Device Activation Code

```
% Load the reference (password) audio signal
[audio_signal_ref, fs_ref] = audioread('bonjour.wav');  % Replace with the actual password signal file

% Load the test signal
[audio_signal_test, fs_test] = audioread('bonjour.wav');  % Replace with the test signal file

% Check if sampling rates match
if fs_ref ~= fs_test
   disp('Warning: Sampling rates do not match!');
   return;
end

% Normalize both the reference and test signals
audio_signal_ref = audio_signal_ref / max(abs(audio_signal_ref));  % Normalize reference
audio_signal_test = audio_signal_test / max(abs(audio_signal_test));  % Normalize test signal

% Parameters
F1 = 50;  % Frequency for bandpass filter (adjust as needed)

% --- Process Password Signal ---
% Bandpass filter F1 for reference signal
bpFilt1_ref = designfilt('bandpassiir', 'FilterOrder', 4, ...
                         'HalfPowerFrequency1', F1-5, ...
                         'HalfPowerFrequency2', F1+5, ...
                         'SampleRate', fs_ref);
Yf1_ref = filter(bpFilt1_ref, audio_signal_ref);
Pyf1_ref = mean(Yf1_ref.^2);  % Power of the filtered signal F1
```

```
% Bandpass filter F1*3 for reference signal
bpFilt2_ref = designfilt('bandpassiir', 'FilterOrder', 4, ...
                    'HalfPowerFrequency1', 3*F1-5, ...
                    'HalfPowerFrequency2', 3*F1+5, ...
                    'SampleRate', fs_ref);
Yf2_ref = filter(bpFilt2_ref, audio_signal_ref);
Pyf2_ref = mean(Yf2_ref.^2);  % Power of the filtered signal F1*3

% Calculate Thresholds from the reference signal (password)
Th1 = 0.95 * Pyf1_ref;  % Threshold for filtered signal F1
Th2 = 0.95 * Pyf2_ref;  % Threshold for filtered signal F1*3

% Display the reference thresholds
disp(['Threshold Th1 (Reference): ', num2str(Th1)]);
disp(['Threshold Th2 (Reference): ', num2str(Th2)]);

% --- Process Test Signal ---
% Bandpass filter F1 for test signal
bpFilt1_test = designfilt('bandpassiir', 'FilterOrder', 4, ...
                    'HalfPowerFrequency1', F1-5, ...
                    'HalfPowerFrequency2', F1+5, ...
                    'SampleRate', fs_test);
Yf1_test = filter(bpFilt1_test, audio_signal_test);
Pyf1_test = mean(Yf1_test.^2);  % Power of the filtered signal F1

% Bandpass filter F1*3 for test signal
bpFilt2_test = designfilt('bandpassiir', 'FilterOrder', 4, ...
                    'HalfPowerFrequency1', 3*F1-5, ...
                    'HalfPowerFrequency2', 3*F1+5, ...
                    'SampleRate', fs_test);
Yf2_test = filter(bpFilt2_test, audio_signal_test);
```

```
Pyf2_test = mean(Yf2_test.^2);  % Power of the filtered signal F1*3

% Display the power of the filtered test signals
disp(['PYF1 (Test): ', num2str(Pyf1_test)]);
disp(['PYF2 (Test): ', num2str(Pyf2_test)]);

% --- Debugging Outputs ---
disp('--- Debugging Outputs ---');
disp(['Power of Filtered Reference Signal (F1): ', num2str(Pyf1_ref)]);
disp(['Power of Filtered Reference Signal (F1*3): ', num2str(Pyf2_ref)]);

disp(['Power of Filtered Test Signal (F1): ', num2str(Pyf1_test)]);
disp(['Power of Filtered Test Signal (F1*3): ', num2str(Pyf2_test)]);

% --- Activation Check ---
disp(['Comparing test power to thresholds:']);
disp(['Pyf1_test: ', num2str(Pyf1_test), ' | Th1: ', num2str(Th1)]);
disp(['Pyf2_test: ', num2str(Pyf2_test), ' | Th2: ', num2str(Th2)]);

% Compare powers
if Pyf1_test > Th1 && Pyf2_test > Th2
   disp('Activation: Password Accepted');
else
   disp('Activation: Password Rejected');
end
```