Smart Music Reactive LED Strip

Yok Jye Tang
School of Electrical and
Computer Engineering
205 Dreese Labs
2015 Neil Ave.
Columbus, OH 43210
Email: tang.1121@osu.edu

Srinivasan Subramaniyan
School of Electrical and
Computer Engineering
205 Dreese Labs
2015 Neil Ave.
Columbus, OH 43210
Email: subramaniyan.4@osu.edu

Johnathon Aurand
School of Electrical and
Computer Engineering
205 Dreese Labs
2015 Neil Ave.
Columbus, OH 43210
Email: aurand.15@osu.edu

Abstract—**For the past few years, smart systems play an important role in our day-to-day life. With ease of availability of smartphones, sensors and actuators, new smart home applications are built on a day-to-day basis. This report briefs the design of a Smart Light-Emitting Diode (LED) strip which reacts to music and has other custom modes. Different user modes have been defined which can be operated with the LED strips. The Arduino microcontroller and the Sound senors are also integrated for the complete design. The work is pure embedded system based application and all the Embedded systems design concepts were used for the implementation at different stages.**

Index Terms—**Smart Home, LED Strip, microcontroller**

I. Introduction

With the Introduction of Google Home Mini and Amazon's Alexa, voice based AI is getting used in many applications. One of the major applications of Internet of Things is a smart home. Several research projects have been proposed in recent times to make smart home applications that turn off the lights, fans and air conditioners. Very few applications have been developed for Music control. Our work is a combination of voice recognition and a music reaction based LED strip through the development of an application in Android to integrate both of them. The LED strip can be configured to various modes, which can be configured using the android app within designed parameters. The LED strip can be controlled by the application directly by using the appropriate selections or through voice commands. The LED Strip can be used on many occasions such as birthdays parties or for family reunions and can be configured to their respective modes. Chan.et.al [7] proposed a voice based smart home application which can control entertainment systems through Alexa. Alexander.et.al [8] used LED strips to automate drivers perception. This helps us to understand how the LED strip can be configured. Meena.et.al [9] develops an application which changes music based on moods. We adopted this idea to introduce different modes in the LED strips. The modes are described in the Android software development section.

The rest of this article is organized as follows. Section II gives the details about the overall system architecture. Section III briefs the technical approaches in the paper. The proposed unified architecture is described in Section IV addresses the experiments and results. Section V concludes the article.

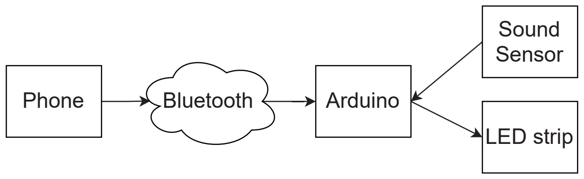


Fig. 1. Block diagram of the smart music reactive LED strip.

II. System Architecture

This section presents the system architecture of our smart music reactive LED strip. The basic setup of the system will be introduced. The actual hardware circuit setup will also be included followed by some discussions and analyses.

Fig. 1 shows the block diagram of our smart music reactive LED strip. In our design, we control the LED strip using the Arduino. The settings of individual LEDs, which include color and brightness, can be changed by the Arduino's output. By using the internal clock of the Arduino, we can further develop more advanced mode settings, such as fading and blinking etc., through adjusting the delays between changing color and brightness. Additionally, the Arduino connects to the sound sensor. The sound sensor reads the sound intensity of the music played on a standalone speaker. Based on the sound intensity readings, the Arduino will change the colors of LEDs according to different sound intensity thresholds. The detail design approaches to set up the modes and sound intensity thresholds is discussed in Section III: Arduino Software Development. The app written for an android device communicates the Arduino via a Bluetooth transceiver.

This project requires multiple hardware components and sensors to achieve the final design goal. It includes: one Arduino Mega board, breadboard, LED strip (50 LEDs), SEN-12642 SparkFun sound sensor, HC-05 Bluetooth transceiver, and wires/cables. Fig 2(a) shows the overall system architecture. The detail of the circuit is shown in Fig. 2(b). The pins of each hardware component are connected to the Arduino's ports using wires. The detail configurations for the LED strip, SEN-12642 and HC-05 will be included in the next section. As shown in Fig 2(a), the Arduino board is powered by USB cable connected to the Laptop. All the hardware components

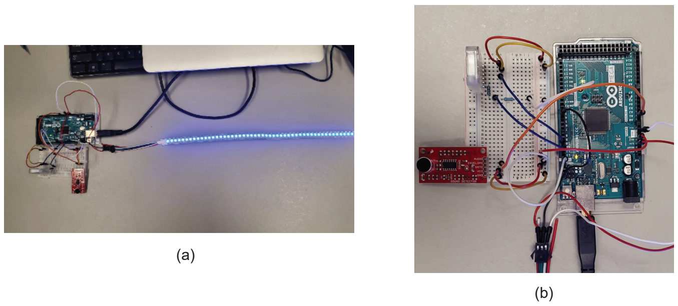


Fig. 2. Hardware circuit of the smart music reactive LED strip. a) overall system architecture; b) detail circuit connections

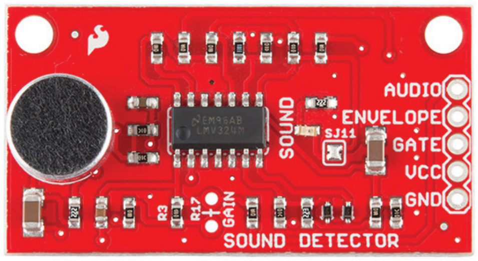


Fig. 3. SEN-12642 SparkFun sound sensor [3].

can be powered by using 3.3-5 voltage inputs.

III. Technical Approaches

The main goal of this project is to design a smart music reactive LED strip, which is able to be controlled by using app or voice commands and, at the same time, can react to music. To achieve the final goal, the tasks that need to be completed include: Arduino setup and sensor integration, voice recognition, Android software development and Arduino software development. This section covers the technical approaches for completing each task.

A. Arduino Setup and sensor Integration

In this sub-section, we discuss three major hardware components that are included in our Arduino setup. They are LED strip (50 LEDs), SEN-12642 SparkFun sound sensor, HC-05 Bluetooth transceiver. The detailed approaches to use/control these components are discussed as follows.

1) LED strip: In this project, we use WS2812B individually addressable Red Green Blue (RGB) LED strip. The LED strip consists of 50 LEDs [1]. Each LED has its own setting and can be adjusted individually. The detail of the LED strip can be found in Fig. 2(a). It consists of three wire connections: VCC, GND, DATA. The VCC and GND are connected to the 3.3/5 V pin and the ground pin of the Arduino board, respectively. The data wire is connected to one of the digital pins of the Arduino board and it is configured as an 'OUTPUT'. By generating different digital output signal to the data pin, we can control the color and the brightness of each LED in the LED strip.

To program the LED strip on Arduino board, we employ the 'FastLED' library [2]. An 'CRGB' type array with 50 entries is created. This array needs to be initialized using the 'FastLED.addLeds' function, which configures the pin number of the Arduino board that connects to the LED strip and the total number of LEDs in the strip. To set the RGB value of the i-th LED, the i-th entry of the array can be programmed by using the 'CRGB(R,G,B)' function. The function takes three inputs: Red (R), Green (G), Blue (B). Each input is an integer ranging from 0 to 255. Assume the array is named as 'LEDs'. To set the color of the 5-th LED to violet, the programming is done as 'LEDs[4] = CRGB(143,0,255)'. Besides, we can also set the brightness of the LED by adjusting the input value of the 'maximizeBrightness(brightness)' object of the 'CRGB' type. The input of this object is also an integer value and it ranges from 0 to 255, where '255' is the maximum brightness and it is set as default. Takes the setting of 5-th LED as the same example. If we want to lower down the brightness of the 5-th LED to around 50%, the programming is done as 'LEDs[4].maximizeBrightness(128)'. To allow the LED strip response to the settings that have been made to each LED, we need to add the 'FastLED.show()' function.

In summary, to control the LED strip, three major functions are used: 'CRGB(R,G,B)', 'maximizeBrightness(brightness)' and 'FastLED.show()'. The first two functions control the RGB value and brightness of each LED, respectively. The last function makes the LED strip responses to the new settings that haven been made to each LED. Without adding the 'FastLED.show()' function, the LED strip will not have any response. By adjusting the number of LEDs we change and/or inserting delays, we can develop different modes, such as Running, Fading, Blinking, etc.

2) SEN-12642 SparkFun sound sensor: Figure 3 shows the layout of the SparkFun sound sensor. The sound intensity value of the music is captured by the microphone located at the left. The sound sensor consists of 5 pins. To able to read the sound intensity, we just need 3 of them: VCC, GND and Envelope. The Envelope pin is connected to one of the analog pins of the Arduino board. The analog pin does not need to be initialized. The sound intensity value can be directly collected by using the 'analogRead(SOUND_SENSOR_PIN)' function [3], where 'SOUND_SENSOR_PIN' denotes the pin number of the sound sensor. The sound intensity collected by using this method can be read as an integer value. Then, by assigning different thresholds, which will be detailed in the Arduino software development section, the RGB value of each LED can be changed accordingly.

3) HC-05 Bluetooth transceiver: To communicate with the Arduino via the Bluetooth, a Bluetooth transceiver is needed. For this project, we use the HC-05 Bluetooth transceiver [4]. The transceiver consists of 6 pins. The VCC and GND are connected to the 3.3/5 V and GND pins, respectively, of the Arduino. The receiver (RX) pin is connected to the transmitter (TX) pin of the Arduino through the voltage divider, and the TX pin is directly connected to RX pin of the Arduino. Note that the pins to be used as TX and RX in the Arduino are

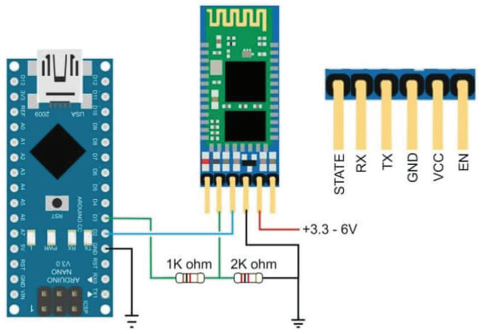


Fig. 4. HC-05 Bluetooth transceiver [4].

Algorithm 1 Read Bluetooth data in the Arduino

```
1: while(BTSerial.available()≤0) ;
2: data = BTSerial.read();
```

chosen by the user. Before starting the Arduino programming, the HC-05 needs to be configured in Slave mode. This is because, in our design, the Arduino is controlled by the phone. The steps for the Slave configuration can be found in [5]. The enable (EN) pin needs to be set high during the configuration. The configuration is carried out through the Serial Monitor by selecting “Both NL and CR” and set the baud rate to “38400”.

After the Slave configuration is done, the Arduino can receive the data sent by the phone using the HC-05 transceiver. The phone needs to first pair with the HC-05. After the Bluetooth connection is made, the phone can send the data to the Arduino. The Android codes of establishing the Bluetooth connecting and send/receive data have been taught in this class and will not be repeated in this report. To read the received data on the Arduino, the ‘SoftwareSerial.h’ library needs to be employed. A Bluetooth serial variable is initialized as ‘SoftwareSerial BTSerial(RX, TX)’, where ‘BTSerial’ is the variable name and the initialization needs to include the RX and TX pin numbers of the Arduino. Then, in the main function, the data sent by the phone can be assessed by using two lines of codes in the Algorithm 1. The first line of code is added to keep checking until the incoming data is received. If a byte of data is received, it will move to the next line. The value of the received data can be read by using ‘.read()’ and assigned to an integer ‘data’ variable [6].

B. Voice Recognition

The voice recognition task can be broken down into multiple stages, as illustrated in the Figure 5. As soon as the record button is clicked, it goes through the following stages. The permissions are already initialized in the manifest files for recording audio and to use the microphone. The input taken from the microphone is converted to text. The Android speech recognizer class [10] is used for this purpose. The converted texted is passed to the keyword spotting module. Keyword spotting can be implemented using ML model for real-time speech analysis of complex sentences. However, since Voice recognition of smart music LED reactive control only contains

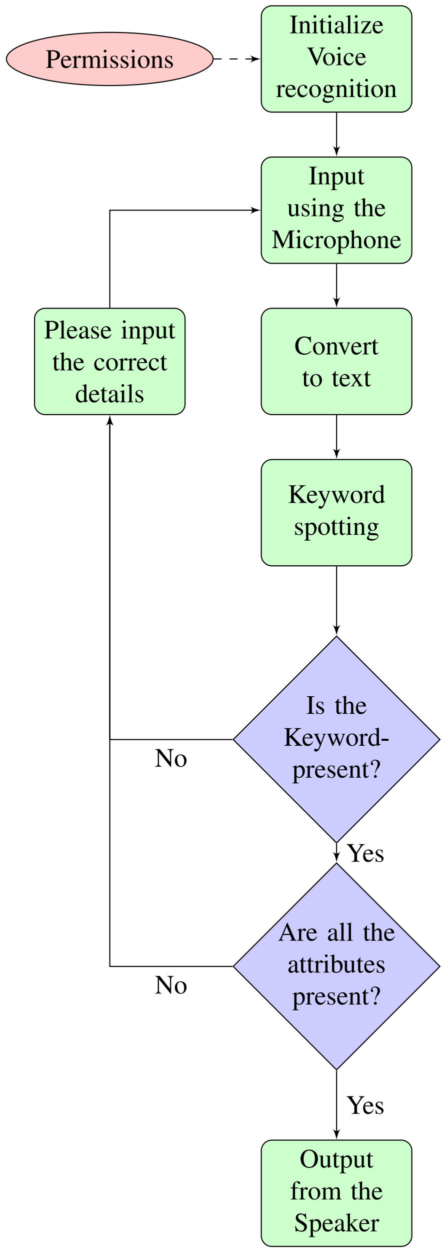

Permissions
Initialize Voice recognition
Input using the Microphone
Convert to text
Keyword spotting
Please input the correct details
Is the Keyword-present?
No
Yes
Are all the attributes present?
No
Yes
Output from the Speaker

Fig. 5. Flowchart for Voice Recognition

a few commands, as described in Table I, a dictionary is for the implementation. The key value pair was used for the corresponding keywords. The key for a particular voice command was transmitted via Bluetooth to the Arduino board, which is connects the LED strip.

The speech recognition has two layers of abstraction, since some times partial keywords can lead to incorrect detection. For example, if an input command is given as turn on the power button, the application would send a speech output stating it is an incorrect command. The two levels of abstraction makes this possible. The first keyword spotting would identify the word ”On ” and it would perform the second layer of searching. Since the keyword ”LED” is absent, the voice

TABLE I
VOICE COMMAND MODES

Mode	Command
1	Turn on the LED
2	Turn off the LED
3	Set the brightness
4	Change the Color
5	Change to Music Reactive Mode

recognition does not go through. If the input was ”change the color” the output toasted would be ”Please specify a valid color after going through the documentation. This is only possible since two layers of hash mapping was used for the voice recognition purpose. The text2speech [11] is also simultaneously implemented, which plays the toast message as the output in the speaker.

C. Android Software Development

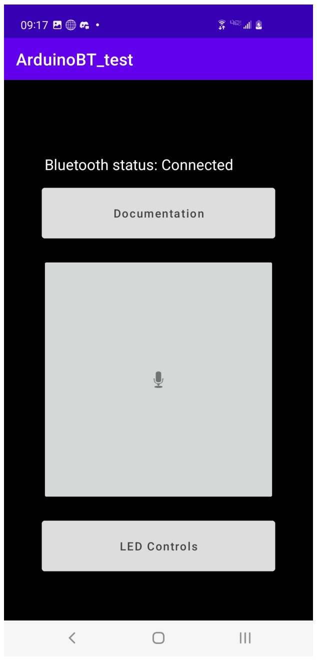
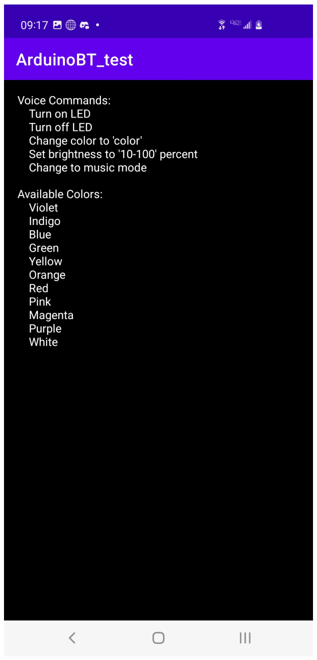


Fig. 6. Main Screen



Voice Commands:
Turn on LED
Turn off LED
Change color to 'color'
Set brightness to '10-100' percent
Change to music mode

Available Colors:
Violet
Indigo
Blue
Green
Yellow
Orange
Red
Pink
Magenta
Purple
White

Fig. 7. Documentation Screen

The Main Screen, shown in Figure 6, is designed to be able to easily activate the voice commands without the need to look at the screen. This is accomplished by having the size of the voice commands button be large and centered on the screen. Additionally, the Main Screen contains a button to take the user to the Documentation Screen, shown in Figure 7, which shows the list of available voice commands, and a button to take the user to the on-screen LED Controls, shown in Figure 8, which contains buttons to turn on and off the LED strip from the screen and buttons which takes the user to sub-menus that allow further customization.

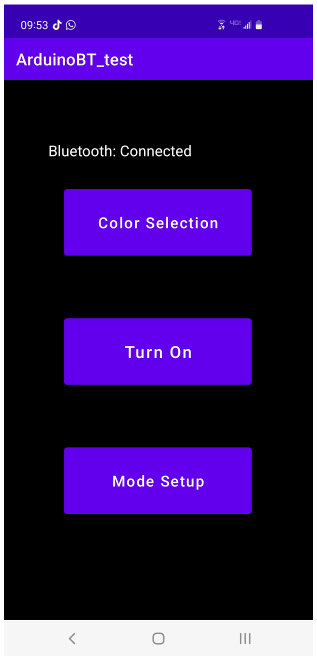
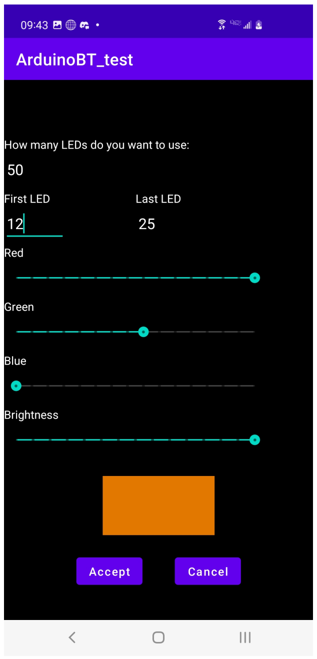


Fig. 8. LED Controls Screen



How many LEDs do you want to use:
50
First LED 12
Last LED 25

Fig. 9. Color Selection Screen

By clicking the Color Selection button on the LED Controls Screen, the user is directed to the Color Selection Screen shown in Figure 9. This screen contains the following:

- A text box to enter the number of LEDs to use on the strip
- A text box to enter the first LED to change color
- A text box to enter the last LED to change color
- A slider bar to change the amount of Red in the color
- A slider bar to change the amount of Blue in the color
- A slider bar to change the amount of Green in the color
- A slider bar to change the brightness of the LEDs
- An area to show the approximate color determined from the slider bars
- A button to send the settings to the Arduino board
- A button to return the user to the previous screen

Also on LED Controls Screen is the Mode Setup button. This takes the user to the Mode Screen which defaults to no selected mode, shown in Figure 10.

Modes: This screen contains a drop-down selector to choose customization options for the different modes. This was done since only one mode can operate at a time. There are four different modes:

- Music Reactive Mode, shown in Figure 11
- Running Mode, shown in Figure 12
- Fading Mode, shown in Figure 13
- Blinking Mode, shown in Figure 14

Each of the modes contain an Activate/Deactivate toggle switch, which will deactivate the toggles on the other modes when taken to Activate on one mode, and a Submit button,

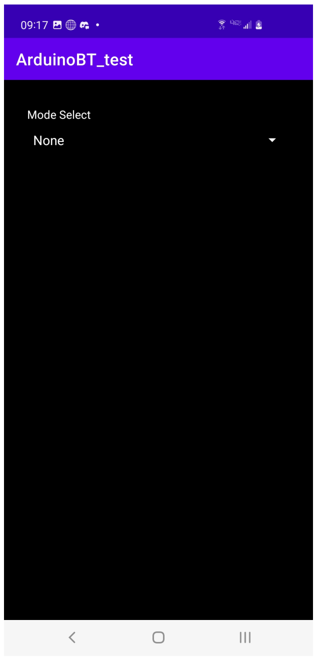


Fig. 10. No Selected Mode

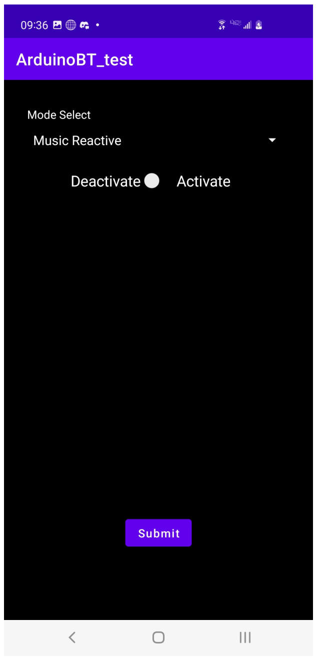


Fig. 11. Music Reactive Mode

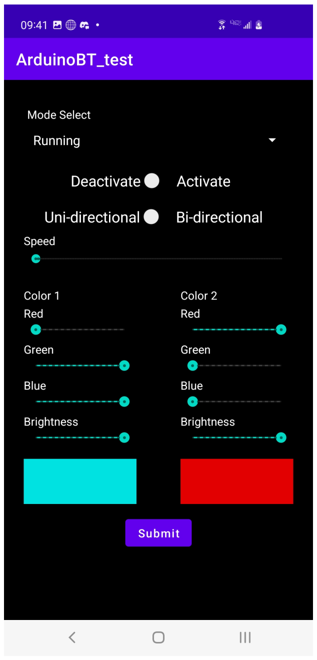


Fig. 12. Running Mode

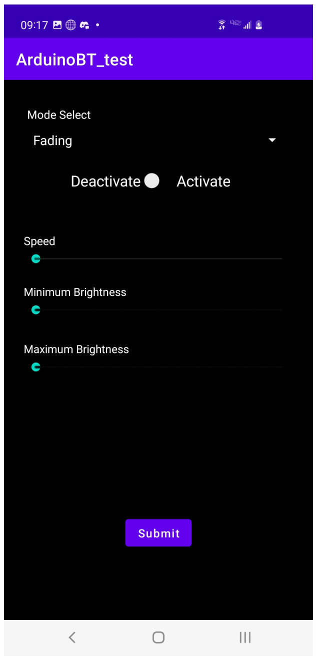


Fig. 13. Fading Mode

which will send the inputted data for the mode to the Arduino board.

The Music Reactive Mode causes the LEDs to change colors based on the intensity of the sound that the sound sensor detects.

The Running Mode causes the LED strip to change between two different colors by sequentially changing one LED at a time with a user set delay between each LED until all the LEDs are the first selected color then doing it again until all the LEDs are the second selected color. Running Mode contains a Uni-directional/Bi-directional toggle switch which controls the direction of the second color. In either Uni-directional or Bi-directional mode the first color will travel from the start of the strip to the end. In Uni-directional mode the second color will also travel from the start of the strip to the end, starting when the first color reaches the end. In Bi-directional mode the second color will travel from the end of the strip to the beginning, starting when the first color reaches the end of the strip. Running mode also contains slider bars to select the first and second colors by controlling the amount of red, green, or blue and the brightness of each color. It also has an area under first and second colors slider bars that shows the approximate color selected.

The Fading Mode changes the brightness of the LEDs from a user set minimum brightness to a user set maximum brightness and back again continuously. Each cycle is controlled by a user set speed parameter.

Finally, Blinking Mode alternates between two different brightness levels at a user set speed. Both the Fading Mode and Blinking Mode contain a slider bar the control the speed at which each cycle takes and slider bars to control the minimum

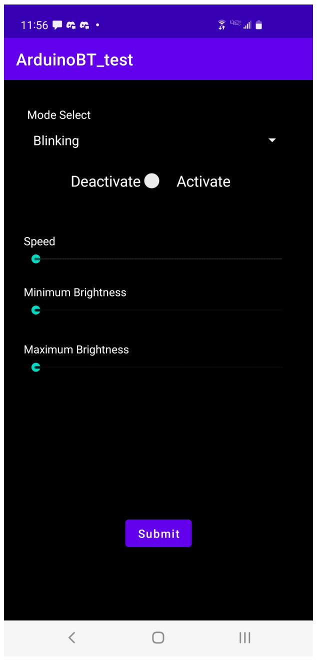


Fig. 14. Blinking Mode

and maximum brightness of the LED strip in each cycle.

When all the modes are deactivated the LED strip will the the most recently inputted color setup from the Color Selection Screen, or a default blue color if no colors were selected from the screen.

D. *Arduino Software Development*

The Arduino code needs to be able to accept commands from the Bluetooth device, receive the sound intensity from the sound sensor, and control the LED strip. In order to accomplish these tasks the Arduino code runs in an infinite loop with two main sections, Communications and Mode Control.

Communications: The first thing the communication section does is check if there is any incoming data. If there isn't any data, the section is skipped. If there is data, that data element is expected to be an operation code (OP code) by design. From this point the code splits into different section based on the OP code. Each of sections will then read any further data it expects to receive for that OP code. The OP codes used and data bytes read are shown in Table II. After the bytes are read, arrays are updated for the Mode Control section to use in controlling the LED strip. Turning on or off the LEDs and changing the color don't need any updates so they are handled immediately.

TABLE II
COMMUNICATION CODES

Op Code	Effect	Data Elements
0	Turn off LEDs	No further Data
1	Turn on LEDs	No further Data
2	Change Color	RRRRBBBB GGGGIIII Number of LEDs to use First LED to change Last LED to change
3	Music Reactive Mode	XXXXXXXA
4	Running Mode	XXXXXXDA Color 1: RRRRGGGG Color 1: BBBBIIII Color 2: RRRRGGGG Color 2: BBBBIIII XSSSSSSS
5	Fading Mode	SSSSSSSA Maximum: IIIIIIII Minimum: IIIIIIII
6	Fading Mode	SSSSSSSA Maximum: IIIIIIII Minimum: IIIIIIII

Where: R bits are red, B bits are blue, G bits are green, I bits are brightness, A bit is an activation code (0 for off and 1 for on), D bit is a direction code (uni-directional/bi-directional), S bits are speed (which changes delay), and X bits are ignored

Mode Control: The second part of the Arduino code controls the modes. First it receives the desired mode from the Communication Section. Then it splits into different sections depending on the mode selected. For Music Reactive Mode, it reads the sound intensity value coming from the sound sensor. Then it changes the colors based on the value received, as shown in Table III. For Running Mode, it adjusts each LED one at a time with a delay in between set by the speed setting. For Fading Mode, it adjusts the brightness slowly from the minimum to the maximum setting with a delay between each change. Finally for Blinking Mode, it alternates between the minimum and maximum brightness settings with a delay.

TABLE III
SOUND SENSOR THRESHOLDS

Range	Effect
[0-50]	No Change from Normal
(50-80]	(Red+50) Modulus 256
(80-110]	(Green+50) Modulus 256
(110-140]	(Blue+50) Modulus 256
(140-170]	(Red+100) Modulus 256 (Green+100) Modulus 256
(170-200]	(Red+100) Modulus 256 (Blue+100) Modulus 256
(200-260]	(Green+100) Modulus 256 (Blue+100) Modulus 256
>260	(Red+200) Modulus 256 (Green+200) Modulus 256 (Blue+200) Modulus 256

IV. EXPERIMENTS AND RESULTS

This section presents the experiments that we have done for this project. Our design consists of three main functionalities: turn on/off the LED strip, LED strip customization, LED modes (Music Reactive, Running, Fading, Blinking), voice commands. These functionalities are tested.

To set up the experiments, the procedure below is followed:

1) Integrate all hardware components as in Fig. 2.
2) Upload the Arduino code to the Arduino board.
3) Download the Android app developed by our group.
4) Start testing.

The results of each testing are detailed and analyzed as below:

A. *Turn on/off the LED strip*

Turning on/off the LED strip can be done by pressing the 'Turn on/off' button in LED control screen. The testing of this functionality was carried out in two different ways.

In the first testing, we turned on the LED strip immediately by pressing the 'Turn on/off' button after we opened the app. Since the LED strip was not previously set to any color, the color of the LED strip turned into blue (default color). After we pressed the button again, the LED strip was turned off.

In the second testing, we first turned on the LED strip and set the LED strip to a random color. Then we turned off the LED strip. After that, we turned on the LED strip again. At this time, the color of LED strip was what we set before.

From the two testings carried out above, we can ensure that the response of the LED strip to this functionality is what we were expected.

B. *LED strip customization*

In our design, we able to set the color and/or brightness of each LED in the LED strip. Fig. 15 shows one of the LED customization examples in our design. It can be observed that the LED strip can consist of three different colors. The first 14 LEDs are set to Red color (R:255, G:0, B:0). The (15-25)-th LED is set to Green color (R:0, G:255, B:0). The

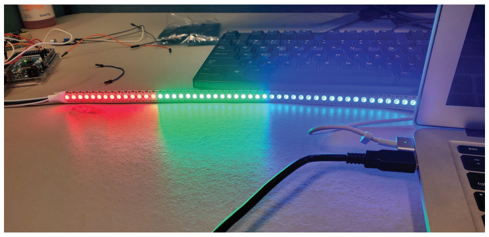


Fig. 15. LED customization example.

last 25 LEDs are set to Blue color (R:0, G:0, B:255). The brightness of each LED is set to 70%. Although the LED can be customized as expected, some colors are not perfectly displayed. For instance, as shown in Fig. 15, the color of the middle LEDs is not a perfect Aqua color. This is because the LED strip put more stronger effect on the Green color over the Blue one. To solve this problem, we need to purchase a much higher quality LED strip.

C. LED modes

For our design, four LED modes are developed: 1) Music Reactive mode; 2) Running mode; 3) Fading mode; 4) Blinking mode. Each mode is tested independently.

For each mode, the testing is carried out under two scenarios. In the first scenario, we activate the mode immediately after opens the app. In this case, since we have not set any color yet, the color of the LED strip is blue color (default) and any mode effect is carried out based on this color. When the mode is deactivated, the LED strip turns into solid blue color. In the second scenario, we first customize the color of the LEDs and then activates the mode. In this case, since we have set the color, the mode effect is carried out on top of the customized color. When the mode is deactivated, the LED strip turns back into the customized color without any effect.

In addition, the results of each mode testing is discussed and analyzed in more details as follows:

1) Music Reactive mode
   To test this mode, we played random music on the phone and place the phone's speaker near to microphone of the sound sensor. After we moved the toggle switch to 'activate' and press the 'submit' button, the color of the LED strip was changed according to the sound intensity of the music. If the music is louder, the color of the LED strip will change more aggressively. Then, after we moved the toggle switch to 'deactivate' and press the 'submit' button, the color of the LED strip turns into solid default/customized color. In our experiment, this mode worked as expected. However, we realize that we need place the speaker near to microphone of the sound sensor in order to get a satisfied result. To improve the range, the sensitivity of the sound sensor needs to be further increased. This can be done by replacing the current sound sensor to a more advanced one.
2) Running mode
   In this mode, we tested two 'running' directions: Uni-directional and Bi-directional. For the Uni-directional, the first color appears in sequence from one end to the other. When the first color reaches the last LED, the second color will repeat the same effect. For the Bi-directional, the first color also appears in sequence and starts from one end to the other. When the first color reaches the last LED, the second color will repeat the same effect but in the different direction. The RGB values of those two colors are set independently together with its own brightness. The 'running' speed, which determine how fast can a color runs from one end to the other, was set to half. After we moved the toggle switch to 'activate' and press the 'submit' button, the 'running' effect of both directions worked as expected. However, when doing the experiment of this mode, the delays between the color changes of each LED, which is the speed, needs to be fined tuned. If the delay is too fast, the 'running' effect can not be distinguished and it is just a solid color. If the delay is too slow, the Arduino can not response to the new input command sent by the app. Through try and error, we realized that the ranges of the delay should be set between 25 and 150 milliseconds.
3) Fading/blinking mode
   The fading and blinking modes were tested in the same format. For the fading mode, the brightness is changed gradually. For the blinking mode, the brightness is changed immediately. The results of testing these two modes were turned out as expected. The delay corresponding to the speed was also fined turned. The delay setting used in the Running mode is employed by these two modes.

D. Voice commands

To test out the voice command, we pressed the voice command button in the main screen and say the words that initiates the commands. In this project, we only consider four voice commands: turn on/off the LED strip, change the color of the LED strip, change the brightness of the LED strip, and activate/deactivate the Music Reactive mode.

To make sure everything is setup correctly, the voice command was tested in two scenarios. In the first scenario, we say the correct words as documented in the Documentation Screen and see whether if the LED strip can response to our command. In the second scenario, we purposely say the wrong words and see whether if the wrong command can be detected.

The details of using the correct and wrong commands to initiate each functionality are included as follows:

- Turn on/off the LED strip
  To initiate this functionality, we needs to say: "turn on the LED". In the experiment, it worked perfectly. When we only said the word 'on', the following response was received: "Please enter a valid command". The same results are obtained when turning of the LED strip.
- Change the color of the LED strip
  To initiate this functionality, we needs to say: "change color to" any specific color. The available colors in our

library are listed in the Documentation Screen. This command has been tested and it worked perfectly. We also tested the command by saying: ”change the color”. Since we didn’t specify any specific color, the following response was received: ”Please go through the document and provides the available color”.

- Change the brightness of the LED strip
  To test this command, we said: ”set the brightness to twenty percent”. As result, the LED strip turns dimmer. Then we said: ”set the brightness to hundred percent”. After that, the LED strip become brightener. Besides, we also tested the wrong command by saying: ”set the brightness”. Since we didn’t mention the percentage of the brightness, the following response was received: ”Please mention how much do you want to set the brightness to”.
- Activate/deactivate the Music Reactive mode
  To test this command, we said: ”change to music reactive mode”. As result, the Music Reactive mode was initiated immediately. When we play the music near to the sound sensor’s microphone, the color of the LED strip changes.

V. Conclusion and Future work

In this project, we proposes a smart music reactive LED strip, which is able to be controlled by using app or voice and, at the same time, can react to the music. All the tasks that we have planned to include in this project have been successfully completed. They include: turn on/off the LED strip, LED strip customization, Music Reactive mode, command the LED strip using the voice commands. Besides, we have also developed a few LED modes for the user to play around. All of these functionalities can done by using our app. The layout of the app is clean and the functionality of each button/switch is straightforward to the user.

For the future works, more functionalities can be added to our design. For instances, adding more LED modes and allow the voice recognition to detect more complex user’s commands. Besides, buying a higher quality LED strip can also improve the color accuracy of the LED strip.

Acknowledgment

The authors would like to thank the professor, Dr. Wang, for the guidance offered, the classmates for their feedback during the presentation and the TA, Xianhao, for his valuable comments and feedback.

References

[1] Amazon. Retrieved April 30, 2022, from https://www.amazon.com/dp/B08H26FFK5/ref=twister_B08H27HQYF?_encoding=UTF8&amp;th=1.

[2] Fastled animation library. FastLED LED animation library for Arduino. (n.d.). Retrieved April 30, 2022, from https://fastled.io/.

[3] Sound detector hookup guide. (n.d.). Retrieved April 30, 2022, from https://learn.sparkfun.com/tutorials/sound-detector-hookup-guide.

[4] Amazon. Retrieved April 30, 2022, from https://www.amazon.com/dp/B01G9KSAF6?psc=1&ref=ppx_yo2ov_dt_b_product_details

[5] Dejan, (2022, February 17). How to configure and pair two HC-05 bluetooth modules as master and slave: At commands. How To Mechatronics. Retrieved April 30, 2022, from https://howtomechatronics.com/tutorials/arduino/how-to-configure-pair-two-hc-05-bluetooth-module-master-slave-commands/.

[6] Dejan, (2022, February 18). Arduino and HC-05 Bluetooth Module Complete tutorial. How To Mechatronics. Retrieved April 30, 2022, from https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/.

[7] Yue, C. Z., & Ping, S. (2017, April). Voice activated smart home design and implementation. In 2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST) (pp. 489-492). IEEE.

[8] Meschtscherjakov, A., Döttlinger, C., Kaiser, T., & Tscheligi, M. (2020, April). Chase Lights in the Peripheral View: How the Design of Moving Patterns on an LED Strip Influences the Perception of Speed in an Automotive Context. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (pp. 1-9).

[9] Talele, Meena, et al. ”SMART MUSIC PLAYER USING MOOD DETECTION.”

[10] Android Speech recognizer, https://developer.android.com/reference/android/speech/SpeechRecognizer

[11] TextToSpeech https://developer.android.com/reference/android/speech/tts/TextToSpeech