time

Digital Signal Processing

คือ การประมวลผลสัญญาณดิจิทัล แต่แหล่งกำเนิดมาจากสัญญาณอนาล็อกซึ่งจะต้องผ่านกระบวนการแปลงสัญญาณอนาล็อกเป็นดิจิทัล (Analog-to-Digital Conversion - ADC) หรือการดิจิไทซ์ (digitization) ซึ่งประกอบด้วยการสุ่มตัวอย่าง (sampling) และการควอนไทซ์ (quantization) ให้อยู่ในรูปดิจิทัลก่อนที่จะทำการประมวลผลต่อไป



การประยุกต์ใช้งาน Fast Fourier Transform สำหรับใช้การวิเคราะห์ความถี่คลื่นเสียง

เริ่มจากการรับอินพุตของสัญญาณ ในช่วงระยะเวลานึง ตัวอย่างเช่นรับในช่วงระยะเวลา 50 millisecond เป็นต้น โดยภายในระยะเวลาดังกล่าวจะมีการเก็บข้อมูลสัญญาณดิจิทัลที่เข้ามา ซึ่งเราจะนำข้อมูลนั้นไปเข้าฟังก์ชัน Fast Fourier Transform เพื่อแปลง Fourier โดยผลลัพธ์ที่ได้จะเป็นโดเมนความถี่ของอินพุตนั้นๆ หลังจากได้โดเมนความถี่มาแล้ว จะทำการหาความถี่สูงสุด เพื่อนำไปเปรียบเทียบเพื่อหาตัวโน้ต

การไล่ระดับความถี่ของโน้ตดนตรี

โดยทั่วไปแล้วการเรียกระดับไล่เสียงของตัวโน้ตตามที่นิยมกันจะเรียก โด, เร, มี, ฟา, ซอล, ลา, ที หรือแทนด้วยตัวอักษรภาษาอังกฤษ C, D, E, F, G, A, B โดยที่โน้ตตัว “โด” (C) เป็นระดับเสียงต่ำไล่ไปจนถึงโน้ตตัว “ที” (B) ซึ่งเป็นระดับเสียงสูง ความถี่ของตัวโน้ตแต่ละตัวเป็นดังนี้

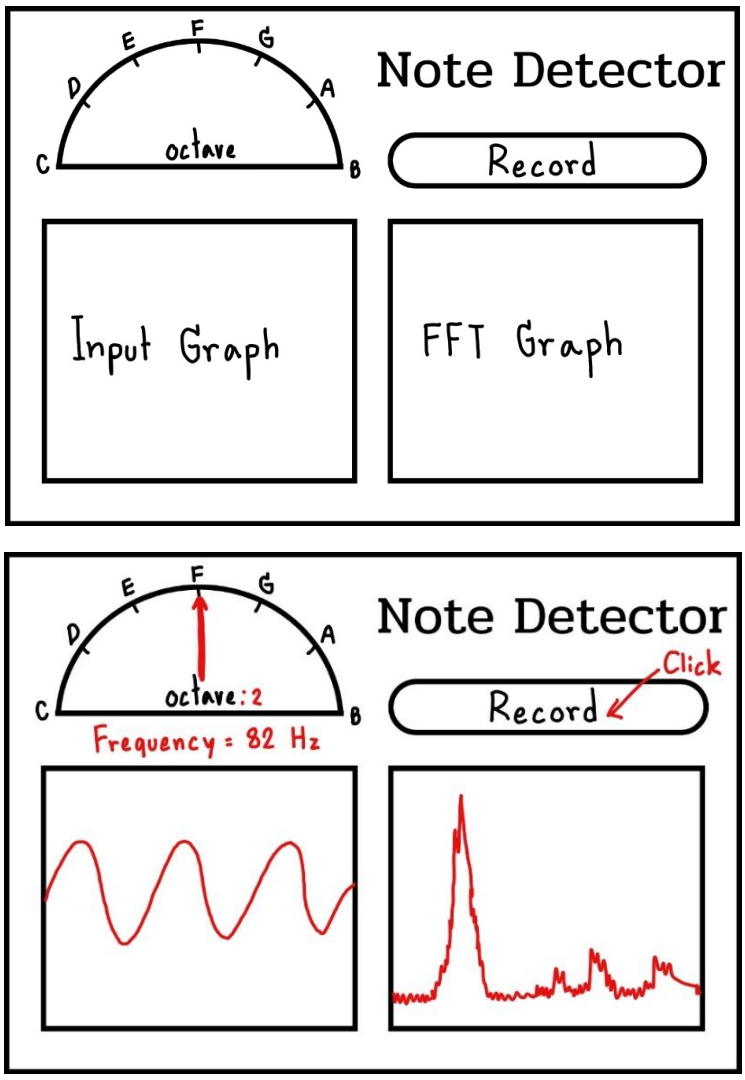
- โน้ตตัว “โด” มีความถี่ 262 Hz
- โน้ตตัว “เร” มีความถี่ 294 Hz
- โน้ตตัว “มี” มีความถี่ 330 Hz
- โน้ตตัว “ฟา” มีความถี่ 349 Hz
- โน้ตตัว “ซอล” มีความถี่ 370 Hz
- โน้ตตัว “ลา” มีความถี่ 440 Hz
- โน้ตตัว “ที” มีความถี่ 495 Hz

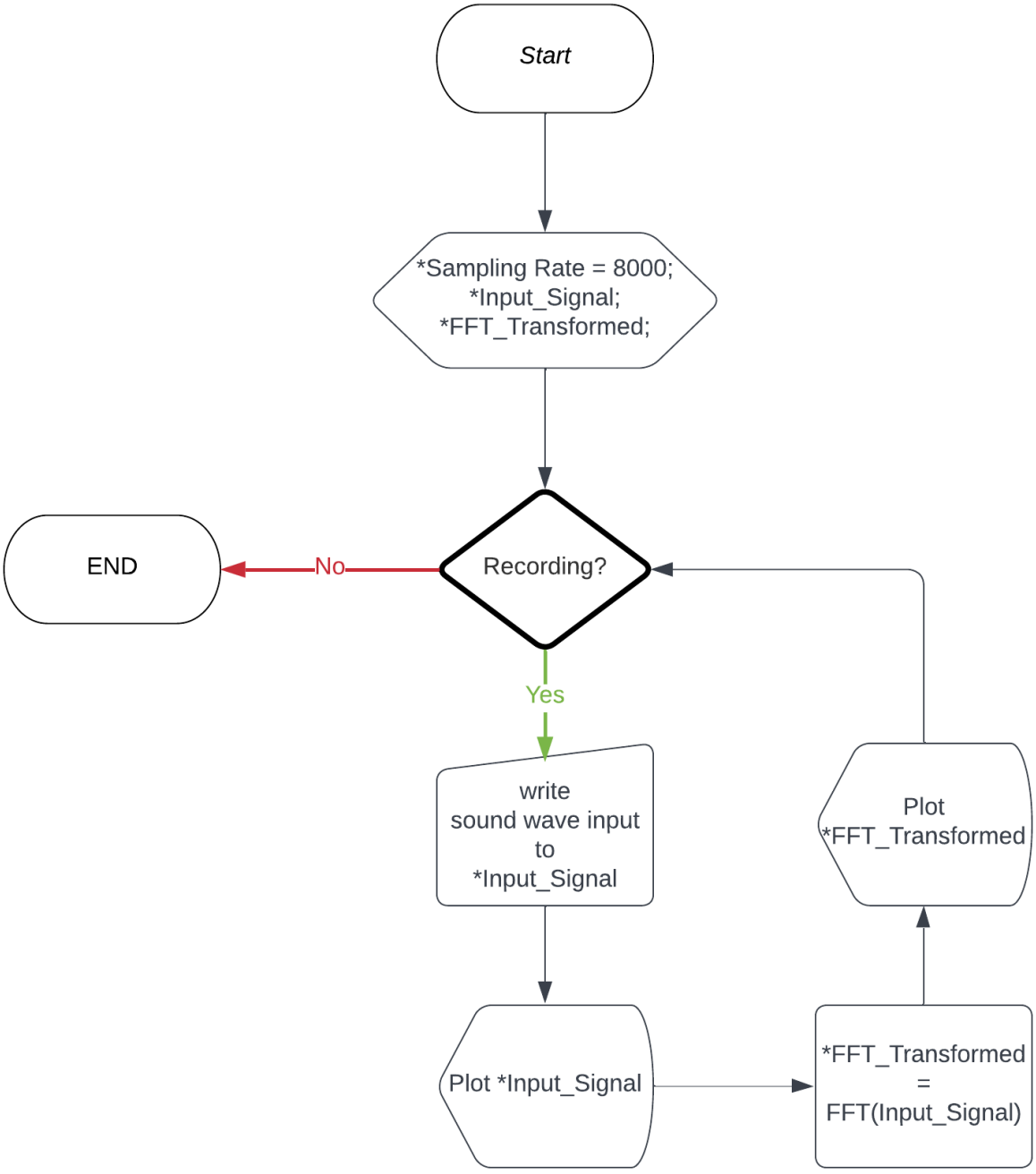
โดยในการวิเคราะห์ความถี่คลื่นเสียงเราได้ทำการใช้ตัวโน้ต และความถี่ดังตารางนี้

	OCTAVE 0	OCTAVE 1	OCTAVE 2	OCTAVE 3	OCTAVE 4	OCTAVE 5	OCTAVE 6	OCTAVE 7	OCTAVE 8
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz	523.25 Hz	1046.50 Hz	2093.00 Hz	4186.01 Hz
C#/Db	17.32 Hz	34.65 Hz	69.30 Hz	138.59 Hz	277.18 Hz	554.37 Hz	1108.73 Hz	2217.46 Hz	4434.92 Hz
D	18.35 Hz	36.71 Hz	73.42 Hz	146.83 Hz	293.66 Hz	587.33 Hz	1174.66 Hz	2349.32 Hz	4698.63 Hz
D#/Eb	19.45 Hz	38.89 Hz	77.78 Hz	155.56 Hz	311.13 Hz	622.25 Hz	1244.51 Hz	2489.02 Hz	4978.03 Hz
E	20.60 Hz	41.20 Hz	82.41 Hz	164.81 Hz	329.63 Hz	659.25 Hz	1318.51 Hz	2637.02 Hz	5274.04 Hz
F	21.83 Hz	43.65 Hz	87.31 Hz	174.61 Hz	349.23 Hz	698.46 Hz	1396.91 Hz	2793.83 Hz	5587.65 Hz
F#/Gb	23.12 Hz	46.25 Hz	92.50 Hz	185.00 Hz	369.99 Hz	739.99 Hz	1479.98 Hz	2959.96 Hz	5919.91 Hz
G	24.50 Hz	49.00 Hz	98.00 Hz	196.00 Hz	392.00 Hz	783.99 Hz	1567.98 Hz	3135.96 Hz	6271.93 Hz
G#/Ab	25.96 Hz	51.91 Hz	103.83 Hz	207.65 Hz	415.30 Hz	830.61 Hz	1661.22 Hz	3322.44 Hz	6644.88 Hz
A	27.50 Hz	55.00 Hz	110.00 Hz	220.00 Hz	440.00 Hz	880.00 Hz	1760.00 Hz	3520.00 Hz	7040.00 Hz
A#/Bb	29.14 Hz	58.27 Hz	116.54 Hz	233.08 Hz	466.16 Hz	932.33 Hz	1864.66 Hz	3729.31 Hz	7458.62 Hz
B	30.87 Hz	61.74 Hz	123.47 Hz	246.94 Hz	493.88 Hz	987.77 Hz	1975.53 Hz	3951.07 Hz	7902.13 Hz

รายละเอียดการออกแบบ

Lo-fi Ui

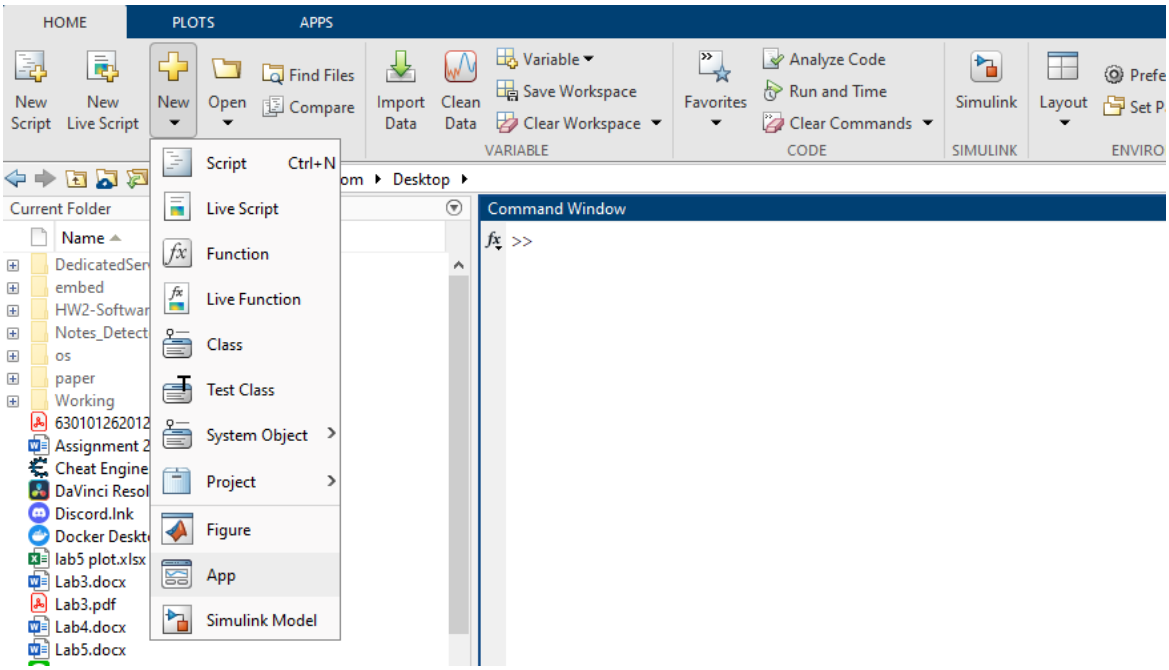



Start
*Sampling Rate = 8000;
*Input_Signal;
*FFT_Transformed;
Recording?
No
END
Yes
write
sound wave input
to
*Input_Signal
Plot *Input_Signal
*FFT_Transformed
=
FFT(Input_Signal)
Plot
*FFT_Transformed

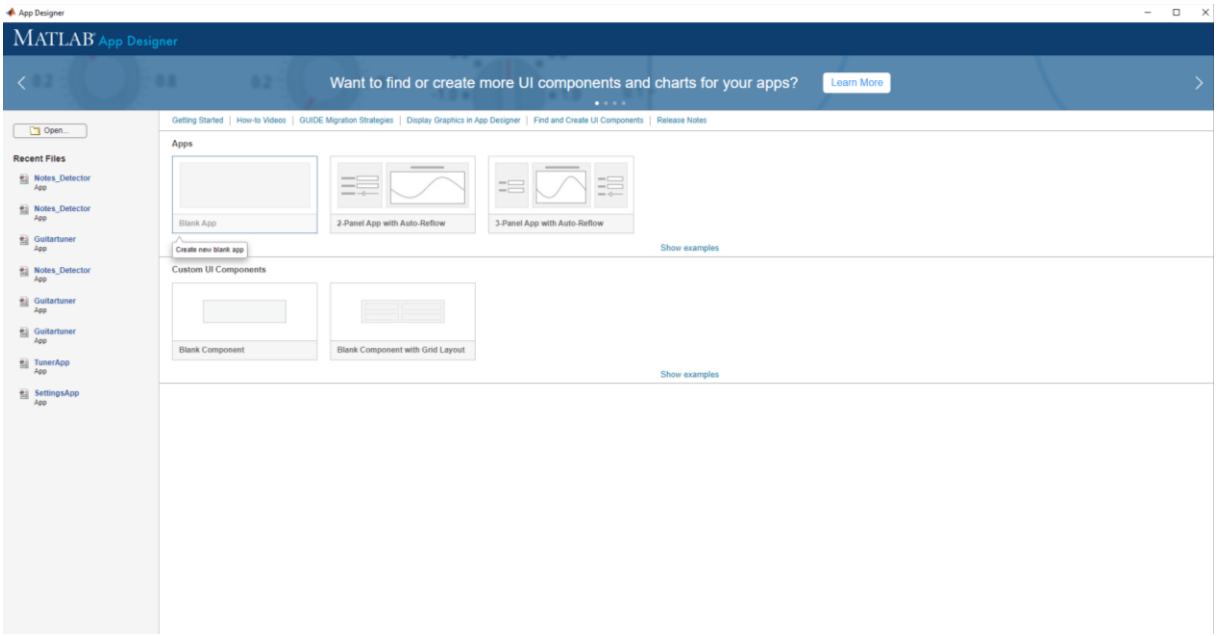
การดำเนินการ

การเริ่มใช้งาน MATLAB App Designer

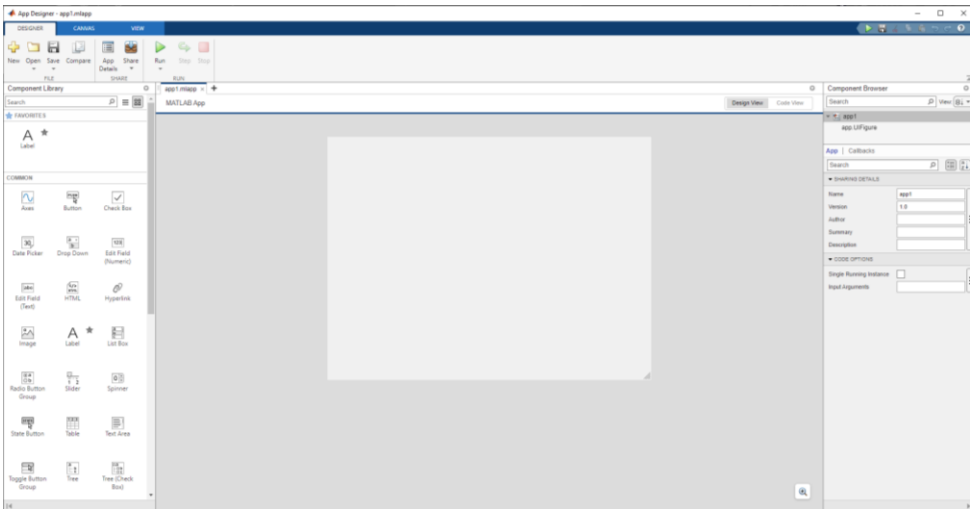
1. สร้างไฟล์งานสำหรับพัฒนาแอพลิเคชันด้วย MATLAB โดยการคลิกไปที่ New > App



2. เข้าสู่หน้าเลือก Template ซึ่งเราจะไม่ใช้ Template จึงกดไปที่ Blank App

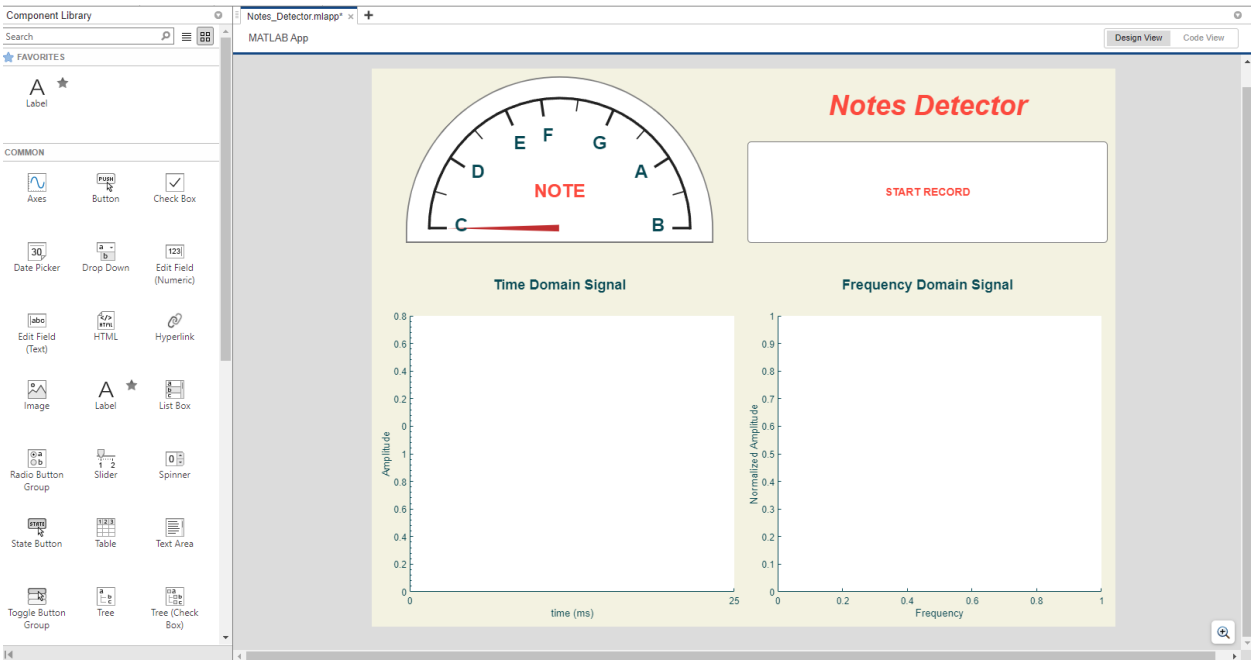


3. จะได้หน้า App Designer มาเริ่มทำการสร้าง App ได้เลย

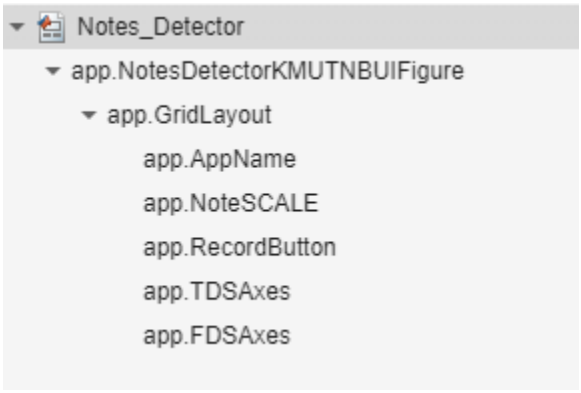


เริ่มต้นสร้าง Application

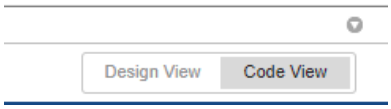
4. ทำการวาง Component สำเร็จรูปตามตำแหน่งที่ออกแบบไว้ใน Lo-Fi UI



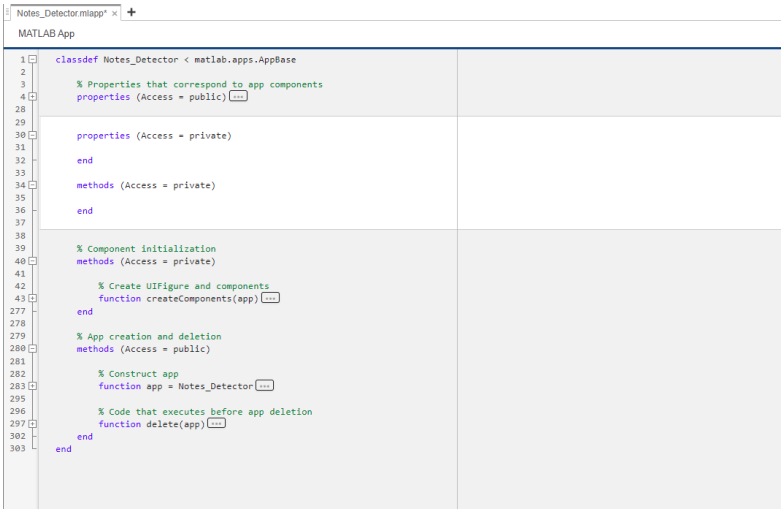
- โดยมีรายละเอียด Component ต่างๆดังนี้
    - AppName ชื่อแอพ Notes Detector
    - NoteSCALE หน้าปัดแสดงตัว Note
    - RecordButton ปุ่มกดอัดเสียง
    - TDSAxes Time Domain Signal Axes
    - FDSAxes Frequency Domain Signal



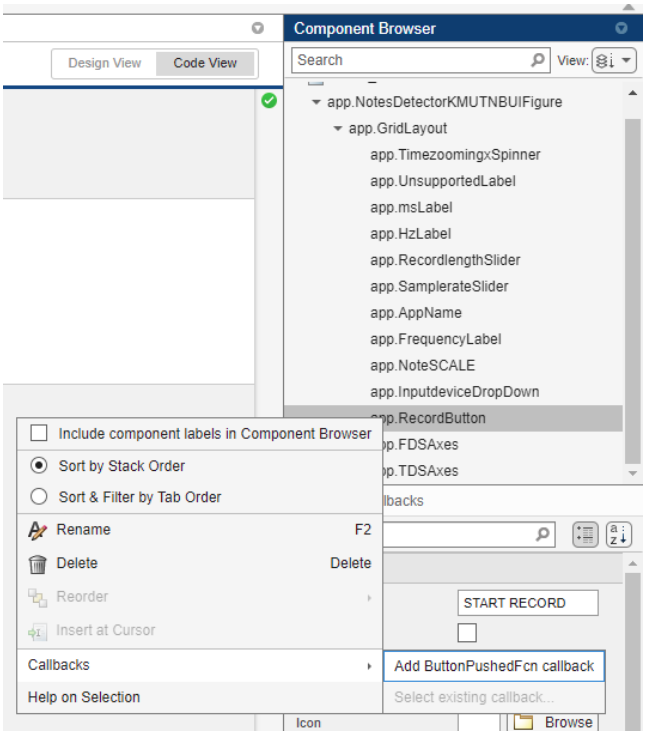
5. จากนั้นไปเขียน Code กำกับการทำงานโดยการกดไปที่ Code View



จะเจอหน้า Text editor ที่มีส่วน Text ที่มีพื้นหลังสีขาวและสีเทา ส่วนของสีเทาคือส่วนที่เราแก้ไขใน Code View ไม่ได้ จำพวกเช่น การตั้งค่าสี Label, Front, XLim ของ Axes เป็นต้น ซึ่งค่าพวกนี้เป็นค่าตั้งต้นที่จะแสดงเมื่อเปิด App ต้องไปแก้ไขที่แต่ละ Component ที่หน้า Design View ส่วนของสีขาวคือส่วนที่เราสามารถแก้ไขได้โดยเราจะเขียนการทำงาน App ในส่วนนี้



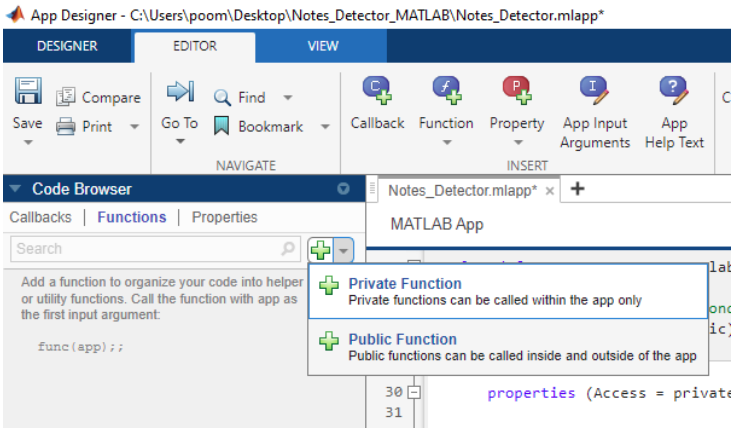
- เบื้องต้น เราสามารถเพิ่มการทำงานการรองรับของ Event หรือ Action ต่างๆเมื่อผู้ใช้กระทำการใดๆกับแต่ละ Component เช่น เมื่อผู้ใช้กดปุ่ม จะเริ่มการอัดเสียง เป็นต้น โดยการกดไปที่แต่ละ Component ที่ต้องการ เลือก Callbacks > Add ... callback



แล้วจะได้ Callback Function มาใช้งานให้เราเขียนกำกับการทำงานได้เลย

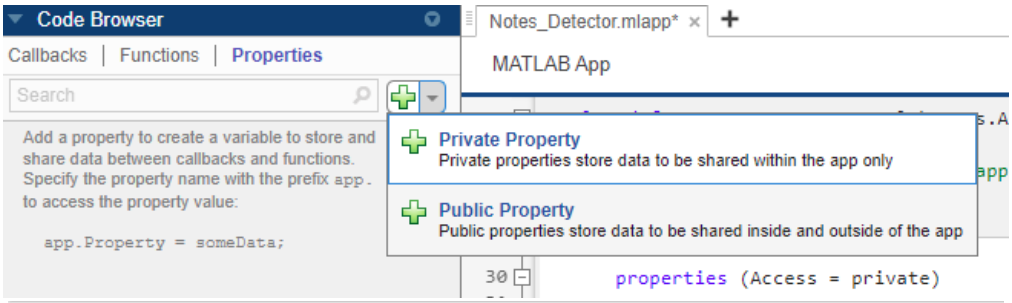
```
% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: RecordButton
    function RecordButtonPushed(app, event)

    end
end
```

- การเพิ่ม Function หรือ Method ทั่วไปไว้เรียกใช้งานคือ ที่ Code Browser กดไปที่ Function > ปุ่ม + สีเขียว > Private/Public Function



```
properties (Access = private)

end

methods (Access = private)


    function results = func(app)

    end
end
```

- การเพิ่ม Properties หรือ Attribute ก็เช่นกัน ที่ Code Browser กดไปที่ Properties > ปุ่ม + สีเขียว > Private/Public Property โดยการเรียกใช้ method หรือ property ที่สร้างมาจะเรียกใช้โดยการใส่ app. นำหน้าตามด้วยชื่อ method หรือ property นั้นๆ




```
properties (Access = private)

    Value1 = 0; % Description
end

methods (Access = private)


    function method1(app)
        app.Value1 = 99;
    end
end
```

6. หลังจากนั้น ทำการเขียน method และ Callback ให้สำเร็จตามที่ออกแบบการทำงานไว้ โดยมีส่วนนอกเหนือจากการสร้างอัตโนมัติของ App Designer คือส่วนที่ที่เขียนเพิ่มมา ดังนี้

```
    properties (Access = private)

        frequencies = [16.35 17.32 18.35 19.45 20.6 21.83 23.12 24.5 25.96
    27.5 29.14 30.87 32.7 34.65 36.71 38.89 41.2 43.65 46.25 49 51.91 55
    58.27 61.74 65.41 69.3 73.42 77.78 82.41 87.31 92.5 98 103.83 110
    116.54 123.47 130.81 138.59 146.83 155.56 164.81 174.61 185 196
    207.65 220 233.08 246.94 261.63 277.18 293.66 311.13 329.63 349.23
    369.99 392 415.3 440 466.16 493.88 523.25 554.37 587.33 622.25 659.25
    698.46 739.99 783.99 830.61 880 932.33 987.77 1046.5 1108.73 1174.66
    1244.51 1318.51 1396.91 1479.98 1567.98 1661.22 1760 1864.66
    1975.53 2093 2217.46 2349.32 2489.02 2637.02 2793.83 2959.96
    3135.96 3322.44 3520 3729.31 3951.07 4186.01 4434.92 4698.63
    4978.03 5274.04 5587.65 5919.91 6271.93 6644.88 7040 7458.62
    7902.13];
        notes = ["C0" "Db0" "D0" "Eb0" "E0" "F0" "Gb0" "G0" "Ab0" "A0" "Bb0"
    "B0" "C1" "Db1" "D1" "Eb1" "E1" "F1" "Gb1" "G1" "Ab1" "A1" "Bb1" "B1" "C2"
    "Db2" "D2" "Eb2" "E2" "F2" "Gb2" "G2" "Ab2" "A2" "Bb2" "B2" "C3" "Db3" "D3"
    "Eb3" "E3" "F3" "Gb3" "G3" "Ab3" "A3" "Bb3" "B3" "C4" "Db4" "D4" "Eb4" "E4"
    "F4" "Gb4" "G4" "Ab4" "A4" "Bb4" "B4" "C5" "Db5" "D5" "Eb5" "E5" "F5" "Gb5"
    "G5" "Ab5" "A5" "Bb5" "B5" "C6" "Db6" "D6" "Eb6" "E6" "F6" "Gb6" "G6" "Ab6"
    "A6" "Bb6" "B6" "C7" "Db7" "D7" "Eb7" "E7" "F7" "Gb7" "G7" "Ab7" "A7" "Bb7"
    "B7" "C8" "Db8" "D8" "Eb8" "E8" "F8" "Gb8" "G8" "Ab8" "A8" "Bb8" "B8"];
        NoteIndex = 0;

        Arec = audiorecorder;
        plotPause = true;

        InputsInfo = audiodevinfo().input;
```

```
        TotolDevice = audiodevinfo(1);

    end

    methods (Access = private)

        function plotAudio(app)

            while ~app.plotPause

                recordblocking(app.Arec, .025);

                audio = getaudiodata(app.Arec);

                Tscale = linspace(0,.025,length(audio));

                plot(app.TDSAxes,Tscale,audio);

                Fs = app.Arec.SampleRate;              % Sampling frequency
                L = length(audio);                     % Length of signal

                Y = fft(audio);

                P2 = abs(Y/L);
                P1 = P2(1:L/2+1);
                P1(2:end-1) = 2*P1(2:end-1);

                f = Fs*(0:(L/2))/L;
                plot(app.FDSAxes,f,P1);

```

```
                app.updateDetectedNote(f,P1);

            end
        end

        function updateDetectedNote(app,freqes,ampts)

            [~,index] = max(ampts);
            maxFreq = freqes(index);

            A = repmat(app.frequencies,[1 length(maxFreq)]);
            [~,closestIndex] = min(abs(A-maxFreq'));
            app.NoteIndex = closestIndex;

            app.NoteLabel.Text = app.notes(app.NoteIndex);
            app.NoteSCALE.Value = mod(app.NoteIndex-1,12);
            app.FrequencyLabel.Text = string(maxFreq)+" Hz";

        end


        function stopPlot(app)

            app.RecordButton.Text="START RECORD";
            app.RecordButton.BackgroundColor="#ffffff";
            app.plotPause = true;

        end
    end

```

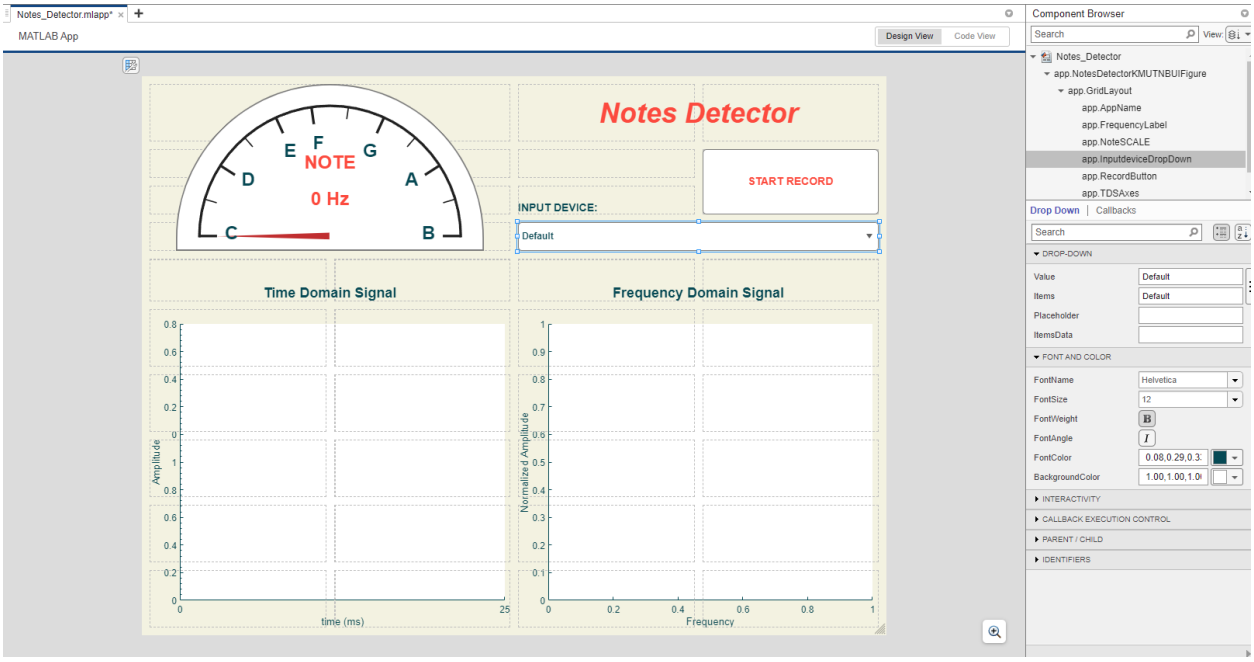
และส่วนของ Callback จาก Component ต่างๆดังนี้

```
methods (Access = private)

        % Button pushed function: RecordButton
        function RecordButtonPushed(app, event)
            if app.RecordButton.Text == "START RECORD"
                app.RecordButton.Text="RECORDING...";
                app.RecordButton.BackgroundColor="#f0f0f0";

                app.plotPause = false;
                app.plotAudio();

            elseif app.RecordButton.Text == "RECORDING..."
                app.stopPlot();
            end
        end

        % Value changed function: InputdeviceDropDown
        function InputdeviceDropDownValueChanged(app, event)
            value = app.InputdeviceDropDown.Value;

            deviceID = -1;

            if value ~= "Default"
                for i = 1:app.TotolDevice

                    if strcmp(app.InputsInfo(i).Name,value)
                        deviceID = app.InputsInfo(i).ID;
                        break;
```

```
            end

        end
    end

    if audiodevinfo(1,deviceID,8000,8,1)
        clear app.Arec;
        app.Arec = audiorecorder(8000,8,1,deviceID);
    else
        beep;
        app.InputdeviceDropDown.Value = "Default";
        app.Arec = audiorecorder;
    end


end

% Drop down opening function: InputdeviceDropDown
function InputdeviceDropDownOpening(app, event)
    clear app.InputsInfo app.TotolDevice
    audiodevreset;
    app.InputsInfo = audiodevinfo().input;
    app.TotolDevice = audiodevinfo(1);

    devicelist = {'Default'};

    for c = 1:app.TotolDevice
        devicelist{1,c+1} = app.InputsInfo(c).Name;
    end
```

```
            app.InputdeviceDropDown.Items = devicelist;


            app.stopPlot();
        end
    end
```

7. จากการทดลองใช้งานใช้ไมโครโฟนที่ต่อไว้ตั้งแต่แรกจะรับเสียงได้ปรกติ แต่ถ้าใช้กีตาร์ต่อเพิ่มเข้าคอมพิวเตอร์ผ่านอแดปเตอร์จะไม่สามารถรับกีต้าร์เนื่องจาก lib จะใช้อุปกรณ์ default จึงมีความคิดที่จะเพิ่ม feature การเลือก Input Device เข้ามา โดยขั้นต้นจะสร้างเป็น Dropdown Box เอาไว้ก่อน

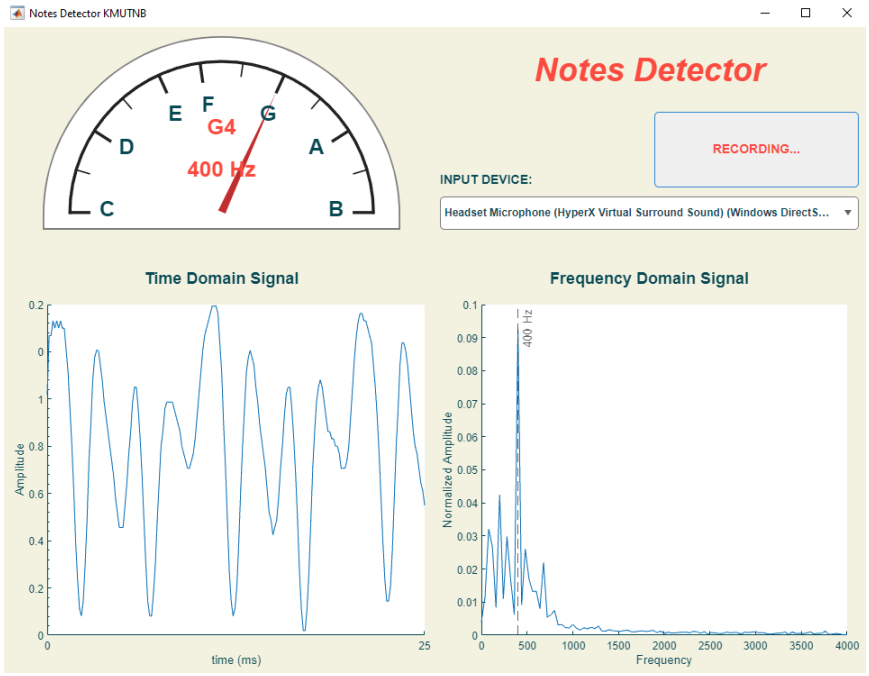


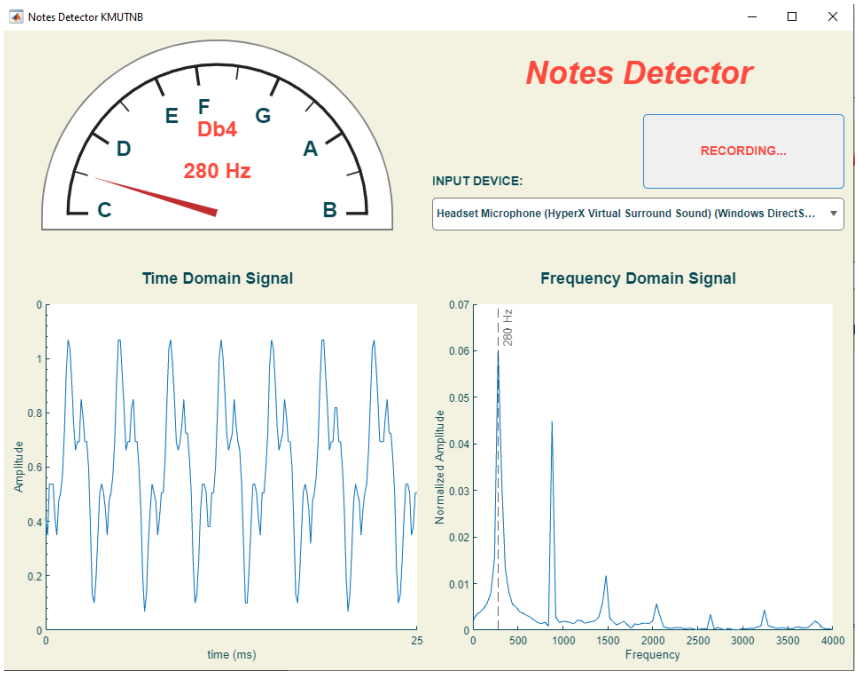
8.จากนั้นทำการเขียน Callback ของ Dropdown ที่สร้างมา ให้ไปเปลี่ยน Device ID ที่เราจะใช้สำหรับวิเคราะห์เสียง

```
function InputdeviceDropDownValueChanged(app, ~)
    value = app.InputdeviceDropDown.Value;
```

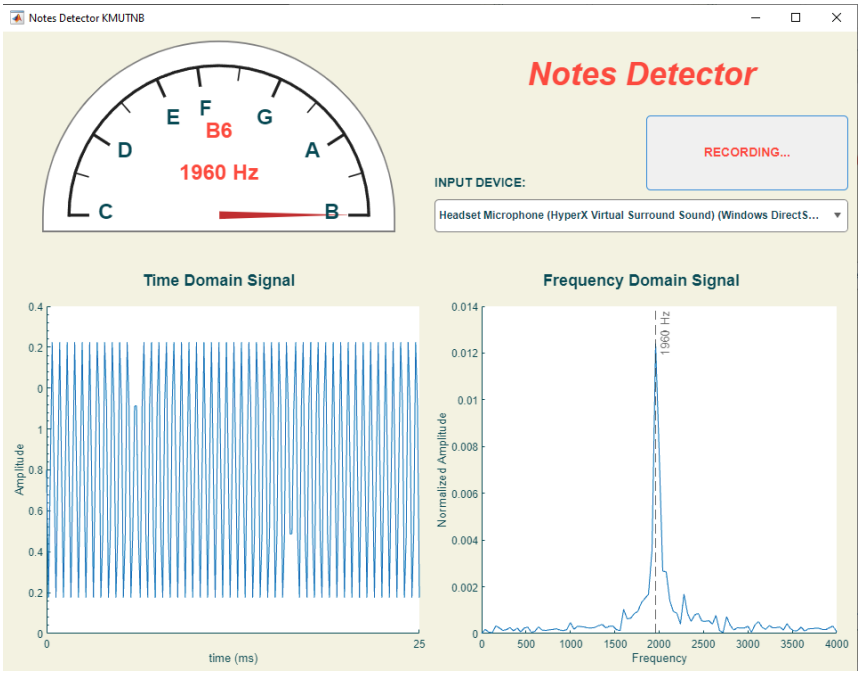
```
    app.deviceID = -1;

    if value ~= "Default"
        for i = 1:app.TotolDevice

            if strcmp(app.InputsInfo(i).Name,value)
                app.deviceID = app.InputsInfo(i).ID;
                break;
            end

        end
    end

    app.setAudiorecorder()

end
```

จากนั้นเราจะสามารถเลือก อุปกรณ์ในการรับเสียงได้แล้ว

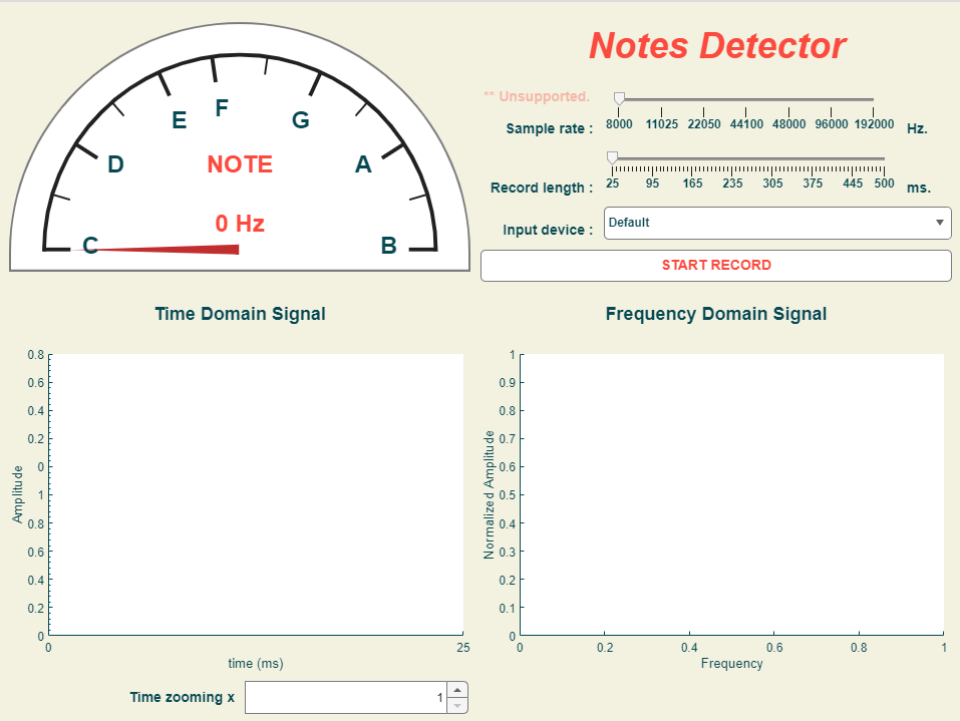




9.เมื่อทดลองใช้พบว่า เมื่อป้อนความถี่ต่ำ Application จะตรวจสอบโน้ตผิดจากความเป็นจริง ส่วน เมื่อป้อนความถี่สูงจะตรวจสอบถูก แต่จะดูรูปลักษณะสัญญาณได้ไม่ชัดเจน



จึงมีการเพิ่ม feature การ Zoom เพื่อให้ดูลักษณะคลื่นได้ชัดขึ้น รวมถึง feature การเปลี่ยน Sample rate และ ความยาวของข้อมูลเสียงที่เราจะนำมาวิเคราะห์ เพื่อให้การวิเคราะห์ความถี่มีความแม่นยำและละเอียดมากขึ้น ซึ่งได้ Application หน้าตาสุดท้ายได้ต่างจากที่ออกแบบไว้ตอนแรกนิดหน่อยดังรูป



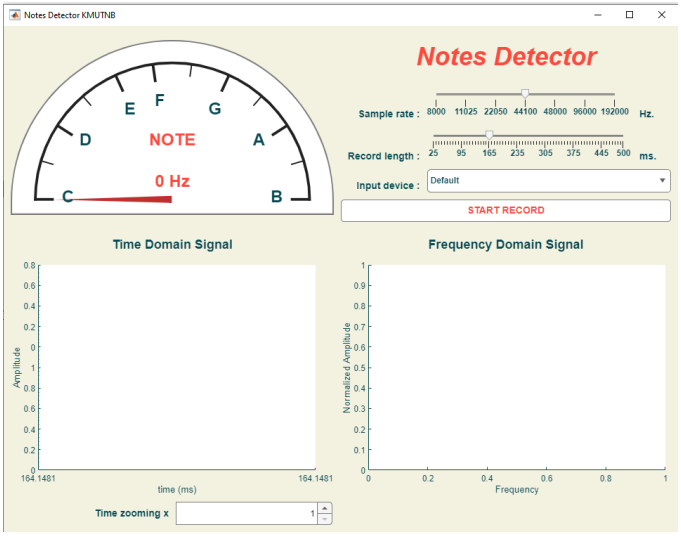
และ Code ต่างๆเกี่ยวกับ Application ของเราสามารถเข้าไปดูทั้งหมดได้ที่

https://github.com/SattapoomTulyasuk/Notes_Detector_MATLAB.git

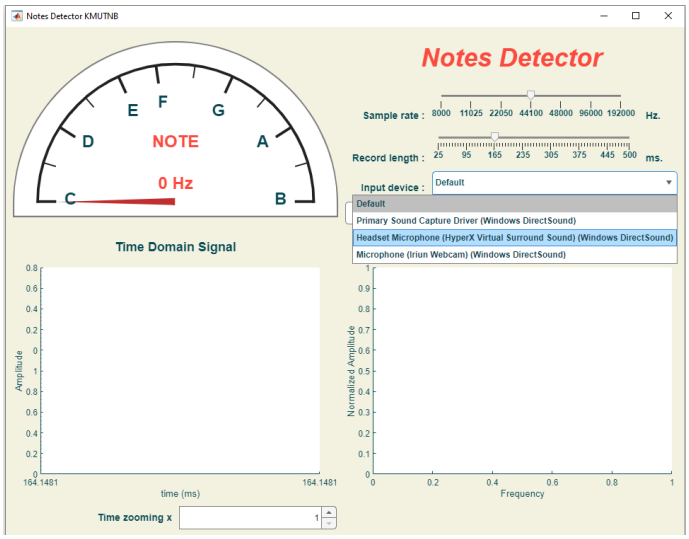
การใช้งานและสรุปผล

วิธีการใช้งาน

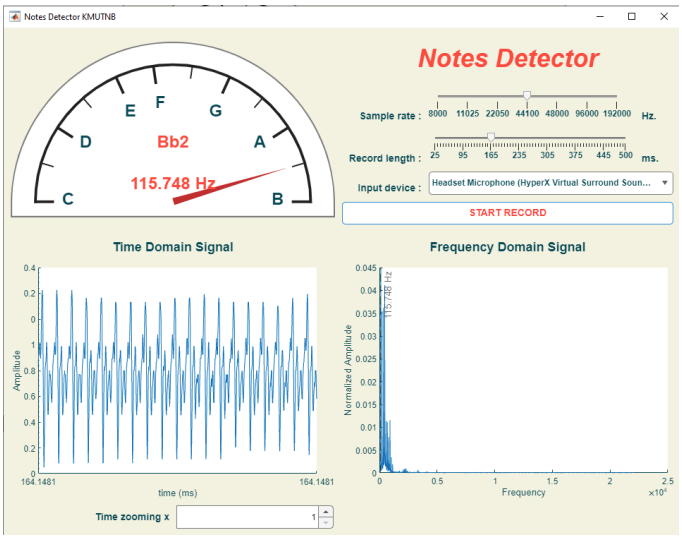
1.ตั้งค่า Sampling Rate (ความถี่ในการเก็บข้อมูล) และ Record Length (ระยะเวลาของเสียงที่จะนำมาวิเคราะห์) ให้เหมาะสมกับเครื่องดนตรี



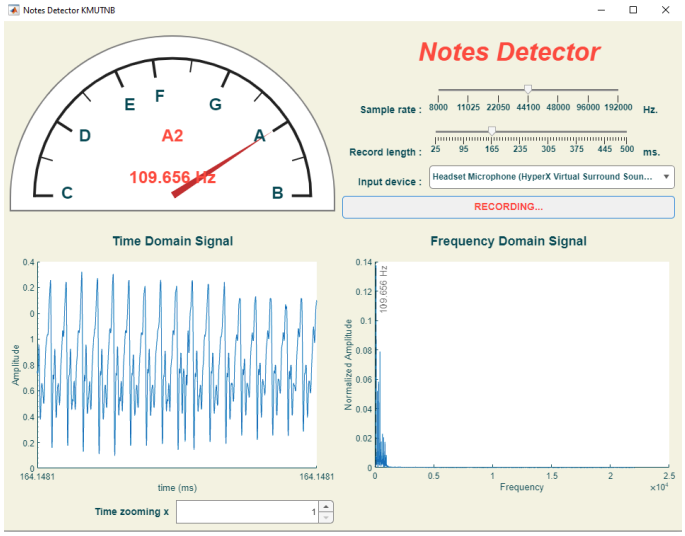
2.เลือกอุปกรณ์รับเสียงที่ Input device



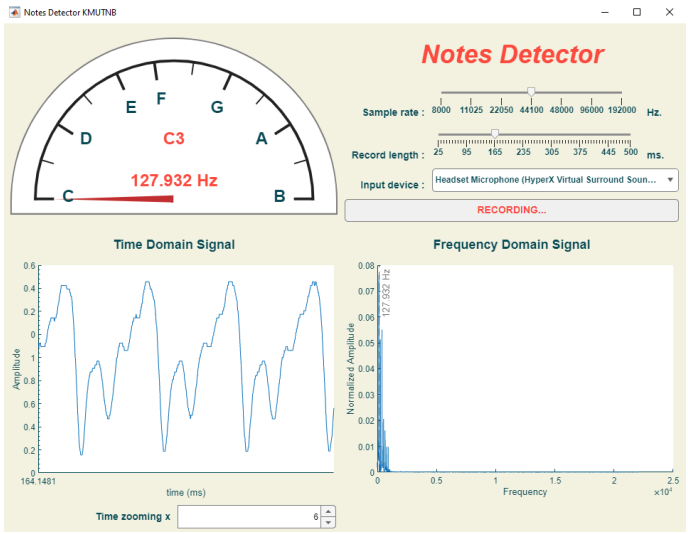
3.เริ่มบันทึกเสียงโดยการกด Start Record



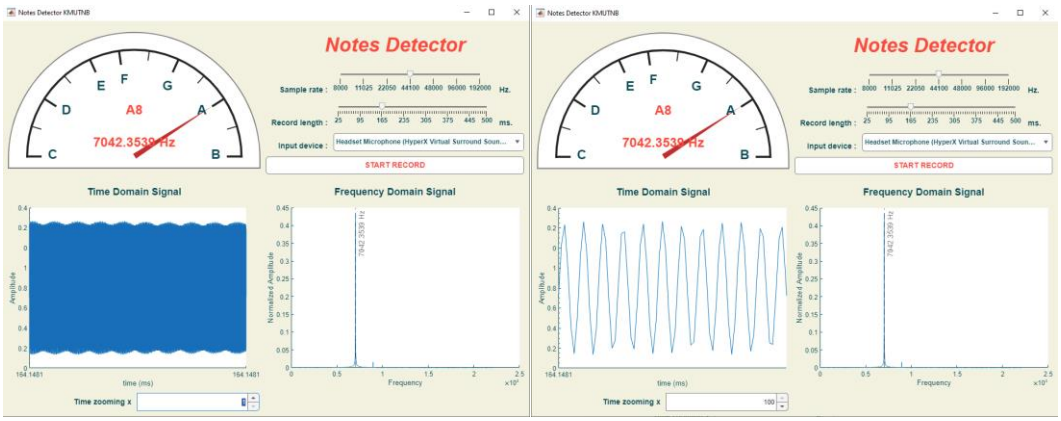
4.ดูคลื่นสัญญาณ Time Domain ทางด้านซ้าย และ Frequency Domain ทางด้านขวา ตัวโน้ตและความถี่ที่วิเคราะห์ได้ที่หน้าปัดด้านซ้ายบน โดยหน้าปัด และกราฟทั้งสองจะอัพเดทเรื่อย ๆ ทุกๆควาเวลาเท่ากับ Record length ที่ตั้งไว้ จนกว่าจะกดหยุดอัด



5.หากดูลักษณะของรูปคลื่นแต่ละคาบไม่ชัดสามารถ Zoom แกนเวลาได้โดยการเพิ่มตัวคูณที่ Time Zooming



6.หยุด Record โดยการกดที่ปุ่มเดิม เพื่อดูสัญญาณสุดท้ายล่าสุด และยังใช้ตัวคูณเวลาได้เหมือนเดิม



ปัญหาที่พบและวิธีการแก้ไข

ปัญหา

- เมื่อมีอุปกรณ์ IO ที่เกี่ยวกับเสียงเชื้อมต่อหลายตัวพร้อมกันบางครั้งไม่สามารถวิเคราะเสียงที่มาจากอุปกรณ์ที่ต้องการได้
- เกิดข้อจำกัดของ Fast Fourier Transform คือเมื่อ Sample rate และ ความยาวข้อมูลที่นำมาวิเคราะห์ต่ำ ความละเอียดในการวิเคราะห์จะต่ำ จนช่วงที่มีความถี่ต่ำจะวิเคราะห์ได้ผิดพลาด
- เมื่อจะดูลักษณะคลื่นของสัญญาณที่รับเข้ามาใน Time Domain จะดูได้ยากเมื่อมีความถี่สูง

วิธีแก้

- เพิ่ม Feature การเลือก Input Device
- เพิ่ม Feature การตั้งค่า Sample rate และระยะเวลาการเก็บข้อมูลเสียง
- เพิ่ม Feature การซูมในแกนเวลาของกราฟ Time Domain

แหล่งงานและข้อมูลอ้างอิง

- https://mixbutton.com/mixing-articles/music-note-to-frequency-chart/
- https://www.mathworks.com/help/matlab/ref/fft.html
- https://www.mathworks.com/help/matlab/ref/audiorecorder.html
- https://www.mathworks.com/help/matlab/ref/audiorecorder.getaudiodata.html?s_tid=doc_ta
- https://www.mathworks.com/help/matlab/ref/linspace.html?s_tid=doc_ta
- https://www.mathworks.com/company/newsletters/articles/introduction-to-object-oriented-programming-in-matlab.html
- https://www.mathworks.com/videos/series/building-apps-in-matlab.html
- https://guitargearfinder.com/guides/how-to-connect-guitar-to-pc/
- https://www.youtube.com/watch?v=khZRquNckHc
- https://youtu.be/a03-5mr5yn4