MICROPROCESSORS

Voice recorder, player and spectral analysis using Arduino

Coordinator:

Prof. Mircea Giurgiu, PhD

Student:

Bogdan Andreea Laura

Group 2331, eng

Year III

Introduction

For this project, the requirements were to implement a circuit which would have the purpose of detecting and recording the user`s voice, storing the sound, then playing it back and performing a spectral analysis of the sound, using Arduino.

To be able to make this task possible, these are the required components:

- *Arduino UNO board;*
- *ISD1820 sound sensor;*
- *IR sensor;*
- *Speaker (optional).*

To also display the sound signal spectrum, we will need to attach a *32x8 LED matrix* to the circuit, so that when sound is detected the LEDs on each column of the matrix will light up according to the volume of the sound.

Unfortunately, I wasn`t able to implement a fully functional circuit, only a mock-up version of it, due to the fact that certain needed components couldn`t be found in Tinkercad/Proteus, and neither in online libraries, so proper tests and simulations couldn`t be completed. With this being said, I had to focus more on the theoretical side than the practical one, thus this project is mostly research-based.

Theoretical background

1.1 Working principle

Figure 1. ISD1820 [1]

The ISD1820 Voice Recorder Module is based on the ISD1820 IC, which is a single chip Voice recorder IC for single message record and playback. A major feature of the ISD1820 Voice Recorder Module is that is can store the messages in its non-volatile memory and can be configured to store messages of length between 8 Seconds to 20 Seconds. [3]

When the user waves across the IR Sensor, which is a component used to detect movement, the Voice Recorder starts registering the user`s voice.

Then, by feeding audio signal to the LED matrix, it will filter out seven frequency bands centered around 63Hz, 160Hz, 400Hz, 1,000Hz, 2,500Hz, 6,250Hz, and 16,000Hz . The seven frequencies are peak detected and multiplexed to the output to provide a DC representation of the amplitude of each band. These DC values will be read by the microcontroller (in our case the Arduino board) analog input and output the spectrum to the LED Matrix so that it can be displayed on the screens. [2]

1.2 Applications

- Security Systems
- Accident voice recordings
- Record message during collisions [3]
- Various other applications for personal entertainment

Implementation

For the implementation of a voice recorder and player using Arduino I have used the circuit below [4]:

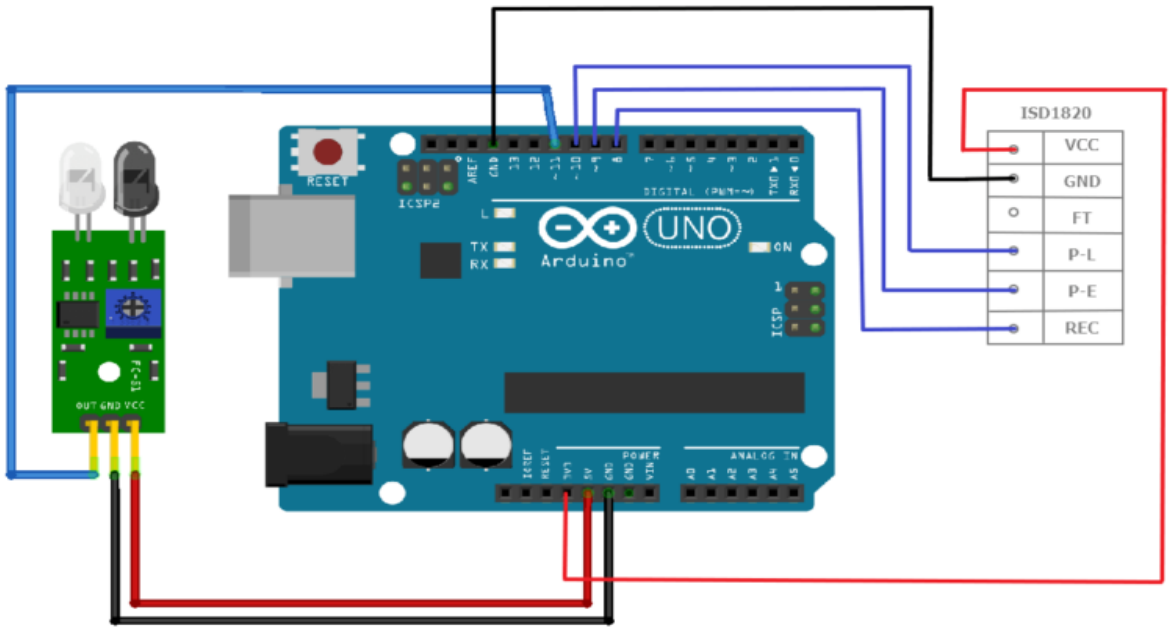


Figure 2. Circuit for voice recorder and player using Arduino UNO

1.3 Used components

Nr. crt	Component name
1.	*Arduino UNO*
2.	*ISD 1820 Voice Recorder*
3.	*IR Sensor*

The ISD1820 voice recorder and player would be able detect and record sounds on it own, but by using a microcontroller such as the Arduino UNO, it makes the process much easier to control and oversee.

1.4 Pin connections

- **For ISD1820:**

ISD1820 Voice Recorder	Arduino
VCC	3.3 V
GND	GND
REC	8
P-E	9
P-L	10

- **For IR sensor:**

IR Sensor	Arduino
VCC	5 V
GND	GND
OUT	11

1.5 Pin workings

- ***VCC***– 3.3V power supply
- ***GND***– Power ground
- ***REC*** – The REC input is an active-HIGH record signal. The module starts recording whenever REC is HIGH. This pin must remain HIGH for the duration of the recording. REC takes precedence over either playback (PLAYL or PLAYE) signal.
- ***PLAYE – Playback***, Edge-activated: When a HIGH-going transition is detected on continues until an End-of-Message (EOM) marker is encountered or the end of the memory space is reached.
- ***PLAYL – Playback***: Level-activated, when this input pin level transits for LOW to HIGH, a playback cycle is initiated.

- ***Speaker Outputs – The SP+ and SP-*** : pins provide direct drive for loudspeakers with impedances as low as 8Ω.
- ***MIC – Microphone Input***: the microphone input transfers its signals to the on-chip preamplifier.
- **FT – Feed Through**: This mode enables the Microphone to drive the speaker directly.
- ***P-E*** – Play the records endlessly.[1]

1.6 How to use it

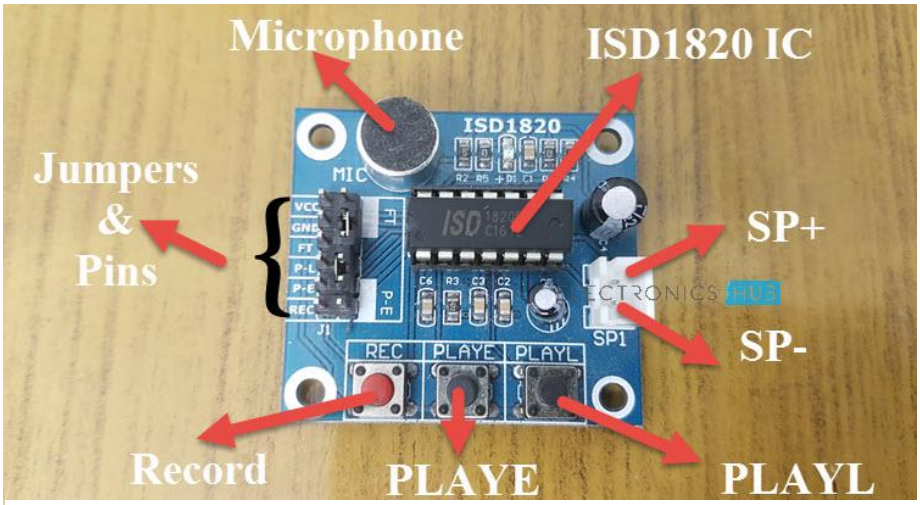


Figure 3. ISD1820-explained [3]

When the user presses the ***REC*** button then the ***RECLED*** will light. In order to keep recording, the REC button must be pressed during the entire duration of the recording. When the user sees fit to end it, the REC button can be released.

In order to enter *PLABACK* mode, which means the sound that was recorded will be played back, the user will need to perform the following actions:

- The button ***PLAYE:*** will need to be pressed by the user only one time, and all of the records will be played until the pre-recorded sound comes to an end.
- The button ***PLAYL***: a playback cycle is initiated. It needs to be pressed continuously, until the user wants to stop the playback record or end it.
- ***P-E mode***: when short P-E jumper the record will playback repeatedly until jumper off or power down.
- ***FT mode***, when short FT jumper, that means that everything that is spoken into the *MIC* will be directly played back to *Speaker*.

When there is no object in front of the ***IR Sensor***, its output is LOW and Arduino does nothing.

When there is an object in front of the ***IR Sensor***, its output becomes HIGH and Arduino then starts recording a message by making the REC Pin HIGH for about 10 Seconds.

3.5. Circuit implemented in Proteus 8 Professional

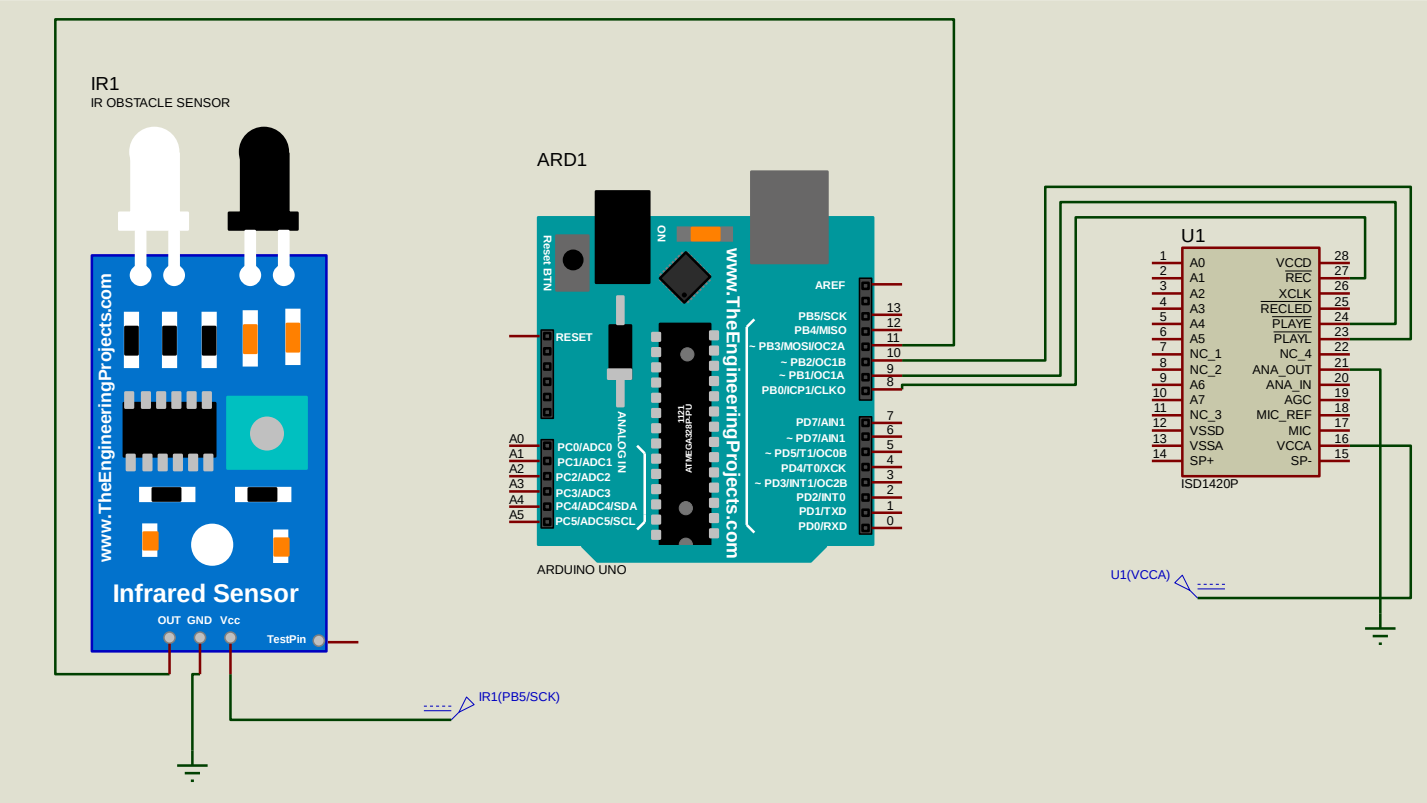


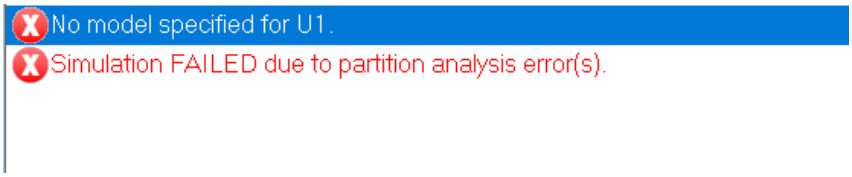
Figure 4. Circuit implemented in Proteus

As it can be observed in the figure above I tried to implement a voice recorder and player using Arduino UNO, in a program called Proteus 8 Professional.

The pin connections are the same as the ones presented and enumerated in *Figure 2 (pg.4).*

Unfortunately, this circuit is not fully functional due to the fact that Proteus does not have a library for the Voice recorder ISD1820, and I couldn`t find one online to import into the program either. The only left option was to use a component similar to the required one, ISD1420P, which has more or less the same functions and specifications as ISD1820, according to their datasheets [5][6].

What makes this circuit unfunctional is the fact that ISD1420P cannot be simulated in Proteus. It`s mostly used as decoration. When trying to run the simulation, the following error is presented:



3.6. Sound spectral analysis

The circuit can be modified in order to display the spectrum of the sound signal.

In order to do this, a *32x8 LED matrix* will need to be attached to the Arduino UNO board, which has and ADC pin, which is use for converting input audio signal into digital samples, so that when sound is detected the LEDs on each column of the matrix will light up according to the frequency of the sound. [6]

By feeding audio signal to the LED matrix, it will filter out seven frequency bands centered around 63Hz, 160Hz, 400Hz, 1,000Hz, 2,500Hz, 6,250Hz, and 16,000Hz . The seven frequencies are peak detected and multiplexed to the output to provide a DC representation of the amplitude of each band. These DC values will be read by the microcontroller (in our case the Arduino board) analog input and output the spectrum to the LED Matrix so that it can be displayed on the screens

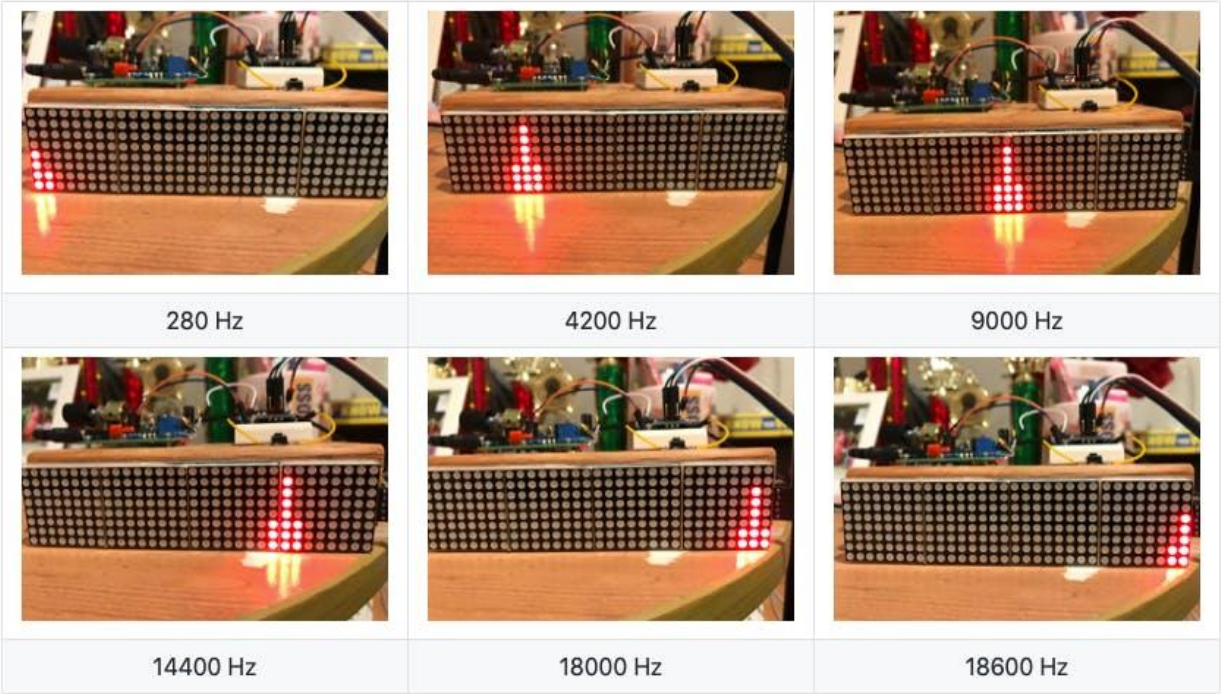


Figure 5.Example of how the audio spectrum analysis would look like at different frequencies [6]

4. Appendix

Code and its explanation [4]

```
#define IR  11     // connects the OUT of the IR sensor to pin 11 of the Arduino board
#define REC 8      // connects the REC pin of the ISD1820 to pin 8 of the Arduino board
#define PLAYE 9    // connects the PLAYE pin of the ISD1820 to pin 9 of the Arduino board
#define PLAYL 10    // connects the PLAYL pin of the ISD1820 to pin 10 of the Arduino board
void setup(){
 pinMode(IR, INPUT);   //defines the IR pin of the IR Sensor as input
 pinMode(REC, OUTPUT);    // defines the REC pin of the ISD1820 as output

 Serial.begin(9600);
}
void loop(){
 int i = digitalRead(IR);
 if(i == 0)
 {
   Serial.println("The user waved!");        //the IR sensor detected motion
   digitalWrite(REC, 1);                // a recording of the voice has started
   delay(10000);                        //it will record for 10 seconds
   digitalWrite(REC, 0);               //it stops recording
    delay(1000);                       //it waits 1 second
    digitalWrite(PLAYE, 1);              // the recording is played back
   delay(10000);
   digitalWrite(PLAYE, 0);              //the recording stops
 }
}
```

5. Conclusion

Taking everything into account, what can be remembered from everything that has been mentrioned in this project is that, it would be quite easy to record and playback a voice or a sound using Arduino Uno and a ISD1820 voice recorder.

This has been a fun and interesting topic to gather information on, and I wish it would've been possible for me to make the circuit fully functional in Proteus or Tinkercad.

As a future project I would like to implement this circuit in real life, including with a sound spectral analyzer, and see how it truly works.

Bibliography

[1] https://create.arduino.cc/projecthub/gadgetprogrammers/diy-auto-voice-record-and-playback-7a47d7

[2] https://duino4projects.com/arduino-based-bi-color-led-matrix-audio-spectrum-visualizer/

[3] https://www.electronicshub.org/interfacing-isd1820-voice-recorder-module-with-arduino/

[4] https://www.factoryforward.com/isd1820-arduino-voice-recordplayback-tutorial/

[5] ISD1820 datasheet https://datasheetspdf.com/pdf/786127/ETC/ISD1820/1

[6] ISD1420P https://www.futurlec.com/Others/ISD1420P.shtml

[7] https://create.arduino.cc/projecthub/shajeeb/32-band-audio-spectrum-visualizer-analyzer-902f51