EE 113D LAB 1B
Lab 2

Khyle Calpe
405016683

EE113D: Digital Signal Processing Design

Lab 2: Voice Transformation

OBJECTIVE

The objective of Lab 2 is to implement a DSP player & recorder by configuring the I/O pin & clock configurations of the Nucleo board and coding with inbuilt functions to sample and transform voice recordings. The lab consists of 4 parts: voice recording, reverb transformation, up sampling, and down sampling. In all of the components, the lab requires the use of signal processing techniques to properly modify the voice samples.

DATA

Voice Sampling

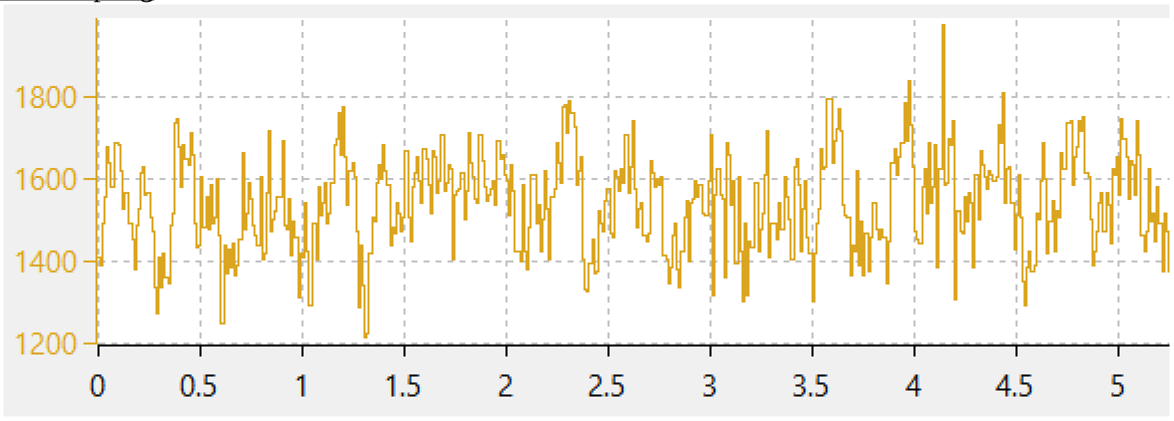
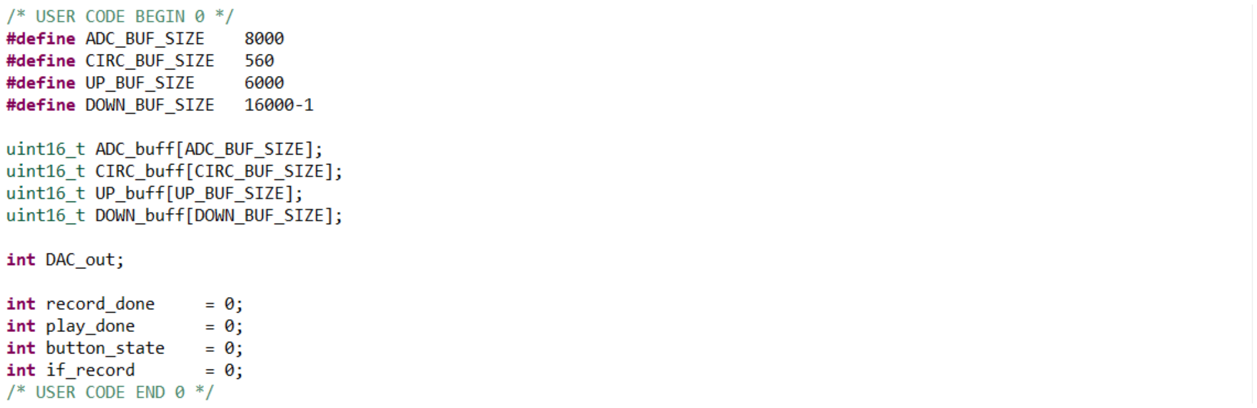
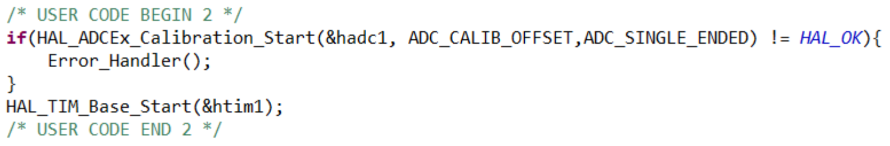


Fig. 1: Sample of voice recording with the word "demo" at 8kHz.

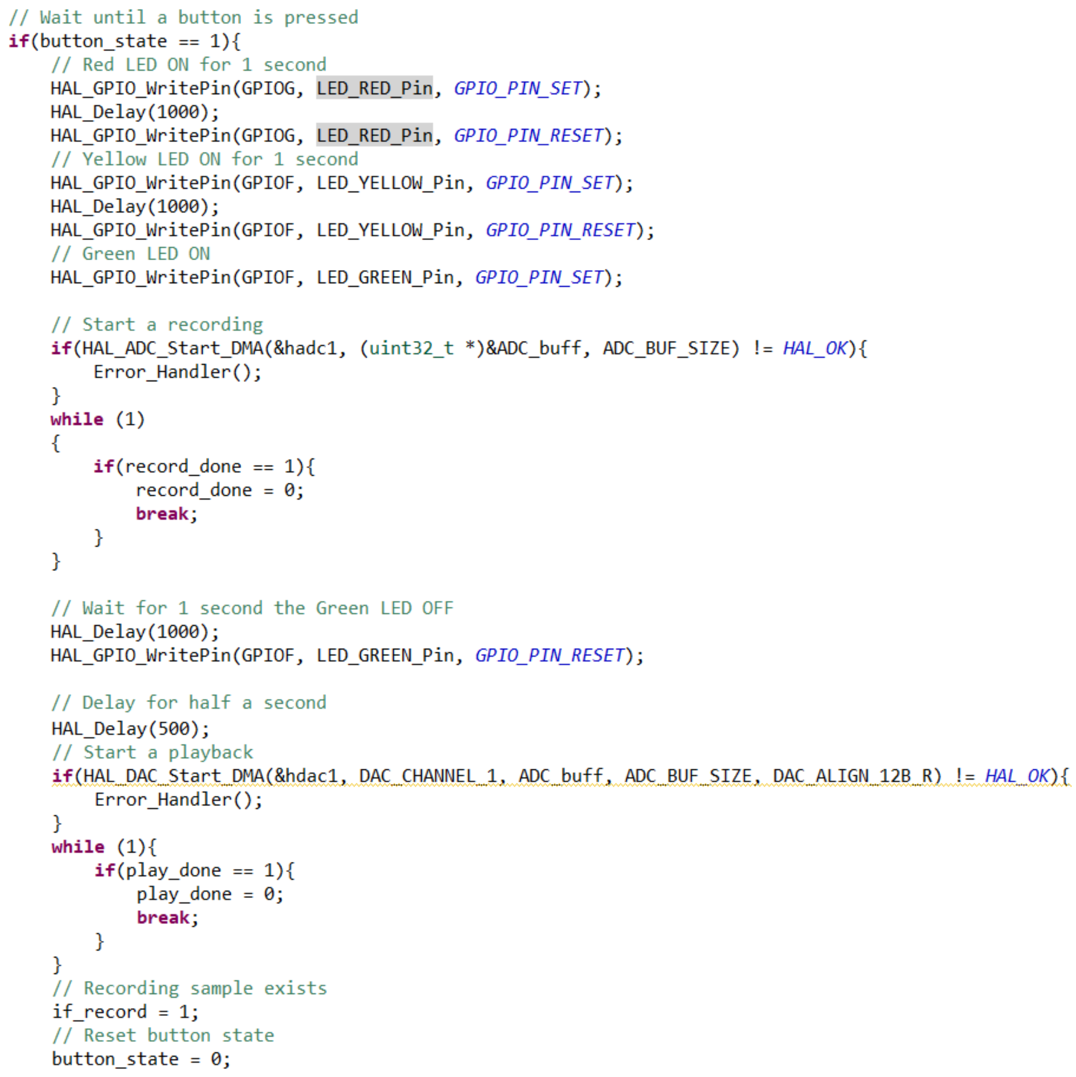
CODE:



```
/* USER CODE BEGIN 0 */
#define ADC_BUF_SIZE    8000
#define CIRC_BUF_SIZE   560
#define UP_BUF_SIZE     6000
#define DOWN_BUF_SIZE   16000-1

uint16_t ADC_buff[ADC_BUF_SIZE];
uint16_t CIRC_buff[CIRC_BUF_SIZE];
uint16_t UP_buff[UP_BUF_SIZE];
uint16_t DOWN_buff[DOWN_BUF_SIZE];

int DAC_out;

int record_done     = 0;
int play_done       = 0;
int button_state    = 0;
int if_record       = 0;
/* USER CODE END 0 */
```



```
/* USER CODE BEGIN 2 */
if(HAL_ADCEx_Calibration_Start(&hadc1, ADC_CALIB_OFFSET,ADC_SINGLE_ENDED) != HAL_OK){
    Error_Handler();
}
HAL_TIM_Base_Start(&htim1);
/* USER CODE END 2 */
```



```
// Wait until a button is pressed
if(button_state == 1){
    // Red LED ON for 1 second
    HAL_GPIO_WritePin(GPIOG, LED_RED_Pin, GPIO_PIN_SET);
    HAL_Delay(1000);
    HAL_GPIO_WritePin(GPIOG, LED_RED_Pin, GPIO_PIN_RESET);
    // Yellow LED ON for 1 second
    HAL_GPIO_WritePin(GPIOF, LED_YELLOW_Pin, GPIO_PIN_SET);
    HAL_Delay(1000);
    HAL_GPIO_WritePin(GPIOF, LED_YELLOW_Pin, GPIO_PIN_RESET);
    // Green LED ON
    HAL_GPIO_WritePin(GPIOF, LED_GREEN_Pin, GPIO_PIN_SET);

    // Start a recording
    if(HAL_ADC_Start_DMA(&hadc1, (uint32_t *)&ADC_buff, ADC_BUF_SIZE) != HAL_OK){
        Error_Handler();
    }
    while (1)
    {
        if(record_done == 1){
            record_done = 0;
            break;
        }
    }

    // Wait for 1 second the Green LED OFF
    HAL_Delay(1000);
    HAL_GPIO_WritePin(GPIOF, LED_GREEN_Pin, GPIO_PIN_RESET);

    // Delay for half a second
    HAL_Delay(500);
    // Start a playback
    if(HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, ADC_buff, ADC_BUF_SIZE, DAC_ALIGN_12B_R) != HAL_OK){
        Error_Handler();
    }
    while (1){
        if(play_done == 1){
            play_done = 0;
            break;
        }
    }
    // Recording sample exists
    if_record = 1;
    // Reset button state
    button_state = 0;
```

Reverb Effect

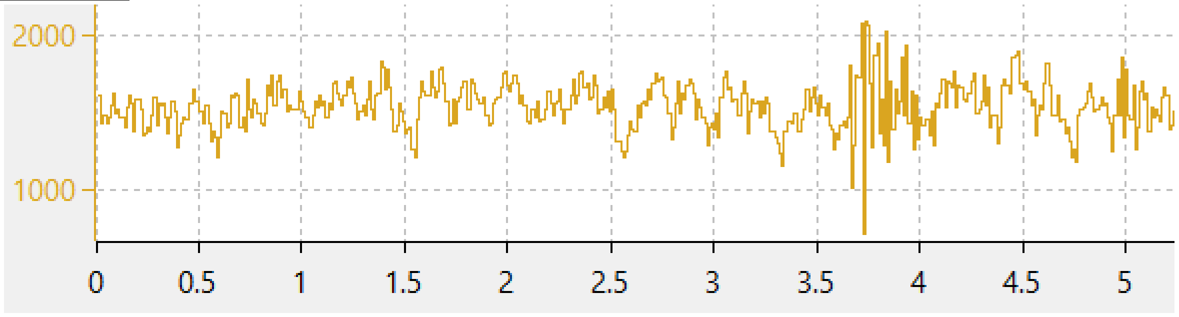


Fig. 2: Sample of voice recording transformed with a reverb effect.

CODE:

```
}else if(button_state == 2){
    // Check if a recording sample exists
    if(!if_record) continue;
    // Green LED ON & Red LED OFF to signal activity
    HAL_GPIO_WritePin(GPIOF, LED_GREEN_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOG, LED_RED_Pin, GPIO_PIN_RESET);
    // Reverb Effect
    for(int i=0; i<ADC_BUF_SIZE; i++){
        // Reverb + Delay
        DAC_out = ADC_buff[i] + (2/3.f) * CIRC_buff[i%CIRC_BUF_SIZE];
        if(HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, &DAC_out, 1, DAC_ALIGN_12B_R) != HAL_OK){
            Error_Handler();
        }
        while (1){
            if(play_done == 1){
                play_done = 0;
                break;
            }
        }
        // Circular buffer for delay
        CIRC_buff[i%CIRC_BUF_SIZE] = ADC_buff[i];
    }
    // Reset button state
    button_state = 0;
```

Up Sampling

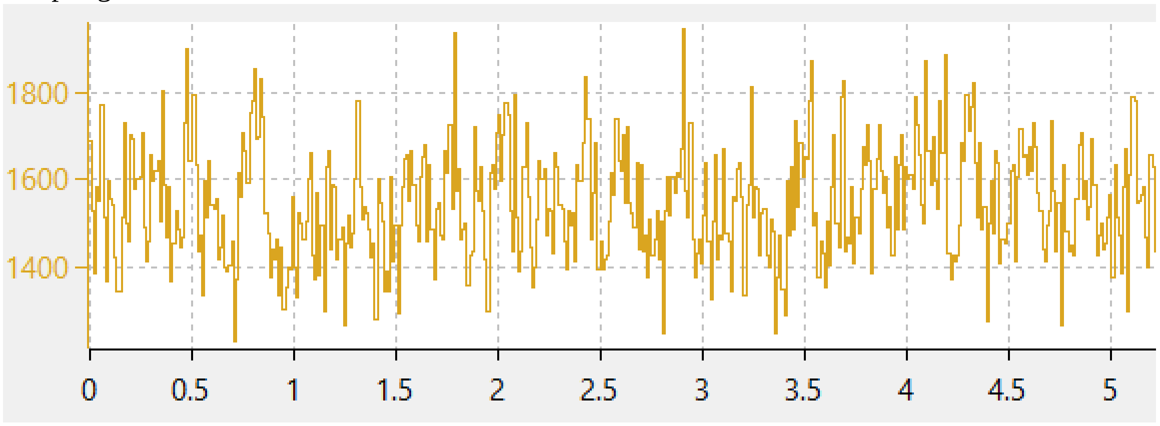


Fig. 3: Interpolated sample of voice recording speeded up by a factor of a third.

CODE:

```c
}else if(button_state == 3){
    // Check if a recording sample exists
    if(!if_record) continue;
    // Green LED ON & Red LED OFF to signal activity
    HAL_GPIO_WritePin(GPIOF, LED_GREEN_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOG, LED_RED_Pin, GPIO_PIN_RESET);
    // Fill the upsampled array with the recording
    for(int i=0; i<ADC_BUF_SIZE/4; i++){
        UP_buff[3*i] = ADC_buff[4*i];
    }
    // Interpolate the upsampled array
    int point_1, point_2, point_3;
    for(int i=0; i<ADC_BUF_SIZE/4; i++){
        point_1 = ADC_buff[4*i+1];
        point_2 = ADC_buff[4*i+2];
        point_3 = ADC_buff[4*i+3];
        UP_buff[3*i+1] = (1 / 3.f) * (point_1 - point_2) + point_1;
        UP_buff[3*i+2] = (2 / 3.f) * (point_2 - point_3) + point_2;
    }
    // Start a playback
    if(HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, UP_buff, UP_BUF_SIZE, DAC_ALIGN_12B_R) != HAL_OK){
        Error_Handler();
    }
    while (1){
        if(play_done == 1){
            play_done = 0;
            break;
        }
    }
    // Reset button state
    button_state = 0;
```

Down Sampling

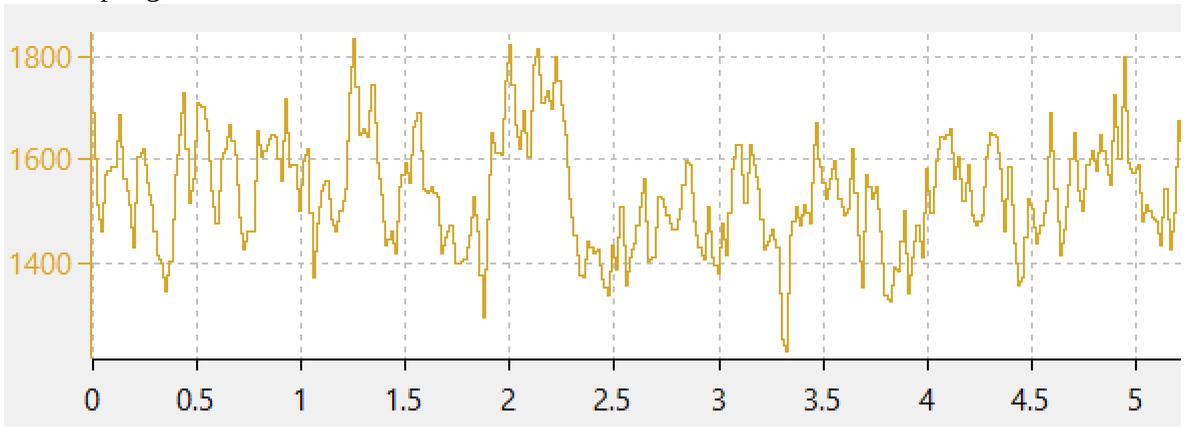


Fig. 4: Interpolated sample of voice recording slowed down by a factor of a half.

CODE:

```
}else if(button_state == 4){
    // Check if a recording sample exists
    if(!if_record) continue;
    // Green LED ON & Red LED OFF to signal activity
    HAL_GPIO_WritePin(GPIOF, LED_GREEN_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOG, LED_RED_Pin, GPIO_PIN_RESET);
    // Fill the downsampled array with the recording
    for(int i=0; i<ADC_BUF_SIZE; i++){
        DOWN_buff[2*i] = ADC_buff[i];
    }
    // Interpolate the downsampled array
    for(int i=0; i<ADC_BUF_SIZE-1; i++){
        DOWN_buff[2*i+1] = (1 / 2.f) * (DOWN_buff[2*i+2] + DOWN_buff[2*i]);
    }
    // Start a playback
    if(HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, DOWN_buff, DOWN_BUF_SIZE, DAC_ALIGN_12B_R) != HAL_OK){
        Error_Handler();
    }
    while (1){
        if(play_done == 1){
            play_done = 0;
            break;
        }
    }
    // Reset button state
    button_state = 0;
}
// Green LED OFF & Red LED ON to signal inactivity
HAL_GPIO_WritePin(GPIOF, LED_GREEN_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOG, LED_RED_Pin, GPIO_PIN_SET);
```

RESULTS

The results of the lab followed expectations and met the requirements to a sufficient degree. For example, the voice recordings and transformation audio outputs are clear, albeit with some environmental noise. With some windowing techniques applied to the Fourier Transform of the signals, the speaker would output clearer and less noisy sound. However, physical limitations of the equipment proved to be quite difficult to work with and costed significant time lost to debugging. For instance, the lack of proper solder connections between the microphone and the amplifier to the rest of the circuit resulted in unreliable audio measurements and distorted audio outputs. To solve this issue, the wire connections between the microphone and the rest of the circuit were routinely checked for loose connections. For future labs, soldering might be required if too many devices have crimp connections.