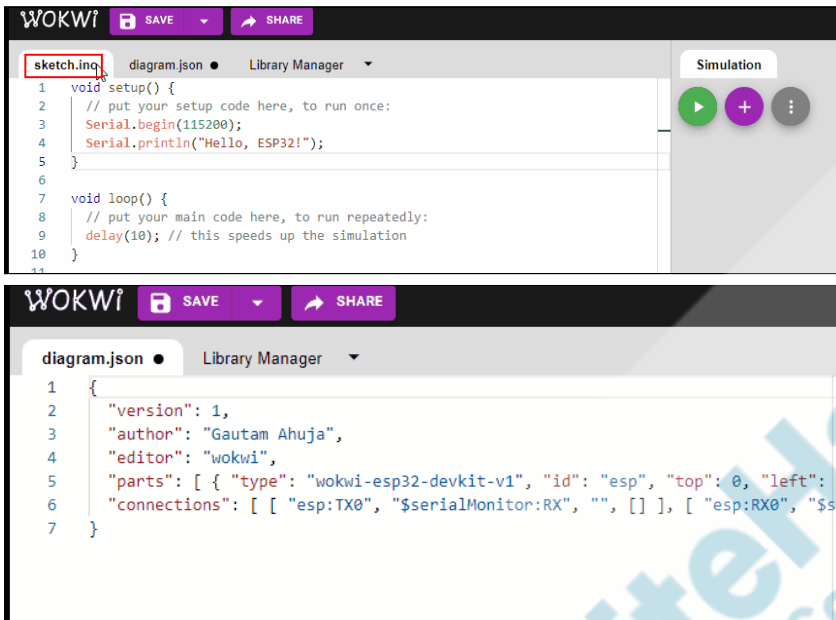


Topic	Smart Clock IV	
Class Description	Students will learn how to add timer and alarm clock features to the smart clock.	
Class	PRO C268	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> Learn to take input from users using an encoder. Learn to code timers and alarm clocks for smart clocks. 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Smartphone Student Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen 	
Class structure	Warm-Up Teacher-Led Activity Student-Led Activity Wrap-Up	10 mins 15 mins 15 mins 10 mins
WARM-UP SESSION - 10 mins		
Teacher Action		Student Action
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? Following are the WARM-UP session deliverables: <ul style="list-style-type: none"> Greet the student. 		ESR: Hi, thanks! Yes, I am excited about it! Click on the slide show tab and present the slides

<ul style="list-style-type: none"> • Revision of previous class activities. • Quizzes. 	
<p align="center">WARM-UP QUIZ Click on In-Class Quiz</p>	
<p>Activity Details</p> <p>Following are the session deliverables:</p> <ul style="list-style-type: none"> • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring more interest in students. 	
<p align="center">TEACHER-LED ACTIVITY 15 mins</p>	
<p align="center">Teacher Initiates Screen Share</p>	
<ul style="list-style-type: none"> • Learn to take input from users using Encoder 	
Teacher Action	Student Action
<p>Do you remember what we learned in the previous class?</p> <p>Can you tell me how we achieved it?</p> <p>Great. You are revising very well.</p> <p>Do you have any questions from the previous class?</p> <p><i>Note: If the student has any doubts, clarify the doubts.</i></p>	<p>ESR: Yes.</p> <p>ESR: Varied.</p> <p>ESR: Varied</p>
<p>What more features can we add to our smart clock?</p> <p>Yes. So, let's get started.</p> <p>Open the wokwi simulator and replace all the files</p>	<p>ESR: Alarm clock, Timer.</p>

downloaded from [Teacher Activity 1](#).

Note: Refer to Lesson C260.



The screenshot shows the Wokwi IDE interface. The top panel displays the 'sketch.ino' file with the following code:

```
1 void setup() {
2   // put your setup code here, to run once:
3   Serial.begin(115200);
4   Serial.println("Hello, ESP32!");
5 }
6
7 void loop() {
8   // put your main code here, to run repeatedly:
9   delay(10); // this speeds up the simulation
10 }
11
```

The bottom panel displays the 'diagram.json' file with the following JSON code:

```
1 {
2   "version": 1,
3   "author": "Gautam Ahuja",
4   "editor": "wokwi",
5   "parts": [ { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 0, "left": 0
6   "connections": [ [ "esp:TX0", "$serialMonitor:RX", "", [] ], [ "esp:RX0", "$se
7 }
```

Note: Follow the below steps and involve the student as well while doing so.

We will start with an alarm clock. Can you tell me how an alarm clock works?

ESR: Varied.

Yes. User sets the time they want to be alerted. Then, when this time matches the real-time, the clock plays a sound to alert.

So we need input from the user to ask about the alarm time. Can we accept input from the user using the hardware we already have?

ESR: Varied.

Yes, the input can be taken using an **encoder** and the selected value can be printed on the **LCD** device.

How many times and for what different inputs will we ask the user?

ESR: Varied.

Yes, we are supposed to, again and again, ask for hours, minutes, and seconds in our program.

Is there a way to efficiently use the code?

ESR: Varied.

Yes. You are right. We can use functions.

Let's create a function **set_value()** to store input entered by the user.

Also, to make sure the input is valid and since the upper and lower limit for the hour, minute, and seconds differ, we use **min_val** and **max_val** variables as parameters to this function.

This function also returns the value saved hence the return type is **int**.

```
int set_value(int min_val , int max_val){  
  
}
```

Initially the value must be 0.

```
int value = 0;
```

The user will choose the desired value using an encoder. Do we have any function that stores the value selected by the encoder?

ESR: Varied.

Yes. **encoder()**

ESR: Varied.

Which variable is updated by it?

Yes, the **counter** variable.

So we reset the counter to 0.

```
counter = 0;
```

Then in a while loop until the user presses the push button, we print the selected input(**counter** variable) on the display.

```
while (true){  
    lcd_print(0,1,"  ");  
    lcd_print(0,1,String(counter));  
}
```

The counter variable range will keep on changing depending on the type of user input .i.e. Hour range is 0-23 and so on. Hence, we constrain it using the input parameters **min_val** and **max_val**.

```
counter = constrain(counter , min_val ,  
max_val);
```

Also, we break the **loop** if the push button is pressed.

```
button.loop();  
if (button.isPressed())  
    break;
```

Lastly, we update the **variable value** to **counter** (the input from the encoder), return **value** and **counter** must be set back to 0 to enable the mode selection of the smart clock.

```
value = counter;  
counter = 0;  
return value;
```

Reference Code:

```
int set_value(int min_val , int max_val){  
  
    int value = 0;  
    counter = 0;  
  
    while(true){  
        button.loop();  
        if (button.isPressed())break;  
  
        counter = constrain(counter , min_val ,  
max_val);  
  
        lcd_print(0,1," ");  
        lcd_print(0,1,String(counter));  
  
    }  
  
    value = counter;  
    counter = 0;  
  
    return value;  
}
```

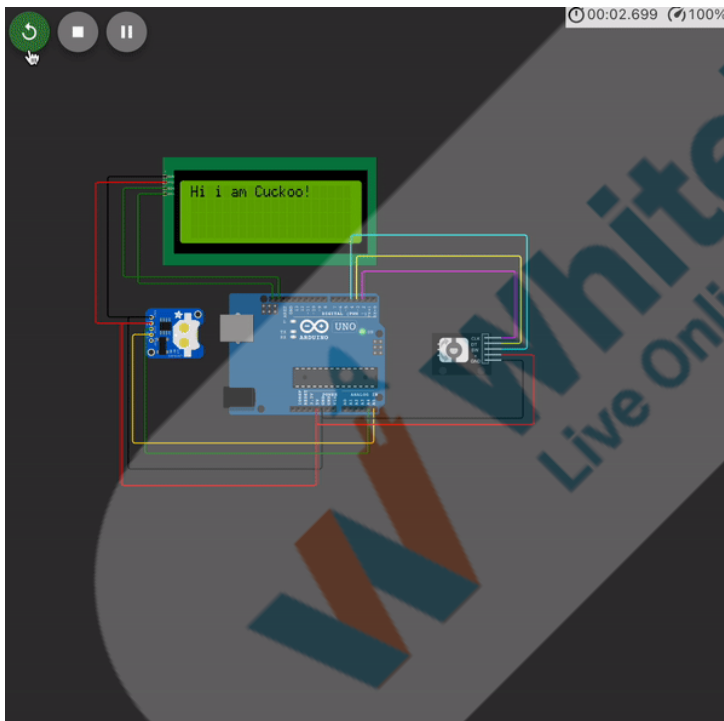
Let's check this method by asking for input in the **set_alarm()** function.

Also, we will need variables to store.

```
// alarm variables  
int alarm_hours = 0;  
int alarm_minutes = 0;
```

```
void set_alarm() {  
  lcd_print(0,0,"Enter hours : ");  
  alarm_hours = set_value(0,23);  
  Serial.print(alarm_hours);  
}
```

Reference Output:



<https://s3-whjr-curriculum-uploads.whjr.online/a33a8686-5334-4267-a45a-61e9155deee9.gif>

We observe two issues here. Can you tell me what are they?

ESR: Varied.

Yes. First, it flickers a lot. Let's fix that.

Do you remember from last class how we resolved the flickering issue?

Yes, by creating another variable we can resolve this issue.

Let's do so. We create the variable **last_counter** and it must be different than the actual **counter** variable showing that the user has rotated the encoder and changed it and hence change the display.

```
int last_counter = -1;
```

```
if (last_counter != counter){  
    lcd_print(0,1,"  ");  
    lcd_print(0,1,String(counter));  
    last_counter = counter;  
}
```

Then, we reset it back to -1.

```
last_counter = -1;
```

Reference Code:

ESR: By creating another variable.


```
int set_value(int min_val, int max_val) {
    int value = 0;
    int last_counter = -1;
    counter = 0;

    while (true) {
        button.loop();
        if (button.isPressed())
            break;

        counter = constrain(counter, min_val, max_val);
        if (last_counter != counter) {
            lcd_print(0, 1, " ");
            lcd_print(0, 1, String(counter));
            last_counter = counter;
        }
    }
    value = counter;
    counter = 0;
    last_counter = -1;
    return value;
}
```

What's the other issue?

ESR: Varied.

The encoder value can be set only between 0 and 3 even after the constrain is set between 0 and 23. Why?

ESR: Varied.

It is because of the **constrain()** command used in the **encoder()**.

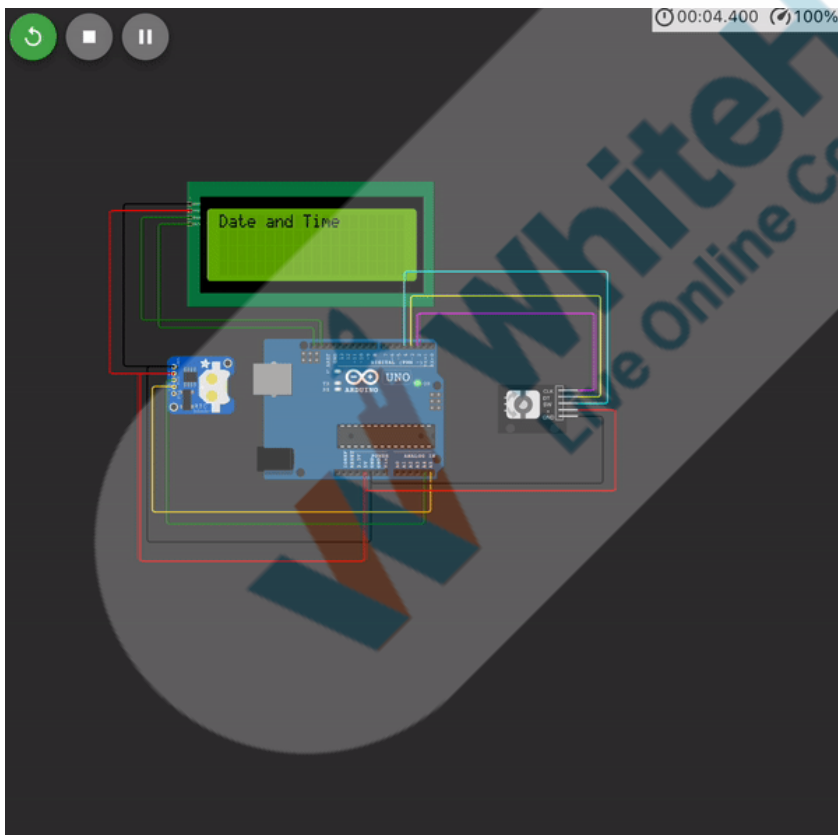
Let's update that.

Remove the **constrain** instruction from **encoder()** and put it in **mode_selector()** as the range 0 to 3 i.e. 4 options are for mode selection.

Reference Code:

```
void encoder() {  
  prev_counter = counter;  
  if (digitalRead(dt) == HIGH) counter++;  
  else counter--;  
  flag = 1;  
}  
  
void mode_selector() {  
  counter = constrain(counter, 0, 3);  
  if (prev_counter != counter && flag == 1) {  
    if (counter == 0) {  
      lcd.clear();  
    }  
  }  
}
```

Reference Output:



https://s3-whjr-curriculum-uploads.whjr.online/3b576755-e1d7-4294-8b9d-33141afd495e.gif	
Perfect! We are able to accept the input from users using an encoder. Let's continue. Are you excited to do this now?	ESR: Yes.
Teacher Stops Screen Share	
STUDENT-LED ACTIVITY- 15 mins	
<ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Full Screen. 	
Student Initiates Screen Share	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Learn to code features like an alarm clock and countdown timer. 	
Teacher Action	Student Action
<p>Let's get started.</p> <p>What should we do?</p> <p>In set_alarm(), we must accept the minutes input from the user.</p> <pre>lcd_print(0,0,"Enter Minutes : "); alarm_minutes = set_value(0,59);</pre> <p>After that, we just display a message to the user that such an alarm is set.</p> <pre>// printing data lcd_print(0,0,"Alarm is set for ");</pre>	<p>The student opens the wokwi simulator and replaces all the files downloaded from Student Activity 1.</p>

```
lcd_print(0,1,String(alarm_hours));  
lcd_print(3,1," Hours and ");  
lcd_print(0,2,String(alarm_minutes));  
lcd_print(3,2," Minutes");  
delay(5000);
```

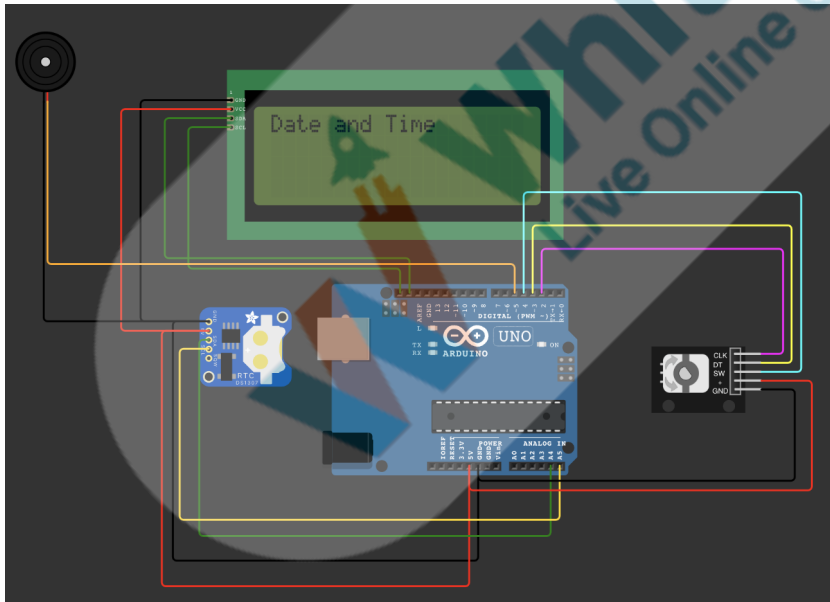
Next, what should be done?

We want to keep checking the alarm time in real-time and if it matches, play sound using a buzzer.

Perfect. Let's add the buzzer hardware and make its connections.

GND of **Buzzer** to **GND** of **RTC/Arduino** and **VCC** of **Buzzer** to pin 5 of Arduino.

Reference Image:



Next, we create a function to play the buzzer.

ESR: Varied.

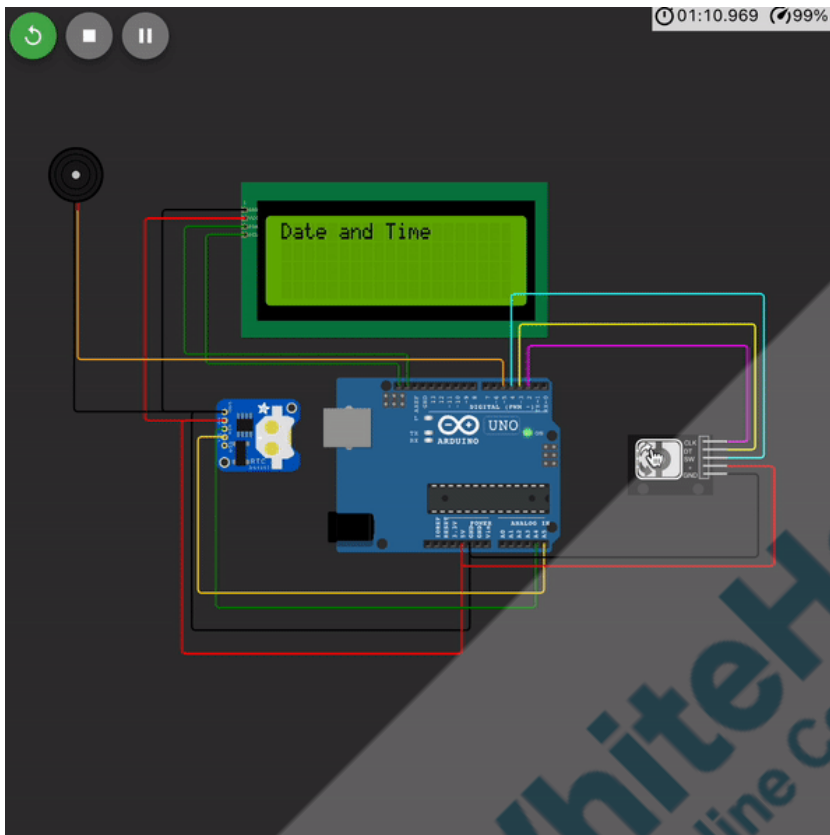
```
byte buzzer_pin = 5;
```

```
void play_buzzer(int frequency , int  
duration){  
    tone(buzzer_pin , frequency);  
    delay(duration);  
    noTone(buzzer_pin);  
}
```

Next, we check it with real-time in **loop()** and ring the buzzer at alarm time along with resetting the alarm variables back to 0.

```
if (alarm_minutes == minute &&  
alarm_hours == hour){  
    play_buzzer(1000 , 5000);  
    // resetting variables  
    alarm_hours = 0;  
    alarm_minutes = 0;  
}
```

Reference Output:



ESR: Varied.

<https://s3-whjr-curriculum-uploads.whjr.online/dbdea4df-d8ac-409b-88e5-95282a191674.gif>

Now, we will work on the timer.

How does the timer work?

Yes. It starts as per the user entered time value and keeps on decreasing until it is 00:00.

Again, we ask the user about the timer value.

```
int countdown_minutes = 0;
int countdown_seconds = 0;
// setting minutes
```

```
lcd_print(0,0,"Enter Minutes : ");
countdown_minutes = set_value(0 , 59);
// setting seconds
lcd_print(0,0,"Enter Seconds : ");
countdown_seconds = set_value(0 , 59);
```

Until either the minutes or seconds decrease to 0, we will continue to display the timer on the LCD. Even if seconds reach 0, we need to check the minutes and appropriately decrease it by a unit and update the seconds to 59.

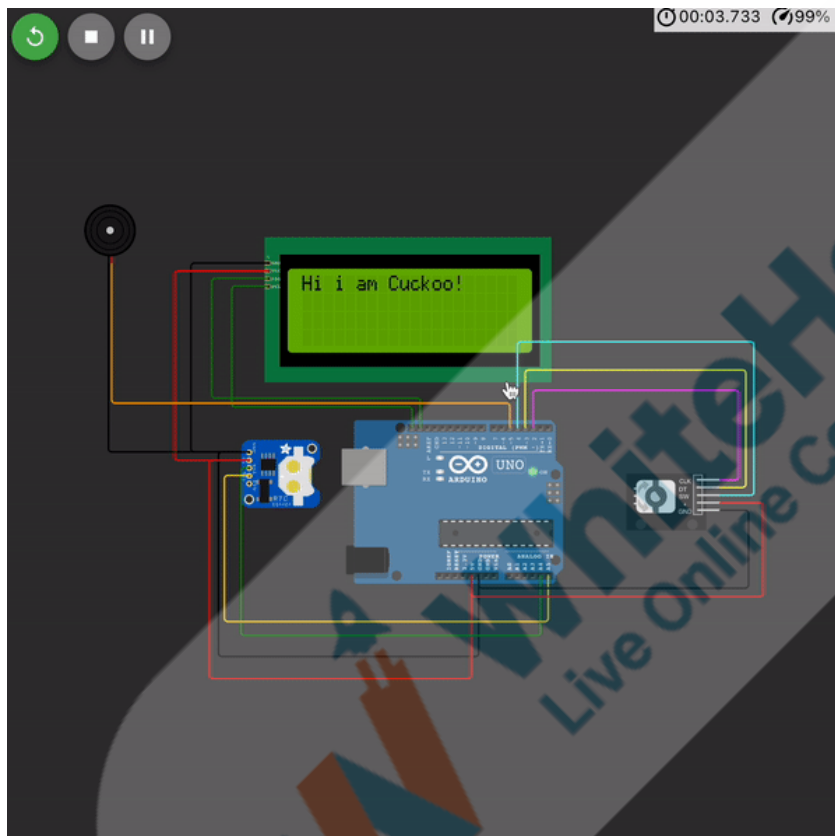
Also, once it reaches 00:00, we will display **Countdown Over** on the display and play the buzzer to alert.

```
// start the stopwatch
while (countdown_minutes != 0 ||
countdown_seconds != 0){

    String t = String(countdown_minutes) + ":"
               + String(countdown_seconds);
    lcd.clear();
    lcd_print(0,0,"Countdown Timer");
    lcd_print(0,1,t);
    if (countdown_seconds == 0){
        if (countdown_minutes != 0){
            countdown_seconds = 59;
            countdown_minutes--;
        }
    }
    countdown_seconds--;
    delay(1000);
```

```
}  
  
lcd.clear();  
lcd_print(0,0,"Countdown over");  
play_buzzer(200 , 3000);
```

Reference Output:



<https://s3-whjr-curriculum-uploads.whjr.online/a73dbb57-a888-4a15-a9ad-d5dcd5ab0db9.gif>

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 10 mins

Activity details

© 2022 - WhiteHat Education Technology Private Limited.

Note: This document is the original copyright of WhiteHat Education Technology Private Limited.
Please don't share, download or copy this file without permission.

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ

Click on In-Class Quiz




Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
<p>You get “hats-off” for your excellent work!</p> <p>In the next class, we will learn to use another display hardware LED Dot Matrix.</p>	<p><i>Make sure you have given at least 2 hats-off during the class for:</i></p> <div> <div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div> </div>

PROJECT OVERVIEW DISCUSSION

Refer to the document below in Activity Links Sections

Teacher Clicks

✕ End Class

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Previous class code	https://github.com/procodingclass/PRO-C267-REFERENCE-CODE
Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/PRO-C268-STUDENT-TEMPLATE
Teacher Reference 2	Reference Code	https://github.com/procodingclass/PRO-C268-REFERENCE-CODE
Teacher Reference 3	Project	https://docs.google.com/document/d/1E-13xxDf40FogTDZ2jSUI9YQ8lD974u1dfs3JFgoVDI/edit
Teacher Reference 4	Project Solution	https://github.com/procodingclass/PRO-C268-PROJECT
Teacher Reference 5	In-Class Quiz	https://docs.google.com/document/d/1kM6sQm1HsdcZl6d5_wEKGerwPX4l4gS7pp6g4CO1a6M/edit
Student Activity 1	C-268 Student Template	https://github.com/procodingclass/PRO-C268-STUDENT-TEMPLATE