Topic	**Smart Clock III**	
Class Description	**Students will learn how to add an LCD device and show the time as well as add a stopwatch feature to the smart clock.**	
Class	**PRO C267**	
Class time	**50 mins**	
Goal	● Understand the **LCD** device ● Learn to use LCD Display with Arduino	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ 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**

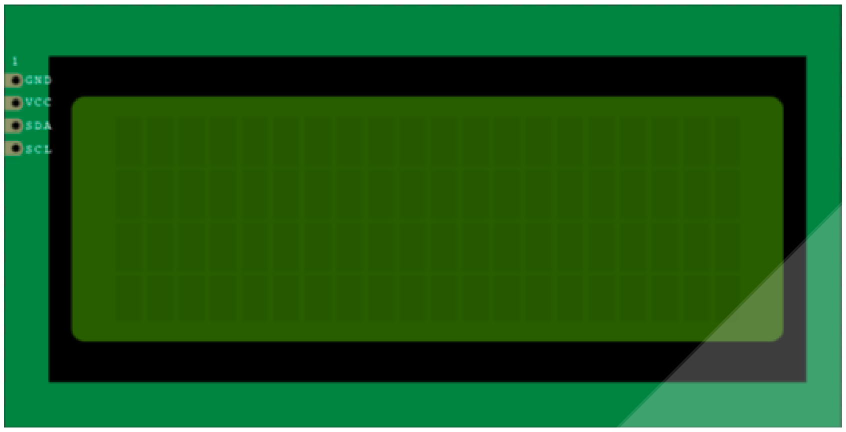
WARM-UP SESSION - 10 mins

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:** ● Greet the student.	**ESR**: Hi, thanks! Yes, I am excited about it! Click on the slide show tab and present the slides

● Revision of previous class activities. ● Quizzes.	
WARM-UP QUIZ Click on In-Class Quiz	
Activity Details **Following are the session deliverables:** ● Appreciate the student. ● Narrate the story by using hand gestures and voice modulation methods to bring more interest in students.	
TEACHER-LED ACTIVITY 15 mins	
Teacher Initiates Screen Share	
● **Understand the LCD Display device** ● **Learn to use LCD with Arduino**	
Teacher Action	**Student Action**
Do you remember what we learned in the previous class? Can you tell me how we achieved it? Great. You are revising very well. Do you have any questions from the previous class? *Note: If the student has any doubts, clarify the doubts.*	**ESR**: Yes. **ESR**: Varied. **ESR**: Varied
How are we displaying the time on our smart clock? Can we use something better?	**ESR:** On the console/serial. **ESR:** Yes. We can add a hardware device for display.

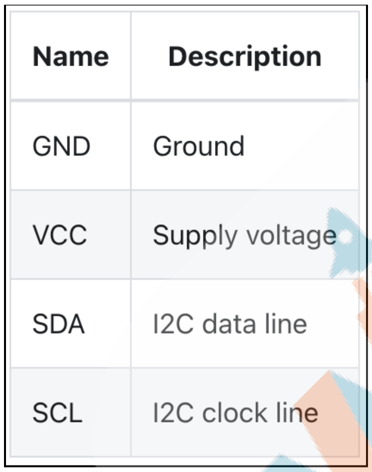
Great. Do you know about any display hardware? Superb! LCD, and LED displays are commonly used by TVs, smartphones, etc. These devices help you represent information in visual formats. Today, we will learn about **LCD**. LCD stands for **Liquid Crystal Display**. It is a type of flat panel display which uses liquid crystals. Every picture is made of millions of pixels. In LCDs, these pixels are lit by a backlight, which is switched on and off electronically. We have two types of LCD hardware: 1. **LCD2004** This is a display with 4 lines and 20 characters on each line. 2. **LCD1602** This is a display device with 2 lines and 16 characters per line.	**ESR:** Varied.
Which one would be better for our smart clock? **2004** as it provides more number of lines. The two-line display is not sufficient to display the mode of the smart clock and ask for input as well.	**ESR**: Varied.
Great. We will use LCD2004 with Arduino. Are you excited?	**ESR:** Yes.

Let’s see what it looks like.



Click here to view this image.

It has 4 pins.



Name	Description
GND	Ground
VCC	Supply voltage
SDA	I2C data line
SCL	I2C clock line

The **default I2C address** of the LCD2004 module is **0x27**.

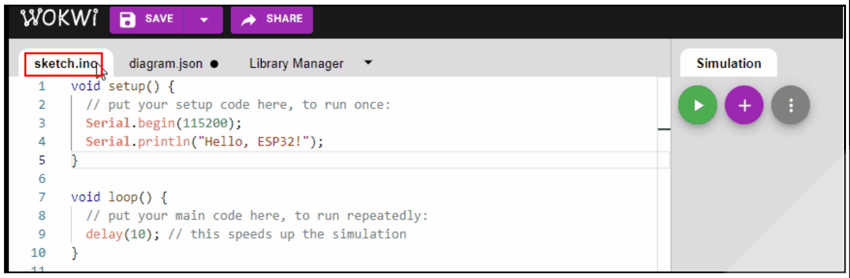
Next, how do we get started with it?

ESR: Add the LCD2004 hardware and make its connections.

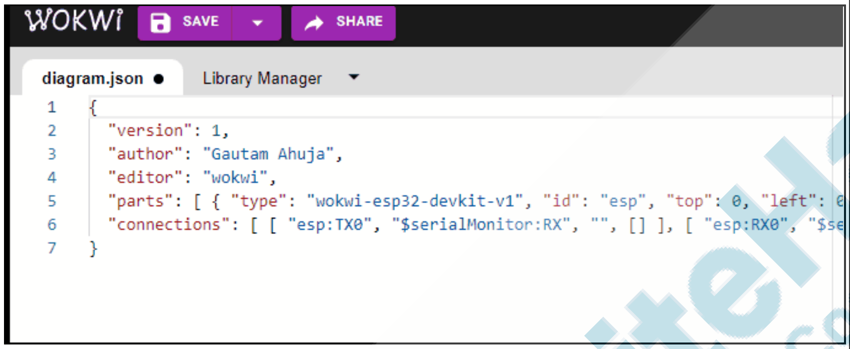
Great. Let’s start.

Open the wokwi simulator and replace all the files downloaded from Teacher Activity 1.

Note: Refer to Lesson C260.



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("Hello, ESP32!");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(10); // this speeds up the simulation
}
```



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

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

First, we create the circuit diagram.
Step 1: Select the components:

- 1 x **LCD 20x4 (I2C)**

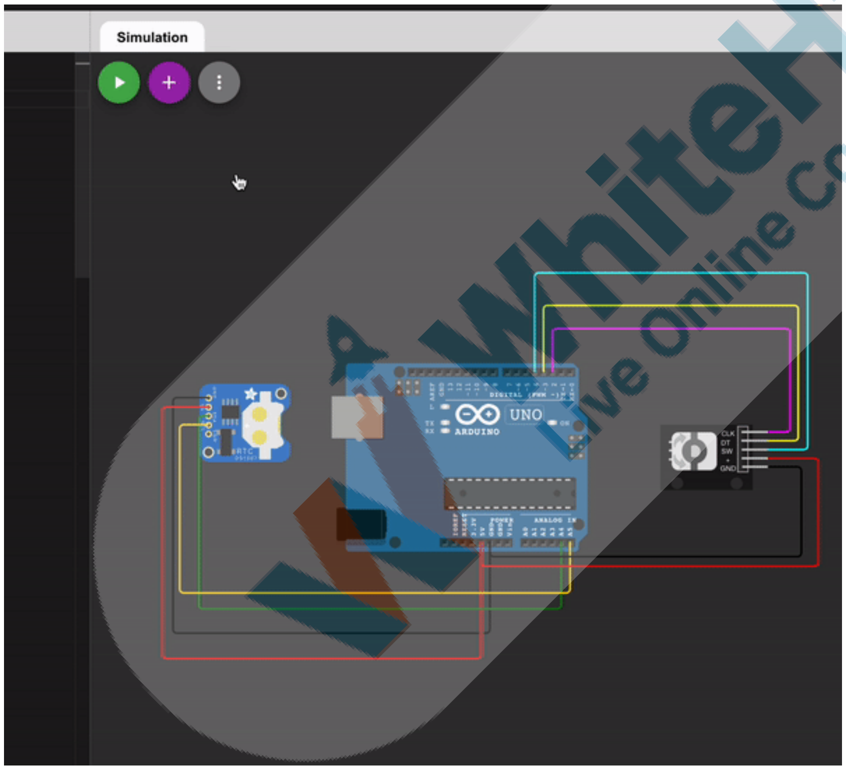
Step 2: Make the connections:

The circuit of this project consists of an **Arduino** Controller, and an **LCD 20x4 (I2C)**.

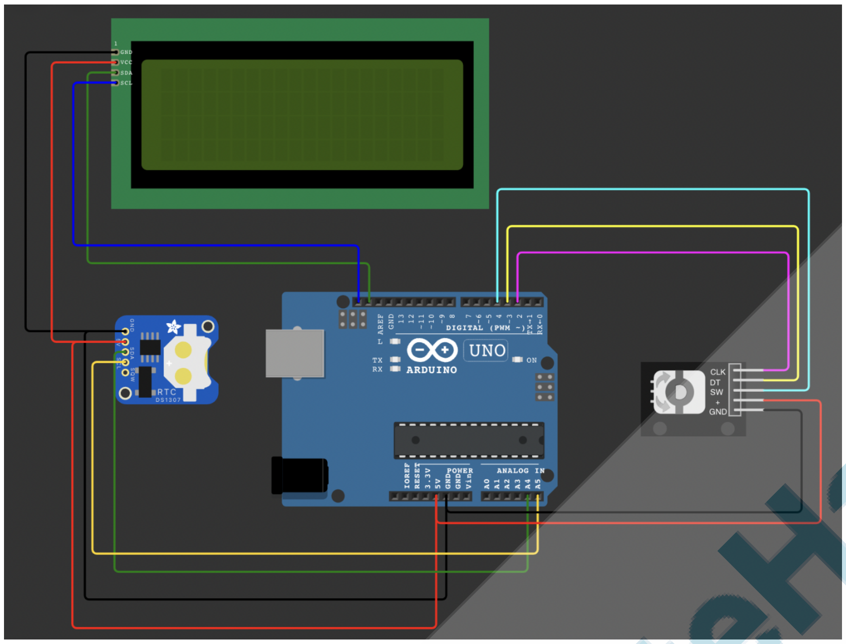
LCD 20x4 (I2C)	**Arduino PIN**
VCC	VCC
GND	GND

SDA	A4
SCL	A5

*Note: Wire color can be changed by **clicking over it** and selecting the color, or via the **diagram.json** file. Go to the **diagram.json** wire and change the color of the wire. Any design changes or color changes can be done via the diagram.json file. Keep the track of the component and change the design settings.*
Also, you can connect the VCC and GND of LCD2004 to RTC's VCC and GND instead of Arduino to avoid jumbling of wires.

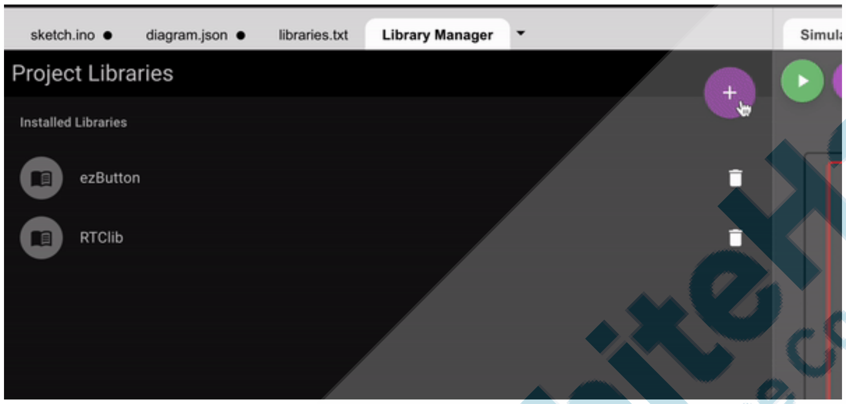


https://s3-whjr-curriculum-uploads.whjr.online/e7613499-c5e1-4541-9ef9-5e299bf6adb.gif

Reference image:

Next, what do we do?
 	
Correct. Let's code.
1. We added the library file to work with **LCD2004**.
`#include <LiquidCrystal_I2C.h>`
 	
2. Next, we create an object of the LCD by setting its address and the number of characters and lines.
`// lcd object : setting register address 0x27`
`LiquidCrystal_I2C lcd(0x27 , 20 , 4);`
 	
3. Then, in **setup(),** we initialize the LCD device and also turn on its backlight.
`// initializing LCD`
` lcd.init(); // initialize the lcd.`
	ESR: Code to work with LCD.

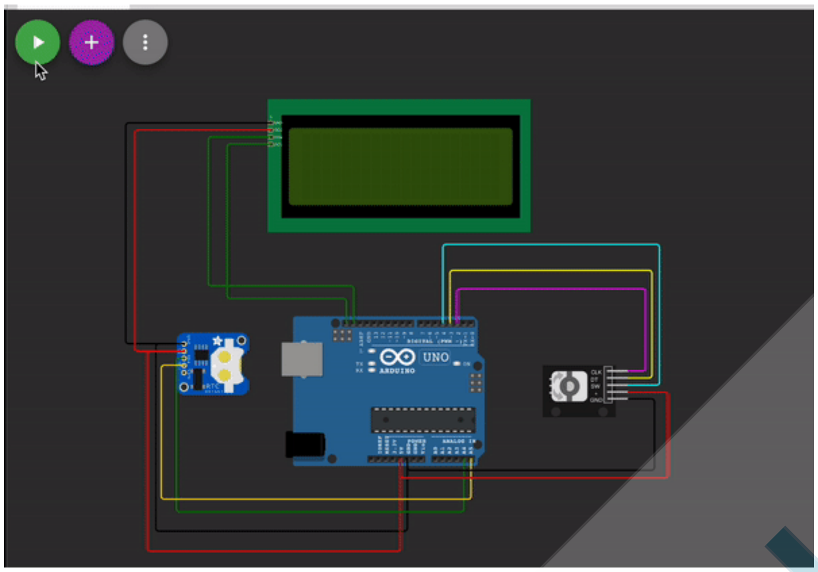
```
 lcd.clear();  // clears the screen and
moves the cursor to the top left corner.
 lcd.backlight();  // turn on the
backlight.
```

4. Also, add reference **LiquidCrystal I2C** in the Library manager file to avoid errors while working with the **LiquidCrystal_I2C.h** header file.



https://s3-whjr-curriculum-uploads.whjr.online/df438a24-c012-4be8-a96a-1998aaba29ff.gif

Reference output:



https://s3-whjr-curriculum-uploads.whjr.online/e7a90988-8fe3-40e1-aaf3-6bb1c9a31681.gif

5. Next, let's see how we display messages on the LCD. We reposition the cursor to the desired location x and y for displaying our message using **setCursor()**.
   To print the message, **print()** instruction is used.
   Also, for convenience and efficient code writing, we create our method to print messages on the LCD device.

```
void lcd_print(int x , int y , String
message){
 lcd.setCursor(x,y);
 lcd.print(message);
}
```

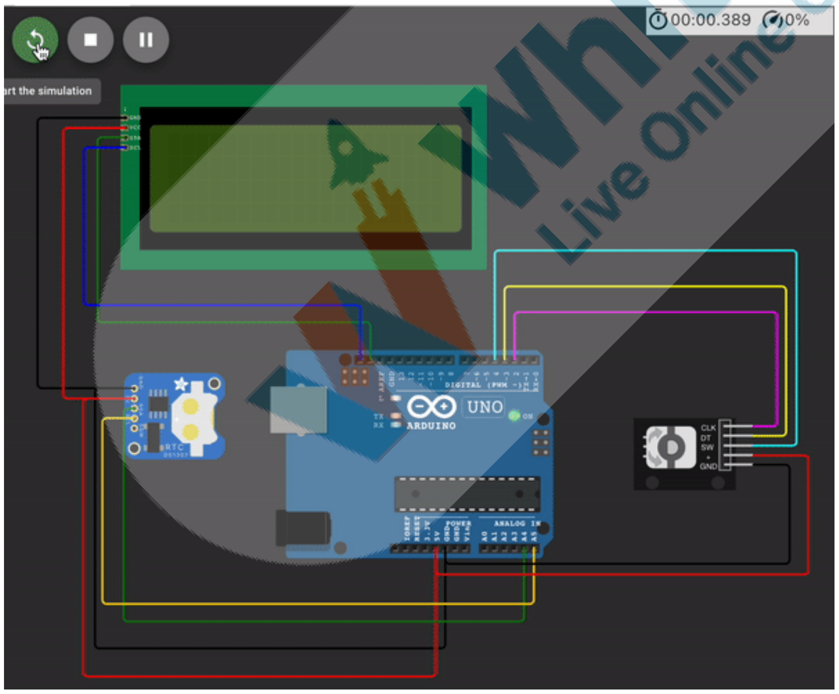
6. Next in **setup()**, we display a welcome message on our display device by clearing the previous message using **clear()** instructions.
Can you tell me why we add the **delay()**?

Great.

```
//  initial messages
 lcd.clear();
 lcd_print(0,0,"RTC found");
 delay(1000);
 lcd_print(0,0,"Hi i am Cuckoo!");
 delay(2000);
```

ESR: We want the message to be seen by the user and hence we add a pause using **delay()**.

Reference Output:



https://s3-whjr-curriculum-uploads.whjr.online/a42fb04b-b868-4585-b90c-11310b8e7e5a.gif
Next, what should be displayed on the device?

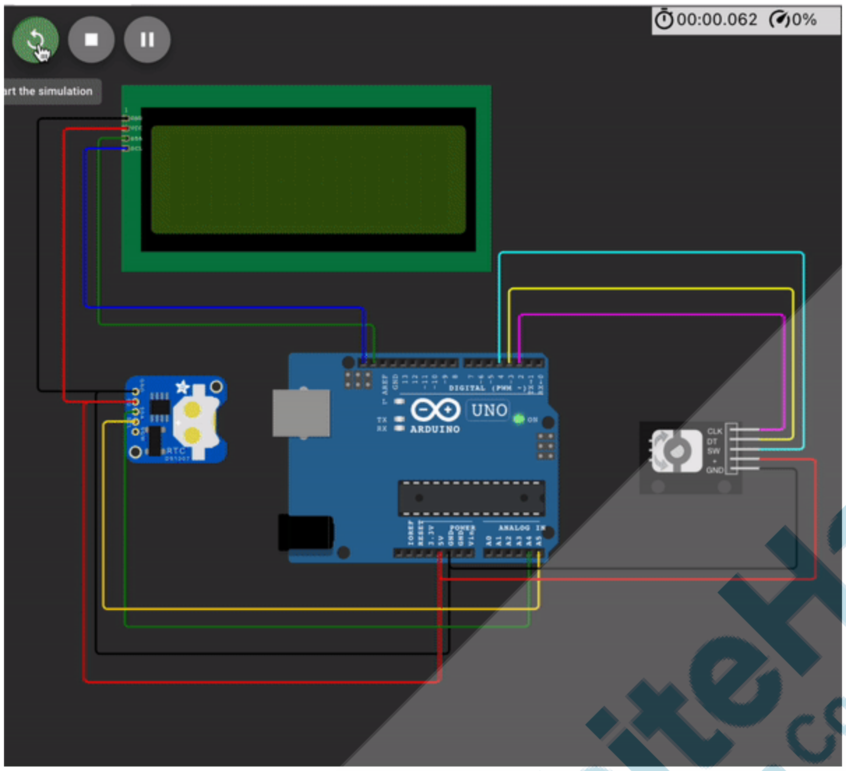
Correct. So we replace the serial print instructions to LCD print instructions in **mode_selector()**.

Reference Code:

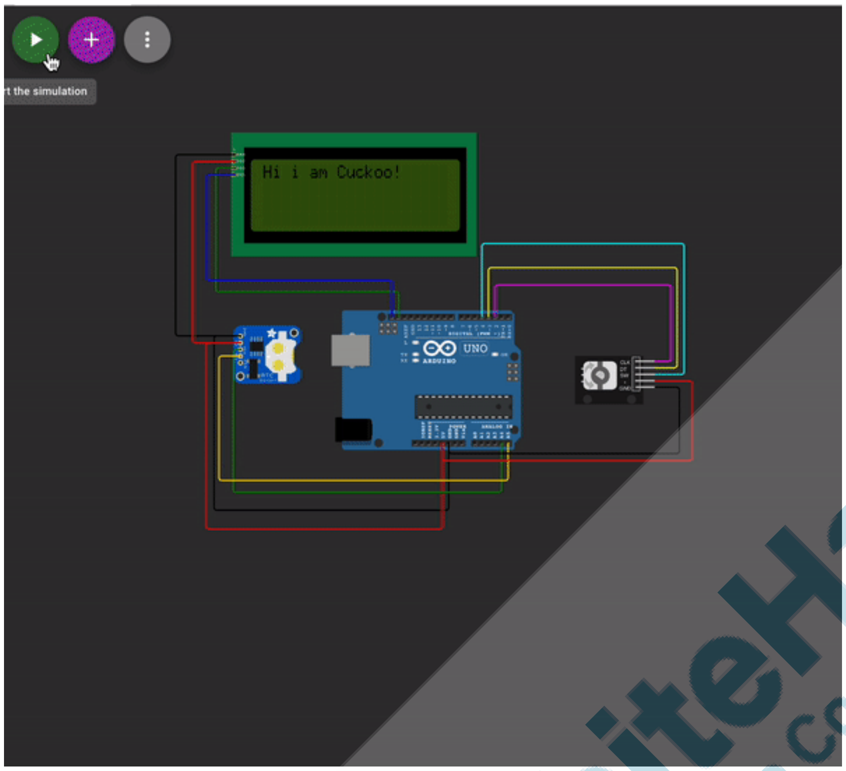
```
lcd.clear();
lcd_print(0,0,"Date and Time");

lcd.clear();
lcd_print(0,0,"Set Alarm");

lcd.clear();
lcd_print(0,0,"Stopwatch");

lcd.clear();
lcd_print(0,0,"Countdown Timer");
```

Reference Output:

ESR: The different modes of our smart clock.

 00:00.062
0%
art the simulation
UNO
ARDUINO
DIGITAL (PWM ~)
ANALOG IN
POWER
RTC
CLK
DT
SW
GND https://s3-whjr-curriculum-uploads.whjr.online/2076e3d3-500b-4675-a52f-a5c76c238d0e.gif
 	
 Then, what should happen when you select the mode?
 	
 Bingo! Let's replace the serial print instructions to function calls.
 	
 Reference Code:
 `void select_mode(){`
 `  if (counter == 0)get_time();`
 `  else if (counter == 1)set_alarm();`
 `  else if (counter == 2)stopwatch();`
 `  else countdown();`
 `}`
	ESR: It should perform the task.

```
void get_time(){

}
```
```
void set_alarm(){

}
```
```
void stopwatch(){

}
```
```
void countdown(){

}
```
 Reference Output:	

 https://s3-whjr-curriculum-uploads.whjr.online/dd5a5717-bf40-42cd-9b35-3df048683384.gif	
Perfect. Let's add the functionalities to our smart clock. Are you excited to do this now?	**ESR**: Yes.

Teacher Stops Screen Share

STUDENT-LED ACTIVITY- 15 mins

- **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.**

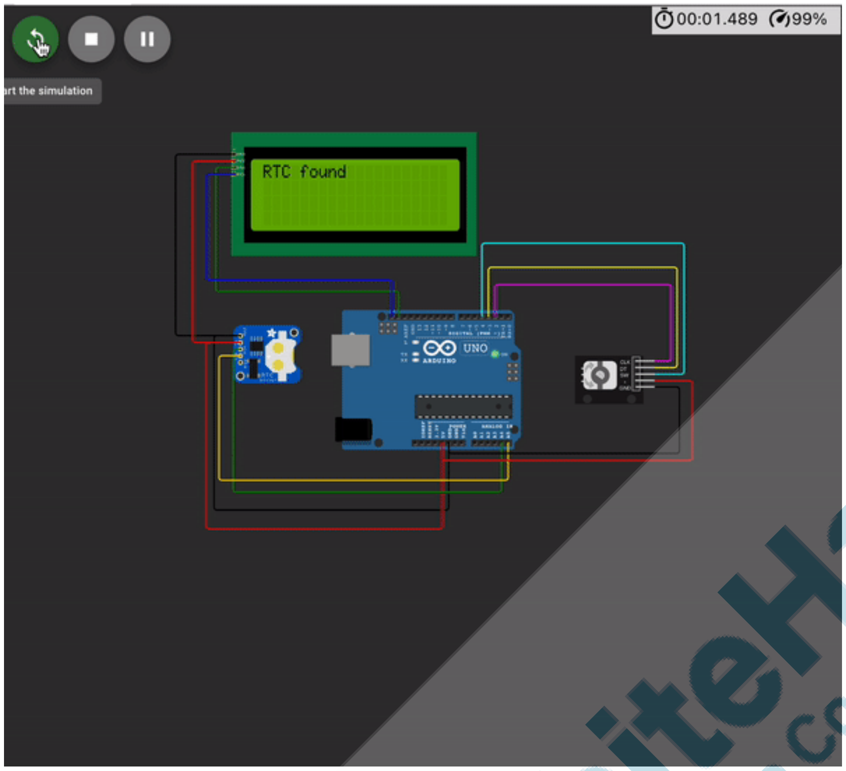
Student Initiates Screen Share

<u>ACTIVITY</u>

- **Learn to code features like display time and stopwatch on the LCD.**

Teacher Action	Student Action

Let's get started.
 	
Do you think the modes are working fine?
 	
Note: The student observes that the modes are not displayed until the encoder is rotated.
 	
So how do we fix that?
The mode selection is based on variables, **flag** and **prev_counter** (added in C266).
 	
Hence, we update them.
 	
`int prev_counter = -1;`
`int flag = 1;`
 	
Reference Output:
	Student opens the wokwi simulator and replaces all the files downloaded from Student Activity 1.
	
	ESR: No.
	
	ESR: Varied.



https://s3-whjr-curriculum-uploads.whjr.online/aead8670-719e-4650-b258-504d05608f9b.gif

Next, we want to show the time on our display. Let’s do so.

Note: Students have already learned to work with RTC to get the current date and time.

We create a function **current_time()** to get the current date and time as shown below:

```
//  clock variables
int year = 0;
int month = 0;
int day = 0;
int hour = 0;
int minute = 0;
int second = 0;
```

```
void current_time(){

 //  getting current date and time
 DateTime current = rtc.now();
 year = current.year();
 month = current.month();
 day = current.day();
 hour = current.hour();
 minute = current.minute();
 second = current.second();
}
```

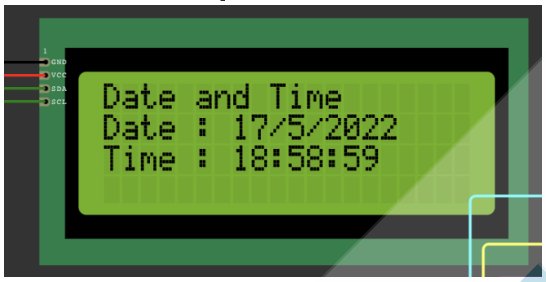
Next, let's code the functionality to display time on LCD.

What we want in the program is to keep updating and display the time. Hence, we use an infinite loop to get the current time, format it using string instructions and call the print method of the LCD to display it on the device.

Reference Code:

```
void get_time(){
while(true){
   //  getting current time
   current_time();
     String current_date = "Date : " +
String(day) + "/" + String(month) +
                 "/" + String(year);
     String current_time = "Time : " +
String(hour) + ":" + String(minute) +
                 ":" + String(second);
     lcd_print(0,0,"Date and Time");
     lcd_print(0,1,current_date);
     lcd_print(0,2,current_time);
}
}
```

Reference Output:



```
Date and Time
Date : 17/5/2022
Time : 18:58:59
```

But, there is a problem. We can't change the mode and try different features.

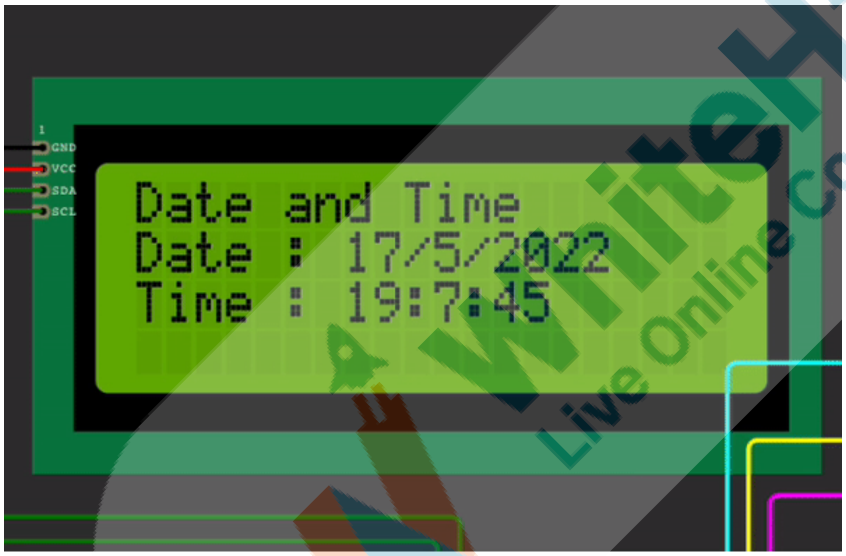
To solve this, we need to break the loop again when the push button on the encoder is clicked.

Reference Code:

```
   //  looping button
   button.loop();
   if (button.isPressed()){
     lcd.clear();

     //  let's run the mode_selector
     prev_counter = -1;
     flag = 1;

     //  break the loop
     break;
}
```

get_time() Reference Code:

```
void get_time(){
 while(true){
   //  looping button
   button.loop();
   if (button.isPressed()){
     lcd.clear();

     //  let's run the mode_selector
     prev_counter = -1;
     flag = 1;

     //  break the loop
     break;
   }

   //  getting current time
   current_time();
     String current_date = "Date : " +
String(day) + "/" + String(month) +
                 "/" + String(year);
```

```
    String current_time = "Time : " +
String(hour) + ":" + String(minute) +
                ":" + String(second);
    lcd_print(0,0,"Date and Time");
    lcd_print(0,1,current_date);
    lcd_print(0,2,current_time);
  }
}
```
 	
Reference Output:
 	

Date and Time
Date : 17/5/2022
Time : 19:7:45
 	
https://s3-whjr-curriculum-uploads.whjr.online/75ea8c85-56cf-4c45-bf67-2311305ed838.gif
 	
Next, we work on the stopwatch feature.
 	
How does a stopwatch work?
 	
Perfect. So we initialize our hour, minute and second
	ESR: It starts from 00:00:00 And keeps counting until we stop it.

variables to 0 for each.

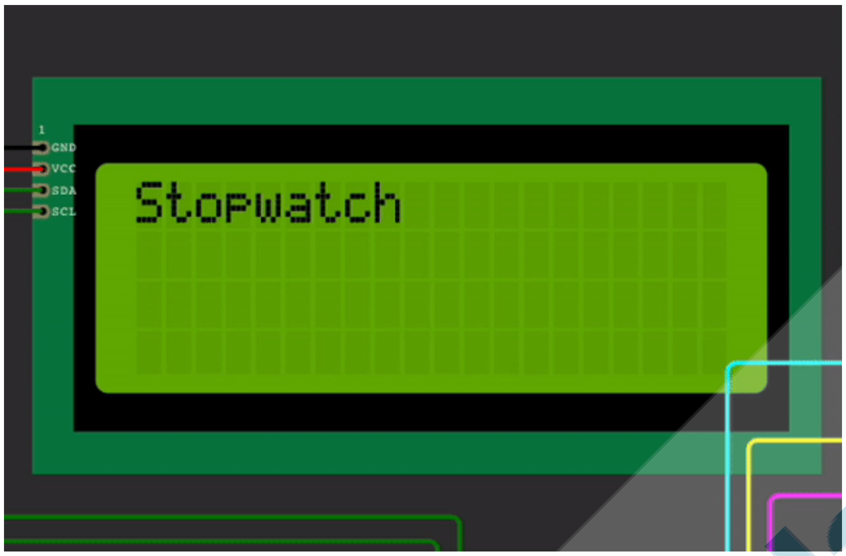
```
int stopwatch_hour=0, stopwatch_minute=0,
stopwatch_second = 0;
```

After every second of real time, the second will increase by one unit of the stopwatch.

Next, in the **while loop** again, we get the current time and we create a string of hour, minute and second using string instructions and update the display on the LCD along with increasing one second on the stopwatch.

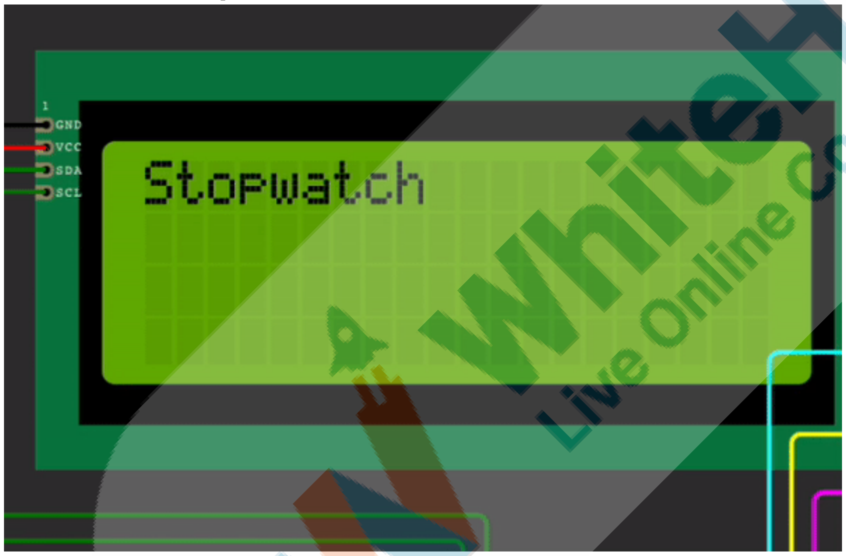
```
while (true){
 current_time();
     String stopwatch_time =
String(stopwatch_hour) +
                             " : " +
String(stopwatch_minute) +
                             " : " +
String(stopwatch_second);
     lcd.clear();
     lcd_print(0,0,"Stopwatch");
     lcd_print(0,1,String(stopwatch_time));
     stopwatch_second++;
}
```

Reference Output:

 https://s3-whjr-curriculum-uploads.whjr.online/194772b7-b532-4f17-9878-ad69a9a53541.gif
 	
But, what is wrong here?
 	
Yes. The display is flickering a lot and the timer is very fast.
 	
How do we solve this?
 	
We use the variable **last_second** to check with the real time if a second has been completed and if it has, then only increment the seconds.
 	
Reference Code:
`int last_second = 0;`
 	
`  if (abs(second - last_second) >= 1){`
`    last_second = second;`
`    String stopwatch_time =`
`String(stopwatch_hour) +`
	ESR: Varied.
	
	ESR: Varied.

```
                                                     " : " +
String(stopwatch_minute) +
                                                     " : " +
String(stopwatch_second);
         lcd.clear();
         lcd_print(0,0,"Stopwatch");
         lcd_print(0,1,String(stopwatch_time));
         stopwatch_second++;
      }
```

Reference Output:



https://s3-whjr-curriculum-uploads.whjr.online/04cf9c46-efc0-4960-89b8-a1145df133de.gif

Does it work fine?

No, seconds go beyond 60. Similarly, hours and minutes too can go beyond 24 and 60.

So we need to check for the calculation of time. After every 60 seconds, one minute is passed and reset the seconds back to 0. Similarly, the hour and the minute too.

```
//  condition check
   if (stopwatch_second > 59){
     stopwatch_second = 0;
     stopwatch_minute++;
   }
   else if (stopwatch_minute > 59){
     stopwatch_minute = 0;
     stopwatch_hour++;
   }
   else if (stopwatch_hour > 24){
     stopwatch_second, stopwatch_minute,
stopwatch_hour = 0;
}
```

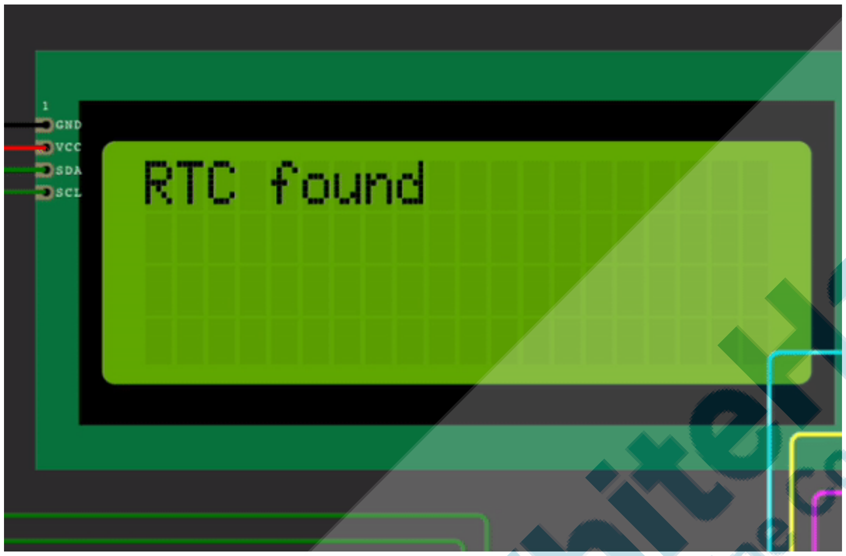
Also, we want to break the **loop** and allow change of the mode.

```
 //  breaking loop if button pressed
   button.loop();
   if (button.isPressed()){
     prev_counter = -1;
     flag = 1;

     delay(5000);
     lcd.clear();
     break;
}
```

stopwatch() Reference Code:

```
void stopwatch(){

 int stopwatch_hour, stopwatch_minute,
stopwatch_second = 0;
 int last_second = 0;

 while(true){
   //  breaking loop if button pressed
   button.loop();
   if (button.isPressed()){
     prev_counter = -1;
     flag = 1;

     delay(5000);
     lcd.clear();
     break;
   }

   //  tracking current time to get the
'seconds' variable
   current_time();
```

```
   //  stopwatch algo
   if (abs(second - last_second) >= 1){
     last_second = second;
     String stopwatch_time =
String(stopwatch_hour) +
                              " : " +
String(stopwatch_minute) +
                              " : " +
String(stopwatch_second);
     lcd.clear();
     lcd_print(0,0,"Stopwatch");
     lcd_print(0,1,String(stopwatch_time));
     stopwatch_second++;
   }

   //  condition check
   if (stopwatch_second > 59){
     stopwatch_second = 0;
     stopwatch_minute++;
   }
   else if (stopwatch_minute > 59){
     stopwatch_minute = 0;
     stopwatch_hour++;
   }
   else if (stopwatch_hour > 24){
     stopwatch_second, stopwatch_minute,
stopwatch_hour = 0;
```

```
    }
  }
}
```
 Reference Output:  RTC found
https://s3-whjr-curriculum-uploads.whjr.online/2500955a-e894-4494-a38d-1a7613686e96.gif
Great. Our smart clock shows time as well as stopwatch works fine. Good job today!	

Teacher Guides Student to Stop Screen Share	
WRAP-UP SESSION - 10 mins	
Activity details **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**
You get "hats-off" for your excellent work! In the next class, we will add the functionalities of the timer and alarm clock.	*Make sure you have given at least 2 hats-off during the class for:* Creatively Solved Activities +10 Great Question +10 Strong Concentration +10

PROJECT OVERVIEW DISCUSSION
Refer to the document below in Activity Links Sections

Teacher Clicks ✖ End Class

WhiteHat Jr
Live online Coding for Kids

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Previous class code	https://github.com/procodingclass/PRO-C266-TEACHER-REFERENCE-CODE
Teacher Activity 2	LCD2004	https://docs.wokwi.com/parts/wokwi-lcd2004
Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/PRO-C267-STUDENT-TEMPLATE
Teacher Reference 2	Reference Code	https://github.com/procodingclass/PRO-C267-REFERENCE-CODE
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/a7e521b3-0be4-457b-a7de-89115420f8cb.pdf
Teacher Reference 4	Project Solution	https://github.com/procodingclass/PRO-C267-PROJECT-SOLUTION
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/77e8969b-6693-4b12-a1d8-fccf883e5d44.pdf
Student Activity 1	C-267 Student Template	https://github.com/procodingclass/PRO-C267-STUDENT-TEMPLATE
Student Activity 2	LCD2004	https://docs.wokwi.com/parts/wokwi-lcd2004