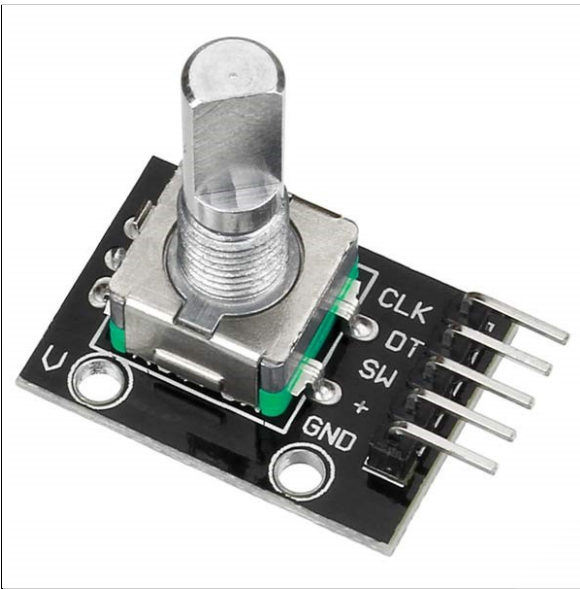


Topic	Smart Clock II	
Class Description	Students will learn how to add a rotary encoder and select the mode to work with more features of the smart clock.	
Class	PRO C266	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> • Understand Rotary Encoder KY040 • Understand the principle of mode selection • Understand Interrupts • Learn to use Rotary Encoder with Arduino 	
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?		ESR: Hi, thanks! Yes, I am excited about it!

Following are the WARM-UP session deliverables: <ul style="list-style-type: none"> • Greet the student. • Revision of previous class activities. • Quizzes. 	Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Activity Details Following are the session deliverables: <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. 	
TEACHER-LED ACTIVITY 15 mins	
Teacher Initiates Screen Share	
<ul style="list-style-type: none"> • Understand Rotary Encoder KY040 • Understand the principle of mode selection • Understand Interrupts • Learn to use Rotary Encoder 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? <i>Note: If the student has any doubts, clarify the doubts.</i>	ESR: Yes. ESR: Varied. ESR: Varied

What feature/s did we add to our smart clock in the last class?	ESR: Checking the time.
What more can be added?	ESR: Timer, Alarm, Stopwatch.
Great! We will work on these features of smart clocks in this class.	
Let's see how we can add these features.	
How many features are we planning to add?	ESR: Three.
To do so, what kind of hardware can be used?	ESR: One that supports multiple options.
Do you know any?	ESR: Yes/No.
We will use an Encoder for this functionality.	
Let's understand the Encoder and its working.	
A rotary encoder is a type of position sensor which is used for determining the angular position of a rotating shaft.	
It generates an electrical signal, either analog or digital.	
This is what it looks like.	



It is commonly used in car music players and computer devices such as a mouse. Do you know where else you might have seen these?

ESR: Varied

It's similar to a **potentiometer**.

But why aren't we using potentiometers?

ESR: Varied.

A **rotary encoder** can essentially be used for the same purpose as the potentiometer. However, a potentiometer typically has a default point past which the shaft cannot rotate, while a rotary encoder can rotate in one direction for as long as we want. To reset the position reading, we can push the shaft down to press the reset button. The rotary encoder we have produces digital signals.

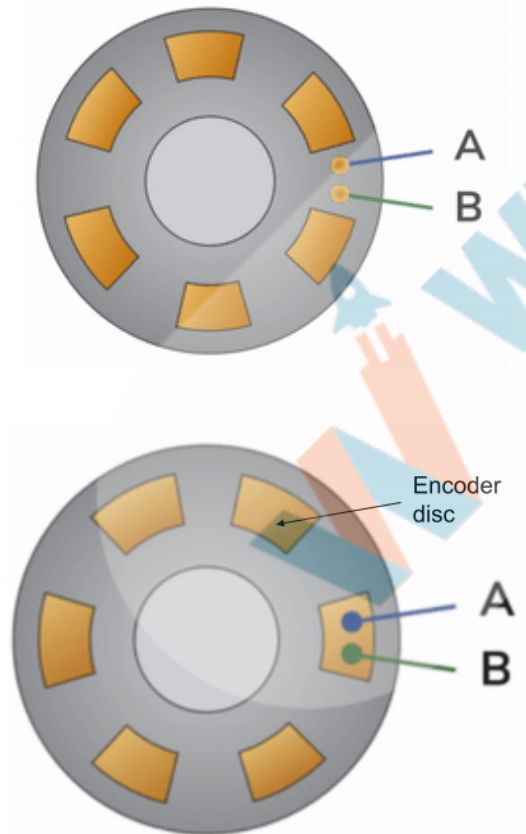
Now, let's have a look at the pins of the encoder:

1. **CLK:** Output A (Digital)

2. **DT:** Output B (Digital)
3. **SW:** Switch
4. **+**: VCC — Voltage supply
5. **GND:** Ground

Let's see how CLK and DT pins work with the encoder.

A rotary encoder calculates the angular position of a rotating shaft using a series of **square wave pulses**. It has two contact pins called A(CLK) and B(DT) that are 90 degrees out of phase. When the rotary encoder is manually rotated, pins A and B make contact with the disc and generate the pulse. By counting the number of pulses, we can know the rotary position.



<p>To check the direction and if the pins are rotating clockwise or counterclockwise, refer to the below gifs:</p> <p>Anti-clockwise: https://s3-whjr-curriculum-uploads.whjr.online/b3f84159-a2f2-4424-b713-14f39eeba0a1.mp4</p> <p>Clockwise: https://s3-whjr-curriculum-uploads.whjr.online/7f81b79c-3365-445f-86ce-180ddd2937af.mp4</p> <p><u>The Keyes KY-040 Rotary Encoder</u> The Keyes KY-040 is a rotary encoder and an input device that indicates how much the knob has rotated and what direction it is rotating in.</p> <p>It's a great device for stepper, servo motor control, to control devices like digital potentiometers.</p> <p>Great. Are you excited to use it?</p>	<p>ESR: Yeah!</p>
<p>How do we get started with it?</p> <p>Great. Let's start.</p> <p>Open the wokwi simulator and replace the file 'sketch.ino' and 'diagram.json' downloaded from Teacher Activity 1.</p> <p><i>Note: Refer to Lesson C260.</i></p>	<p>ESR: Add the encoder hardware and make its connections.</p>

WOKWI SAVE SHARE

sketch.ino diagram.json Library Manager

```

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

```

Simulation ▶ + ⋮

WOKWI SAVE SHARE

diagram.json Library Manager

```

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.

First, we create the circuit diagram.

Step 1: Select the components:

- 1 x KY-040 Rotary Encoder

Step 2: Make the connections:

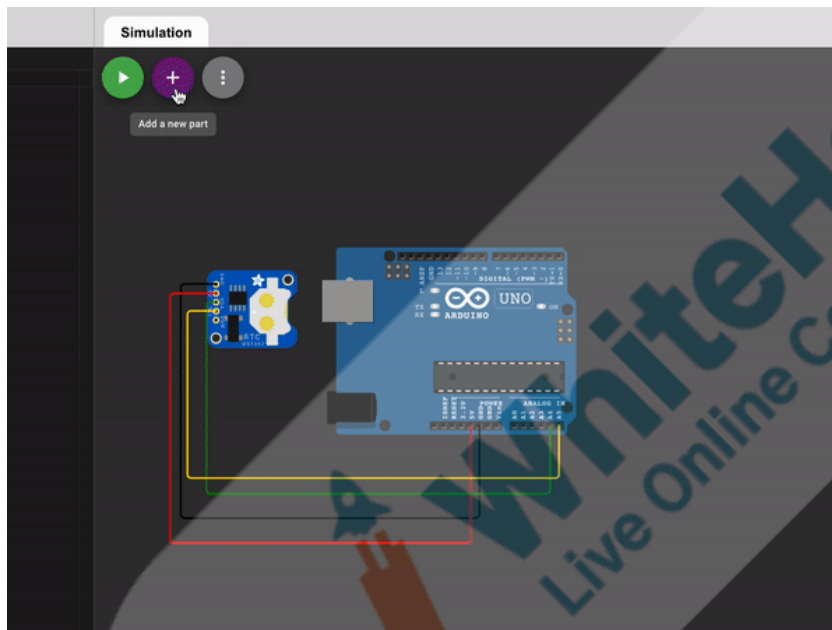
The circuit of this project consists of an **Arduino Controller**, and a **KY-040 Rotary Encoder**.

KY-040 Rotary Encoder	Arduino PIN
CLK	2
DT	3
SW	4
GND	GND

VCC/+

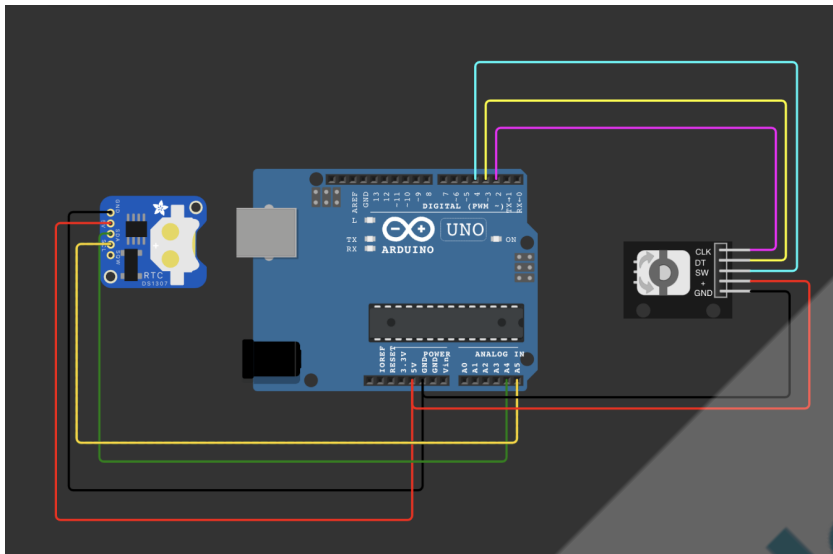
5V

*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.*



<https://s3-whjr-curriculum-uploads.whjr.online/c7b78bc6-0c90-42e6-a284-edced6d9dac7.gif>

Reference image:



Before we start coding, let's also learn about **Interrupts**.

Example:

<https://s3-whjr-curriculum-uploads.whjr.online/79e983ee-1bc6-49d5-a3f2-2c48b24a82d2.mp4>

Interrupt is a signal emitted by either hardware or software when a process or an event needs immediate attention. It alerts the processor to perform a high priority process which causes the current working process to be paused and it resumes once the interrupt is handled.

Hardware Interrupts occur when an external event, like a pin goes high or low.

Software Interrupts occur with software instruction.

How does it work?

When the event or interrupt happens, the processor takes immediate notice, saves its execution state, runs a small chunk of code (often called the **Interrupt Handler** or **Interrupt Service Routine**), and then returns to whatever it was doing before.

To work with interrupts, we have a few instructions.

1. attachInterrupt(interrupt, function, mode)

Specifies a function to call when an external interrupt occurs.

interrupt: the number of the interrupt (int)

function: the function to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.

mode defines when the interrupt should be triggered. Four constants are predefined as valid values:

- a. **LOW** to trigger the interrupt whenever the pin is low,
- b. **CHANGE** to trigger the interrupt whenever the pin changes value
- c. **RISING** to trigger when the pin goes from low to high,
- d. **FALLING** for when the pin goes from high to low.

2. digitalPinToInterrupt(pin)

Pin number of the interrupt, which tells the microprocessor which pins to monitor. The pin depends on the microcontroller being used.

Why do we use interrupts?

Interrupts are useful for making things happen automatically in microcontroller programs and can help solve timing problems. A good task for using an interrupt might be reading a rotary encoder, and monitoring user input.

If you wanted to make sure that a program always catches the pulses from a rotary encoder, and never misses a pulse, it would be difficult to write a program to manage it,

as it would have to constantly check the sensor (DT and CLK pins) lines of the encoder. In such cases, using an interrupt is more efficient while not missing the pulses.	
Superb! Now, let's learn to use Encoder with Arduino. 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	
<u>ACTIVITY</u> <ul style="list-style-type: none"> • Learn to use the Rotary Encoder with Arduino. 	
Teacher Action	Student Action
<p>Let's get started.</p> <p>Let's code now.</p> <ol style="list-style-type: none"> 1. We define the pins for the encoder connections. <pre>// encoder variables byte clk = 2; byte dt = 3; byte sw = 4;</pre> <p><i>Note: Use pins 2 or 3 for CLK as Interrupts are supported on pins 2 and 3 on Arduino Uno.</i></p>	<p>Student opens the wokwi simulator and replaces the files downloaded from Student Activity 1.</p>

2. Now, we need an interrupt to be connected to the **CLK** pin so that as soon as it goes LOW, the function **encoder()** should be called and executed. We do this in **setup()**

```
attachInterrupt(digitalPinToInterrupt(  
clk) , encoder , FALLING);
```

3. Let's now define the **encoder()**.

```
void encoder() {  
    if (digitalRead(dt) ==  
HIGH) counter++;  
    else counter--;  
    counter = constrain(counter , 0 , 3);  
}
```

We check that if we get a HIGH signal on the DT pin, the next option should be selected and hence we increase the counter variable value otherwise decrease the counter value as it has made an anti-clockwise movement.

constrain() instruction allows setting lower and upper limits for an encoder.

4. Next, we create the **mode_selector()** function to check for the counter value, and based on that the functionality of the smart clock is executed.

Note: Implementation of specific features of smart clocks will be done in the next class.

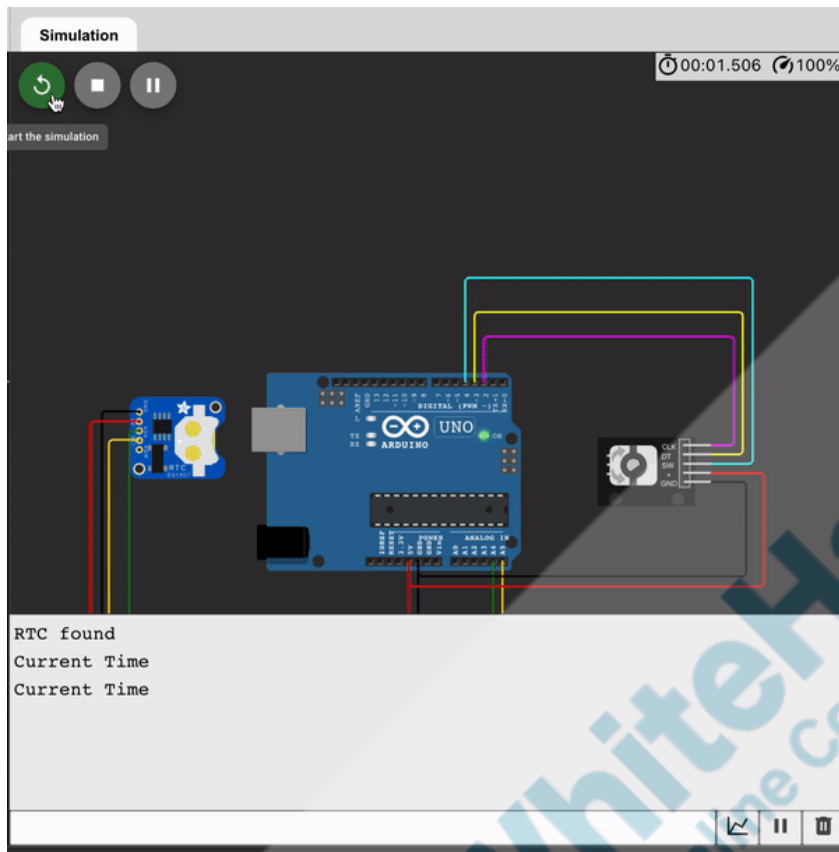
```
void mode_selector() {  
    if (counter == 0) {  
        Serial.println("Current Time");
```

```
}  
  
else if (counter == 1){  
    Serial.println("Set Alarm");  
}  
  
else if (counter == 2){  
    Serial.println("Stopwatch");  
}  
  
else if (counter == 3){  
    Serial.println("World clock");  
}  
  
}
```

5. Call the **mode_selector()** function in **loop()** to update the status of the encoder.

```
mode_selector();
```

Reference Output:



<https://s3-whjr-curriculum-uploads.whjr.online/fe909c9e-c923-4a58-8fe4-bec47142ec6d.gif>

It works. But it displays the selected mode repeatedly.
So how do we solve it?

We will use a flag variable which will be 0 in the beginning and will be set to 1 when it is rotated. Also, we will check the value of the flag when we execute the selected mode.

Update the functions **encoder()** and **mode_selector()**:

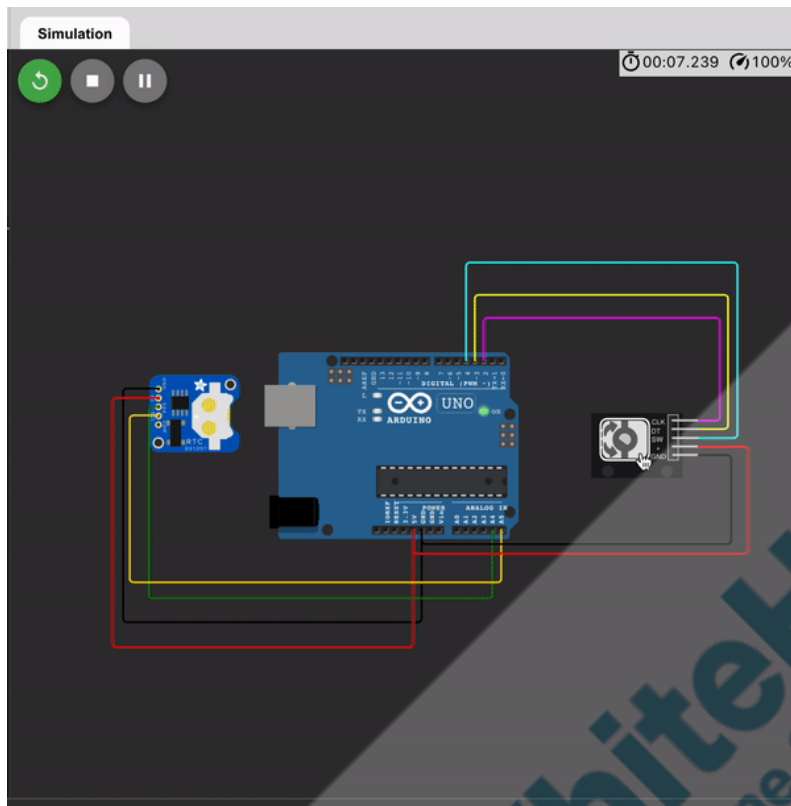
```
int flag = 0;
```

```
void encoder() {
```

ESR: Varied.

```
if (digitalRead(dt) == HIGH) counter++;  
else counter--;  
counter = constrain(counter, 0, 3);  
flag = 1;  
}  
  
void mode_selector() {  
  
if ( flag == 1) {  
    if (counter == 0) {  
        Serial.println("Current Time");  
    }  
    else if (counter == 1) {  
        Serial.println("Set Alarm");  
    }  
    else if (counter == 2) {  
        Serial.println("Stopwatch");  
    }  
    else if (counter == 3) {  
        Serial.println("World clock");  
    }  
    flag = 0;  
}  
}
```

Reference output:



<https://s3-whjr-curriculum-uploads.whjr.online/46a85379-9479-410a-ac1b-1948f05d9a61.gif>

Note: Teacher demonstrates the issue around the extremes. i.e. the first and last option getting repeated.

What do you see here?

ESR: The options are getting selected

Are the options getting selected properly?

ESR: No

So, how do we solve it?

ESR: Varied

We use the **prev_counter** variable to store the previous value of counter. So that when it increases/decreases even beyond the extremes, the previous value will help us solve it.

Reference code:

```
int prev_counter = 0;
```

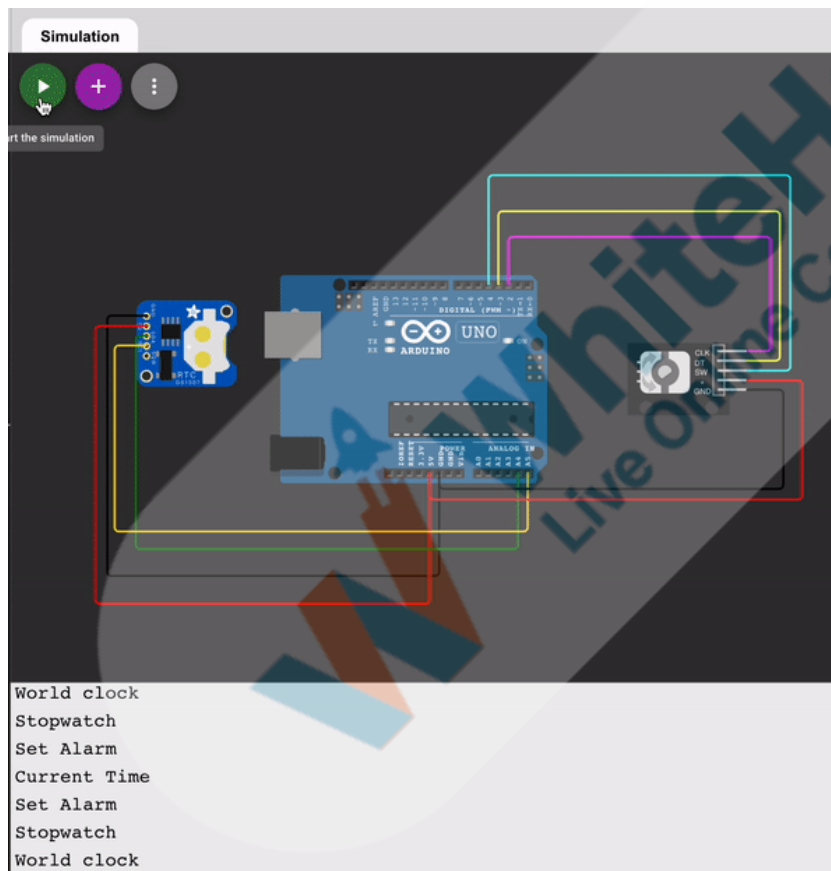
```
void encoder() {
  prev_counter = counter;
  if (digitalRead(dt) == HIGH) counter++;
  else counter--;
  counter = constrain(counter, 0, 3);
  flag = 1;
}

void mode_selector() {

  if (prev_counter != counter && flag ==
1) {
    if (counter == 0) {
      Serial.println("Current Time");
    }
    else if (counter == 1) {
      Serial.println("Set Alarm");
    }
    else if (counter == 2) {
      Serial.println("Stopwatch");
    }
  }
}
```

```
}  
  
else if (counter == 3) {  
    Serial.println("World clock");  
}  
  
flag = 0;  
}  
}
```

Reference output:



<https://s3-whjr-curriculum-uploads.whjr.online/32f5cb82-a227-44f4-8912-dcd076c5addf.gif>

6. To efficiently work with the push button on the encoder we use **ezButton** as the library provides in-built functionality.

Note: Students have already learned about ezButton in C253. Hence, the teacher must ask students questions about ezButton here.

```
#include<ezButton.h>
```

```
ezButton button(sw);
```

```
setup()
```

```
button.setDebounceTime(25);
```

```
loop():
```

```
button.loop();
```

```
if (button.isPressed())
```

```
    select_mode();
```

```
void select_mode() {
```

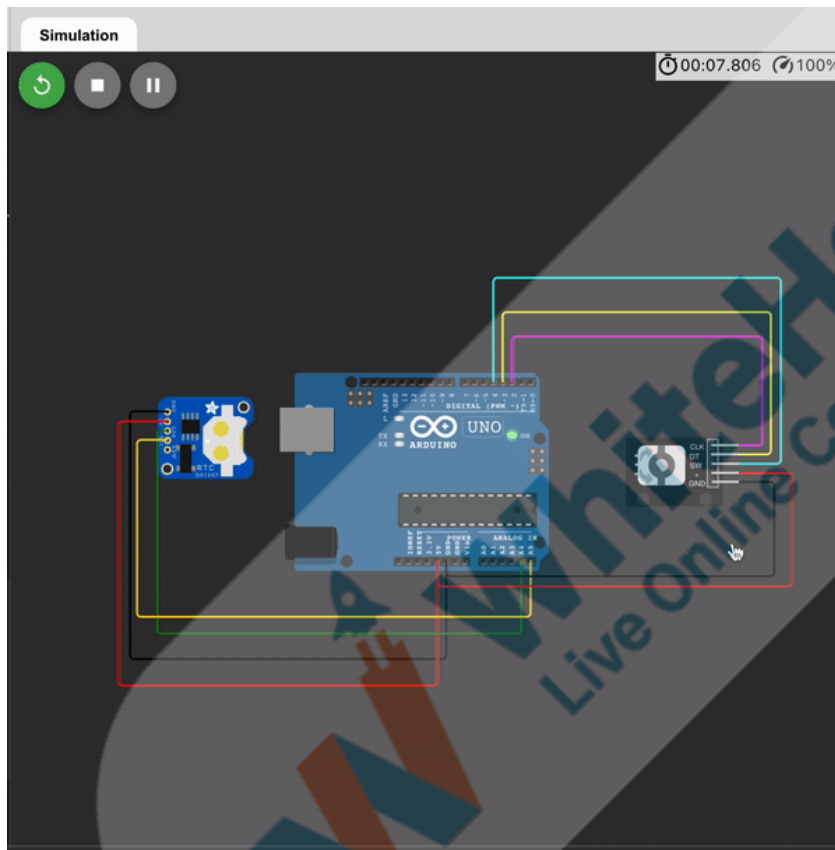
```
    if (counter ==  
0) Serial.println("Date and Time mode  
is selected");
```

```
    if (counter ==  
1) Serial.println("Set Alarm mode is  
selected");
```

```
    if (counter ==  
2) Serial.println("Stopwatch mode is  
selected");
```

```
if (counter ==  
3) Serial.println("Countdown Timer mode  
is selected");  
}
```

Reference output:



<https://s3-whjr-curriculum-uploads.whjr.online/5012a3fb-f523-4404-a624-9ed02a24335a.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 add the functionalities of the stopwatch, timer, and alarm clock and also add display hardware.</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/P-RO-C265-Clock-with-RTC
Teacher Activity 2	KY-040 Encoder	https://docs.wokwi.com/parts/wokwi-ky-040
Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/P-RO-C266-STUDENT-TEMPLATE
Teacher Reference 2	Reference Code	https://github.com/procodingclass/P-RO-C266-TEACHER-REFERENC-E-CODE
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/05bd57a0-9a7a-433e-b737-f32103595aea.pdf
Teacher Reference 4	Project Solution	https://github.com/procodingclass/P-RO-C266-Project-Solution.git
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/0afab6f5-3226-4f65-a3c6-f2144dbd73dd.pdf
Student Activity 1	C-266 Student Template	https://github.com/procodingclass/P-RO-C266-STUDENT-TEMPLATE
Student Activity 2	KY-040 Encoder	https://docs.wokwi.com/parts/wokwi

		-ky-040
--	--	-------------------------

