فربد فولادی-98243045

عرفان رفیعی اسکوئی-98243027

در ابتدا ما به دو متغیر و یک آرایه نیاز داریم تا در ادامه از این ها استفاده کنیم.

یک Sampling Rate و یک window length که اولی برای محاسبه k ها و coef مورد استفاده قرار می گیرد و دومی نیز برای قسمت بندی بافر استفاده می شود که سیگنال خوانده شده در آن قرار دارد.

یک آرایه به نام Sampling Data نیز داریم که اندازه آن برابر با 5000 (چون یک مقداری از سیگنال باید خوانده شود تا ورودی تشخیص داده شود پس این عدد را انتخاب می کنیم تا از خواندن حداقل سیگنال ورودی اطمینان حاصل کنیم) است و برای این استفاده می شود که مقدار آنالوگی که خوانده می شود را در خودش نگهداری کند و در ادامه با window length آن را از بافر بخوانیم.

مقدار sampling rate را چون ما در هر بار که سمپل می گیریم بالا ترین فرکانس تقربا 1630 است و دو برابر آن 3200 می باشد تقریبا و ما چون باید حداقل دو برابر آن را سمپل گرفت ما یک عدد انتخابی 5000 را در نظر گرفتیم که 20 الی 25 درصد از حداقل سمپل نیز بالا تر باشد تا مقدار خطا به صفر برسد.

در ادامه مقدار k هر ROW و COL را و دوبرابر آن را حساب می کنیم متناسب با فرمولی که در صورت پروژه داده شده بود.

بعد ار محاسبات k ها مقادیری برای coef هر k تعریف می کنیم که در main آن ها را با رابطه داده شده مقدار دهی می کنیم.

به دو تابع uint8_t Seven_Segment_Decoder(uint8_t num) و void goertzel() نیاز داریم که اولی برای نشان دادن عدد مد نظر و دومی برای پیدا کردن عدد مد نظر است.

در تابع Seven_Segment_Decoder با استفاده از یک سویچ کیس اعداد و علامت ها را نشان می دهیم.

تابع ()goertzel نیز زمانی صدا زده می شود که ما در آرایه در index ای از ارایه باشیم که بخش پذیر بر window length باشد در غیر این صورت index یا یکی زیاد می شود یا برابر با صفر می شود که سیگنال جدید را بگیرد.

در تابع ()goertzel در ابتدا با کمک تابع Power توان هر coef را متناسب با رابطه داده شده حساب می کند.

```
float Power(float coef)
{
  float Q0, Q1, Q2;
  Q2 = 0;
  Q1 = 0;
  Q0 = 0;

  for(int i = Index - Window_Lenth + 1; i <= Index; i++){
    Q2 = Q1;
    Q1 = Q0;
    Q0 = (coef * Q1) - Q2 + (float)Sampelling_Data[i];
  }
  return Q1 * Q1 + Q2 * Q2 - (coef * Q1 * Q2);
}
```

در ادامه با ازای هر row و col یک آرایه 4 تایی داریم که توان هر یک از coef های row را در خودش نگهداری می کند و در ادامه ماکسیمم آن را انتخاب می کند.

```
float myPowers_row[4];
float myPowers_col[4];
uint8_t ROW_MAX = 0, COL_MAX = 0;

myPowers_row[0] = Power(coef_k697) + Power(coef_k_mul697);
myPowers_row[1] = Power(coef_k770) + Power(coef_k_mul770);
myPowers_row[2] = Power(coef_k852) + Power(coef_k_mul852);
myPowers_row[3] = Power(coef_k941) + Power(coef_k_mul941);
for(int j = 0; j < 3; j++)
  if(myPowers_row[j] <= myPowers_row[j + 1])
    ROW_MAX = j + 1;
```

به همین صورت برای col ها.

```
myPowers_col[0] = Power(coef_k1209) + Power(coef_k_mul1209);
myPowers_col[1] = Power(coef_k1336) + Power(coef_k_mul1336);
myPowers_col[2] = Power(coef_k1477) + Power(coef_k_mul1477);
myPowers_col[3] = Power(coef_k1633) + Power(coef_k_mul1633);
for(int j =0; j < 3; j++)
  if(myPowers_col[j] <= myPowers_col[j + 1])
    COL_MAX = j + 1;
```

در ادامه با چک کردن MAX_ROW و MAX_COL عدد متناظر با آن عدد یا کرکتر را با تابع Seven_Segment_Decoder نمایش می دهیم و در output قرار می دهیم.