Smart Clock-III

What is our GOAL for this CLASS?

In this class, we learned how to add an LCD device and show the time as well as add a stopwatch feature to the smart clock.

What did we ACHIEVE in the class TODAY?

- Understood LCD device
- Learned to use LCD Display with Arduino

Which CONCEPTS/ CODING BLOCKS did we cover today?

- **Concepts**: Transmitting data, receiving data, infinite loops, sequencing of code, and displaying messages on LCD devices.
- **Coding blocks:** Serial.begin(), delay(), lcd.init(), lcd.clear(), lcd.backlight(), lcd.setCursor(), lcd.print(), year(), month(), day(), hour(), minute(), second()

How did we DO the activities?

1. **Learned 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:
   LCD2004- This is a display with 4 lines and 20 characters on each line.
   LCD1602- This is a display device with 2 lines and 16 characters per line.

   What it looks like.

2. **Learned how to use LCD2004 with Arduino:**
   a. Open the wokwi simulator and create a new Arduino Uno project.
   b. Download the code from the template and replace all the files.
   c. Then, create the circuit diagram.
   Step 1: Select the components:

   - 1 x **LCD 20x4 I2C**

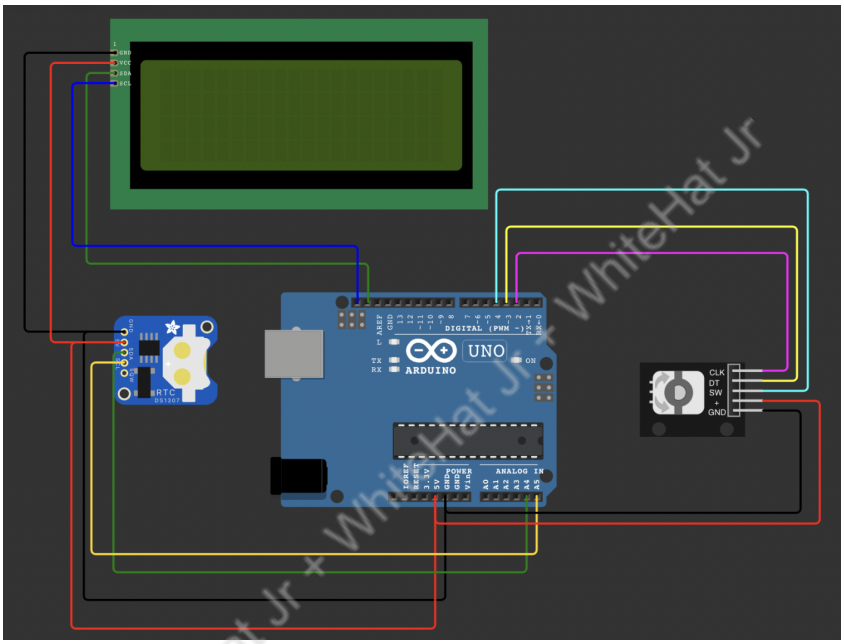
   Step 2: Make the connections:

   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

Reference image:



d. Add the library file to work with LCD2004.

```
#include <LiquidCrystal_I2C.h>
```

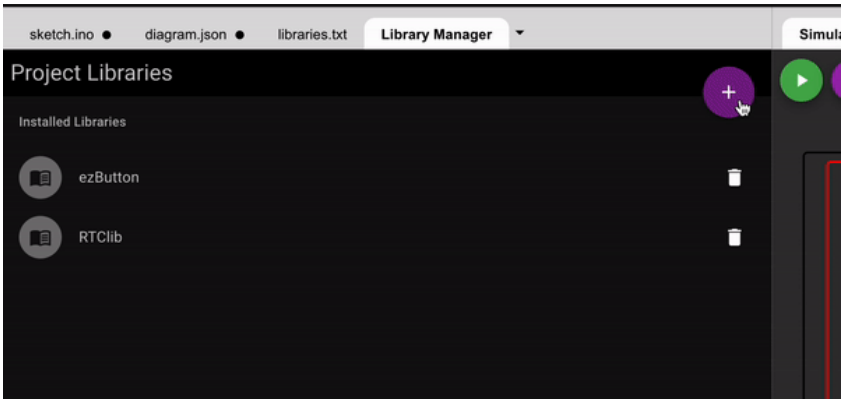
e. Next, create an object of the LCD by setting its address and the number of characters and lines.

```
// lcd object : setting register address 0x27
LiquidCrystal_I2C lcd(0x27 , 20 , 4);
```

f. In **setup()**, initialize the LCD device and also turn on its backlight.

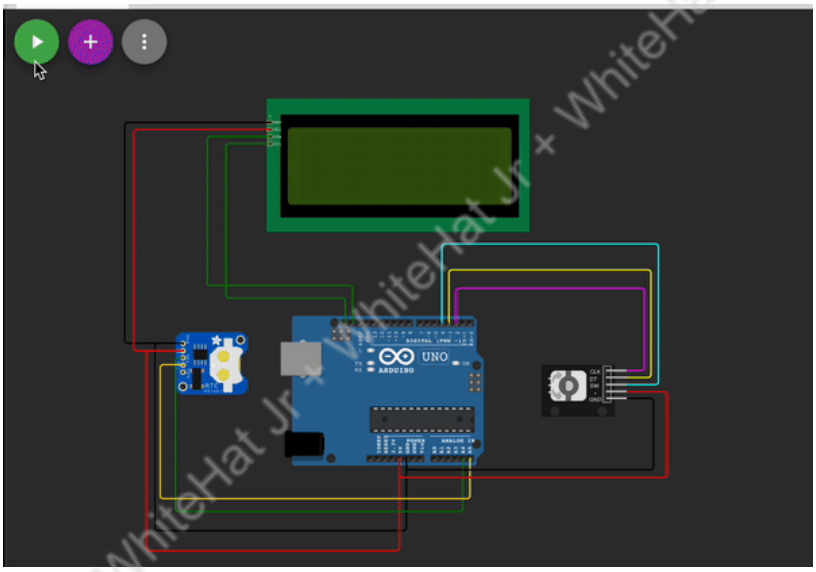
```
// initializing LCD
lcd.init();  // initialize the lcd.
lcd.clear();  // clears the screen and moves the cursor to the top left corner.
lcd.backlight();  // turn on the backlight.
```

g. 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

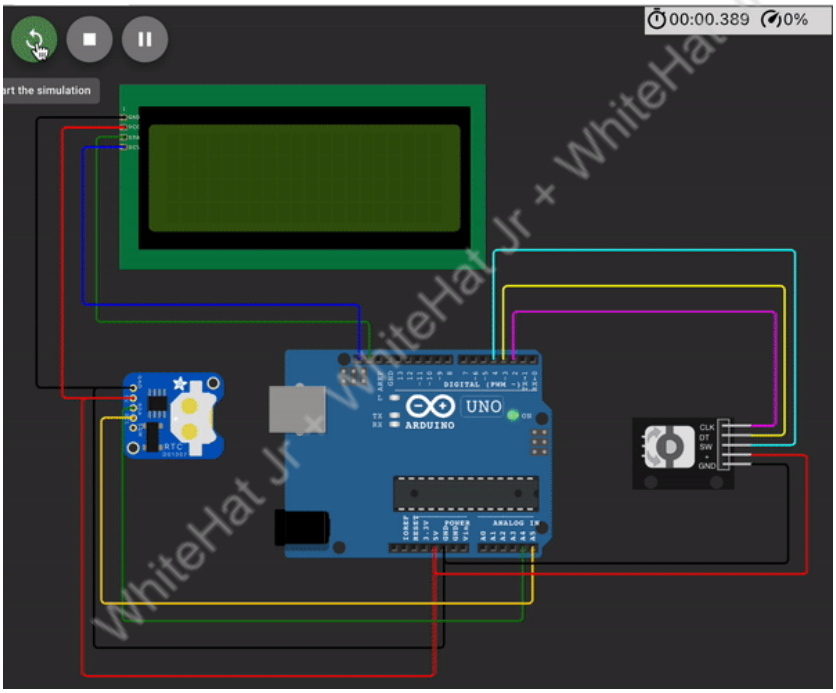
h. Next, display messages on the LCD. To do so, reposition the cursor to the desired location x and y for displaying our message using **setCursor()**.
To print the message, **print()** instruction is used.
Also, for convenience and efficient code writing, create our method to print messages on the LCD device.

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

i. Next in **setup**(), display a welcome message on our display device by clearing the previous message using **clear**() instructions.

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

Reference Output:



https://s3-whjr-curriculum-uploads.whjr.online/a42fb04b-b868-4585-b90c-11310b8e7e5a.gif

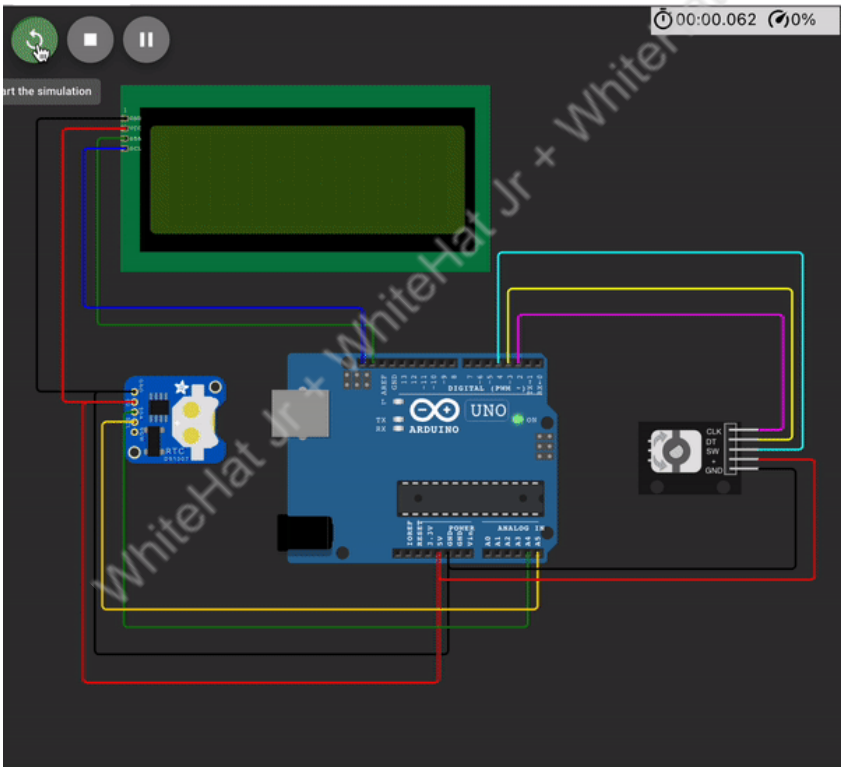
j. Replace the serial print instructions to LCD print instructions in **mode_selector()**.

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:



https://s3-whjr-curriculum-uploads.whjr.online/2076e3d3-500b-4675-a52f-a5c76c238d0e.gif

k. Now, 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();
}
```

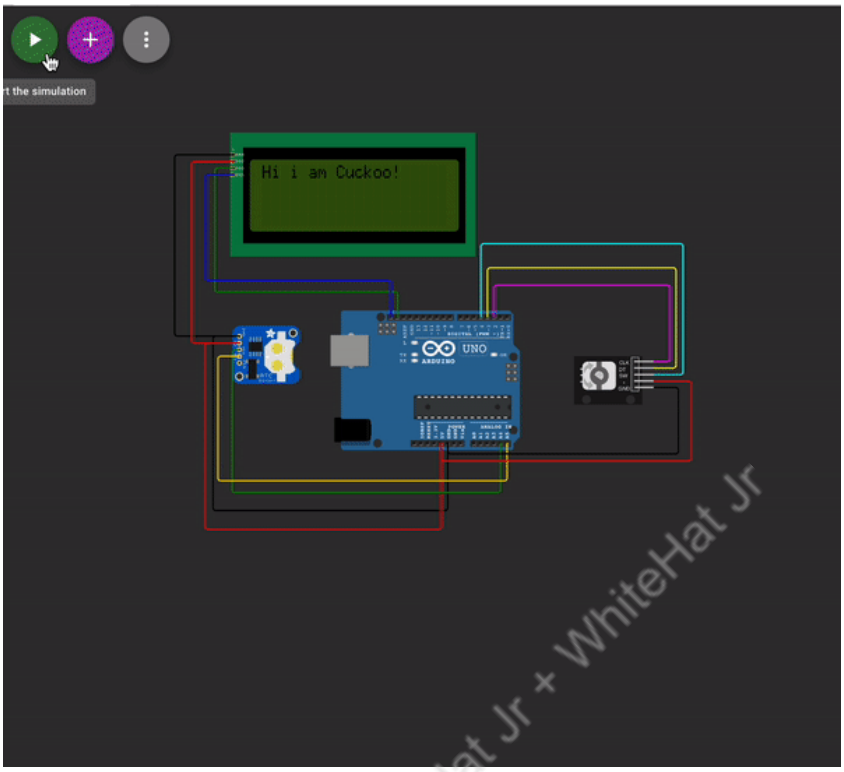
```
void get_time(){


}
```

```
void set_alarm(){


}
```

```
void stopwatch(){


}
```

```
void countdown(){


}
```

Reference Output:

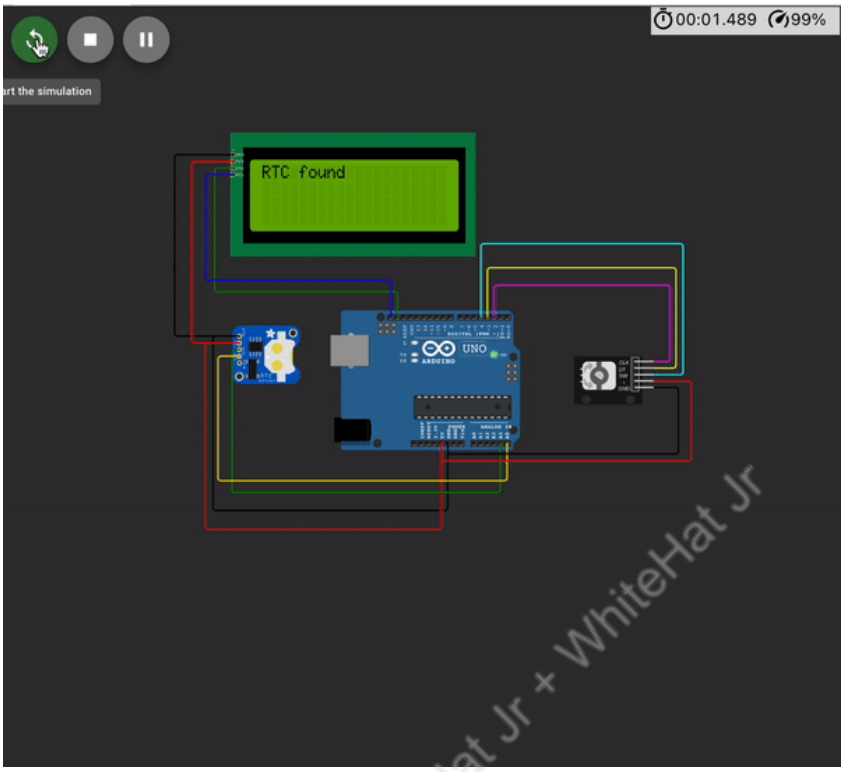


https://s3-whjr-curriculum-uploads.whjr.online/dd5a5717-bf40-42cd-9b35-3df048683384.gif

l. Observed that the modes are not displayed until the encoder is rotated. The mode selection is based on variables **flag** and **prev_counter** (added in C266). Hence, update them.

```
int prev_counter = -1;
int flag = 1;
```

Reference Output:



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

m. 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();
```

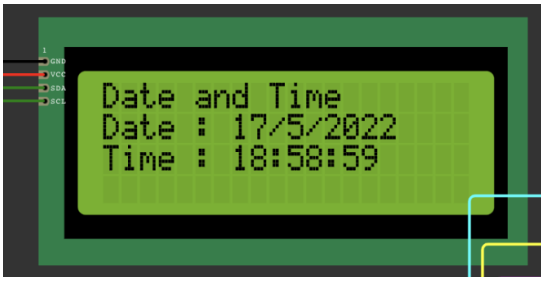
```
year = current.year();
month = current.month();
day = current.day();
hour = current.hour();
minute = current.minute();
second = current.second();
}
```

n. Next, code the functionality to display time on LCD.
We want the program to keep updating and displaying 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:



o. To allow change of the mode and try different features, we need to break the loop when again the push button on the encoder is clicked.

Reference Code:

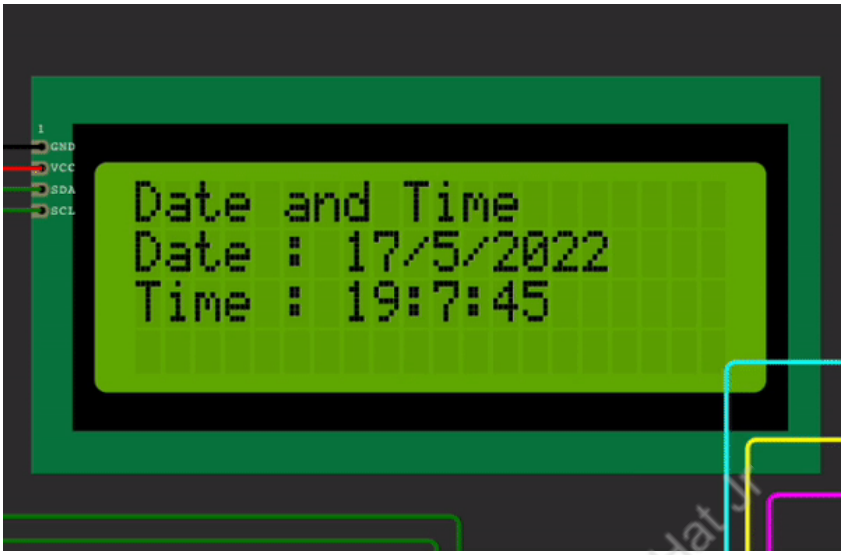
```
// looping button
button.loop();
if (button.isPressed()){
  lcd.clear();

  // 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:



https://s3-whjr-curriculum-uploads.whjr.online/75ea8c85-56cf-4c45-bf67-2311305ed838.gif

p. Next, we work on the **stopwatch** feature. Initialize our hour, minute and second variables to 0 for each.

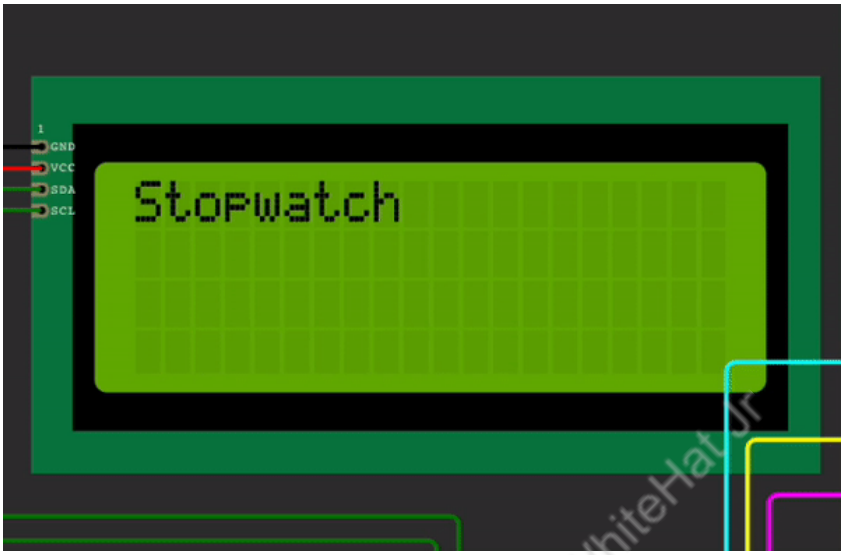
```
int stopwatch_hour=0, stopwatch_minute=0, stopwatch_second = 0;
```

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

q. We observed that the display is flickering a lot and the timer is very fast. To 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) +
                    " : " + 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

r. Now, we then observed that the seconds go beyond 60. Similary, 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.

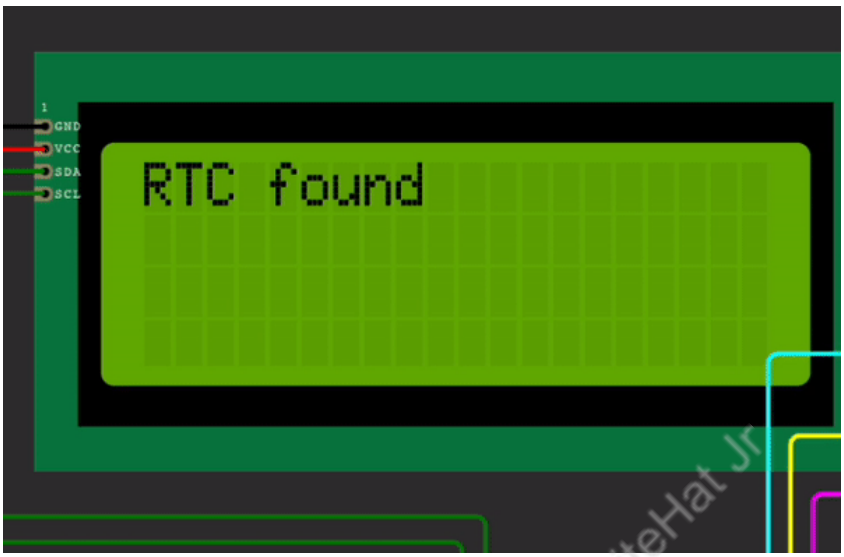
```
// breaking loop if button pressed
  button.loop();
  if (button.isPressed()){
    prev_counter = -1;
    flag = 1;

    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:



https://s3-whjr-curriculum-uploads.whjr.online/2500955a-e894-4494-a38d-1a7613686e96.gif

What's NEXT?

In the next class, we will add the functionalities of the timer and alarm clock.

Expand Your Knowledge

Read more about the LCD2004 here.