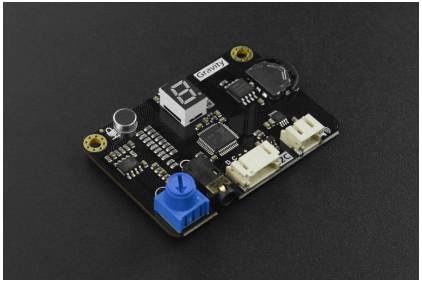
SKU:DFR0699 (https://www.dfrobot.com/product-2359.html)



(https://www.dfrobot.com/product-2359.html)

Introduction

Gravity: I2C Voice Recorder Module Pro is the latest all-in-one voice interaction module from DFRobot. Integrated with recording and playback function, users can directly press the onboard button to record or use a controller with Gravity I2C interface to control it. The module can store 10 segments of 100s audio and comes with built-in power amplifier. The recorded files can be played by the earphone/speaker interface, which allows you to conveniently build voice interaction projects.

Specification

- Compatibility

Development Board	Compatible	Not Compatible	Not Tested

Development Board	Compatible	Not Compatible	Not Tested
Arduino AVR	✔		
micro:bit	✔		
mPython	✔		
Firebeetle-ESP32	✔		

- Recording Function:
  - Can store up to 10 segments of 100s audio
- Speech Synthesis:
  - Synthesis -999999999.999999999 to 999999999.999999999
- Electrical Performance:
  - Operating Voltage: 3.3~5V
- Physical Characteristics:
  - Dimension:
  - Mounting Hole: 4 x M3
- Control Interface: I2C
  - I2C Address: 0x30-0x39
- Output Interface:

- 3.5mm Headphone Port
- PH2.0 Speaker Port

Board Overview

1. Microphone

Please aim the microphone at your mouth when speaking.

2. LED Indicator

   - Off: No recording at the current number
   - Yellow: There is a recording at the current number
   - Red: Is recording
   - Green: Is playing
   - Flashing in red: Is deleting

3. Digital Tube Display

   - Selected recording number
   - Remaining available recording time(in recording)

4. Toggle Switch

   - Control the recording and playing.

5. PH2.0 Speaker Port

   - Connect the small speaker, integrated amplifier output
   - PH2.0 speaker port and 3.5mm headphone port cannot be connected at the same time

- PH2.0 speaker port and 3.5mm headphone port cannot be connected at the same time.

6. I2C Interface
    - Connect to mainboard, power supply, or control via I2C
7. 3.5 Headphone Jack
    - Connect headphones or speakers
8. Potentiometer Knob
    - Adjust the volume

Tutorial

Connection

1. Connect the Gravity-4P I2C interface to the main control board I2C interface. If you don't need I2C control, leave the blue and green wires unconnected.

- Red: 3.3~5V +
- Black: GND -
- Blue: I2C SCL
- Green: I2C SDA

2. Connect the 3.5mm headphone jack or the PH2.0 speaker port, the two cannot be connected at the same time, otherwise the audio will not be played normally.

How to Use

The module function can be easily adjusted by flipping the button for recording and playback.

The module function can be easily adjusted by flipping the button for recording and playback.

Recording

1. Toggle the switch up and down to select the number that can store the recording. For the number with no recording, the LED does not light up.
2. Long press the toggle switch, when the LED turns red, recording starts.
3. The number x10 displayed on the digital tube indicates the remaining available recording time. If it displays 8, it means it can record for about 80 seconds.
4. Any operation of the toggle switch will stop recording.

Playback

1. Flip the switch up and down to select the number with recording. For the number with a recording, the LED shows yellow.
2. Short press the toggle switch, the LED will turn green and start playing
3. Any operation of the toggle switch will stop recording.

Deletion

1. Flip the switch up and down to select the number with recording
2. The number with a recording, the LED is yellow
3. Long press the toggle switch, the LED light flashes red quickly, start deleting
4. After the LED flashes, the deletion is complete, and the current number can be re-recorded

I2C

1. Connect according to the wiring instructions
2. Download **the library file** and install. (About how to install the library? (https://www.arduino.cc/en/Guide/Libraries#.UxU8mdzF9H0))

2. Download the library file and install (About how to install the library: (https://www.ardumotor.com/Guide/Libraries-for-Arduino-IDE.html))

I2C Address Settings

The default I2C address of the recording and playback module is 0x30, which can be manually modified within the range of 0x30-0x39.

1. When power off, press and hold the toggle switch, power on at this time
2. When the LED light is white, it means entering the I2C address setting mode
3. The number displayed by the digital tube is the low bit of the I2C address, such as: 2 means the set address is 0x32; 3 means the set address is 0x33.
4. Dial the toggle switch to the address you want to set, press the toggle switch to save.

I2C Recording/Playback

- Recording

Recording sample code function:

1. Select recording 0
2. Delete recording 0
3. Countdown 3 seconds to start recording
4. Stop after 20 seconds of recording

```
#include "DFRobot_VoiceRecorder.h"

#define I2C_ADDRESS  0x30 // default I2C address 0x30
DFRobot_VoiceRecorder_I2C voicerecorder(&Wire, I2C_ADDRESS);

void setup()
{
  Serial.begin(115200);
  while (voicerecorder.begin() != 0) {
    Serial.println("i2c device number error!");
    delay(1000);
  } Serial.println("i2c connect success!");

  Serial.println("Delect Voice 0");
  voicerecorder.setVoiceNumber(VOICE_NUMBER_0);

  Serial.println("Delete Voice");
  voicerecorder.deleteVoice();

  for (int8_t n = 3; n > 0; n--)
  {
    Serial.println(n);
    delay(1000);
  }

  Serial.println("Recode Start");
  voicerecorder.recordvoiceStart();

  for (int8_t n = 20; n > 0; n--)
  {
```

```
  {
    Serial.println(n);
    delay(1000);
  }


  voicerecorder.recordVoiceEnd();
  Serial.println("Recode End");
}

void loop()
{
}
```

- Playback

Recording sample code function:

1. Start looping from recording 0
2. Skip if there is no recording

```
#include "DFRobot_VoiceRecorder.h"


#define I2C_ADDRESS  (0x30)   //I2C Address
DFRobot_VoiceRecorder_I2C voicerecorder(&Wire, I2C_ADDRESS);

void setup() {
  Serial.begin(115200);

  while (voicerecorder.begin() != 0)
  {
    Serial.println("i2c device number error!");
    delay(1000);
  }
  Serial.println("i2c connect success!");
}

void loop() {
  for (uint8_t voiceNum = 0; voiceNum < 10; voiceNum++)
  {
    voicerecorder.setVoiceNumber(voiceNum);                 //Set number
    Serial.println("setVoiceNumber " + String(voiceNum));

    if (VOICE_NONE == voicerecorder.playVoiceStart())       //Start playing, and check if it is empty
    {
      Serial.println(String(voiceNum) + " is empty");       //Enter the next number if it is empty
      delay(3000);
      continue;
    }
```

```
    Serial.println("VoiceStart");
    while (voicerecorder.getNowState() == VOICE_PLAYING)          //Show remaining available time, exist when ending
    {
      Serial.println("TimeRemaining " + String(voicerecorder.getTimeRemaining()));
      delay(1000);
    }

    delay(3000);

  }
}
```

I2C Speech Synthesis

There are two modes for speech synthesis.

- ICE_SYNTHESIS_MODE

Input -123.987

Output "negative one hundred twenty three nine eight seven"

Can be used to directly read sensor data, measurement parameters, etc.

The valid range of this mode is 9 integers and 9 decimals. If the integer is out of the range, the synthesis will be stopped, and if the decimal is out of the range, the tail will be truncated.

- VOICE_REPLACE_MODE

Input 15071097

Output "one five zero seven one zero nine seven"

Can be used to read student ID, phone number, etc.

The effective length of this mode is 32 characters, non-numeric characters are automatically skipped, and an error will be reported if the character string is too long.

VOICE_SYNTHESIS_MODE Example

Read the input value of A0, output the original value and the calculated voltage value.

```
#include "DFRobot_VoiceRecorder.h"


#define I2C_ADDRESS  (0x30)   //I2C Address
DFRobot_VoiceRecorder_I2C voicerecorder(&Wire, I2C_ADDRESS);

uint16_t AnalogRaw0;
float AnalogVolt0;

void setup()
{
  Serial.begin(115200);

  while (voicerecorder.begin() != 0)
  {
    Serial.println("i2c device number error!");
    delay(1000);
  }
  Serial.println("i2c connect success!");
}

void loop()
{
  AnalogRaw0 = analogRead(A0);
  AnalogVolt0 = 5.0 * AnalogRaw0 / 1024;

  Serial.println(String(AnalogRaw0));
  voicerecorder.VoiceSynthesis(CHINESE_LANGUAGE, String(AnalogRaw0), VOICE_SYNTHESIS_MODE);
  while (voicerecorder.getNowState() != VOICE_NONE);
  delay(1000);
```

```
  Serial.println(String(AnalogVolt0, 2));
  voicerecorder.VoiceSynthesis(CHINESE_LANGUAGE, String(AnalogVolt0, 2), VOICE_SYNTHESIS_MODE);
  while (voicerecorder.getNowState() != VOICE_NONE);
  delay(3000);
}
```

VOICE_REPLACE_MODE Example

Read the Fibonacci sequence out

```
#include "DFRobot_VoiceRecorder.h"

#define I2C_ADDRESS  (0x30)   //I2C Address
DFRobot_VoiceRecorder_I2C voicerecorder(&Wire, I2C_ADDRESS);

uint32_t a[3] = {0, 0, 1};

void setup()
{
  Serial.begin(115200);

  while (voicerecorder.begin() != 0)
  {
    Serial.println("i2c device number error!");
    delay(1000);
  }
  Serial.println("i2c connect success!");
}

void loop()
{
  Serial.println(String(a[2]));
  voicerecorder.VoiceSynthesis(CHINESE_LANGUAGE, String(a[2]), VOICE_REPLACE_MODE);
  a[0] = a[1], a[1] = a[2];
  a[2] = a[0] + a[1];

  while (voicerecorder.getNowState() != VOICE_NONE);
  delay(3000);
}
```

Mind+ Programming

- Click the link to download the Mind+ software (http://mindplus.cc/download-en.html).
- About how to program on Mind+, please refer to https://mindplus.dfrobot.com/microbit (https://mindplus.dfrobot.com/microbit).

Arduino C Reference

```
/**
 * @brief initialization parameters for i2c
 * @return 0 or 1, 0 is i2c begin success, 1 is i2c begin error
 */
uint8_t begin();

/**
 * @brief Set button mode
 * @param BUTTON_MODE_ON   0x00
 * @param BUTTON_MODE_OFF  0x01
 */
void setButtonMode(uint8_t mode);

/**
 * @brief Set light mode
 * @param LIGHT_MODE_OFF   0x00
 * @param LIGHT_MODE_ON    0x01
 */
void setLightMode(uint8_t mode);

/**
 * @brief set voice number
 * @param VOICE_NUMBER_0   0x00
 * @param VOICE_NUMBER_1   0x01
 * @param VOICE_NUMBER_2   0x02
 * @param VOICE_NUMBER_3   0x03
 * @param VOICE_NUMBER_4   0x04
 * @param VOICE_NUMBER_5   0x05
 * @param VOICE_NUMBER_6   0x06
 * @param VOICE_NUMBER_7   0x07
```

```
 * @param VOICE_NUMBER_7   0x07
 * @param VOICE_NUMBER_8   0x08
 * @param VOICE_NUMBER_9   0x09
 * @return state
 *    VOICE_SYNTHESISING   is speech synthesis state
 *    VOICE_PLAYING        is playing state
 *    VOICE_RECORDING      is recording state
 *    VOICE_NONE           is idle condition set number success
 */
uint8_t setVoiceNumber(uint8_t number);

/**
 * @brief get i2c device address
 * @return i2c device address
 */
uint8_t getI2CAddress();

/**
 * @brief get now state
 * @return state
 *    VOICE_SYNTHESISING   is speech synthesis state
 *    VOICE_PLAYING        is playing state
 *    VOICE_RECORDING      is recording state
 *    VOICE_NONE           is idle condition
 */
uint8_t getNowState(void);

/**
 * @brief get Button Mode
 * @return Mode
 *    BUTTON_MODE_ON   0x00
 *    BUTTON_MODE_OFF  0x01
 */
uint8_t getButtonMode(void);

/**
```

```
/**
 * @brief get light Mode
 * @return Mode
 *    LIGHT_MODE_OFF   0x00
 *    BUTTON_MODE_OFF  0x01
 */
uint8_t getButtonMode(void);

/**
 * @brief get voice number
 * @return number
 *    VOICE_NUMBER_0   0x00
 *    VOICE_NUMBER_1   0x01
 *    VOICE_NUMBER_2   0x02
 *    VOICE_NUMBER_3   0x03
 *    VOICE_NUMBER_4   0x04
 *    VOICE_NUMBER_5   0x05
 *    VOICE_NUMBER_6   0x06
 *    VOICE_NUMBER_7   0x07
 *    VOICE_NUMBER_8   0x08
 *    VOICE_NUMBER_9   0x09
 */
uint8_t getVoiceNumber(void);

/**
 * @brief get voice state
 * @return state 0 is Current position has audio 1 is empty
 */
uint8_t getVoiceState(void);

/**
 * @brief get time reamaining
 * @return The time range is 0-100
 */
uint8_t getTimeRemaining(void);
```

```
/**
 * @brief start record
 * @return VOICE_SUCCESS is start record ,VOICE_BUSY is repeat recording or playback,VOICE_NONE is Audio already exists, delete an
 */
uint8_t recordvoiceStart(void);

/**
 * @brief start play
 * @return VOICE_SUCCESS is start play ,VOICE_BUSY is repeat recording or playback,VOICE_NONE is no songs in the current number
 */
uint8_t playVoiceStart(void);

/**
 * @brief delete voice
 * @return VOICE_SUCCESS is delete success ,VOICE_BUSY is repeat recording or playback,VOICE_NONE is no songs in the current numbe
 */
uint8_t deleteVoice(void);

/**
 * @brief End of the tape
 * @return VOICE_SUCCESS is end success ,VOICE_NONE is no begin record
 */
uint8_t recordVoiceEnd(void);

/**
 * @brief End play
 * @return VOICE_SUCCESS is end success ,VOICE_NONE is no begin record
 */
uint8_t playVoiceEnd(void);

/**
 * @brief speech synthesis
 * @param language is  CHINESE_LANGUAGE     0x01
 *                     ENGLISH_LANGUAGE     0x02
```

```
 * @param number range is (999999999 to -999999999)
 * @return VOICE_SUCCESS        is speech synthesis success
 *         VOICE_BUSY           is recording or playing. Please finish recording or playing first
 *         VOICE_SYNTHESISING   is In speech synthesis
 *         DATA_ERROR           is data error
 *         MODE_ERROR           is mode error
 */
uint8_t VoiceSynthesis(uint8_t language ,int64_t number);


/**
 * @brief speech synthesis
 * @param language is  CHINESE_LANGUAGE     0x01
 *                     ENGLISH_LANGUAGE     0x02
 * @param string   is  Input string the scope is determined by the pattern
 * @param mode     is  VOICE_SYNTHESIS_MODE         range (999999999.999999999  to  -999999999.999999999)
 *                     VOICE_REPLACE_MODE           Nine-bit string
 * @return VOICE_SUCCESS        is speech synthesis success
 *         VOICE_BUSY           is recording or playing. Please finish recording or playing first
 *         VOICE_SYNTHESISING   is In speech synthesis
 *         DATA_ERROR           is data error
 *         MODE_ERROR           is mode error
 */
uint8_t VoiceSynthesis(uint8_t language ,String string ,uint8_t mode);
```