Lenovo Educational Board Advanced Manual

Contents
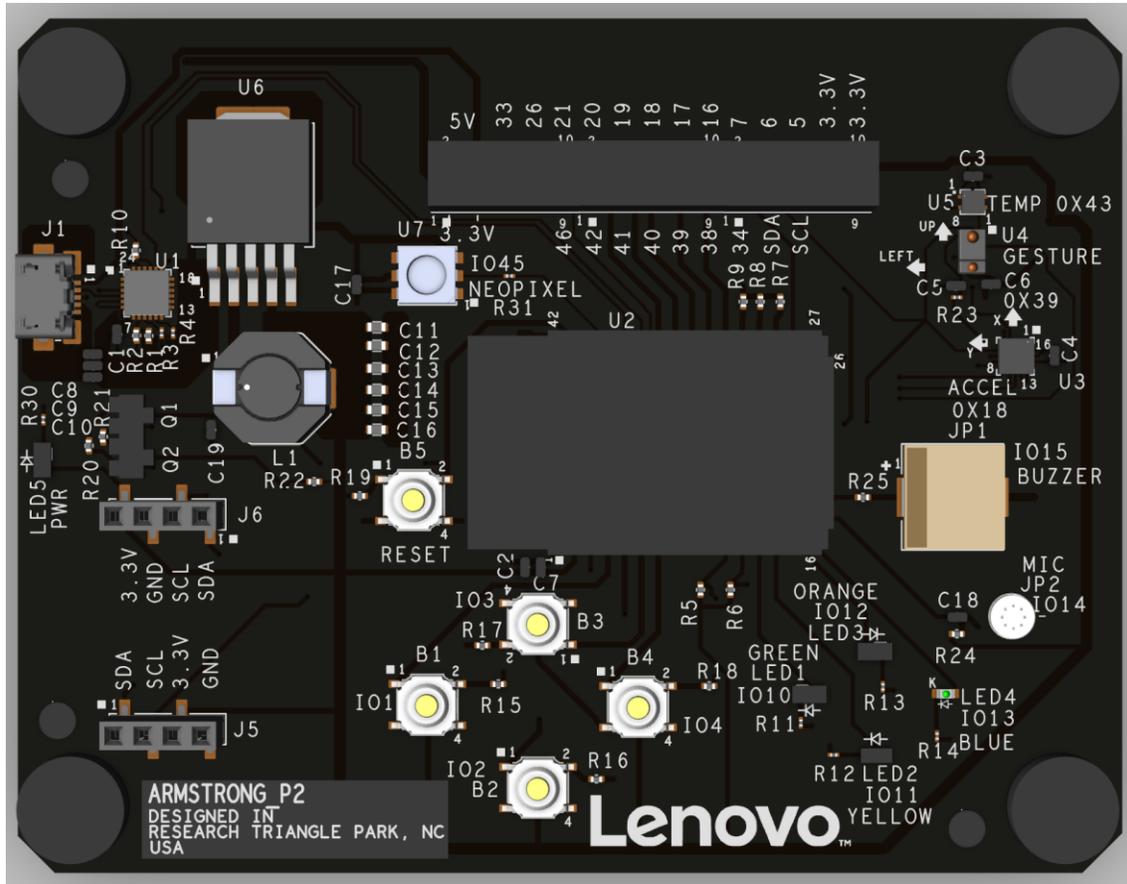
Components Schematic

Reference ID	Component	Function
B1	Button 1	Input button
B2	Button 2	Input button
B3	Button 3	Input button
B4	Button 4	Input button
B5	Button 5	Reset
IO10	LED 1	Green LED Output
IO11	LED 2	Yellow LED Output
IO12	LED 3	Orange LED Output
IO13	LED 4	Blue LED Output
IO14	Microphone	Audio Input/Output
IO15	Buzzer	Audio Output
IO45	Neopixel	LED Output
J1	Port 1	USB port
L1		Inductor
U1		
U2	CPU	
U3	Accelerometer	
U4	Gesture Sensor	
U5	Temperature Sensor	
U6		
U7	Neopixel	

Lesson A – 1: How's The Weather?

OVERVIEW

In this lesson we will:

- Build our own weather station

- Utilize the temperature, light, and humidity sensor to report weather conditions to the Serial Monitor

Teacher Guide

In this lesson, students will be interacting with the ambient light, temperature, and humidity sensors. We learned about all of these sensors in previously lessons, but in this lesson, we will use them together. Once the code is uploaded, we suggest bringing your students outside if it is not raining to see the difference in temperature, humidity, and ambient light compared to that in the classroom.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?
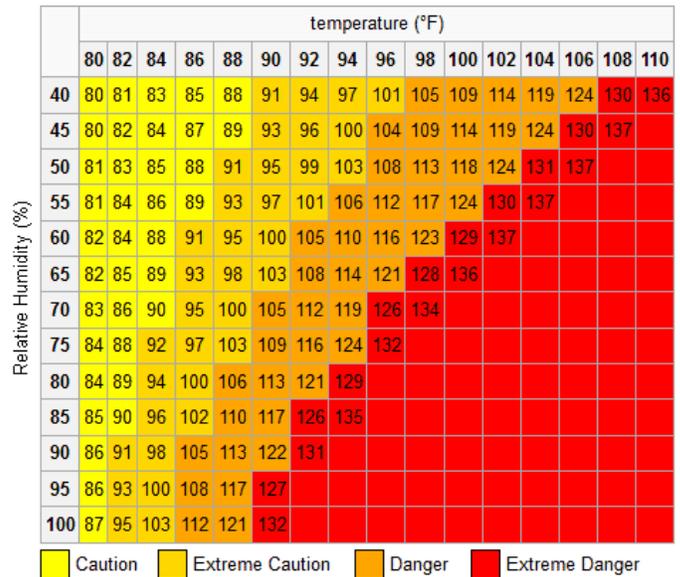
3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

In previous lessons, we have learned about the temperature and ambient light sensors. In this lesson, we will start using the humidity data that is also available from the temperature sensor. While temperature and humidity data are useful, we can use ambient light to draw some predictions into current weather conditions. Using this information, we can make our best prediction of the current weather. However, to take your weather station a step further, you can include heat index and dew point. This metrics will make your weather station have more data as these can be calculated using chart and more if statements in the loop() function. Additionally, we recommend using the following website to calculate dew point which can then be used to show you have comfortable that you would feel. Please feel free to add this to your program using additional if statements. The URL: https://www.omnicalculator.com/physics/dew-point#:~:text=How%20do%20I%20calculate%20dew,by%20the%20temperature%20plus%20243.04.

temperature (°F)																
Relative Humidity (%)	80	82	84	86	88	90	92	94	96	98	100	102	104	106	108	110
40	80	81	83	85	88	91	94	97	101	105	109	114	119	124	130	136
45	80	82	84	87	89	93	96	100	104	109	114	119	124	130	137	
50	81	83	85	88	91	95	99	103	108	113	118	124	131	137		
55	81	84	86	89	93	97	101	106	112	117	124	130	137			
60	82	84	88	91	95	100	105	110	116	123	129	137				
65	82	85	89	93	98	103	108	114	121	128	136					
70	83	86	90	95	100	105	112	119	126	134						
75	84	88	92	97	103	109	116	124	132							
80	84	89	94	100	106	113	121	129								
85	85	90	96	102	110	117	126	135								
90	86	91	98	105	113	122	131									
95	86	93	100	108	117	127										
100	87	95	103	112	121	132										

Caution   Extreme Caution   Danger   Extreme Danger

Dew Point °F	How it Feels	Relative Humidity @ 90°F
Greater than 80°	Severely uncomfortable, Maybe deadly for asthma patients	65% and higher
75° to 80°	Extremely uncomfortable	62%
70° to 74°	Very humid, quite uncomfortable	52% to 60%
65° to 69°	Somewhat uncomfortable	44% to 52%
60° to 64°	OK for most, but notice the humidity	37% to 46%
55° to 59°	Comfortable	38% to 41%
50° to 54°	A little uncomfortable	31% to 37%
less than 49°	Very dry	30%

Procedure

1. **Create a new blank sketch.**

2. **Include the libraries, objects, and variables.**

```
#include <Wire.h>   // I2C library
#include "ens210.h" // ENS210 library
#include <SparkFun_APDS9960.h> // ambient light library

// create objects
SparkFun_APDS9960 apds = SparkFun_APDS9960();
ENS210 ens210;

uint16_t ambient_light = 0; // ambient light variable
```

3. Input the following **setup()** function to initialize temperature and ambient light sensors:

```
void setup() {
  // Enable serial
  Serial.begin(9600);

  // Enable I2C
  Wire.begin();

  // Enable ENS210 and APDS
  ens210.begin();
  apds.init();
  apds.enableLightSensor(false);
}
```

4. Begin inputting the following **loop()** function to get temperature, humidity, and ambient light data:

```
void loop() {
  // creates variables for temperature and humidity and measures new values
  int t_data, t_status, h_data, h_status;
  ens210.measure(&t_data, &t_status, &h_data, &h_status);
  Serial.print("Temp: ");
  Serial.print( ens210.toFahrenheit(t_data, 10) / 10.0 );
  Serial.println(" F");
  Serial.print("Humidity: ");
  Serial.print( ens210.toPercentageH(h_data,1)); Serial.println(" %RH");
  // reads ambient light value
  apds.readAmbientLight(ambient_light);
  Serial.print("Ambient Light: "); Serial.print(ambient_light);
  Serial.println();
  Serial.println();
```

5. Finish inputting the following **loop()** function to display predicted weather conditions:

```
// prints weather condition predictions based on humidity and ambient light
if(ambient_light < 12000 && h_data > .75)
{
  Serial.println("It is probably raining");
}
else if(ambient_light < 20000)
{
  Serial.println("It is cloudy");
}
else if(ambient_light > 50000)
{
  Serial.println("It is sunny");
}
else
{
  Serial.println("There is a mix of sun and clouds");
}
Serial.println();
// waits 2 seconds
delay(2000);
}
```

Result

You have created a basic functioning weather station! Try testing it by covering the light sensor with your hand and by breathing on the temperature sensor lightly. Additionally, if it is not raining, you can test the weather station outside for different temperature, humidity, and ambient light readings.

Now It's Your Turn!

- Can you get real weather data from online to compare the accuracy of your station with?

- Can you create a new variable (like dewpoint or heat index) that can be calculated with the weather information that you have?

- Can you display how comfortable you would feel based on temperature and humidity?

Lesson A – 2: If This, Then That

OVERVIEW

In this lesson we will:

- Learn what IFTTT is

- Set up IFTTT and connect it to Arduino

- Test the triggering event

Teacher Guide

In this lesson, students will be creating various accounts to access new features via a website called IFTTT. Some of these new features include receiving emails, receiving phone calls, receiving SMS text messages, and posting Tweets at the push of a button. For this lesson, students must first create a IFTTT account which is detailed in the instructions. Then, students must register for an email account if they do not already have one or a Twitter account (or both). Please note that phone calls are only available in the U.S. and to send text messages you must have an android device.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

IFTTT, stands for If This Then That, is a free web-based service to create chains of simple conditional statements, called **applets**. An applet is triggered by changes that occur within other web services such as Gmail, Facebook, Telegram, Instagram, or Pinterest. For example, an applet may send an e-mail message if the user tweets using a hashtag or copy a photo on Facebook to a user's archive if someone tags a user in a photo.

IFTTT is an IoT Platform provide us a free-web based service, helps in connecting different apps and devices with each other. Here, we are using ESP32 Wi-Fi module which delivers advance features for IoT projects. ESP32 can be configured as Access Point (AP) mode and Station (STA) mode, but we are using it in Station mode for this project. And, for sending the email notification we are using IFTTT cloud service. Also, you can do various tasks using IFTTT like sending SMS, twitter tweets and many more.
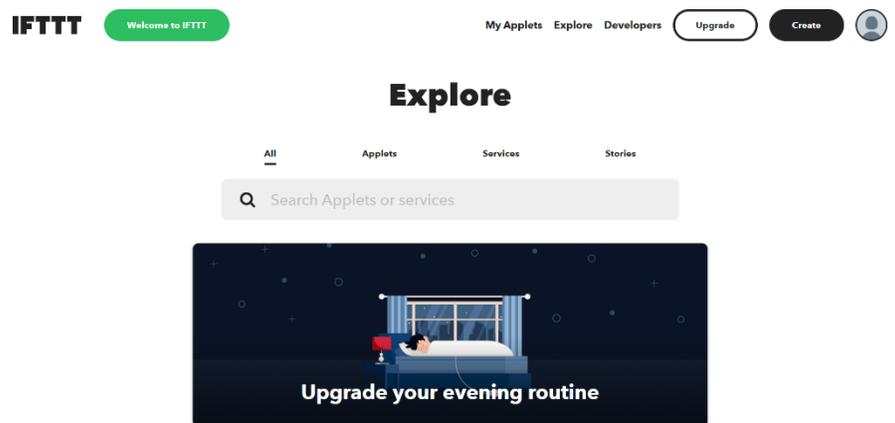
For programming ESP32 we are using Arduino IDE and IFTTT applet to send an E-mail. Likewise, you are using IFTTT for sending data through ESP32, you can also use Adafruit IO for the same, below are some projects using ESP32, IFTTT and Adafruit IO.

Procedure

This project will simply connect your Arduino board to the Webhooks Applet, so every time you press a specific button, you will receive either an email (button at pin 1), phone call (button at pin 2), SMS Text (button at pin 3), or send a Tweet (button at pin 4). Please note to use the phone call service, you must be within the U.S. Additionally, to use the SMS text you must have an android device to use. Both receiving an email and sending a tweet should work regardless assuming you create an account.

To begin this project, you will need to register an IFTTT account and create multiple IFTTT Actions:

1. **Visit IFTTT.com and click on the "Get Started" button from the top right corner then create an account.**

2. **Follow the registration instructions and verify your account.**

Email

1. After you have created your account, in the top right corner select "Create"

2. Click on the "Add" button and search for Webhooks.

3. Click on Webhooks and click on Connect

Web Request – a message that is communicated between a webserver and a client and this helps to display the user's interface

---

4. Select "Receive a web request"

5. Name the Event "Email" and click Create trigger

6. Click on the "Add" button and search for "Email"

7. Select "Email"

8. Select "Send me an email"

9. Input the following information:

10. Click on Create action

11. **Finally, click Continue then Finish.**

Phone Call

1. **In the top right corner select "Create"**

2. **Click on the "Add" button and search for Webhooks.**

3. **Click on Webhooks and click on Connect**

4. **Select "Receive a web request"**

5. **Name the Event "Call" and click Create trigger**

6. **Click on the "Add" button and search for "phone"**

7. **Select "Phone Call (US only)"**

8. **Select "Call my phone"**

9. **Input the following information:**
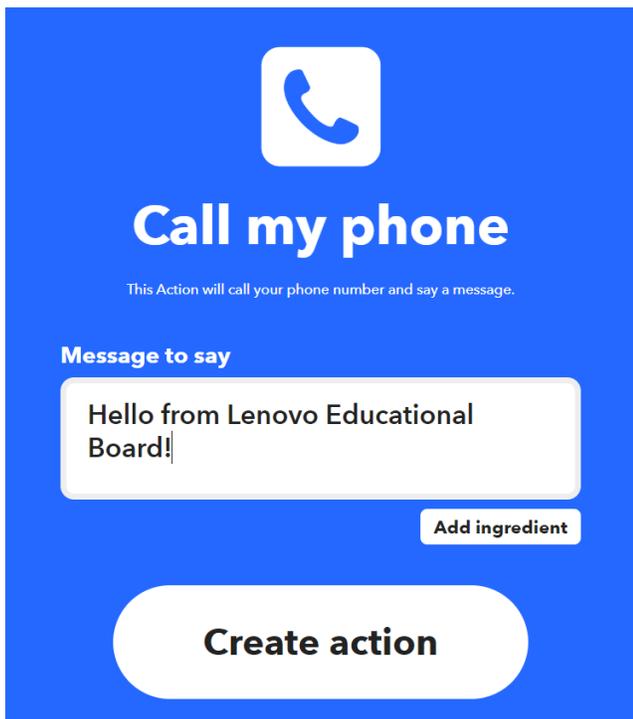
Call my phone

This Action will call your phone number and say a message.

Message to say

Hello from Lenovo Educational Board!

Add ingredient

Create action

10. **Click on Create action**

11. Finally, click Continue then Finish.

SMS Text Message

1. In the top right corner select "Create"

2. Click on the "Add" button and search for Webhooks.

3. Click on Webhooks and click on Connect

4. Select "Receive a web request"

5. Name the Event "SMS" and click Create trigger

6. Click on the "Add" button and search for "sms"

7. Select "Android SMS"

8. Select "Send an SMS"

9. Input the following information:

10. Click on Create action

11. Finally, click Continue then Finish.

Tweet

1. In the top right corner select "Create"

2. Click on the "Add" button and search for Webhooks.

3. Click on Webhooks and click on Connect

4. Select "Receive a web request"

5. Name the Event "Tweet" and click Create trigger

6. Click on the "Add" button and search for "twitter"

7. Select "Twitter"

8. Select "Post a Tweet"

9. Connect your Twitter account.

10. Input the following information:

11. Click on Create action

12. Finally, click Continue then Finish.

Connecting Applets

After you have successfully created the Webhooks Action, now it's time to get your Applet key and instructions to trigger an event.

1. Click on Search from the top panel

2. Type in Webhooks and select Services

4. Click on Documentation from the right

5. This page will show you a unique key for your ID

6. Note: Keep this key confidential because it will give anyone access to all your applets.

7. Input "Email" in the {event} under the "Make a Post or get web request"

8. Copy and paste the URL on to a new tab to test the trigger of the event

9. You should get a message indicating "Congratulations! You've fired the Email event"

10. Please repeat steps 7-9 for event names: "Call", "SMS", and "Tweet"

Arduino Code

1. Include the following libraries, variables, objects, and constants:

```
// include the following libraries
#include <WiFi.h>
#include <HTTPClient.h>
#include <Preferences.h>
Preferences preferences;

String ssid = "";
String password =  "";
// declare button pins
const int BUTTON = 1;
const int BUTTON2 = 2;
const int BUTTON3 = 3;
```

2. Input the getWifi() function to read WiFi information from the EEPROM:

```
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

3. Input the following setup() function:

```
void setup() {
  // set baud rate
  Serial.begin(115200);
  delay(200);
  // connect to wifi
  getWifi();
  WiFi.begin(ssid, password);
  Serial.println("Connecting to WiFi..");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("Connected to the WiFi network");
  // set pinModes for buttons to input
  pinMode(BUTTON, INPUT);
  pinMode(BUTTON2, INPUT);
  pinMode(BUTTON3, INPUT);
  pinMode(BUTTON4, INPUT);
```

This function allows the Lenovo Educational Board to connect to Wi-Fi and set the buttons to input.

4. Input the following **loop()** function:

```
void loop() {
  // if a button is pressed send an email, phone call, sms, or tweet
  if (digitalRead(BUTTON) == HIGH)
  {
    openURL("https://maker.ifttt.com/trigger/Email/json/with/key/**INSERTKEYHERE**");
  } // make sure to replace "**INSERTKEYHERE**" with your key
  if (digitalRead(BUTTON2) == HIGH)
  {
    openURL("https://maker.ifttt.com/trigger/Call/json/with/key/**INSERTKEYHERE**");
  } // make sure to replace "**INSERTKEYHERE**" with your key
  if (digitalRead(BUTTON3) == HIGH)
  {
    openURL("https://maker.ifttt.com/trigger/SMS/json/with/key/**INSERTKEYHERE**");
  } // make sure to replace "**INSERTKEYHERE**" with your key
  if (digitalRead(BUTTON4) == HIGH)
  {
    openURL("https://maker.ifttt.com/trigger/Tweet/json/with/key/**INSERTKEYHERE**");
  } // make sure to replace "**INSERTKEYHERE**" with your key
}
```

Inside this function, make sure to replace the last section of the URL with your personal key where each button calls the function to open the URL.

5. Input the openURL function:

```
// opens a url using a get request
void openURL(String url)
{
  if ((WiFi.status() == WL_CONNECTED))
  { //Check the current connection status
    HTTPClient http;
    http.begin(url); // Specify the URL
    int httpCode = http.GET(); // Make the request
    if (httpCode > 0)
    { //Check for the returning code
      String payload = http.getString();
      Serial.println(payload);
    }
    else
    {
      Serial.println("Error on HTTP request");
    }
    http.end(); //Free the resources
  }
}
```

This function fires the web request we created on IFTTT by using a get request.

A get request is used to request data which is used to then trigger the next event (i.e., email, phone call, sms, or tweet).

Result

You can now send emails, phone calls, sms, and tweets to yourself via the 4 buttons on your board.

Now It's Your Turn!

- Can you send yourself an email if the temperature passes a certain threshold?

- Can you send a phone call if you read a certain gesture?

Lesson A – 3: IFTTT Play Tune

OVERVIEW

In this lesson we will:

- Utilize IFTTT to play a song on the Lenovo Educational Board

- Utilize IFTTT to use Amazon Alexa to communicate with the Lenovo Educational Board

Teacher Guide

In this lesson, students will be utilizing the Adafruit IO library which can be installed in the library manager to control the Lenovo Educational Board's LEDs and other devices. After installing this library, students will need to create an Adafruit IO account which will be used later in the lesson. Additionally, students will need to download the Amazon Alexa app and create an Amazon account in preparation for this lesson.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

Following from the last lesson, we will continue to explore what the Lenovo Educational Board can do coupled with the IFTTT website. For this lesson, the board will receive data from the IFTTT service to create an audio effect. The way that we will communicate between IFTTT and the Lenovo Educational Board will be a Adafruit IO Feed. Adafruit IO is a library that we will use for this lesson and contains Feeds for IFTTT to post to and the Lenovo Educational Board to monitor. In this lesson, we will use these feeds to determine when to play the Tetris song after we have spoken to Amazon Alexa to trigger the event.

Procedure

1. Click here to open the Adafruit IO website. Here is the URL: https://io.adafruit.com/

2. Create an account and click Feeds.

3. Click the New Feed button

4. Input the following information:

Create a new Feed   ✕

Name

| tetris |

Maximum length: 128 characters. Used: 6

Description

| plays tetris song |

            Cancel   **Create**

5. Open IFTTT.com and log in to your account.

6. Click "Create" in the upper right corner then click the "Add" button and search "Amazon"

7. Select "Amazon Alexa" then select "Say a specific phrase"

8. Click "Connect" and log in or create an Amazon account.

9. Input the following information:

What phrase?

| Trigger tetris |

Create trigger

10. Click on Create trigger

11. Click the "Add" button in the Then That section.

12. Search for "Adafruit" and select Adafruit.

13. Select "Send data to Adafruit IO" and create or log in to your Adafruit account.

14. Click "Authorize" on the pop-up window by scrolling down.

15. In following field select the following Feed name and input the following information:
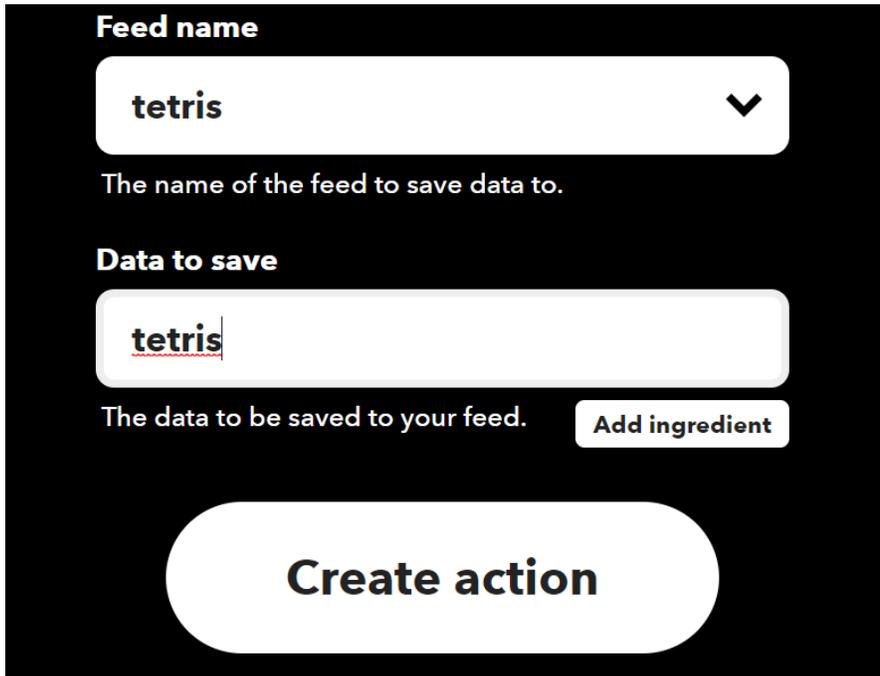
16. Finally, click Continue then Finish.

17. Now, open the Arduino IDE and create a new blank sketch.

18. Click Tools then Manage Libraries...
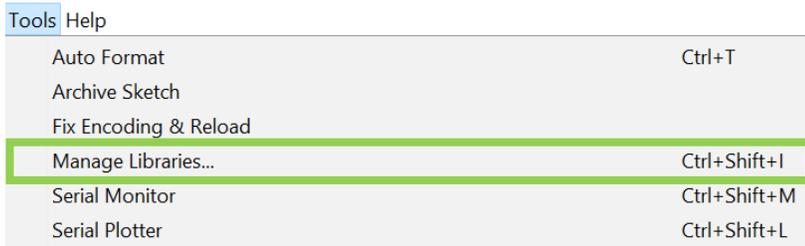
19. Type Adafruit IO Arduino in the search box and scroll down.
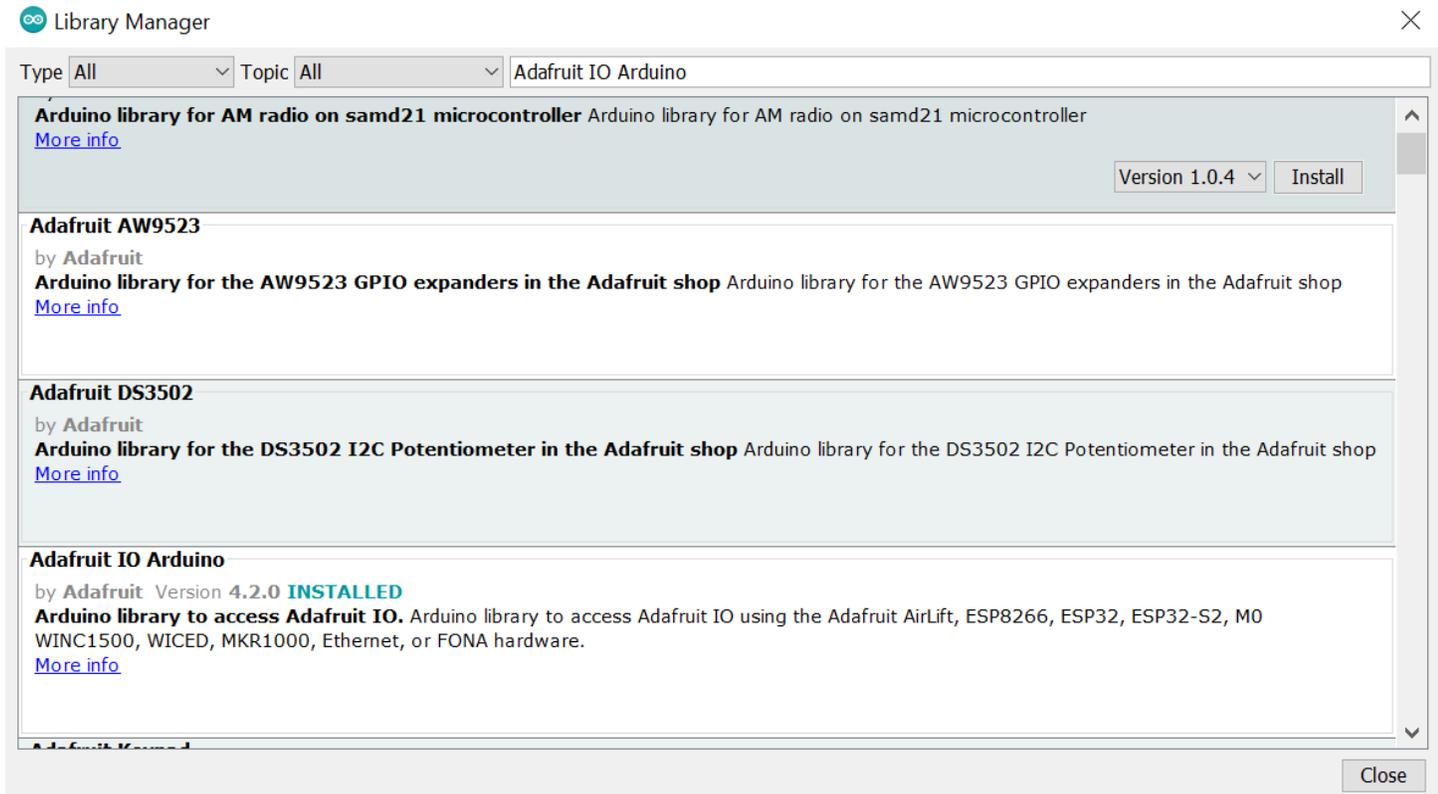
20. Click Install and then Close.

21. Include the following libraries and object, then, input your Adafruit username and key:

```
// include the following libraries
#include "AdafruitIO_WiFi.h"
#include <Preferences.h>
Preferences preferences;

#define IO_USERNAME "*****" // input adafruit username here
#define IO_KEY "*****" // input adafruit key here
String ssid = "";
String password = "";
```

Make sure to input your Adafruit username and API key.

22. To find your key please open your Adafruit Feed and click the key icon:

23. Set up the feed for the Lenovo Educational Board to receive information from IFTTT:

```
// set up feed
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, ssid.c_str(), password.c_str());
AdafruitIO_Feed *tetris_feed = io.feed("tetris");
int c = 0;
```

24. Input the following pins and music information:

```
// buzzer pin and note frequencies
#define PIEZO_PIN  (15)
#define _R      (0)
#define _C0     (16.35)
#define _CS0    (17.32)
#define _D0     (18.35)
#define _DS0    (19.45)
#define _E0     (20.60)
#define _F0     (21.83)
#define _FS0    (23.12)
#define _G0     (24.50)
#define _GS0    (25.96)
#define _A0     (27.50)
#define _AS0    (29.14)
#define _B0     (30.87)
#define _C1     (32.70)
#define _CS1    (34.65)
#define _D1     (36.71)
#define _DS1    (38.89)
#define _E1     (41.20)
#define _F1     (43.65)
#define _FS1    (46.25)
#define _G1     (49.00)
#define _GS1    (51.91)
#define _A1     (55.00)
#define _AS1    (58.27)
#define _B1     (61.74)
#define _C2     (65.41)
#define _CS2    (69.30)
#define _D2     (73.42)
#define _DS2    (77.78)
#define _E2     (82.41)
#define _F2     (87.31)
#define _FS2    (92.50)
#define _G2     (98.00)
#define _GS2    (103.83)
#define _A2     (110.00)
#define _AS2    (116.54)
#define _B2     (123.47)
#define _C3     (130.81)
#define _CS3    (138.59)
#define _D3     (146.83)
#define _DS3    (155.56)
#define _E3     (164.81)
#define _F3     (174.61)
#define _FS3    (185.00)
#define _G3     (196.00)
#define _GS3    (207.65)
#define _A3     (220.00)
```

```
#define _AS3     (233.08)
#define _B3      (246.94)
#define _C4      (261.63)
#define _CS4     (277.18)
#define _D4      (293.66)
#define _DS4     (311.13)
#define _E4      (329.63)
#define _F4      (349.23)
#define _FS4     (369.99)
#define _G4      (392.00)
#define _GS4     (415.30)
#define _A4      (440.00)
#define _AS4     (466.16)
#define _B4      (493.88)
#define _C5      (523.25)
#define _CS5     (554.37)
#define _D5      (587.33)
#define _DS5     (622.25)
#define _E5      (659.25)
#define _F5      (698.46)
#define _FS5     (739.99)
#define _G5      (783.99)
#define _GS5     (830.61)
#define _A5      (880.00)
#define _AS5     (932.33)
#define _B5      (987.77)
#define _C6      (1046.50)
#define _CS6     (1108.73)
#define _D6      (1174.66)
#define _DS6     (1244.51)
#define _E6      (1318.51)
#define _F6      (1396.91)
#define _FS6     (1479.98)
#define _G6      (1567.98)
#define _GS6     (1661.22)
#define _A6      (1760.00)
#define _AS6     (1864.66)
#define _B6      (1975.53)
#define _C7      (2093.00)
#define _CS7     (2217.46)
#define _D7      (2349.32)
#define _DS7     (2489.02)
#define _E7      (2637.02)
#define _F7      (2793.83)
#define _FS7     (2959.96)
#define _G7      (3135.96)
#define _GS7     (3322.44)
#define _A7      (3520.00)
#define _AS7     (3729.31)
#define _B7      (3951.07)
#define _C8      (4186.01)
#define _CS8     (4434.92)
#define _D8      (4698.63)
#define _DS8     (4978.03)
#define _E8      (5274.04)
#define _F8      (5587.65)
#define _FS8     (5919.91)
#define _G8      (6271.93)
#define _GS8     (6644.88)
#define _A8      (7040.00)
#define _AS8     (7458.62)
#define _B8      (7902.13)
#define BPM      (120.0)
#define POLY_DELTA (14400)
```

Please feel free to copy and paste the above notes and the frequency of Hertz they represent.

25. Input the following arrays that store the notes and times:

```
// lead notes to play
float lead_notes[] = {
  // part 1
  _E5, _B4, _C5, _D5, _C5, _B4, _A4, _A4, _C5, _E5, _D5,
_C5, _B4, _B4, _C5, _D5, _E5, _C5, _A4, _A4, _R,
  _D5, _F5, _A5, _G5, _F5, _E5, _C5, _E5, _D5, _C5, _B4,
_B4, _C5, _D5, _E5, _C5, _A4, _A4, _R,

  // part 2
  _E4, _C4, _D4, _B3, _C4, _A3, _GS3, _B3,
  _E4, _C4, _D4, _B3, _C4, _E4, _A4, _A4, _GS4, _R
};
// durations for lead notes
float lead_times[] = {
  // part 1
  1.0, 0.5, 0.5, 1.0, 0.5, 0.5, 1.0, 0.5, 0.5, 1.0, 0.5,
0.5, 1.0, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
  1.5, 0.5, 1.0, 0.5, 0.5, 1.5, 0.5, 1.0, 0.5, 0.5, 1.0,
0.5, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,

  // part 2
  2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0,
  2.0, 2.0, 2.0, 2.0, 1.0, 1.0, 1.0, 1.0, 3.0, 1.0
};
```

```
// bass notes to play
float bass_notes[] = {
  // part 1
  _E2, _E3, _E2, _E3, _E2, _E3, _E2, _E3, _A1, _A2, _A1, _A2, _A1, _A2, _A1, _A2, _GS1,
_GS2, _GS1, _GS2, _GS1, _GS2, _GS1, _GS2, _A1, _A2, _A1, _A2, _A1, _B2, _C3, _E3,
  _D2, _D3, _D2, _D3, _D2, _D3, _D2, _D3, _C2, _C3, _C2, _C3, _C2, _C3, _C2, _C3, _B1,
_B2, _B1, _B2, _B1, _B2, _B1, _B2, _A1, _A2, _A1, _A2, _A1, _A2, _A1, _A2,

  // part 2
  _A1, _E2, _A1, _E2, _A1, _E2, _A1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2,
_A1, _E2, _A1, _E2, _A1, _E2, _A1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2,
  _A1, _E2, _A1, _E2, _A1, _E2, _A1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2,
_A1, _E2, _A1, _E2, _A1, _E2, _A1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2, _GS1, _E2
};
// durations for bass notes
float bass_times[] = {
  // part 1
  0.5,  0.5,  0.5,  0.5,  0.5,  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5,  0.5,  0.5,  0.5,  0.5,  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,

  // part 2
  0.5,  0.5,  0.5,  0.5,  0.5,  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
  0.5,  0.5,  0.5,  0.5,  0.5,  0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5
};
```

These arrays hold the notes to be played in order, including the amount of time between each note.

26. Input the counts of notes variables:

```
int lead_note_count = sizeof(lead_notes) / sizeof(float);
int bass_note_count = sizeof(bass_notes) / sizeof(float);
```

27. Input the play_one_note() function:

```
// plays one note on the buzzer
void play_one_note(float frequency, unsigned long duration) {
  unsigned long period = 1000000.0/frequency;
  for (unsigned int cycles=duration/period; cycles>0; cycles--) {
    // half the time on
    digitalWrite(PIEZO_PIN, HIGH);
    delayMicroseconds( period/2 );
    digitalWrite(PIEZO_PIN, LOW);
    delayMicroseconds( period/2 );
  }
  delayMicroseconds(duration % period);
}
```

This function plays the note using the given frequency and duration to play it for.

28. Input the play_two_notes() function:

```
// plays two notes on the buzzer by calling play_one_note
void play_two_notes(float freq1, float freq2, unsigned long duration) {
    for (unsigned long t=0; t<duration; t+=2*POLY_DELTA) {
      play_one_note(freq1, POLY_DELTA);
      play_one_note(freq2, POLY_DELTA);
    }
}
```

This function plays two notes using the given frequency and duration to play it for by calling the play_one_note() function.

29. Input the playTetris() function:

```
// plays the tetris theme song
void playTetris() {
// chooses note to play
  int curr_lead_note = 0;
  int curr_bass_note = 0;
  float curr_lead_note_time_remaining = lead_times[curr_lead_note];
  float curr_bass_note_time_remaining = bass_times[curr_bass_note];
  float lead_freq, bass_freq, note_value;
  unsigned long duration;
  // chooses note duration to play
  while (curr_lead_note < lead_note_count && curr_bass_note < bass_note_count) {
    lead_freq = lead_notes[curr_lead_note];
    bass_freq = bass_notes[curr_bass_note];
    note_value = min(curr_lead_note_time_remaining, curr_bass_note_time_remaining);
    duration = note_value * 1000000 * (60.0/BPM);
    // determines how to play the note
    if (lead_freq > 0 && bass_freq > 0) {
      play_two_notes(lead_freq, bass_freq, duration);
    } else if (lead_freq > 0) {
      play_one_note(lead_freq, duration);
    } else if (bass_freq > 0) {
      play_one_note(bass_freq, duration);
    } else {
      delay( duration/1000 );
    }

    // Advance lead note
    curr_lead_note_time_remaining -= note_value;
    if (curr_lead_note_time_remaining < 0.001) {
      curr_lead_note++;
      curr_lead_note_time_remaining = lead_times[curr_lead_note];
    }

    // Advance bass note
    curr_bass_note_time_remaining -= note_value;
    if (curr_bass_note_time_remaining < 0.001) {
      curr_bass_note++;
      curr_bass_note_time_remaining = bass_times[curr_bass_note];
    }
  }
}
```

This function plays the Tetris theme song once by incrementing through the arrays and playing the notes in order to get the song we want.

30. Input the handleMessage() function:

```
// called when a message is recieved
void handleMessage(AdafruitIO_Data *data) {
  if (c != 0)
  {
    Serial.println("Tetris has been triggered!");
    playTetris();
  }
  c++;
}
```

This function is called whenever we see that the Feed has been triggered in Adafruit IO by IFTTT.

31. Input the getWifi() function to read WiFi information from the EEPROM:

```
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

32. Input the following **setup()** function:

```
void setup() {
  // begin serial
  Serial.begin(115200);

  // set up buzzer
  pinMode(PIEZO_PIN, OUTPUT);

  // connect to adafruit
  Serial.print("Connecting to Adafruit IO");
  io.connect();

  // sets up function to be called when a message is recieved
  tetris_feed->onMessage(handleMessage);

  // connecting to adafruit
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.println(io.statusText());
  tetris_feed->get();
}
```

The buzzer is setup for output and we connect the Tetris Feed to be monitored in the loop() function.

33. Input the following **loop()** function:

```
void loop() {
  // processes data
  io.run();
}
```

The loop function is always looking for data in the Adafruit IO Feed.

Result

Now, when you talk say "Alexa Trigger Tetris" either on the Alexa app on your phone or on a device that had Alexa, you will hear the Tetris theme song be played.

Now It's Your Turn!

- Can you make your Lenovo Educational Board play a different song? (hint: you can search online for other songs created for Arduino)

- Can you turn on the LEDs by using a voice command from Alexa? And then back off?

Lesson A – 4: IFTTT Nightlight

OVERVIEW

In this lesson we will:

- Utilize IFTTT to turn on the Neopixel after sunset

- Utilize IFTTT to turn off the Neopixel after sunrise

- Utilize IFTTT to turn on the LEDs if the weather conditions change to either clear skies or rain

Teacher Guide

In this lesson, students will be utilizing what they learned in the last lesson to have the LEDs and Neopixel light up when certain weather conditions arise.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

Continuing from the last lesson, we will explore another application of what the Lenovo Educational Board can do coupled with the IFTTT website. For this lesson, we will similarly post to the Adafruit IO feed using the IFTTT website. However, we will use the Weather Underground service to monitor sunrise, sunset, and current conditions at a given location. Then, IFTTT will post to the appropriate feed and the board will determine which LED or Neopixel color to turn on or off using this data.

Procedure

1. Click <u>here</u> to open the Adafruit IO website. Here is the URL: https://io.adafruit.com/

2. Log In to your account and click Feeds.

3. Click the New Feed button

4. Input the following information:

5. Create another New Feed.

6. Input the following information:

Create a new Feed   ✕

Name

sunset

Maximum length: 128 characters. Used: 6

Description

sunset reported

Cancel   Create

7. Create another New Feed.

8. Input the following information:

Create a new Feed   ✕

Name

rain

Maximum length: 128 characters. Used: 4

Description

rain reported

Cancel   Create
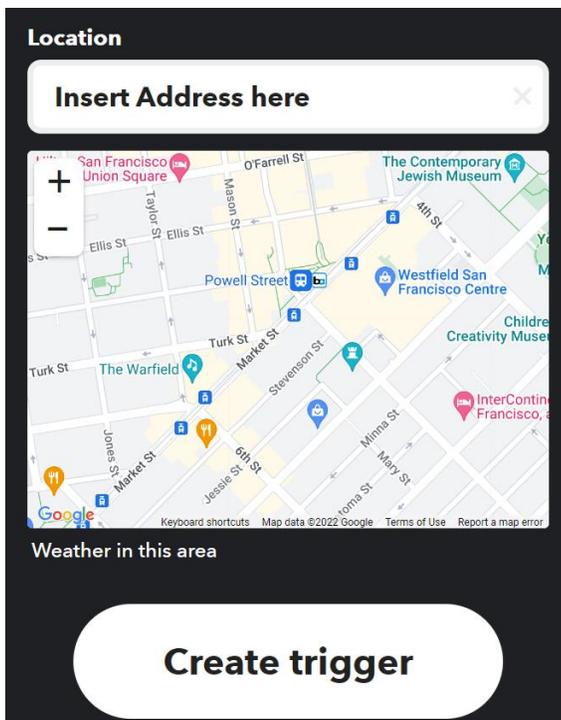
9. Create another New Feed.

10. Input the following information:

11. Open IFTTT.com and log in to your account.

12. Click "Create" in the upper right corner then click the "Add" button and search "Weather"

13. Select "Weather Underground" then select "Sunrise"

14. Click "Connect" and log in or create a Weather Underground account.

15. Input the following information:

16. Click on Create trigger

17. Click the "Add" button in the Then That section.

18. Search for "Adafruit" and select Adafruit.

19. Select "Send data to Adafruit IO" and create or log in to your Adafruit account.

20. In following field select the following Feed name and input the following information:

Feed name

sunrise ▾

The name of the feed to save data to.

Data to save

sunrise

The data to be saved to your feed.   Add ingredient

Create action

21. Finally, click Continue then Finish.

22. Repeat steps 12-21 and replace "Sunrise" with "Sunset"

23. Repeat steps 12-21 two more times but choose "Current condition changes to Clear" and "Current condition changes to Rain" and then fill in either "clearday" or "rain" in the fields above appropriately.

24. Now, open the Arduino IDE and create a new blank sketch.

25. Include the following libraries and input your Adafruit username and key as well as your Wi-Fi name and password:

```
// include the following libraries
#include "AdafruitIO_WiFi.h"
#include <Adafruit_NeoPixel.h>
#include <Preferences.h>

#define IO_USERNAME "*****" // input adafruit username here
#define IO_KEY "*****" // input adafruit key here
String ssid = "";
String password = "";
```

26. To find your key please open your Adafruit Feed and click the key icon:

27. Input the following objects and constants:

```
#define NEOPIXEL_PIN 45 // Neopixel is on pin 45
#define NEOPIXEL_COUNT 1 // There is only 1 RGB LED
#define NEOPIXEL_INDEX 0 // RGB LED is at index 0

// create appropriate objects
Preferences preferences;
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, ssid.c_str(), password.c_str());
Adafruit_NeoPixel Pixel(NEOPIXEL_COUNT, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
```

28. Set up the feed for the Lenovo Educational Board to receive information from IFTTT:

```
// set up feeds
AdafruitIO_Feed *sunrise_feed = io.feed("sunrise");
AdafruitIO_Feed *sunset_feed = io.feed("sunset");
AdafruitIO_Feed *rain_feed = io.feed("rain");
AdafruitIO_Feed *clearday_feed = io.feed("clearday");
```

29. Input the following pins and counters to prevent running during boot:

```
// create LED pins and counters to prevent each from running when booting
const int yellow = 11;
const int blue = 13;
int c = 0;
int i = 0;
int j = 0;
int k = 0;
```

30. Input the sunrise() function:

```
// called when sunrise recieved
void sunrise(AdafruitIO_Data *data) {
  if (c != 0)
  {
    Serial.println("Sunrise has been triggered!");
    // Turn Neopixel off
    Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 0);
    Pixel.show();
  }
  c++;
}
```

This function is called whenever we see that the sunrise Feed has been triggered in Adafruit IO by IFTTT.

31. Input the sunset() function:

```
// called when sunset recieved
void sunset(AdafruitIO_Data *data) {
  if (i != 0)
  {
    Serial.println("Sunset has been triggered!");
    // Turn Neopixel on
    Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 255, 255);
    Pixel.show();
  }
  i++;
}
```

This function is called whenever we see that the sunset Feed has been triggered in Adafruit IO by IFTTT.

32. Input the rain() function:

```
// called when rain recieved
void rain(AdafruitIO_Data *data) {
  if (j != 0)
  {
    Serial.println("Rain has been triggered!");
    // yellow LED off and blue LED on
    digitalWrite(yellow, LOW);
    digitalWrite(blue, HIGH);
  }
  j++;
}
```

This function is called whenever we see that the rain Feed has been triggered in Adafruit IO by IFTTT.

33. Input the clearday() function:

```
// called when clearday recieved
void clearday(AdafruitIO_Data *data) {
  if (k != 0)
  {
    Serial.println("Clearday has been triggered!");
    // blue LED off and yellow LED on
    digitalWrite(blue, LOW);
    digitalWrite(yellow, HIGH);
  }
  k++;
}
```

This function is called whenever we see that the clearday Feed has been triggered in Adafruit IO by IFTTT.

34. Input the getWifi() function to read WiFi information from the EEPROM:

```
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

34. Input the following **setup()** function:

```
void setup() {
  // begin serial
  Serial.begin(115200);
  getWifi();
  // set up LEDs and Neopixel
  pinMode(yellow, OUTPUT);
  pinMode(blue, OUTPUT);
  Pixel.begin();
  Pixel.clear();
  Pixel.show();

  // connect to adafruit
  Serial.print("Connecting to Adafruit IO");
  io.connect();

  // sets up function to be called when a message is received
  sunrise_feed->onMessage(sunrise);
  sunset_feed->onMessage(sunset);
  rain_feed->onMessage(rain);
  clearday_feed->onMessage(clearday);
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.println(io.statusText());
  sunrise_feed->get();
  sunset_feed->get();
  rain_feed->get();
  clearday_feed->get();
}
```

The Neopixel and LEDs are setup for output and we connect the Feeds to be monitored in the loop() function.

35. Input the following **loop()** function:

```
void loop() {
  // processes data
  io.run();
}
```

The loop function is always looking for data in the Adafruit IO Feed.

Result

Now, when the sunsets of rises, the Neopixel will turn on or off accordingly. Additionally, the blue LED will turn on if current condition changes to rain and yellow if the current condition changes to clear.

Now It's Your Turn!

- Can you create new applets using IFTTT to create cool smart devices?

- Can you turn on the LED if you receive an email?

Lesson A – 5: Wordle

OVERVIEW

In this lesson we will:

- Create our own Wordle game

- Learn how to create a basic video game utilizing the Serial Monitor

Teacher Guide

In this lesson, students will be to create their own Wordle game. They will use the Neopixel and Serial Monitor in tandem to make this game work best. For best practice, please tell students to copy and paste the worlde_answers array so they do not know the answers. Additionally, please feel free to modify this as you see fit, but if you add more values to the array instead of replacing, be sure to update all of the random number generator to 1+ the array length. Please note that the Neopixel will flash in the order of letters you put in 5 times with red being the letter is not in the word, blue being letter is in the word in the wrong place, and green is the letter is in the word in the correct place.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

Wordle is a game that rose to popularity at the end of 2021. This game is a five-letter guessing game where you have six attempts to guess correctly. The game let's you know your previous guesses and let's you know if the letter you chose is not in the word, is in the word in the wrong spot, or is in the word in the correct spot. In this section, you will learn simple games like this are complex. This lesson includes some of the most code we have used yet while using multiple variables and arrays to make the code flow smoothly. Additionally, constants and arrays are very common in video games to keep track of different variables and groups which will become more obvious in upcoming lessons. Please note for this wordle game the letters will not disappear if they are already used. Furthermore, if you get a letter in the word in the correct spot the Neopixel will flash green, if a letter is in the word in the wrong spot the Neopixel will flash blue, and if the letter is not in the word the Neopixel will flash red.

Procedure

1. **Create a new blank sketch.**

2. **Input the following libraries, variables and object:**

```
#include <Adafruit_NeoPixel.h>

#define NEOPIXEL_PIN 45 // Neopixel is on pin 45
#define NEOPIXEL_COUNT 1 // There is only 1 RGB LED on DOS
#define NEOPIXEL_INDEX 0 // RGB LED is at index 0

// Create a NeoPixel object
Adafruit_NeoPixel Pixel(NEOPIXEL_COUNT, NEOPIXEL_PIN, NEO_GRB + NEO_KHZ800);
// sets computer guess and number of trials
String finalWord = "";
int trials = 0;
```

3. Input the following array for the Wordle answers:

```java
// array of words that can be chosen
static String wordle_answers[] = {
  "honor",
  "rebut",
  "sassy",
  "awake",
  "blush",
  "focal",
  "today",
  "evade",
  "naval",
  "serve",
  "dwarf",
  "model",
  "karma",
  "stink",
  "grade",
  "quiet",
  "bench",
  "feign",
  "major",
  "sugar",
  "fresh",
  "crust",
  "marry",
  "react",
  "pride",
  "floss",
  "helix",
  "croak",
  "staff",
  "paper",
  "unfed",
  "whelp",
  "outdo",
  "adobe",
  "crazy",
  "repay",
  "digit",
  "crate",
  "spike",
  "mimic",
  "pound",
  "linen",
  "unmet",
  "flesh",
  "forth",
  "first",
  "stand",
  "belly",
  "ivory",
  "print",
  "yearn",
  "drain",
  "bribe",
  "panel",
  "flume",
  "agree",
  "error",
```

```
    "swirl",
    "argue",
    "bleed",
    "delta",
    "flick",
    "front",
    "shrub",
    "biome",
    "start",
    "greet",
    "goner",
    "round",
    "audit",
    "gamma",
    "labor",
    "civic",
    "forge",
    "corny",
    "basic",
    "salad",
    "spicy",
    "spray",
    "essay",
    "spend",
    "motor",
    "alone",
    "hatch",
    "hyper",
    "ought",
    "belch",
    "pilot",
    "comet",
    "jaunt",
    "abyss",
    "growl",
    "dozen",
    "erode",
    "click",
    "world",
    "gouge",
    "briar",
    "chill",
    "clock",
};
```

The above array contains the wordle answer which we recommend copy and pasting as it can become tedious and would lose the fun in the game if you know all of the answers.

4. Input the following array to display correctly guessed letters:

```
// array that displays the correct guessed letters
String guessed[] = {
   "_",
   "_",
   "_",
   "_",
   "_",
   "_",
};
```

This array is used to print out which letters are in the correct spot to the console.

5. Input the following **setup()** function:

```
void setup() {
  // sets baud rate
  Serial.begin(115200);
  // initialize Neopixel
  Pixel.begin();
  Pixel.clear();
  Pixel.show();
  // picks a random word
  int r = random(1,101);
  finalWord = wordle_answers[r];
  Serial.println();
  Serial.println();
  Serial.println("You will have 6 attempts to guess a 5 letter english word.");
  Serial.println();
  Serial.println("WORDLE");
}
```

6. Begin inputting the following **loop()** function:

```
void loop() {
  if(trials < 6) // only have 6 guesses
  {
    while(Serial.available() > 0)
    {
      // reads the wordle guess
      String curGuess = Serial.readString();
      Serial.println();
      // A newline character indicates end of input
      if(curGuess[5] == '\n')
      {
        Serial.print("You Guessed: "); Serial.println(curGuess);
```

The loop section will reset if the user uses all six guesses. This section reads the user guess.

7. Continue inputting the following **loop()** function:

```cpp
// displays the neopixel colors whether the letter is correct or not
for(int i = 0; i < 5; i++)
{
  // displays green if in word in correct spot
  if(curGuess[i] == finalWord[i])
  {
    guessed[i] = finalWord[i];
    Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 255, 0);
    Pixel.show();
    delay(200);
    Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 0);
    Pixel.show();
    delay(500);
  }
  else
  {
    bool found = false;
    for(int j = 0; j < 5; j++)
    {
      // displays blue if in word and not in correct spot
      if(curGuess[i] == finalWord[j])
      {
        Serial.print(finalWord[j]);
        Serial.println(" is in the word");
        Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 255);
        Pixel.show();
        delay(200);
        Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 0);
        Pixel.show();
        delay(500);
        found = true;
      }
      // displays red if not in word
      else if (curGuess[i] != finalWord[j] && j == 4 && !found)
      {
        Pixel.setPixelColor(NEOPIXEL_INDEX, 255, 0, 0);
        Pixel.show();
        delay(200);
        Pixel.setPixelColor(NEOPIXEL_INDEX, 0, 0, 0);
        Pixel.show();
        delay(500);
      }
    }
  }
}
```

This section of the loop outputs the color values to the Neopixel based on whether each letter is in the word in the correct spot or not, or not in the word.

8. Finish inputting the following **loop()** function:

```
        // changes to display correctly guessed letters
        Serial.printf("%s %s %s %s %s", guessed[0], guessed[1], guessed[2],
guessed[3], guessed[4]);
        Serial.println();
        trials++;
        if(curGuess[0] == finalWord[0] && curGuess[1] == finalWord[1] && curGuess[2]
== finalWord[2] && curGuess[3] == finalWord[3] && curGuess[4] == finalWord[4])
        { // win condition
           Serial.println();
           Serial.println("Congratulations! You have guessed correctly!");
           trials = 7;
        }
      }
    }
  }
  else if (trials == 6)
  { // ran out of attempts
    Serial.println();
    Serial.println("Sorry you have run out of attempts!");
    Serial.print("The word was: "); Serial.println(finalWord);
    trials++;
  }
  else
  { // picks a new word for next game
    int r = random(1,101);
    finalWord = wordle_answers[r];
    trials = 0;
    restart();
  }
}
```

This section resets displays the win and lose conditions and picks a new word for resetting the game.

9. Input the following **restart()** function:

```
// resets the wordle game
void restart()
{
  for(int i = 0; i < 5; i++)
  {
    guessed[i] = "_";
  }
  Serial.println();
  Serial.println();
  Serial.println("You will have 6 attempts to guess a 5 letter english word.");
  Serial.println();
  Serial.println("WORDLE");
}
```

This function prints to start a new game and resets the array with correctly guessed letters.

Result

Now, you can play Wordle on your Lenovo Educational Board utilizing the Serial Monitor and Neopixel.

Now It's Your Turn!

- Can you make a Wordle game using HTML for a web browser?

- Can you use the OLED screen to output the guesses on there instead? (we will learn more about the OLED screen in the next lesson)

Lesson A – 6: OLED Display Temperature

OVERVIEW

In this lesson we will:

- Learn how to display information on the OLED display

Teacher Guide

In this lesson, students will require an OLED screen which can be purchased online at the following link for about $3-5 per OLED screen: https://www.amazon.com/Hosyond-Display-Self-Luminous-Compatible-Raspberry/dp/B09C5K91H7/ref=pd_lpo_1?pd_rd_i=B09C5K91H7&psc=1

When attaching the OLED screen to the device please be sure to match the following pins to choose the correct header to input the OLED into. GND > GND / 3.3 V > VCC / SCL > SCL / SDA > SDA

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

What is an OLED display?

OLED stands for organic light emitting diode which is essentially a LED that uses a film of organic compound to emit light. For use in these lessons, when we refer to the OLED we are talking about the display. In recent years OLED screens have become increasingly popular as these use many tiny LEDs to create a picture with crisp detail. OLEDs are common on many everyday devices including tv screens, phone screens, and computer monitors.
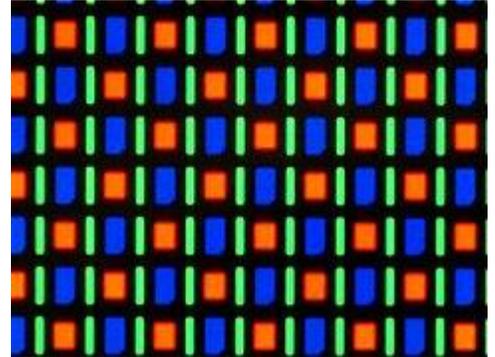
Procedure

1. **Include the following libraries, objects, and variable:**

```cpp
#include <Wire.h>   // I2C library
#include "ens210.h" // ENS210 library
#include <SparkFun_APDS9960.h> // ambient light library
#include <SSD1306Wire.h> // OLED display library

// declare/initialze objects and variables
SSD1306Wire display(0x3c);
SparkFun_APDS9960 apds = SparkFun_APDS9960();
ENS210 ens210;
#define WIRE Wire
uint16_t ambient_light = 0;
```

2. **Input the following setup() function:**

```cpp
void setup() {
  // Enable serial
  Serial.begin(9600);
  // initialize objects
  Wire.begin();
  ens210.begin();
  apds.init();
  apds.enableLightSensor(false);
  display.init();
  display.setContrast(255);
}
```

In the setup() function, we initialize the temperature sensor and the OLED display.

3. Input the following **loop()** function:

```
void loop() {
  // creates variables for temperature, humidity, and
ambient light and measures new values
  int t_data, t_status, h_data, h_status;
  ens210.measure(&t_data, &t_status, &h_data, &h_status);
  apds.readAmbientLight(ambient_light);
  // prints output to OLED display
  display.setLogBuffer(5, 30);
  display.clear();
  display.print("T: ");
  display.print(ens210.toFahrenheit(t_data, 10) / 10.0);
  display.print("F , H: ");
  display.print(ens210.toPercentageH(h_data,1));
  display.println("%");
  display.print("Ambient Light: ");
  display.println(ambient_light);
  display.drawLogBuffer(0, 0);
  display.display();
  delay(2);
}
```

In this section, we receive the most recent data and print in on the OLED screen.

Result

Now, you can see the temperature, humidity, and ambient light data being displayed and updated on the OLED display.

Now It's Your Turn!

- Can you repeat previous lessons that we have completed and print to the OLED display instead?

OVERVIEW

In this lesson we will:

- Learn how internet clocks work

- Learn how to draw on the OLED screen

Teacher Guide

In this lesson, students will require access to the Internet and an OLED screen before proceeding. Additionally, students will learn how to draw lines and circles to draw the watch face on the OLED screen.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

Nowadays, everyone has many devices in their home that are connected to the Internet that display the current time. The devices in your home that you do not need to set the time for use the Internet to get the local time, and set it based on location. While our device does not have your current location to determine your time zone, you will input the offset of your time zone from GMT time to calibrate the time. After that, we will get the current time (in GMT) from websites online to set the analog clock to an exact second.

Procedure

1. **Create a new blank sketch.**

2. **Include the following libraries:**

```
// libraries to be used for OLED display and time
#include <Arduino.h>
#include <WiFi.h>
#include <time.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Preferences.h>
```

3. **Declare the following object, variables, and constants**

```
Preferences preferences;
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &Wire, -1);

String ssid = "";
String password = "";

int GMTOffset = 0;  // GMT Offset counts up from GMT time to your current time
bool setOffset = false; // whether offset is displayed or not
```

Please be sure to input your Wi-Fi name and password in this step.

4. Input the getWifi() function to read WiFi information from the EEPROM:

```
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

5. Input the following setup() function:

```
void setup() {
  // set baud rate
  Serial.begin(115200);
  // initialize display
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  delay(200);
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0,0);
  display.setTextColor(WHITE);
  // connect to WiFi
  WiFi.begin(ssid, password);
  Serial.println("Connecting...");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(1000);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("Connected to Wi-Fi!");
}
```

We connect to Wi-Fi to get the current time from online sources for best accuracy.

6. Begin inputting the following **loop()** function:

```
void loop() {
  while (!setOffset)
  { // indicates whether you have put in your current offset in relation to GMT time
    while(Serial.available() > 0)
    {
      int hours = Serial.parseInt();
      int minutes = Serial.parseInt();
      // A newline character indicates end of input
      if(Serial.read() == '\n')
      {
        // converts to current time
        GMTOffset = (hours * 3600) + (minutes * 60);
        configTime(GMTOffset, 0, "pool.ntp.org","time.nist.gov"); // sets time
        setOffset = true;
      }
    }
  }
}
```

This of the loop() function asks for the offset from GMT in hour,minutes format counting up from GMT time. For example, if your are currently in EST in the United States your offset would be 20,0.

7. Continue inputting the following **loop()** function:

```
time_t rawtime = time(nullptr);
struct tm* timeinfo = localtime(&rawtime);

int radius = 35;
display.drawCircle(display.width()/2, display.height()/2, 2, WHITE);
// draws the clock face
for( int i = 0; i < 360; i= i + 30 )
{
  float angle = i;
  angle = (angle/57.29577951);
  int x1 =(64+(sin(angle)*radius));
  int y1 =(32-(cos(angle)*radius));
  int x2 =(64+(sin(angle)*(radius-5)));
  int y2 =(32-(cos(angle)*(radius-5)));
  display.drawLine(x1,y1,x2,y2,WHITE);
}
```

This section draws the clock face using trigonometry.

8. Finish inputting the following **loop()** function:

```
  // draws the second hand
  float angle = timeinfo->tm_sec*6;
  angle = (angle/57.29577951);
  int x2 = (64+(sin(angle)*(radius)));
  int y2 = (32-(cos(angle)*(radius)));
  display.drawLine(64,32,x2,y2,WHITE);

  // draws the minute hand
  angle = timeinfo->tm_min * 6;
  angle = (angle/57.29577951);
  x2 =(64+(sin(angle)*(radius-3)));
  y2 =(32-(cos(angle)*(radius-3)));
  display.drawLine(64,32,x2,y2,WHITE);

  // draws the hour hand
  angle = timeinfo->tm_hour * 30 + int((timeinfo->tm_min / 12) * 6);
  angle = (angle/57.29577951);
  x2 =(64+(sin(angle)*(radius-11)));
  y2 =(32-(cos(angle)*(radius-11)));
  display.drawLine(64,32,x2,y2,WHITE);

  // shows info to display then resets
  display.display();
  delay(100);
  display.clearDisplay();
}
```

Here, we draw the different hands on the clock which will update every time that we loop through this section as long as we are connected to Wi-Fi.

Result

Now, you have a fully functioning analog clock that displays similar to that of an Apple Watch. Please make sure to read the clock with the Lenovo words facing upwards to read the clock correctly. See below for a picture example.
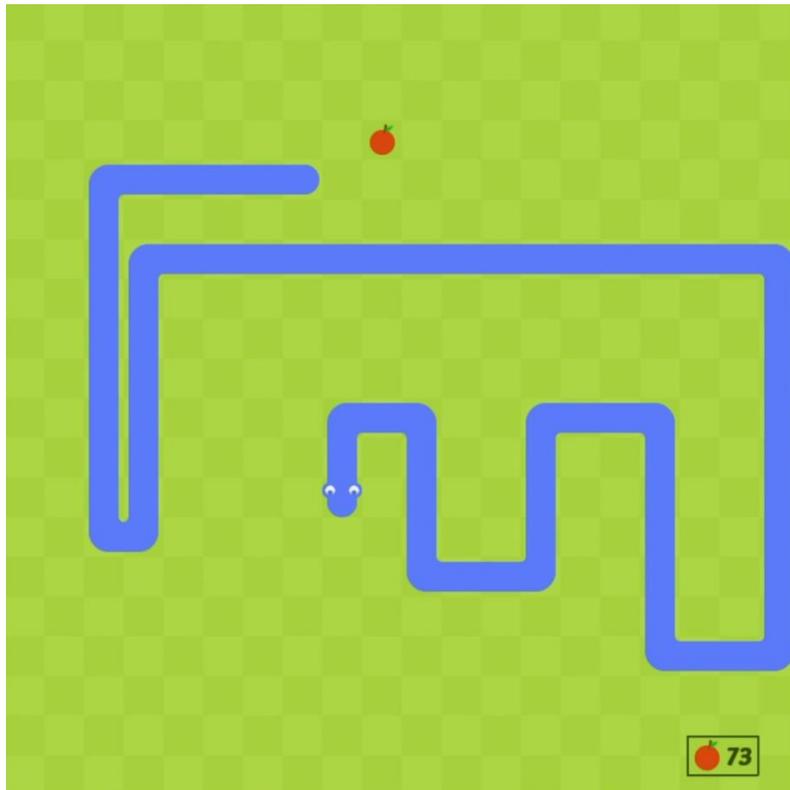
This is how to read the analog clock and the current time would be 9:22 a.m.

Now It's Your Turn!

- Can you add complications to the watch? Complications could include temperature and humidity.

Lesson A – 8: OLED Snake

OVERVIEW

In this lesson we will:

- Learn how to create a more complex moving object game

- Write a program to display a moving object on an OLED display

Teacher Guide

In this lesson, students will utilize the buttons and the OLED screen to create a snake game. This should be one of the more exciting projects that students will have completed yet as they will be rewarded with their learning to play the game once completed.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

Snake is retro video game genre that was created in 1976 as a two-player game called Blockade. While this is one of the classic video games, the first video game created was pong in 1958. Following your skills in the Snake and Tetris lessons, we encourage you to create a pong video game for additional practice. To continue, Snake became popular fast with hundreds of versions being created. Each of these versions was slightly different and the most common snake games you will find today online, will be similar to the snake game that we will create together.

For the snake game, a user controls a square that grows as you eat "fruits" that spawn in random locations across the map. As the user navigates to fruits, they must avoid running into the edges and into the snake as it continues to grow as you eat fruits. The game ends when you run into something or eat all of the fruit. For our version of snake, we will use the button as left (pin 1), down (pin2), up (pin 3), and right (pin 4) to control the direction of the snake's movement.

Procedure

1. **Create a new blank sketch.**

2. **Include the following libraries and objects:**

```
#include <Wire.h>
#include "SSD1306Wire.h"

SSD1306Wire display(0x3c);
#define WIRE Wire
```

3. Start declaring game constants:

```
// Define game constants

// Size of each snake segment
const int BOX_SIZE = 8;
const int MAX_SPEED = 8;
// Width and height of screen
const int SCREEN_WIDTH = 128;
const int SCREEN_HEIGHT = 64;
// Game loop timer, lower value means faster movement
const int TIMER_MAX = 300;
// Location in array for X and Y coordinates
const int X_ARRAY = 0;
const int Y_ARRAY = 1;

int timerCount = TIMER_MAX;
```

These game constants set the size of the objects that we will draw on the screen and the timer data to determine how fast to cycle through the loop function.

4. Finish declaring game constants:

```
// Direction buttons for snake control
int left = 1;
int down = 2;
int up = 3;
int right = 4;

// Speed of the head of the snake
int xSpeed = 1;
int ySpeed = 0;

// Store location of all snake segments
int snakeMap[128][2];
// Index of random fruit location
int fruitLoc;
// Number of fruits eaten
int snakeSize = 0;
// List of all possible fruit locations, used to keep from spawning fruit on top of snake
int fruitLocs[128][2];
// Current fruit X and Y location
int fruitX;
int fruitY;
// Number of fruit remaining to be eaten before game ends
int fruitRemaining = 127;
```

These constants declare the pins for the buttons as well as important information while the snake is moving including direction and fruit information.

5. Input the following **setup()** function:

```
void setup() {
  // Start serial communication for debug
  Serial.begin(115200);
  // Start I2C communication with OLED display
  Wire.begin(8, 9);

  // Set snake starting location
  snakeMap[0][X_ARRAY] = 3;
  snakeMap[0][Y_ARRAY] = 3;

  // Find what locations are available for fruit to spawn
  GetGoodFruit();

  fruitLoc = random(0, fruitRemaining);
  fruitX = (int) (fruitLocs[fruitLoc][X_ARRAY]);
  fruitY = (int) (fruitLocs[fruitLoc][Y_ARRAY]);

  // Assign input buttons
  pinMode(left, INPUT);
  pinMode(down, INPUT);
  pinMode(up, INPUT);
  pinMode(right, INPUT);
  pinMode(45, INPUT);

  // I2C display
  display.init();
  display.setContrast(255);
}
```

In the setup() function, we begin the game by setting up the display and putting in the start location for the fruit and the snake.

6. Begin inputting the following **loop()** function:

```
void loop() {
  // Flag keeps track of if game has ended
  bool gameOver = false;

  // Timer to control game updating speed
  ++timerCount;
  if (timerCount > TIMER_MAX) {
  // Time to update game

  // Move snake body forward
    for (int i = snakeSize; i > 0; i--) {
      snakeMap[i][X_ARRAY] = snakeMap[i - 1][X_ARRAY];
      snakeMap[i][Y_ARRAY] = snakeMap[i - 1][Y_ARRAY];
    }
    snakeMap[0][X_ARRAY] += xSpeed;
    snakeMap[0][Y_ARRAY] += ySpeed;
```

We use this section of the loop to move the snake and use the time to regulate the movement.

7. Continue inputting the following **loop()** function:

```
// Determine if snake has collided with itself
   for(int i = 3; i <= snakeSize; i++){
      if(snakeMap[i][X_ARRAY] == snakeMap[0][X_ARRAY] &&
snakeMap[i][Y_ARRAY] == snakeMap[0][Y_ARRAY]){
         gameOver = true;
      }
   }
```

We use this section of the loop to determine if the snake has collided with itself

8. Continue inputting the following **loop()** function:

```
// Show game over text and score
   if(gameOver){
     display.clear();
     display.setTextAlignment(TEXT_ALIGN_CENTER);
     display.drawString(SCREEN_WIDTH/2, 16, "Game Over");
     display.drawString(SCREEN_WIDTH/2, 32, String("Score: ") + String(snakeSize));
     display.display();
     while (true) {
       delay(10);
     }
   }
```

In this section, we display the game over text if you lost.

9. Continue inputting the following **loop()** function:

```
// Determine if fruit has been eaten
   if (snakeMap[0][X_ARRAY] == fruitX && snakeMap[0][Y_ARRAY] == fruitY) {
     ++snakeSize;
     --fruitRemaining;

     // Show win text if all possible fruit has been eaten
     if(fruitRemaining <= 0){
     display.clear();
     display.setTextAlignment(TEXT_ALIGN_CENTER);
     display.drawString(SCREEN_WIDTH/2, 16, "WOW, You Win!!");
     display.drawString(SCREEN_WIDTH/2, 48, String("Score: ") + String(snakeSize));
     display.display();
     while (true) {
       delay(10);
     }
   }
```

Here, we have the win condition and add to the snake if a fruit is eaten.

10. **Continue inputting the following loop() function:**

```
    // Find possible fruit locations
    GetGoodFruit();

    fruitLoc = random(0, fruitRemaining);
    fruitX = (int) (fruitLocs[fruitLoc][X_ARRAY]);
    fruitY = (int) (fruitLocs[fruitLoc][Y_ARRAY]);

    // Use snake head forward
    snakeMap[snakeSize][X_ARRAY] = snakeMap[snakeSize - 1][X_ARRAY];
    snakeMap[snakeSize][Y_ARRAY] = snakeMap[snakeSize - 1][Y_ARRAY];
  }
```

We use this section to find possible fruit locations that the snake is not located at then move the snake forward.

11. **Continue inputting the following loop() function:**

```
    // Display
    display.clear();
    // Draw snake
    for (int i = 0; i <= snakeSize; i++) {
        display.fillRect(snakeMap[i][X_ARRAY] * BOX_SIZE,
snakeMap[i][Y_ARRAY] * BOX_SIZE, BOX_SIZE, BOX_SIZE);
    }
    // Draw fruit
    display.fillCircle(fruitX * BOX_SIZE + BOX_SIZE / 2, fruitY *
BOX_SIZE + BOX_SIZE / 2, BOX_SIZE / 2);
    // Draw screen outline
    display.drawRect(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT);
    display.display();

    timerCount = 0;
  }
```

This section draws the snake and fruits to the OLED display.

12. Finish inputting the following **loop()** function:

```
  // Get button input
  if (digitalRead(left) && xSpeed == 0) {
    xSpeed = -1;
    ySpeed = 0;
    timerCount = TIMER_MAX;
  } else if (digitalRead(right) && xSpeed == 0) {
    xSpeed = 1;
    ySpeed = 0;
    timerCount = TIMER_MAX;
  }
  else if (digitalRead(up) && ySpeed == 0) {
    ySpeed = -1;
    xSpeed = 0;
    timerCount = TIMER_MAX;
  } else if (digitalRead(down) && ySpeed == 0) {
    ySpeed = 1;
    xSpeed = 0;
    timerCount = TIMER_MAX;
  }

  delay(1);
}
```

This section determines which button is pressed and to move the snake in the correct direction accordingly.

13. Input the GetGoodFruit() function:

```
// Function to find what spaces are not covered by the snake to spawn fruit
void GetGoodFruit() {
  int numHits = 0;
  for (int i = 0; i < 128; i++) {
    bool snakeHit = false;
    int fruitX = (int) (i / 8);
    int fruitY = (int) (i - (fruitX * 8));
    for (int j = 0; j <= snakeSize; j++) {
      if (snakeMap[j][X_ARRAY] != fruitX || snakeMap[j][Y_ARRAY] != fruitY && !snakeHit)
      {
      } else if (!snakeHit) {
        ++numHits;
        snakeHit = true;
      }
    }
    if(!snakeHit){
      fruitLocs[i - numHits][X_ARRAY] = fruitX;
      fruitLocs[i - numHits][Y_ARRAY] = fruitY;
    }
  }
}
```

This function determines where to spawn the fruit based on where the snake is currently located.
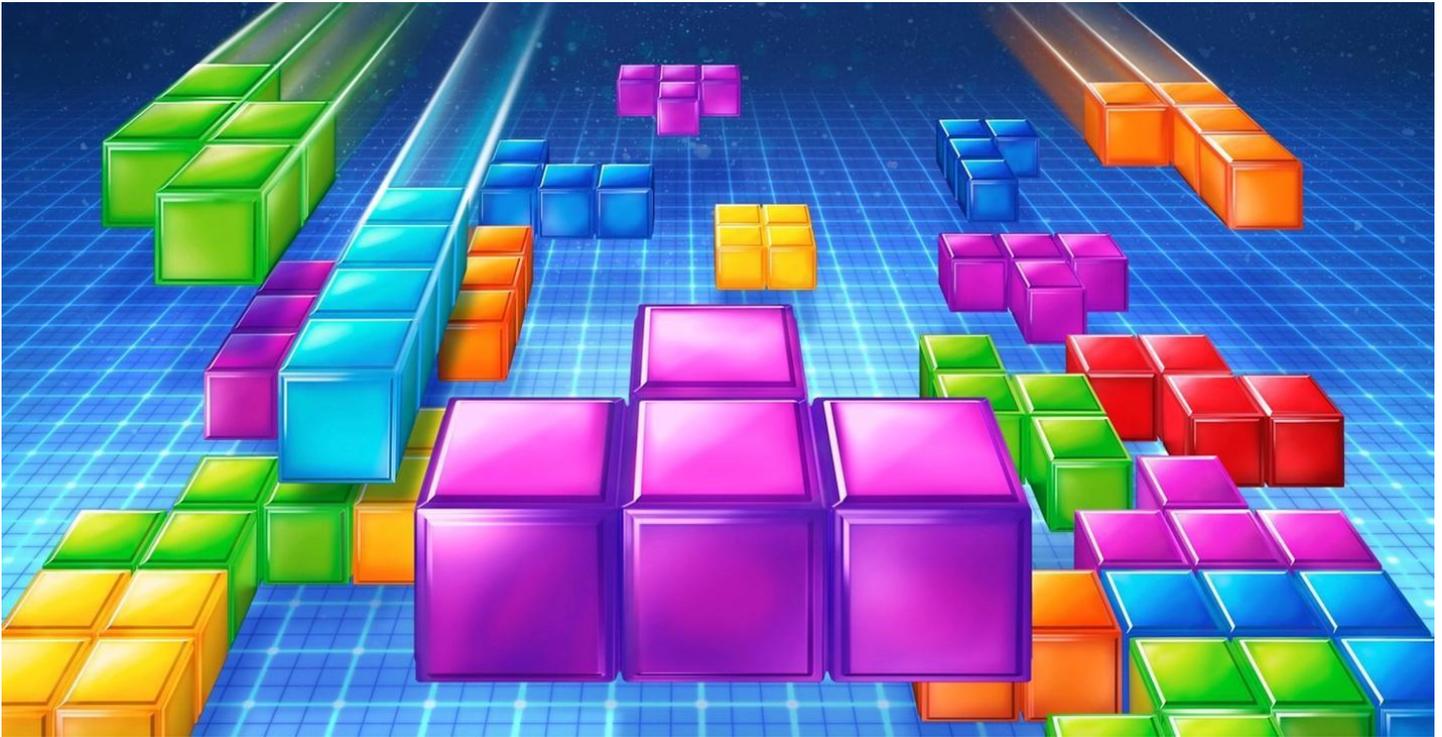
Result

Now, you can play the snake game utilizing the OLED display and four buttons on the Lenovo Educational Board.

Now It's Your Turn!

- Can you create pong game that utilizes the OLED display?

- Can you modify the game to change directions based on the tilt of the board (using the accelerometer) or the gesture sensor?

Lesson A – 9: OLED Tetris

OVERVIEW

In this lesson we will:

- Learn how to create a Tetris video game

Teacher Guide

For this lesson, students will be creating their own Tetris game utilizing the buttons and OLED display. This lesson will add to students' game design abilities.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

Tetris is a video game that was created 1984 by a Soviet software engineer. This game was designed to be a puzzle video game that included moving shapes that would fall to the bottom of the screen. Then, to score points players would complete lines horizontally until the screen was filled to the top. The goal was to score as many points as possible and players tend to score more points the longer that they survive.

This video game was built using simple rules that would make it one of the early great video games, especially since it was thought to improve intelligence. The Tetris video game is widely though of as one of the best-selling video games all time as even to this day there is millions of people who play the game.

Procedure

1. **Create a new blank sketch.**

2. **Input the following libraries and objects:**

```
#include <Wire.h>
#include "SSD1306Wire.h"

SSD1306Wire display(0x3c);
#define WIRE Wire
```

3. **Begin defining game constants:**

```
// Define game constants

// Width and height of screen
const int SCREEN_WIDTH = 128;
const int SCREEN_HEIGHT = 64;
// Game loop timer, lower value means faster movement
const int TIMER_ABS_MAX = 500;
const int TIMER_ABS_MIN = 150;
// How fast does the difficulty increase
const int TIMER_STEP = 10;
int timerMax = TIMER_ABS_MAX;
```

These constants are used to determine the speed of the game and the OLED screen size.

4. Finish defining game constants:

```
// Location in array for X and Y coordinates
const int X_ARRAY = 0;
const int Y_ARRAY = 1;
// Size of individual piece block
const int BLOCK_SIZE = 6;
// Max number of blocks that can fit on the screen
const int MAX_BLOCKS = 210;
const int MAX_BLOCKS_X = 10;
const int MAX_BLOCKS_Y = 21;
const int SHAPE_SIZE = 4;
const int NUM_BLOCK_TYPES = 7;
```

These game constants are used to determine number of blocks that can fit on the screen.

5. Input the following game variables:

```
int randCount = 0;
int randBlocks[7] = {0, 1, 2, 3, 4, 5, 6};

int score = 0;

int timerCount = timerMax;

int dropCount = timerMax;
int dropMax = timerMax;

bool lReady = true;
bool rReady = true;
bool uReady = true;
bool dReady = true;

// Number of blocks currently on the screen
int numBlocks = 0;
int lastBlock = -1;

// Direction buttons for rotation control
int left = 2;
int down = 4;
int up = 1;
int right = 3;
```

These game variables keep track of the timer, blocks on the screen, and buttons that were pressed.

6. **Input the following structs:**

```
struct playerBlock {
  bool shape[SHAPE_SIZE][SHAPE_SIZE];
  bool alive;
  int x;
  int y;
};

struct block {
  bool alive;
  int x;
  int y;
};

struct playerBlock myBlock;
```

These structs keep track of the current block that is dropping on the screen.

7. **Begin inputting the following bool arrays:**

```
bool otherBlocks[MAX_BLOCKS_Y][MAX_BLOCKS_X];

bool Square[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {0, 1, 1, 0},
  {0, 1, 1, 0},
  {0, 0, 0, 0}
};

bool TShape[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {0, 1, 0, 0},
  {1, 1, 1, 0},
  {0, 0, 0, 0}
};

bool LWiggle[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {1, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 0, 0}
};
```

These arrays keep track of all of the blocks on the screen and each type of block that can be randomly chosen.

8. Finish inputting the following bool arrays:

```
bool RWiggle[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {0, 1, 1, 0},
  {1, 1, 0, 0},
  {0, 0, 0, 0}
};

bool LShape[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {1, 0, 0, 0},
  {1, 1, 1, 0},
  {0, 0, 0, 0}
};

bool Bar[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {1, 1, 1, 1},
  {0, 0, 0, 0},
  {0, 0, 0, 0}
};

bool RevLShape[SHAPE_SIZE][SHAPE_SIZE] = {
  {0, 0, 0, 0},
  {0, 0, 1, 0},
  {1, 1, 1, 0},
  {0, 0, 0, 0}
};
```

These arrays keep track of each type of block that can be randomly chosen.

9. Input the following setup() function:

```
void setup() {
  // Start serial communication for debug
  Serial.begin(115200);
  // Start I2C communication with OLED display
  Wire.begin(8, 9);

  RandomShapeCopy(&myBlock);
  myBlock.alive = true;
  myBlock.x = 4;
  myBlock.y = 0;

  // Assign input buttons
  pinMode(left, INPUT);
  pinMode(down, INPUT);
  pinMode(up, INPUT);
  pinMode(right, INPUT);

  pinMode(45, INPUT);

  // I2C display
  display.init();
  display.setContrast(255);
}
```

This section setups the block to begin and sets up the buttons and OLED display.

10. **Begin inputting the following loop() function:**

```cpp
void loop() {
  // Flag keeps track of if game has ended
  bool gameOver = false;

  // Timer to control game updating speed
  ++timerCount;
  ++dropCount;

  if (dropCount > dropMax) {
    myBlock.y++;
    dropCount = 0;
    bool landed = false;
```

This section of the loop() function keeps track on the game update speed and whether the game has ended.

11. **Continue inputting the following loop() function:**

```cpp
    // Check if player shape has collided with floor
    for (int i = 0; i < SHAPE_SIZE; i++) {
      for (int j = 0; j < SHAPE_SIZE; j++) {
        if (myBlock.shape[i][j] && (myBlock.y + i >= MAX_BLOCKS_Y) && !landed) {
          landed = true;
        }
        // Check if player has collided with another block
        if (otherBlocks[myBlock.y + i - 1][myBlock.x + j - 1] && myBlock.shape[i][j]
&& !landed) {
          landed = true;
        }
      }
    }
```

Here, we check to see if a block has collided with the floor or another block.

12. Continue inputting the following **loop()** function:

```
    // If player shape has landed, convert to blocks and reset player shape
    if (landed) {
      myBlock.y--;
      for (int i = 0; i < SHAPE_SIZE; i++) {
        for (int j = 0; j < SHAPE_SIZE; j++) {
          if (myBlock.shape[i][j]) {
            otherBlocks[myBlock.y + i - 1][myBlock.x + j - 1] = true;
          }
        }
      }
      ClearFinishedLines();
      DrawScreen();
```

In this section, we add the block to the array with all of the block on the screen if the block has landed.

13. Continue inputting the following **loop()** function:

```
    // Check if player has lost
    for (int i = 0; i < MAX_BLOCKS_X; i++) {
      if (otherBlocks[0][i]) {
        display.clear();
        display.setTextAlignment(TEXT_ALIGN_CENTER);
        display.drawString(SCREEN_WIDTH / 2, 16, "Game Over");
        display.drawString(SCREEN_WIDTH / 2, 32, String("Score: ") + String(score));
        display.display();
        while (true) {
          delay(10);
        }
      }
    }
```

In this section, we check to see if the player has lost and display the result if so.

14. Continue inputting the following **loop()** function:

```
     // Reset block and increase speed
     myBlock.x = 4;
     myBlock.y = 0;
     RandomShapeCopy(&myBlock);
     if(timerMax > TIMER_ABS_MIN){
       timerMax -= TIMER_STEP;
     }
     if(dropMax > TIMER_ABS_MIN){
       dropMax -= TIMER_STEP;
     }
   }

 } else if (timerCount > timerMax) {
   // Time to update game
   DrawScreen();
   timerCount = 0;
 }
```

Here, we reset the blocks and increase the speed as well as update the game.

15. Continue inputting the following **loop()** function:

```
  // moves the block to the left
  if (digitalRead(left)) {
    if (lReady) {
      bool blocked = false;
      for (int i = 0; i < SHAPE_SIZE; i++) {
        for (int j = 0; j < SHAPE_SIZE; j++) {
          if (myBlock.shape[i][j] && ((myBlock.x + j - 2 < 0) ||
otherBlocks[myBlock.y + i - 1][myBlock.x + j - 2])) {
            blocked = true;
          }
        }
      }
      if (!blocked) {
        myBlock.x--;
      }
      timerCount = timerMax;
    }
    lReady = false;
  } else {
    lReady = true;
  }
```

In this section, we move the block to the left if the left button is pressed.

16. Continue inputting the following **loop()** function:

```cpp
// moves the block to the right
if (digitalRead(right)) {
    if (rReady) {
       bool blocked = false;
       for (int i = 0; i < SHAPE_SIZE; i++) {
          for (int j = 0; j < SHAPE_SIZE; j++) {
             if (myBlock.shape[i][j] && ((myBlock.x + j >= MAX_BLOCKS_X) ||
otherBlocks[myBlock.y + i - 1][myBlock.x + j])) {
                blocked = true;
             }
          }
       }
       if (!blocked) {
          myBlock.x++;
       }
       timerCount = timerMax;
    }
    rReady = false;
} else {
    rReady = true;
}
```

In this section, we move the block to the right if the right button is pressed.

17. Continue inputting the following **loop()** function:

```cpp
// rotates the block
if (digitalRead(up)) {
    if (uReady) {
       timerCount = timerMax;
       RotatePlayerShape();
    }
    uReady = false;
} else {
    uReady = true;
}
```

In this section of the loop(), we rotate the block if the up button is pressed.

18. Finish inputting the following **loop**() function:

```
  // moves the block down until it lands
  if (digitalRead(down)) {
    //if (dReady) {
      bool blocked = false;
      for (int i = 0; i < SHAPE_SIZE; i++) {
        for (int j = 0; j < SHAPE_SIZE; j++) {
          if (myBlock.shape[i][j] && ((myBlock.y + i + 1 >= MAX_BLOCKS_Y) ||
  otherBlocks[myBlock.y + i][myBlock.x + j - 1])) {
            blocked = true;
          }
        }
      }
      if (!blocked) {
        myBlock.y++;
      }
      timerCount = timerMax;
    dReady = false;
  } else {
    dReady = true;
  }

  delay(1);
}
```

In this section if the down button is pressed, we move the current block down until the button is released or it collides with another block.

19. Begin inputting the following ClearFinishedLines() function:

```
void ClearFinishedLines() {
  // clears a line if it is full
  int numLinesFinished = 0;
  bool lineFinished = true;

  for (int i = MAX_BLOCKS_Y - 1; i >= 0; i--) {
    if (i < (MAX_BLOCKS_Y - 1) && lineFinished) {
      ++numLinesFinished;
    }
    lineFinished = true;
    for (int j = 0; j < MAX_BLOCKS_X; j++) {
      if (numLinesFinished > 0) {
        otherBlocks[i + numLinesFinished][j] = otherBlocks[i][j];
      }
      if (!otherBlocks[i][j]) {
        lineFinished = false;
      }
    }
  }
}
```

In this function, we clear a horizontal line if there is blocks all the way across.

20. Finish inputting the following ClearFinishedLines() function:

```
  // Incrememt score, clearing more lines at once gives higher score
  if (numLinesFinished == 1) {
    score += 40;
  } else if (numLinesFinished == 2) {
    score += 100;
  } else if (numLinesFinished == 3) {
    score += 300;
  } else if (numLinesFinished >= 4) {
    score += 1200;
  }
}
```

In this function, we add to the score for finished horizontal lines.

21. Begin inputting the following DrawScreen() function:

```
void DrawScreen() {
  // Display
  display.clear();

  // Draw shape being controlled
  for (int i = 0; i < SHAPE_SIZE; i++) {
    for (int j = 0; j < SHAPE_SIZE; j++) {
      if (myBlock.shape[j][i]) {
        display.drawRect((myBlock.y + j) * BLOCK_SIZE + 1, SCREEN_HEIGHT
- ((myBlock.x + i) * BLOCK_SIZE) - 2, BLOCK_SIZE, BLOCK_SIZE);
      }
    }
  }
```

In this part of the function, we draw the current shape being controlled.

22. Finish inputting the following DrawScreen() function:

```
  // Draw all blocks that have settled
  for (int i = 0; i < MAX_BLOCKS_Y; i++) {
    for (int j = 0; j < MAX_BLOCKS_X; j++) {
      if (otherBlocks[i][j]) {
        display.drawRect((i + 1) * BLOCK_SIZE + 1, SCREEN_HEIGHT - (j + 1)
* BLOCK_SIZE - 2, BLOCK_SIZE, BLOCK_SIZE);
      }
    }
  }
  // Draw screen outline
  display.drawRect(0, 1, SCREEN_WIDTH, SCREEN_HEIGHT - 2);
  display.display();
}
```

In this part of the function, we draw the blocks that have settled and the screen outline.

23. Input the following ShapeCopy() function:

```
void ShapeCopy(struct playerBlock *dest, bool shape[SHAPE_SIZE][SHAPE_SIZE]) {
  // copies a shape
  for (int i = 0; i < SHAPE_SIZE; i++) {
    for (int j = 0; j < SHAPE_SIZE; j++) {
      dest->shape[i][j] = shape[i][j];
    }
  }
}
```

This function copies a shape.

24. Begin inputting the following RandomShapeCopy() function:

```
void RandomShapeCopy(struct playerBlock *dest) {
  // copies a random shape
  int shape = random(NUM_BLOCK_TYPES - randCount);
  switch (randBlocks[shape]) {
    case 0:
      ShapeCopy(&myBlock, Square);
      break;
    case 1:
      ShapeCopy(&myBlock, TShape);
      break;
    case 2:
      ShapeCopy(&myBlock, LWiggle);
      break;
    case 3:
      ShapeCopy(&myBlock, RWiggle);
      break;
    case 4:
      ShapeCopy(&myBlock, LShape);
      break;
    case 5:
      ShapeCopy(&myBlock, Bar);
      break;
    case 6:
      ShapeCopy(&myBlock, RevLShape);
      break;
  }
```

In this part of the function, we chose a copy of a random shape.

25. Finish inputting the following RandomShapeCopy() function:

```
  if (randCount < NUM_BLOCK_TYPES) {
     for (int i = shape; i < NUM_BLOCK_TYPES - randCount; i++) {
       randBlocks[i] = randBlocks[i + 1];
     }
  } else {
     for (int i = 0; i < NUM_BLOCK_TYPES; i++) {
       randBlocks[i] = i;
     }
     randCount = 0;
  }
}
```

In this part of the function, we help chose the random shape.

26. Begin inputting the following RotatePlayerShape() function:

```
void RotatePlayerShape() {
  bool tempArray[SHAPE_SIZE][SHAPE_SIZE];
  bool temp[SHAPE_SIZE][SHAPE_SIZE];
  // rotates the shape
  for (int i = 0; i < SHAPE_SIZE; i++) {
    for (int j = 0; j < SHAPE_SIZE; j++) {
      tempArray[SHAPE_SIZE - j - 1][i] = myBlock.shape[i][j];
      temp[i][j] = myBlock.shape[i][j];
    }
  }
  for (int i = 0; i < SHAPE_SIZE; i++) {
    for (int j = 0; j < SHAPE_SIZE; j++) {
      myBlock.shape[i][j] = tempArray[i][j];
      while (myBlock.shape[i][j] && (myBlock.x + j - 1 < 0) ) {
        myBlock.x++;
      }
      while (myBlock.shape[i][j] && (myBlock.x + j - 1 >= MAX_BLOCKS_X) ) {
        myBlock.x--;
      }
      while (myBlock.shape[i][j] && (myBlock.y + i - 1 >= MAX_BLOCKS_Y) ) {
        myBlock.y--;
      }
    }
  }
```

In this part of the function, we rotate the shape.

27. Finish inputting the following RotatePlayerShape() function:

```
// checks if shape can be rotated or if shape will clip into other shapes
bool landed = false;
// Check if player shape has collided with floor
for (int i = 0; i < SHAPE_SIZE; i++) {
  for (int j = 0; j < SHAPE_SIZE; j++) {
    if (myBlock.shape[i][j] && (myBlock.y + i >= MAX_BLOCKS_Y) && !landed) {
      landed = true;
    }
    // Check if player has collided with another block
    if (otherBlocks[myBlock.y + i - 1][myBlock.x + j - 1] && myBlock.shape[i][j]
&& !landed) {
      landed = true;
    }
  }
}
if(landed)
{
  for (int i = 0; i < SHAPE_SIZE; i++) {
    for (int j = 0; j < SHAPE_SIZE; j++) {
      myBlock.shape[i][j] = temp[i][j];
    }
  }
}
}
```

In this part of the function, we prevent the chape rotation if it will clip off part of the chape by rotating.

Result

Now, you should be able to play Tetris using the left and right buttons (pin 2 and 3) to move the block side to side and the up button (pin 1) to rotate the block. See the picture below for how to hold the board while playing.
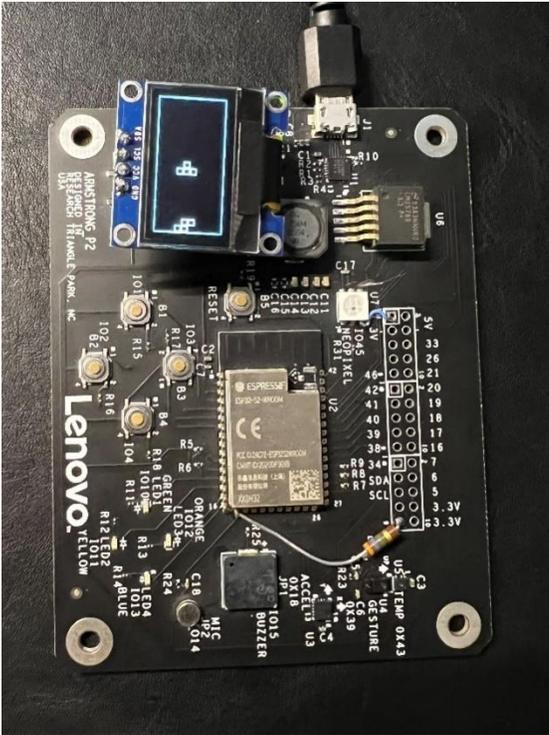
Now It's Your Turn!

- Can you modify Tetris to include different shapes that are not in the original game?

- Can you add new features to Tetris to make it more interesting?

Lesson A – 10: Morse Code Walkie Talkies

OVERVIEW

In this lesson we will:

- Learn what a web socket is

- Create a walkie talkie utilizing a web socket

- Learn more about morse code

Teacher Guide

In this lesson, the students will be interacting with a web socket which will be explained in the background section below. Additionally, for this lesson, students will need to work in pairs and each student will have a different section of code to input. Also, please make sure to have your Wi-Fi name and password ready to give students access to create a web socket.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Background Info

How does a walkie talkie work?

A walkie talkie is a hand-held two-way radio with a transmitter and a receiver. It was designed during World War II for military uses but has since been used for other jobs across industries. A walkie talkie is very similar to a cell phone where it has a microphone and a speaker built into the device with an antenna on top. When one is ready to talk, they hold the device up to their face a press a button to talk into the microphone which is then transmitted to other radios tuned to the same channel via radio waves. When the button is pushed to talk, only one person on the channel can speak, but all of the other radios can hear what is being said via the receiver in the antenna that outputs the sound to the speaker.

What is a web socket?

For this lesson, we will utilize a web socket to communicate across two devices. A web socket allows two-way communication between a client and a server where responses and actions are event driven by the text that is received. Both the client and server devices must be connected to the same Wi-Fi network with the client knowing the IP address of the server to be able to communicate between each other utilizing the radio waves from the Wi-Fi network.
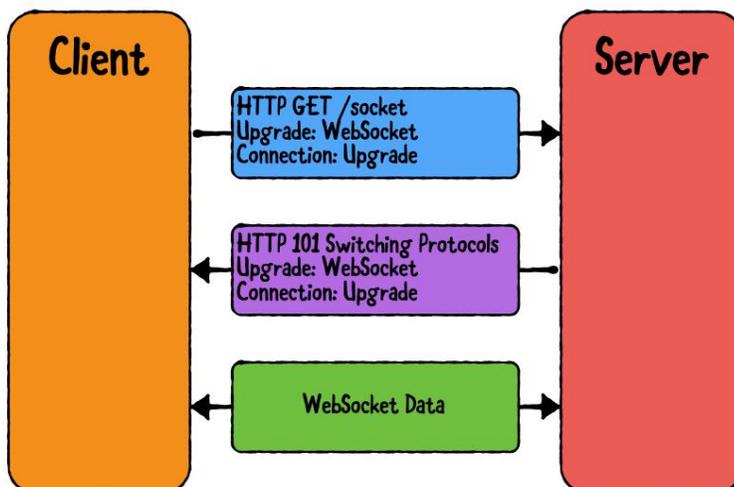
Student 1 Procedure (Server)

1. **Create a new blank sketch.**

2. **Include and download the following libraries:**

```
#include <WiFi.h>
#include <WiFiMulti.h>
#include <WiFiClientSecure.h>
#include <WebSocketsServer_Generic.h>
#include <Preferences.h>
```

3. Define the following web socket constants:

```
// define web socket information
#define WS_PORT 8080
#define  WEBSOCKETS LOGLEVEL  2
```

4. Input the following objects:

```
WiFiMulti WiFiMulti;
WebSocketsServer  webSocket = WebSocketsServer(WS_PORT);
Preferences preferences;
```

5. Input the following constants and variables:

```
// define the LED pin
#define PIN_OUT 10
// define the unit length in ms
#define UNIT_LENGTH 250
// define the button pin to input morse code
#define BUTTON 1
// define the button pin to read morse code
const int buttonTwo = 2;
// define the buton to listen
const int buttonThree = 3;
// determines whether to output to the LED
bool complete = false;
// message in letters
String finalMsg = "";
// used to tell where to send to
uint8_t nums = 0;
// used for translation to and from morse code
static String codex = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890 .,?!:;()@&";
String ssid = "";
String password = "";
```

6. Input the following dots_and_dashes array:

```java
// array where the translations to morse code are stored
static String dots_and_dashes[] = {
  ".-",
  "-...",
  "-.-.",
  "-..",
  ".",
  "..-.",
  "--.",
  "....",
  "..",
  ".---",
  "-.-",
  ".-..",
  "--",
  "-.",
  "---",
  ".--.",
  "--.-",
  ".-.",
  "...",
  "-",
  "..-",
  "...-",
  ".--",
  "-..-",
  "-.--",
  "--..",
  ".----",
  "..---",
  "...--",
  "....-",
  ".....",
  "-....",
  "--...",
  "---..",
  "----.",
  "-----",
  "        ",
  ".-.-.-",
  "--..--",
  "..--..",
  "-.-...",
  "---...",
  "-.-.-.",
  "-.--.",
  "-.--.-",
  ".--.-.",
  ".-...",
};
```

This array stores the morse code translations.

7. Input the following webSocketEvent() function:

```
// receives an event from websocket and processes whether event is text,
disconnected, or connnected
void webSocketEvent(const uint8_t& num, const WStype_t& type, uint8_t * payload,
const size_t& length)
{
  nums = num;
  switch (type)
  {
    case WStype_DISCONNECTED:
    {
      Serial.printf("Disconnected!\n");
      break;
    }
    case WStype_CONNECTED:
      {
        IPAddress ip = webSocket.remoteIP(num);
        Serial.printf("Connected from %d.%d.%d.%d\n", ip[0], ip[1], ip[2], ip[3]);
        break;
      }
    case WStype_TEXT:
    {
      Serial.println();
      Serial.printf("Text Recieved: %s\n", payload);
      String str = (char*)payload;
      outputToLED(str);
      // send data to all connected clients
      // webSocket.broadcastTXT(num, messageFromServer);
      break;
    }
    default:
    {
      break;
    }
  }
}
```

This function determines the event type and chooses the correct case accordingly.

8. Input the getWifi() function to read WiFi information from the EEPROM:

```
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

9. Input the following **setup()** function:

```
void setup()
{
  // set baud rate
  Serial.begin(115200);
  getWifi();
  // sets pins to appropriate mode
  pinMode(BUTTON, INPUT);
  pinMode(buttonTwo, INPUT);
  pinMode(buttonThree, INPUT);
  pinMode(PIN_OUT, OUTPUT);
  digitalWrite(PIN_OUT, LOW);
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  delay(200);
  // connect to Wi-Fi
  WiFiMulti.addAP(ssid.c_str(), password.c_str());
  while (WiFiMulti.run() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(100);
  }
  Serial.println();
  // begin websocket
  webSocket.begin();
  webSocket.onEvent(webSocketEvent);
  // server address and port
  Serial.print("Please put the following IP address in the client sketch: ");
  Serial.print(WiFi.localIP());
  Serial.print(", port: "); Serial.println(WS_PORT);
}
```

The setup() function sets the appropriate pinModes, connects to Wi-Fi, and starts the web socket.

10. Input the following **loop()** function:

```
void loop(){
  // need to be in this mode to receive a message
  while(digitalRead(buttonThree) != HIGH)
  {
    webSocket.loop();
  }
  // press button two after inputting a character to send message
  while (digitalRead(buttonTwo) != HIGH)
  {
    String myMsg = GetLetter();
    Translate(myMsg);
  }
  // send text across web socket
  webSocket.sendTXT(nums, finalMsg);
  finalMsg = "";
  Serial.println();
```

The loop() function is in listening mode until button 3 is pressed and then it looks for input and then once input is finished, it will send the text.

11. **Input the following outputToLED() function:**

```
// outputs string to LEDs in morse code
void outputToLED(String message)
{
  String morseWord = encode(message);
  for(int i = 0; i <= morseWord.length(); i++)
  {
    switch(morseWord[i])
    {
      case '.': // dit
        digitalWrite(PIN_OUT, HIGH);
        delay(UNIT_LENGTH);
        digitalWrite(PIN_OUT, LOW);
        delay(UNIT_LENGTH);
        break;

      case '-': // dah
        digitalWrite(PIN_OUT, HIGH);
        delay(UNIT_LENGTH * 3);
        digitalWrite(PIN_OUT, LOW);
        delay(UNIT_LENGTH);
        break;

      case ' ': // gap
        delay(UNIT_LENGTH);
    }
  }
  digitalWrite(12, HIGH);
  delay(UNIT_LENGTH);
  digitalWrite(12, LOW);
  delay(UNIT_LENGTH * 3);
}
```

This function outputs the text to the LED in morse code.

12. **Input the following GetLetter() function:**

```cpp
// reads input from button to morse code
String GetLetter()
{
  unsigned long START, STOP, TIME;
  String myMsg = "";
  while(true)
  {
    while(digitalRead(BUTTON) == LOW)
    {
      // do nothing until button is pressed
    }
    digitalWrite(PIN_OUT, HIGH);
    START = millis();
    while(digitalRead(BUTTON) == HIGH)
    {
      // do not register input until button is released
    }
    STOP = millis();
    digitalWrite(PIN_OUT, LOW);
    TIME = STOP - START;
    if (TIME > 25)
    {
      myMsg += Dot_or_Dash(TIME);
    }
    while((millis() - STOP) < 750)
    {
      delay(10);
      if (digitalRead(BUTTON) == HIGH)
      {
        break;
      }
    }
    if ((millis() - STOP) >= 750)
    {
      return myMsg;
    }
  }
}
```

This function reads the input from the button into morse code.

13. Input the following Dot_or_Dash() function:

```
// determine if it is a dot or dash
char Dot_or_Dash (unsigned long TIME)
{
  char DoD;
  if (TIME < 500)
  {
    DoD = '.';
  }
  else if (TIME >= 500 && TIME < 5000)
  {
    DoD = '-';
  }
  else
  {
    DoD = ' ';
  }
  return DoD;
}
```

This function determines if a dot or dash is inputted based on how long the button is held.

14. Input the following Translate() function:

```
// translates morse code to letters
void Translate (String myMsg)
{
  for (int i = 0; i < 47; i++)
  {
    if (myMsg == dots_and_dashes[i])
    {
      Serial.print(codex[i]);
      finalMsg += codex[i];
      return ;
    }
  }
  Serial.print("ERROR, I cannot understand you!\n");
  return ;
}
```

This function translates morse code into an individual character.

15. Input the following Encode() function:

```cpp
// translates letters to morse code
String encode(String string)
{
    size_t i, j;
    String morseWord = "";
    for( i = 0; string[i]; i++ )
    {
        for( j = 0; j < 47; j++ )
        {
            if( string[i] == codex[j] )
            {
                morseWord += dots_and_dashes[j];
                break;
            }
        }
        morseWord += " "; // Add tailingspace to separate the chars
    }
    return morseWord;
}
```

This function translates an individual letter into morse code.

Student 2 Procedure (Client)

1. Create a new blank sketch.

2. Include and download the following libraries:

```cpp
#include <WiFi.h>
#include <WiFiMulti.h>
#include <WiFiClientSecure.h>
#include <WebSocketsClient_Generic.h>
#include <Preferences.h>
```

3. Input the following objects:

```cpp
WiFiMulti WiFiMulti;
WebSocketsClient  webSocket;
Preferences preferences;
```

4. Define the following web socket constants:

```
// change this to the ip adress printed to the serial on the server
#define WS_SERVER "XXX.XXX.X.XX"
#define WS_PORT 8080
#define _WEBSOCKETS_LOGLEVEL_ 2
```

Please be sure to input the IP address from the server here before uploading.

5. Input the following constants and variables:

```
// define the LED pin
#define PIN_OUT 10
// define the unit length in ms
#define UNIT_LENGTH 250
// define the button pin to input morse code
#define BUTTON 1
// define the button pin to read morse code
const int buttonTwo = 2;
// define the buton to listen
const int buttonThree = 3;
// determines whether to output to the LED
bool complete = false;
// message in letters
String finalMsg = "";
// used for translation to and from morse code
static String codex = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890 .,?!:;()@&";
String ssid = "";
String password = "";
```

6. Input the following dots_and_dashes array:

```
// array where the translations to morse code are stored
static String dots_and_dashes[] = {
  ".-",
  "-...",
  "-.-.",
  "-..",
  ".",
  "..-.",
  "--.",
  "....",
  "..",
  ".---",
  "-.-",
  ".-..",
  "--",
  "-.",
  "---",
  ".--.",
  "--.-",
  ".-.",
  "...",
  "-",
  "..-",
  "...-",
  ".--",
  "-..-",
  "-.--",
  "--..",
  ".----",
  "..---",
  "...--",
  "....-",
  ".....",
  "-....",
  "--...",
  "---..",
  "----.",
  "-----",
  "        ",
  ".-.-.-",
  "--..--",
  "..--..",
  "-.-...",
  "---...",
  "-.-.-.",
  "-.--.",
  "-.--.-",
  ".--.-.",
  ".-...",
};
```

This array stores the morse code translations.

7. Input the following webSocketEvent() function:

```cpp
bool alreadyConnected = false;
// receives an event from websocket and processes whether event is text,
disconnected, or connnected
void webSocketEvent(const WStype_t& type, uint8_t * payload, const size_t& length)
{
  switch (type)
  {
    case WStype_DISCONNECTED:
    {
      if (alreadyConnected)
      {
        Serial.println("Disconnected!");
        alreadyConnected = false;
      }
      break;
    }
    case WStype_CONNECTED:
    {
        alreadyConnected = true;
        Serial.println("Connected!");
        break;
    }
    case WStype_TEXT:
    {
      Serial.println();
      Serial.printf("Text Recieved: %s\n", payload);
      String str = (char*)payload;
      outputToLED(str);
      break;
    }
    default:
    {
      break;
    }
  }
}
```

This function determines the event type and chooses the correct case accordingly.

8. Input the getWifi() function to read WiFi information from the EEPROM:

```cpp
// get Wifi info
void getWifi()
{
  preferences.begin("credentials", false);
  ssid = preferences.getString("ssid", "");
  password = preferences.getString("password", "");
  if (ssid == "" || password == ""){
    Serial.println("No ssid or password saved.");
  }
  else {
    Serial.println("Successfully read WiFi information.");
  }
}
```

9. Begin inputting the following **setup**() function:

```
void setup()
{
  Serial.begin(115200);
  getWifi();
  // sets pins to appropriate mode
  pinMode(BUTTON, INPUT);
  pinMode(buttonTwo, INPUT);
  pinMode(buttonThree, INPUT);
  pinMode(PIN_OUT, OUTPUT);
  digitalWrite(PIN_OUT, LOW);
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  delay(200);
  Serial.setDebugOutput(true);
  // connect to Wi-Fi
  WiFiMulti.addAP(ssid.c_str(), password.c_str());
  while (WiFiMulti.run() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(100);
  }
```

The setup() function sets the appropriate pinMode and connects to Wi-Fi.

10. Finish inputting the following **setup**() function:

```
  Serial.println();
  // Client address
  Serial.print("WebSockets Client started @ IP address: ");
  Serial.println(WiFi.localIP());
  // server address and port
  Serial.print("Connecting to WebSockets Server @ ");
  Serial.println(WS_SERVER);
  // begin web socket
  webSocket.begin(WS_SERVER, WS_PORT, "/");
  webSocket.onEvent(webSocketEvent);
  // try 5 seconds if no connection
  webSocket.setReconnectInterval(5000);
  // connected to web socket server
  Serial.print("Connected to WebSockets Server @ IP address: ");
  Serial.println(WS_SERVER);
}
```

This section of the setup() function starts the web socket.

11. Input the following **loop()** function:

```
void loop(){
  // need to be in this mode to receive a message
  while(digitalRead(buttonThree) != HIGH)
  {
    webSocket.loop();
  }
  // press button two after inputting a character to send message
  while (digitalRead(buttonTwo) != HIGH)
  {
    String myMsg = GetLetter();
    Translate(myMsg);
  }
  webSocket.sendTXT(finalMsg);
  finalMsg = "";
  Serial.println();
}
```

The loop() function is in listening mode until button 3 is pressed and then it looks for input and then once input is finished, it will send the text.

12. Input the following **outputToLED()** function:

```
// outputs string to LEDs in morse code
void outputToLED(String message)
{
  String morseWord = encode(message);
  for(int i = 0; i <= morseWord.length(); i++)
  {
    switch(morseWord[i])
    {
      case '.': // dit
        digitalWrite(PIN_OUT, HIGH);
        delay(UNIT_LENGTH);
        digitalWrite(PIN_OUT, LOW);
        delay(UNIT_LENGTH);
        break;

      case '-': // dah
        digitalWrite(PIN_OUT, HIGH);
        delay(UNIT_LENGTH * 3);
        digitalWrite(PIN_OUT, LOW);
        delay(UNIT_LENGTH);
        break;

      case ' ': // gap
        delay(UNIT_LENGTH);
    }
  }
  digitalWrite(12, HIGH);
  delay(UNIT_LENGTH);
  digitalWrite(12, LOW);
  delay(UNIT_LENGTH * 3);
}
```

This function outputs the text to the LED in morse code.

13. **Input the following GetLetter() function:**

```
// reads input from button to morse code
String GetLetter()
{
  unsigned long START, STOP, TIME;
  String myMsg = "";
  while(true)
  {
    while(digitalRead(BUTTON) == LOW)
    {
      // do nothing until button is pressed
    }
    digitalWrite(PIN_OUT, HIGH);
    START = millis();
    while(digitalRead(BUTTON) == HIGH)
    {
      // do not register input until button is released
    }
    STOP = millis();
    digitalWrite(PIN_OUT, LOW);
    TIME = STOP - START;
    if (TIME > 25)
    {
      myMsg += Dot_or_Dash(TIME);
    }
    while((millis() - STOP) < 750)
    {
      delay(10);
      if (digitalRead(BUTTON) == HIGH)
      {
        break;
      }
    }
    if ((millis() - STOP) >= 750)
    {
      return myMsg;
    }
  }
}
```

This function reads the input from the button into morse code.

14. Input the following Dot_or_Dash() function:

```c
// determine if it is a dot or dash
char Dot_or_Dash (unsigned long TIME)
{
  char DoD;
  if (TIME < 500)
  {
    DoD = '.';
  }
  else if (TIME >= 500 && TIME < 5000)
  {
    DoD = '-';
  }
  else
  {
    DoD = ' ';
  }
  return DoD;
}
```

This function determines if a dot or dash is inputted based on how long the button is held.

15. Input the following Translate() function:

```c
// translates morse code to letters
void Translate (String myMsg)
{
  for (int i = 0; i < 47; i++)
  {
    if (myMsg == dots_and_dashes[i])
    {
      Serial.print(codex[i]);
      finalMsg += codex[i];
      return ;
    }
  }
  Serial.print("ERROR, I cannot understand you!\n");
  return ;
}
```

This function translates morse code into an individual character.

16. Input the following Encode() function:

```
// translates letters to morse code
String encode(String string)
{
  size_t i, j;
  String morseWord = "";
  for( i = 0; string[i]; i++ )
  {
    for( j = 0; j < 47; j++ )
    {
      if( string[i] == codex[j] )
      {
        morseWord += dots_and_dashes[j];
        break;
      }
    }
    morseWord += " "; // Add tailingspace to separate the chars
  }
  return morseWord;
}
```

This function translates an individual letter into morse code.

Result

Now, you can communicate with each other using the button at pin 3 to receive messages, button at pin 1 to input messages, and the button at pin 2 to send messages. Please note to upload the code for the server section first and to input the IP address into the client sketch before uploading the client.

Now It's Your Turn!

- Can you send weather data to each other across the web socket to compare results?

- Can you do a health check that flashes Neopixel red if client or server is disconnected and flash Neopixel green if client and server are connected?

Lesson A – 11: Don't Stop Dreaming!

OVERVIEW

In this lesson we will:

- Provide ideas for future Arduino projects

Teacher Guide

In this lesson, the students will learn about future project ideas that they can develop.

Tips on Troubleshooting:

If for some reason your board does not produce the expected output, you can troubleshoot by looking at the following factors:

1. **Board Connectivity**

   Is your board connected to your computer?

2. **COM Port**

   Is the IDE configuration set to the right COM port?

3. **PIN number**

   Did you define the PIN number correctly in your program?

Procedure

1. **Create a Pong game**

2. **Create a Simon Says game**

3. **Access the API on NASA's website to flash red when the ISS is overhead**

4. **Utilize IFTTT to create smart home automations**

5. **Create radar detection**

Result

Now, you have some more ideas for projects that can be created using the Lenovo Educational Board. Please feel free to search Google for additional ideas.