# Chapter 8

# Final Implementation

## 8.1 Final System on DSP Board

The final system implemented on ADSP-BF548 realizes following functions.

- Adaptive noise cancellation based on Least Mean Square Algorithm
- Real-time recognition of spoken words
- Result indication via a five-LED array
- YouTube controller

### 8.1.1 LED Indicator

In addition to the result sent back to VisualDSP++, LEDs are employed to indicate recognition result. The DSP board provides 6 general-purpose (amber) LEDs. `LED1` through `LED6` are accessed via the `PG6-PG11` pins of the processor [35].

As is summarized in Table 6.1 on page 59, there are 27 potential recognition outcomes. `LED1` to `LED5` with a capacity of $2^5 = 32$ are designed to represent the recognition result and `LED6` toggles between 'on' and 'off' when new result comes. For example, ☒🟧□🟧□🟧 represents word *zero* ($(10101)_2 = 21$) while ☒□🟧□🟧□ symbolizes word *ten* ($(01010)_2 = 10$). Five zeros ☒□□□□□ mean invalid recognition outcomes.

### 8.1.2 Utilization in Home Automation

We use an Arduino® board as an intermediate between the word recognition system and applications in home automation primarily because of the open-source ecosystem of abundant resources already available to Arduino developers. For example, with the aid of infrared emitter and WIFI module as well as supporting example program, we are able to easily control a TV and send command via wireless communication.

#### Communication between DSP and Arduino

We simply utilize general purpose input / output (GPIO) pins to exchange information between DSP and Arduino. `PG6-PG11` pins connected to `LED1-LED6` are accessible through pin 79-84 of `J1` connector on the back of DSP board. (Fig. A.5 on page 104 illustrates the pin information of `J1` connector.) Five TTL values (either 'high' or 'low') encode the recognition outcome and the toggle of most significant bit indicates the advent of new result.

We also tried to transmit command via UART serial port. It took a long time to configure the corresponding device manager. Eventually, we decided to stick on the GPIO method.

#### Communication between Arduino and PC

Arduino retrieves logic levels of aforementioned pins and then decodes the binary information. At the next step, Arduino sends the decoded result to the WIFI module via the serial port (8 data bits, no parity, one stop bit, 9600 bps). At this stage, PC can directly receive the result through a USB cable.

#### YouTube Controller on PC

A python program keeps listening the serial port (wired connection) or the `Telnet` port (wireless connection). Afterwards, python program simulates a keyboard press according to the received command. Thereby, YouTube on PC becomes controlled by speech command.

#### Other applications

Thanks to the ample Arduino accessories, more home appliances can be easily integrated into our system. For example, air conditioners can be controlled by the infrared signal sent by IR emitter. AC lights can also be controlled by an Arduino with relay module.

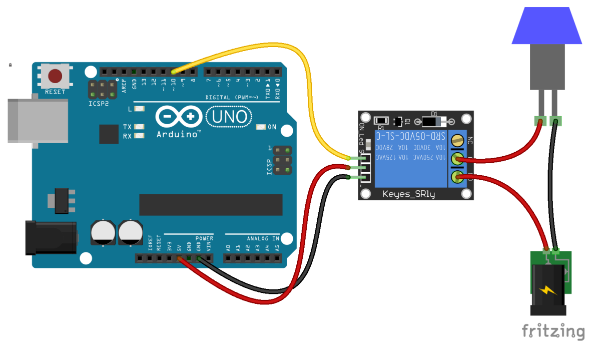

Figure 8.1: Control AC Light using Arduino with Relay Module [36]

## 8.2 Performance on DSP board

### 8.2.1 Feature Extraction & Recognition

We randomly test the final system on DSP for 23 times and compare the outcomes computed by DSP and MATLAB. In term of dataset $n$ ($n = 1, 2, \dots, 23$) and word $k$ ($k = 1, 2, \dots, 27$), let $p_{MATLAB}[n, k]$ denote the probability (natural-log scale) computed by MATLAB and $p_{DSP}[n, k]$ denote the probability (natural-log scale) computed by DSP.

We define the *relative error* of word $k$ in dataset $n$

$$\varepsilon[n,k] = \frac{p_{DSP}[n,k] - p_{MATLAB}[n,k]}{p_{MATLAB}[n,k]} \tag{8.1}$$

and the *Root Mean Square Error* of dataset $n$

$$\epsilon_{RMSE}[n] = \sqrt{\frac{1}{27}\sum_{k=1}^{27}(\varepsilon[n,k])^2} \tag{8.2}$$

For a certain dataset $n$, $\epsilon_{RMSE}[n]$ represents the 'average' of $\varepsilon[n, k]$.

Fig. 8.2 shows that absolute relative error $|\varepsilon[n, k]|$ ranges from $1 \times 10^{-6}$ to $9 \times 10^{-3}$ and dataset 1 has the largest deviation (navy solid line). Fig. 8.3 illustrates the root mean square error $\epsilon_{RMSE}[n]$ of each dataset $n$ and dataset 1 has the largest $\epsilon_{RMSE}[n]$ consistently (navy solid line).

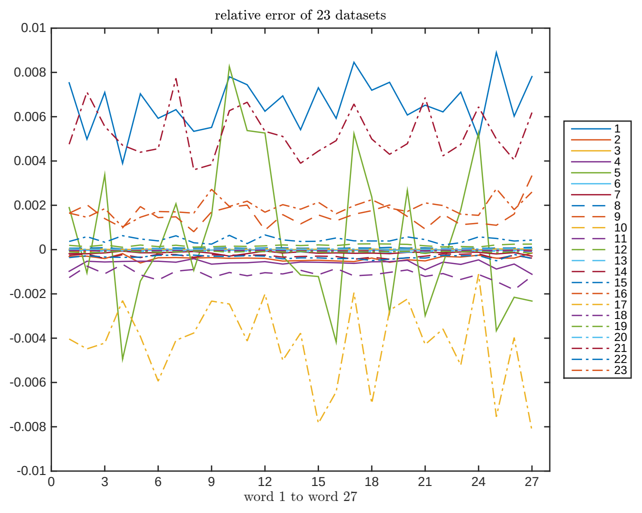

Figure 8.2: $\varepsilon[n, k]$ of 27 words in 23 datasets

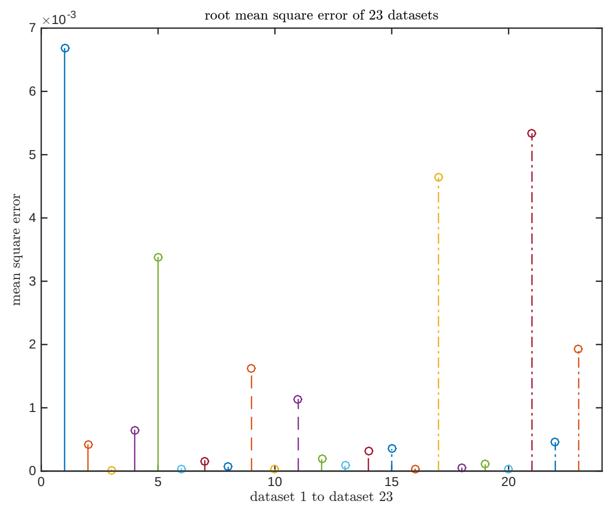

Figure 8.3: $\epsilon_{RMSE}[n]$ of 27 words

In fact, the computation error has no influence on recognition result. Take dataset 1 with the largest root mean square error as an example, Fig. 8.4 depicts the probabilities of 27 words computed by MATLAB (navy circle $\circ$) and DSP (orange cross $\times$). It can be clearly seen that error are much less than the differences between probabilities. Word *one* can be recognized without any interference.

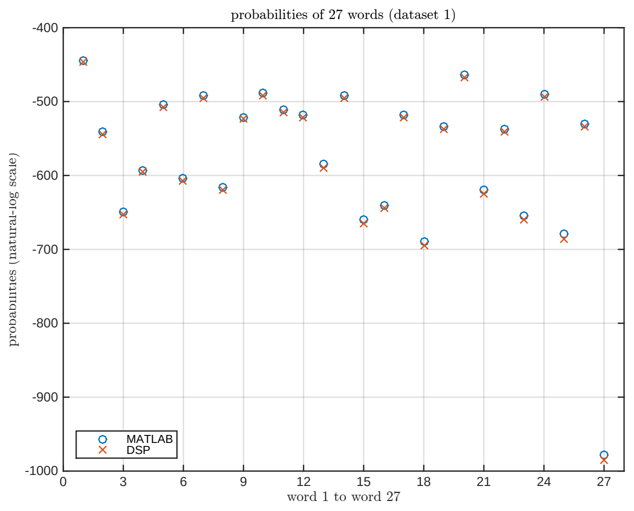

Figure 8.4: 27 probabilities in dataset 1

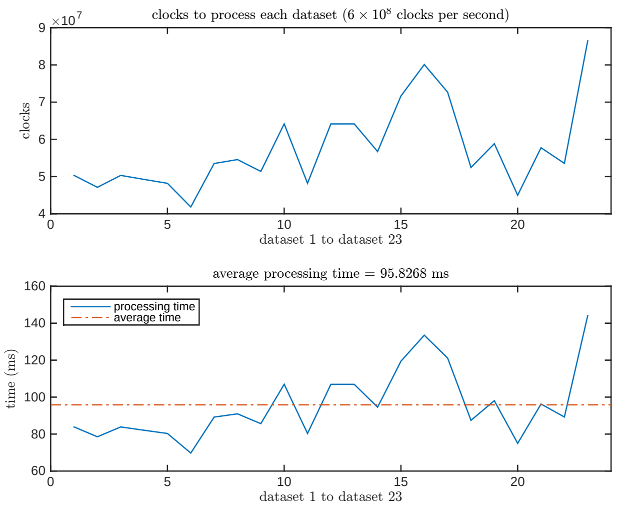

Figure 8.5: Processing time

The clocks required to extract speech features and obtain recognition outcome are plotted in Fig. 8.5. Given $6 \times 10^8$ clocks per second, corresponding processing time can be converted. The average processing time of these 23 datasets is **95.8 ms** $\ll$ 3s (recording time). Processing time varies because different number of frames are passed into MFCC after threshold.