Grove Speech Recognizer Kit for Arduino SKU: 110020108

The Grove Speech Recognizer Kit for Arduino is designed for prototype of Smart Home which includes some basic home elements such as Speech Recognizer, Infrared Emitter. You can learn the functions and applications of Speech Recognizer quickly via this kit, the guideline shows some common demos, let's say you can play music via your speak command 'Play music', or turn on the light according to the corresponding command.

Part List

1. Grove - Speech Recognizer SKU:101020232
2. Grove - MP3 v2.0
3. Grove – RTC
4. Base Shield
5. Grove - Infrared Receiver

6. Grove - Infrared_Emitter

Specification

Parameter	Value/Range
Package size	L:185mm W: 125mm H: 45mm
Gross Weight	222g

Application

How to turn on the TV

Sometimes, we want to control something by our voice, such as TV, air condition. Those devices are controlled by Infrared Remote controller, so we need to know what the code of each button of infrared remote controller first, and then put the code under our program. Finally your voice could trigger those button codes and you can control those devices like infrared remote controller.

Before create this demo you should prepare some necessary components:

- Grove - Infrared Receiver
- Grove - Infrared Emitter
- Grove - Speech Recognizer
- Base Shield
- Arduino UNO
- Infrared Remote controller

Get code of a infrared remote controller via Grove - Infrared Receiver

Download necessary libraries from github : IRSendRev ,pay attention to the path of your libraries: .../arduino/libraries

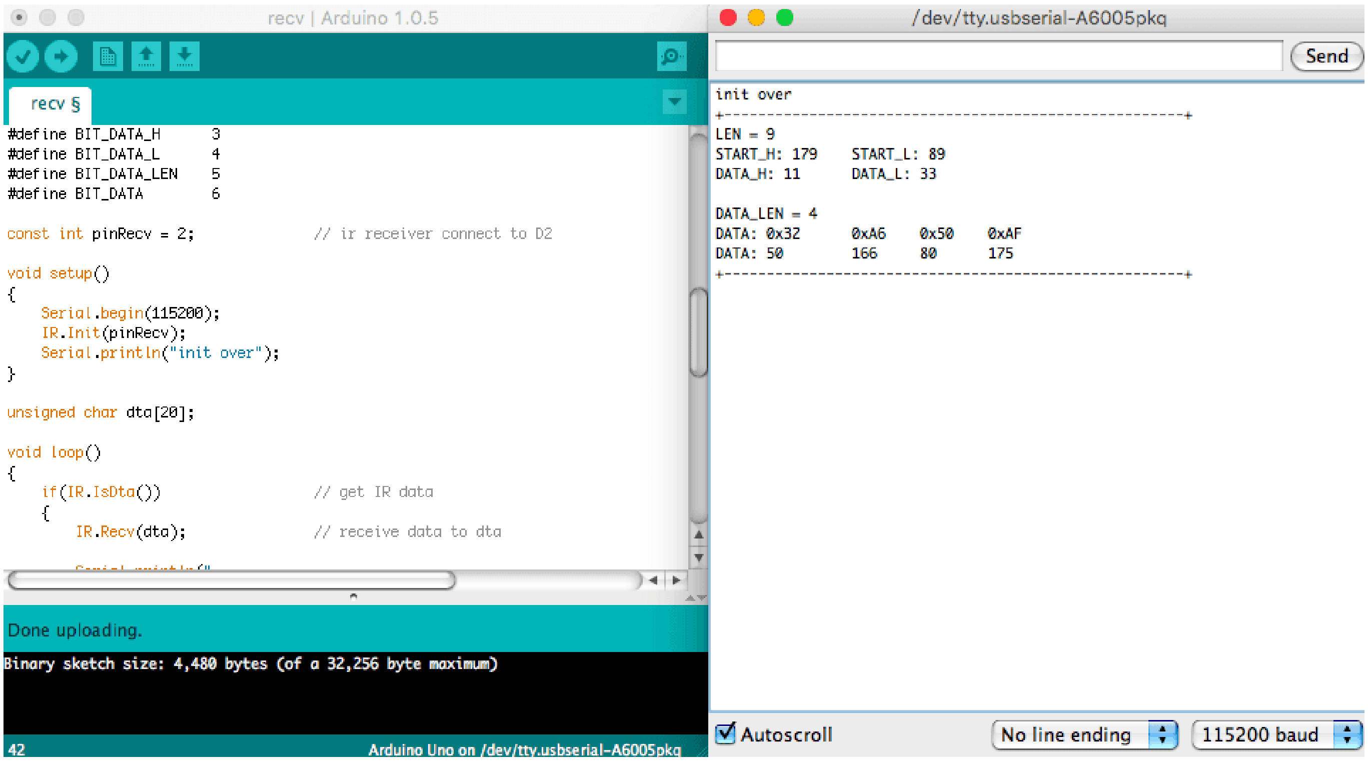
- Now open the example file of "recv" : File --> Sketchbook --> libraries --> IRSendRev --> recv.
- Upload the program to your Arduino UNO.
- Plug Base Shield onto the Arduino UNO,and plug the Grove - Receiver onto the D2 port of Base Shield.

```
const int pinRecv = 2;
```

Also you can change another port while modifying the definement of pin.

- Open the monitor of Arduino UNO.

Press a button of any Infrared Remote Controller, you'll get the detailed information code of the button, see following:



```
#define BIT_DATA_H      3
#define BIT_DATA_L      4
#define BIT_DATA_LEN    5
#define BIT_DATA        6

const int pinRecv = 2;              // ir receiver connect to D2

void setup()
{
    Serial.begin(115200);
    IR.Init(pinRecv);
    Serial.println("init over");
}

unsigned char dta[20];

void loop()
{
    if(IR.IsDta())                  // get IR data
    {
        IR.Recv(dta);               // receive data to dta
```

```
Done uploading.
Binary sketch size: 4,480 bytes (of a 32,256 byte maximum)
42                                  Arduino Uno on /dev/tty.usbserial-A6005pkq
```

```
init over
+------------------------------------------------------+
LEN = 9
START_H: 179    START_L: 89
DATA_H: 11      DATA_L: 33

DATA_LEN = 4
DATA: 0x32      0xA6    0x50    0xAF
DATA: 50        166     80      175
+------------------------------------------------------+
```

Write down the detailed information of the button you want to press. Following are the information of a button:

```
+------------------------------------------------------+
LEN = 9
START_H: 179    START_L: 88
DATA_H: 11     DATA_L: 33

DATA_LEN = 4
DATA: 0x80    0x7E    0x10    0xEF
DATA: 128    126    16    239
+------------------------------------------------------+
```

Modify the IRSend program

Now we can use the previous information code of a button.

- Open the example file of "send" : File --> Sketchbook --> libraries --> IRSendRev --> send.
- Upload the program to your Arduino UNO.
- Plug Base Shield onto the Arduino UNO,and plug the Grove - Receiver onto the D3 port of Base Shield.

First we need modify the default information of the button:

```
unsigned char dtaSend[20];
```

```
void dtaInit()
{
  ............
}
```

Modify those information according to the previous one you wrote:

```
unsigned char dtaSend[20];

void dtaInit()
{
    dtaSend[BIT_LEN]        = 9;            // all data that needs to be sent
    dtaSend[BIT_START_H]    = 179;            // the logic high duration of a
button
    dtaSend[BIT_START_L]    = 88;            // the logic low duration of a button
    dtaSend[BIT_DATA_H]     = 11;            // the logic "long" duration in the
communication
    dtaSend[BIT_DATA_L]     = 33;            // the logic "short" duration in the
communication

    dtaSend[BIT_DATA_LEN]   = 4;            // Number of data which will sent. If
the number is other, you should increase or reduce dtaSend[BIT_DATA+x].

    dtaSend[BIT_DATA+0]     = 128;            // data that will sent
    dtaSend[BIT_DATA+1]     = 126;
    dtaSend[BIT_DATA+2]     = 16;
    dtaSend[BIT_DATA+3]     = 239;
    //dtaSend[BIT_DATA+4]     = 192;
    //dtaSend[BIT_DATA+5]     = 63;
}
```

You can observe that:

```
DATA_LEN = 4
```

So you can note or delete those two lines.

```
   //dtaSend[BIT_DATA+4]     = 192;
    //dtaSend[BIT_DATA+5]     = 63;
```

Of course you can also define many buttons:

```
#include <IRSendRev.h>

#define BIT_LEN         0
```

```
#define BIT_START_H     1
#define BIT_START_L     2
#define BIT_DATA_H      3
#define BIT_DATA_L      4
#define BIT_DATA_LEN    5
#define BIT_DATA        6

const int ir_freq = 38;

unsigned char OpenTV[20];
unsigned char CloseTV[20];
unsigned char IncreaseTemp[20];
unsigned char DecreaseTemp[20];

void OpenTVInit()
{
    OpenTV[BIT_LEN]        = 11;
    OpenTV[BIT_START_H]    = 179;
    /*............ omit ............*/
}

void CloseTVInit()
{
    CloseTV[BIT_LEN]        = 11;
    CloseTV[BIT_START_H]    = 179;
    /*............ omit ............*/
}

void IncreaseTempInit()
{
    IncreaseTemp[BIT_LEN]        = 11;
    IncreaseTemp[BIT_START_H]    = 179;
    /*............ omit ............*/
}

void DecreaseTempInit()
{
    DecreaseTemp[BIT_LEN]        = 11;
    DecreaseTemp[BIT_START_H]    = 179;
   /*............ omit ............*/
}

void setup()
{
    OpenTVInit();
    CloseTVInit();
    IncreaseTempInit();
    DecreaseTempInit();
}

void loop()
{
    IR.Send(OpenTV, 38);
    delay(200);
```

```
        IR.Send(CloseTV, 38);
        delay(200);
        IR.Send(IncreaseTemp, 38);
        delay(200);
        IR.Send(DecreaseTemp, 38);

        delay(2000);
    }
```

Add Speech Recognizer to your IRSend program

- It's very simple to use Speech Recognizer alone, please see the wiki of it first.

We can choose what buttons we need according to the table<ref>Table of return value, Grove - Speech Recognizer</ref> of return value of Speech Recognizer. Following is the table of return value:

Command	Return Value
Turn on the light	1
Turn off the light	2
Play music	3
Pause	4
Next	5
Previous	6
Up	7
Down	8
Turn on the TV	9
Turn off the TV	10
Increase temperature	11
Decrease temperature	12
What's the time	13
Open the door	14
Close the door	15
Left	16
Right	17
Stop	18
Start	19
Mode 1	20

Mode 2	21
Go	22

The following example uses two commands: "Turn of the TV" and "Turn off the TV"

- After reading, we can embed the program of Speech Recognizer in the IRSend program. See the following completed program:

```
#include <IRSendRev.h>
#include <SoftwareSerial.h>

/*========  IR type ========*/

#define BIT_LEN         0
#define BIT_START_H     1
#define BIT_START_L     2
#define BIT_DATA_H      3
#define BIT_DATA_L      4
#define BIT_DATA_LEN    5
#define BIT_DATA        6
const int ir_freq = 38;                  // 38k

/* ========  How many IR buttons you wanna send  ========*/

unsigned char OpenTV[20];
unsigned char CloseTV[20];


/*=========  Choose the pins of Speech Recognizer  ==========*/

#define SOFTSERIAL_RX_PIN  5
#define SOFTSERIAL_TX_PIN  6

SoftwareSerial speech(SOFTSERIAL_RX_PIN,SOFTSERIAL_TX_PIN);


/* =======  How to write the IR data  ========*/
/* ====  You can get those data via IR Recevier  ==== */

void OpenTVInit()
{
    OpenTV[BIT_LEN]        = 9;            // all data that needs to be sent
    OpenTV[BIT_START_H]    = 180;            // the logic high duration of
"OpenTV"
    OpenTV[BIT_START_L]    = 88;            // the logic low duration of "OpenTV"
    OpenTV[BIT_DATA_H]     = 11;            // the logic "long" duration in the
communication
    OpenTV[BIT_DATA_L]     = 33;            // the logic "short" duration in the
communication

    OpenTV[BIT_DATA_LEN]   = 4;            // Number of data which will sent. If
the number is other, you should increase or reduce dtaSend[BIT_DATA+x].
```

```
    OpenTV[BIT_DATA+0]     = 50;            // data that will sent
    OpenTV[BIT_DATA+1]     = 166;
    OpenTV[BIT_DATA+2]     = 80;
    OpenTV[BIT_DATA+3]     = 175;
}

void CloseTVInit()
{
    CloseTV[BIT_LEN]        = 9;            // all data that needs to be sent
    CloseTV[BIT_START_H]    = 178;            // the logic high duration of
"CloseTV"
    CloseTV[BIT_START_L]    = 89;            // the logic low duration of
"CloseTV"
    CloseTV[BIT_DATA_H]     = 10;            // the logic "long" duration in the
communication
    CloseTV[BIT_DATA_L]     = 33;            // the logic "short" duration in the
communication

    CloseTV[BIT_DATA_LEN]   = 4;            // Number of data which will sent. If
the number is other, you should increase or reduce dtaSend[BIT_DATA+x].

    CloseTV[BIT_DATA+0]     = 50;            // data that will sent
    CloseTV[BIT_DATA+1]     = 166;
    CloseTV[BIT_DATA+2]     = 168;
    CloseTV[BIT_DATA+3]     = 87;
}


void setup()
{
    OpenTVInit()
    CloseTVInit()
    Serial.begin(9600);
    speech.begin(9600);
    speech.listen();
}

void loop()
{
    int a=0;

    if(speech.available())
    {
        a = speech.read();    // Read the return value from the Speech Recognizer
        switch (a)
        {
            case 9:                        //  if (return value) then send (IR data)
            IR.Send(OpenTV, 38);
            delay(1000);
            break;
            case 10:
            IR.Send(CloseTV, 38);
            delay(1000);
```

```
                break;
                default:
                break;
            }
        }
    }
```

How to control music

Watch the commands of Speech Recognizer, there're many commands which are related to music, such as "Play music", "Pause", "Stop", "Previous", "Next". So let's do this Speech Music Box!

Download necessary libraries from github: Grove_Serial_MP3_Player_V2.0<ref>library, Grove - MP3 v2.0</ref>, pay attention to the path of your libraries: .../arduino/libraries .

Useful functions about Grove - MP3 v2

There're some useful basic functions of Grove - MP3:

```
PlayPause();    // pause music
PlayResume();   // restart a music
PlayNext();  // next song
PlayPrevious();  // previous song
PlayLoop();  //  loop all songs
SetVolume(uint8_t volume);  // set volume. default value is "0x0E", the range is 0x00 to 0x1E.
IncreaseVolume();  // increase volume
DecreaseVolume();  // decrease volume
```

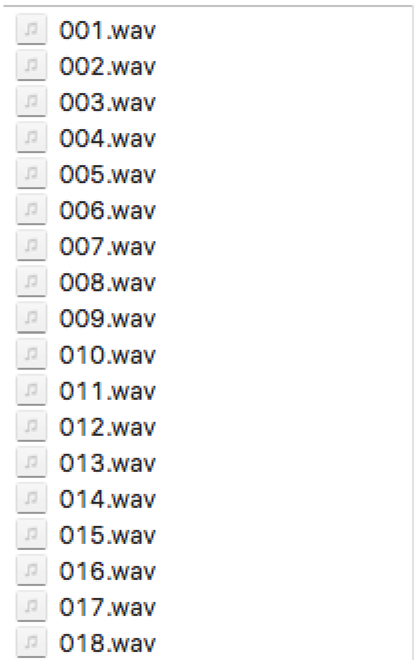
But there're some special functions you need to pay attention:

- SelectPlayerDevice(uint8_t device). *The default device value is 0x02, Select SD card as the player device.*

```
SelectPlayerDevice(0x02);
```

- SpecifyMusicPlay(uint16_t index). *play a song by name.*

  Pay attention to the name of songs, you can set a name like following type:

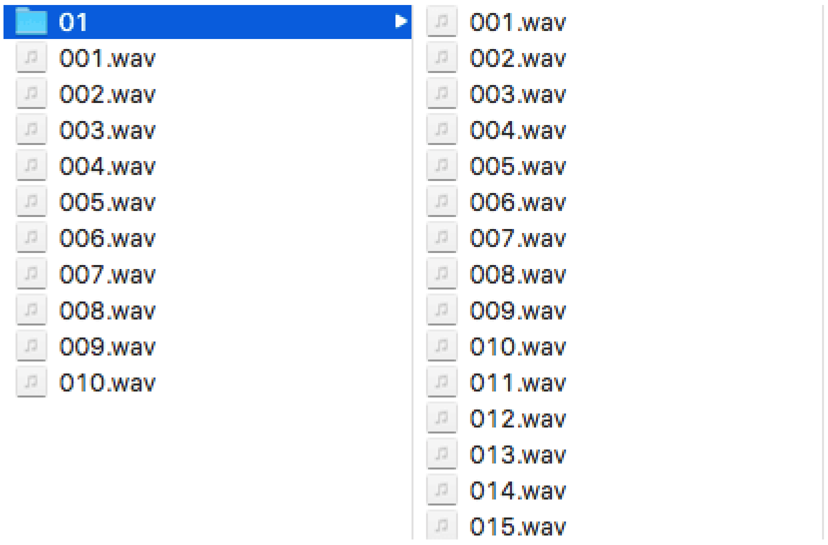

001.wav
002.wav
003.wav
004.wav
005.wav
006.wav
007.wav
008.wav
009.wav
010.wav
011.wav
012.wav
013.wav
014.wav
015.wav
016.wav
017.wav
018.wav

So we can play the song "005" by this function:

```
SpecifyMusicPlay(5);
```

- SpecifyfolderPlay(uint8_t folder, uint8_t index); // play a song by name in a appointed folder

  Furthermore, sometimes we may play music in a new folder, the previous function comes in handy.


01
001.wav
002.wav
003.wav
004.wav
005.wav
006.wav
007.wav
008.wav
009.wav
010.wav

001.wav
002.wav
003.wav
004.wav
005.wav
006.wav
007.wav
008.wav
009.wav
010.wav
011.wav
012.wav
013.wav
014.wav
015.wav

!!!Note The folder index range is 01-99, so the folders' name will only be any number 01 between 99. Pay attention to the number between 1-9, the name of folder should be added to 01-09.

So if we want to play the "005" in the folder "01", we can do this:

```
SpecifyfolderPlay(1,5);
```

- Other attention: Each song has its length of time, so if you want to appoint songs one by one, there've two ways you can choose:

```
delay(length of time);  // delay the length of time until the song is played out

while(QueryPlayStatus() != 0);  //  Return: 0: played out;       1:  not play out
```

Normally, we can use the latter. For example:

```
SpecifyMusicPlay(1);
while(QueryPlayStatus() != 0);
SpecifyMusicPlay(2);
while(QueryPlayStatus() != 0);
SpecifyMusicPlay(3);
while(QueryPlayStatus() != 0);
```

Integrated Program

OK, let's embed Speech Recognizer and Grove - MP3 together. The following demo can control some MP3 functions: play music, pause music, continue music, next song, previous song.

- Copy the program and upload it to your Arduino UNO.
- Plug Base Shield onto the Arduino UNO,and plug the Grove - MP3 v2 onto the D2 port of Base Shield.

```
#include <SoftwareSerial.h>
#include <MP3Player_KT403A.h>

/****** Define the pins of MP3 ******/
SoftwareSerial mp3(2, 3);

/****** Define the pins of Speech Recognizer ******/
#define SOFTSERIAL_RX_PIN  5
#define SOFTSERIAL_TX_PIN  6

SoftwareSerial speech(SOFTSERIAL_RX_PIN,SOFTSERIAL_TX_PIN);


void setup()
{
    mp3.begin(9600);
    speech.begin(9600);
    Serial.begin(9600);
    delay(100);

    SelectPlayerDevice(0x02);       // Select SD card as the player device.
    SetVolume(0x15);                // Set the volume, the range is 0x00 to 0x1E.
}
```

```
void loop()
{
    int a=0;
    if(speech.available())
    {
        a = speech.read();   // Read the return value from the Speech Recognizer
        switch (a)
        {
            case 3:      //  speech command : Play music
            SpecifyMusicPlay(1);   // MP3: play the name of "001"
            break;
            case 4:   //  speech command : Pause
            PlayPause();    // MP3: pause music
            break;
            case 19:   //  speech command : Start
            PlayResume();  // MP3: continue music
            break;
            case 5:   //  speech command : Next
            PlayNext();    // MP3: play next song
            break;
            case 6:   //  speech command : Previous
            PlayPrevious();  // MP3: play previous song
            break;
            default:
            break;
        }

        delay(1000);
    }
}
```

How to broadcast real-time

Did you use Grove - MP3 to broadcast real-time? Let's have a try via Speech Recognizer, Grove - MP3, and Grove - RTC.

Adjust the real-time

Download necessary libraries from github: RTC_DS1307<ref>library, Grove - RTC</ref>, pay attention to the path of your libraries: .../arduino/libraries .

- Open the example file of "SetTimeAndDisplay" : File --> Sketchbook --> libraries --> RTC_DS1307 --> SetTimeAndDisplay.
- Plug Base Shield onto the Arduino UNO,and plug the Grove - RTC onto the I2C of Base Shield.
- Set the right time of RTC.

```
clock.fillByYMD(2016,1,19);//May 23,2016
clock.fillByHMS(15,28,30);//15:28 30"
clock.fillDayOfWeek(Mon);//Saturday
```

- Upload the modified program to your Arduino UNO.

Text-to-speech

As all we known, it has 60 numbers while broadcasting time (0 ~ 59), and we can mouth some words before the MP3 broadcast the time (It's). So we need to add 61 sound files in SD Card.

But there're some tips about SD card you need to pay attention to:

- Form of SD Card: FAT32.
- Format SD Card before deleting any songs; Nothing should to do while adding any songs.
- The sequence of playing is depended on the sequence of song addition in SD Card. *so if we want to play some songs in order, we need to add those songs in SD Card in order.*

By the way, we have ranked 61 sound files in a folder, you can download it and copy it to your SD Card. Of course you may need to format SD Card first.

Name of Sound File	NO. of file in SD Card	Voice Text
000	1th	0
001	2th	1
...	...	...
059	60th	59
060	61th	It's

Integrated Program and broadcast real-time

- Plug Base Shield onto the Arduino UNO; plug Grove - MP3 v2 onto the D2 port of Base shield; plug Grove - Speech Recognizer onto the D5 port of Base Shield; plug Grove - RTC onto the I2C port of Base Shield.
- Copy the following codes on a new sketch of Arduino IDE and upload the program to Arduino UNO.
- Say "HiCell, What's the time" , the MP3 will broadcast the real-time.

```
#include <Wire.h>
#include "DS1307.h"
#include <SoftwareSerial.h>
#include <MP3Player_KT403A.h>

/******* Define the pins of MP3 ******/
SoftwareSerial mp3(2, 3);

/******* Define the pins of Speech Recognizer ******/
#define SOFTSERIAL_RX_PIN  5
```

```
#define SOFTSERIAL_TX_PIN  6

SoftwareSerial speech(SOFTSERIAL_RX_PIN,SOFTSERIAL_TX_PIN);

/******* Define a object of DS1307 class ******/
DS1307 clock;//define a object of DS1307 class

void setup ()
{
  mp3.begin(9600);
  speech.begin(9600);
  clock.begin();
  Serial.begin(9600);
  delay(100);

  SelectPlayerDevice(0x02);       // Select SD card as the player device.
  SetVolume(0x15);   // Set the volume, the range is 0x00 to 0x1E.
}

void loop ()
{
  int a=0;
  speech.listen();   // start to receiver data from the software port of Speech
Recognizer
  if(speech.available())
  {
    a = speech.read();  //  Read the return value from the Speech Recognizer
    if(a==13)
    {
      clock.getTime();   // get the real-time from Grove - RTC
      int b=1+clock.hour;  // get hour data; because the 1th name of song is the
voice "0" , so in order to get the voice "60" (it's) , the number of the name
should be added 1.
      int c=1+clock.minute;  // get hour data; because the 1th name of song is the
voice "0" , so in order to get the voice "60" (it's) , the number of the name
should be added 1.

      mp3.listen();   // start to receiver data from the software port of Grove -
MP3
      SpecifyMusicPlay(61);   // The voice "It's" is the name of "61" song in the
folder of SD card
      while(QueryPlayStatus() != 0);  // play next song before the previous song
is played out
      SpecifyMusicPlay(b);  // play the name of "b" song in the folder of SD card
      while(QueryPlayStatus() != 0);
      SpecifyMusicPlay(c);  //  play the name of "c" song in the folder of SD card
      while(QueryPlayStatus() != 0);
    }
  }
    delay(1000);
}
```

Resource

- Github: IRSendRev
- Github: MP3
- Github: RTC
- Sound files of broadcast

Tech Support

Please submit any technical issue into our forum or drop mail to techsupport@seeed.cc.