Smart Clock-II

What is our GOAL for this CLASS?

In this class, we learned how to add a rotary encoder and select the mode to work with more features of the smart clock.

What did we ACHIEVE in the class TODAY?

- Understood Rotary Encoder KY040
- Understood the principle of mode selection
- Understood Interrupts
- Learned to use Rotary Encoder with Arduino

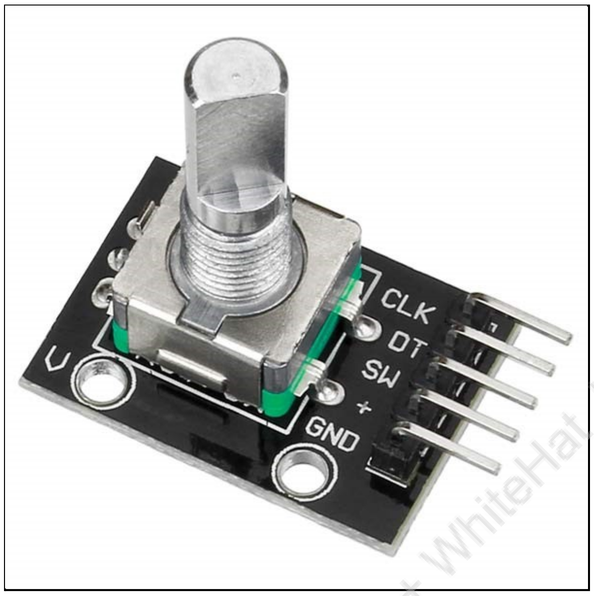
Which CONCEPTS/ CODING BLOCKS did we cover today?

- **Concepts**: Transmitting data, Receiving data, Infinite loops, Sequencing of code, Mode selection on the Rotary Encoder, Interrupts.
- **Coding blocks:** Serial.begin(), Serial.print(), Serial.available(), delay(), Serial.readString(), Serial.setTimeout(), pinMode(), digitalRead(), attachInterrupt(), setDebounceTime(), constrain(), isPressed().

How did we DO the activities?

1. **Learned about Rotary Encoders:**

   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.



   Pins and what they mean:
   1. **CLK:** Output A (Digital)
   2. **DT:** Output B (Digital)
   3. **SW:** Switch
   4. **+:** VCC — Voltage supply
   5. **GND:** Ground

   It is commonly used in car music players and computer devices such as a mouse.

   Working:
   To check the direction and if the pins are rotating clockwise or counterclockwise, refer to the below gifs:

   Anti-clockwise:
   https://s3-whjr-curriculum-uploads.whjr.online/b3f84159-a2f2-4424-b713-14f39eeba0a1.mp4

Clockwise:
https://s3-whjr-curriculum-uploads.whjr.online/7f81b79c-3365-445f-86ce-180ddd2937af.mp4

The Keyes KY-040 Rotary Encoder:
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.

2. **Learned about Interrupts:**
   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.

   To work with interrupts, we have a few instructions.
   a. **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.

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

3. **Learned how to use KY-040 Encoder with Arduino:**
   a. Open the wokwi simulator and create a new Arduino Uno project.
   b. Download the code from the template and replace the files '**sketch.ino**' and '**diagram.json**'
   c. Then, we create the circuit diagram.

Step 1: Select the components:

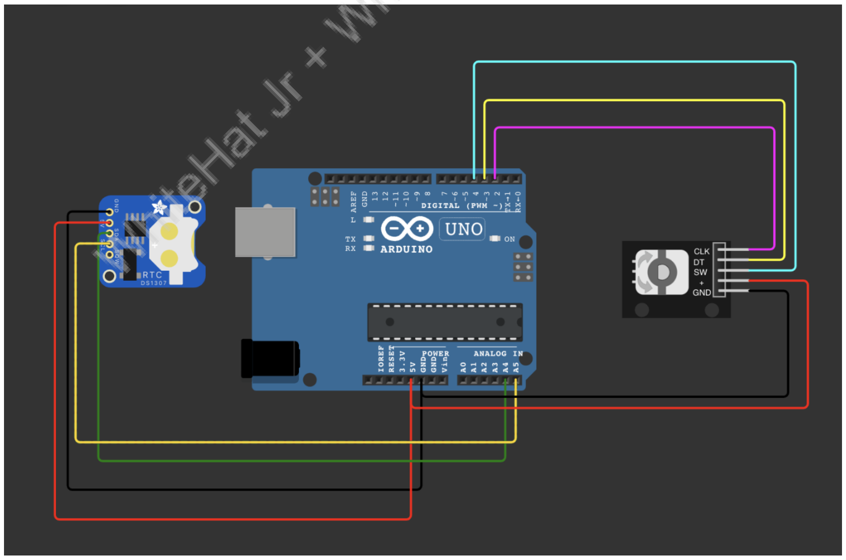
- 1 x **KY-040 Rotary Encoder**

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

Reference image:



d. We defined the pins for the encoder connections.

```
//  encoder variables
byte clk = 2;
byte dt = 3;
byte sw = 4;
```

Note: Use pins 2 or 3 for CLK as Interrupts are supported on pins 2 and 3 on Arduino Uno.

e. Next, an interrupt is connected to the **CLK** pin so that as soon as it goes LOW, the function **encoder()** should be called and executed. We did this in **setup()**

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

f. Next, 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.

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

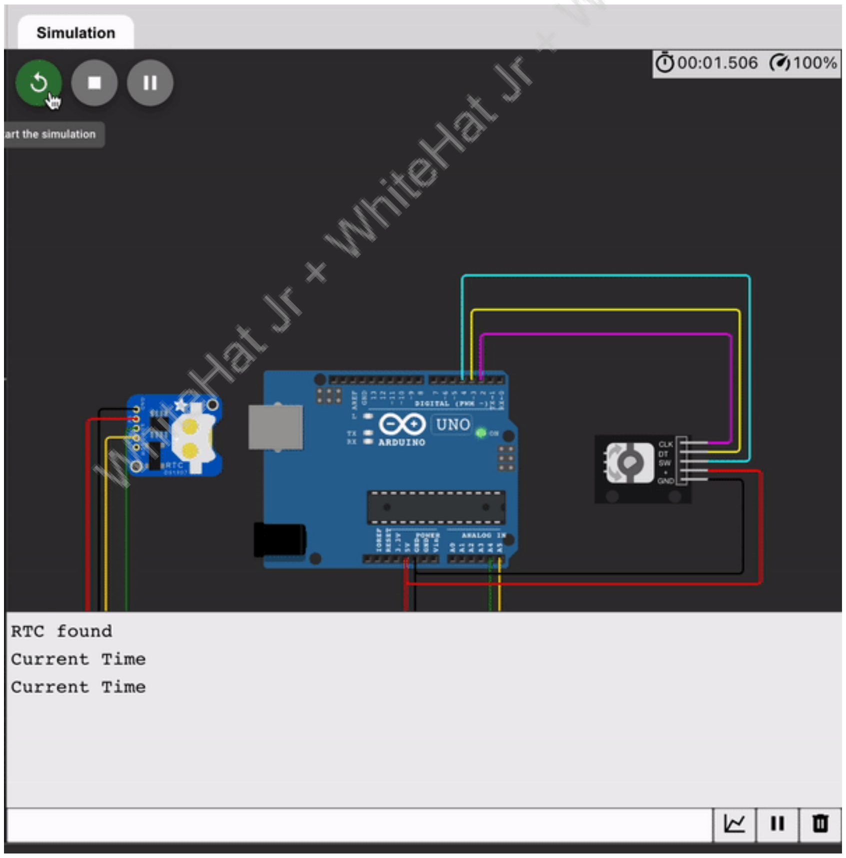
```
void mode_selector(){
  if (counter  ==  0){
    Serial.println("Current Time");
  }
  else if (counter  ==  1){
    Serial.println("Set Alarm");
  }
  else if (counter  ==  2){
```

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

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

```
mode_selector();
```

Reference Output:



```
RTC found
Current Time
Current Time
```

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

i. Next, we solved the issue with the selected mode repeatedly.
We used 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()**:

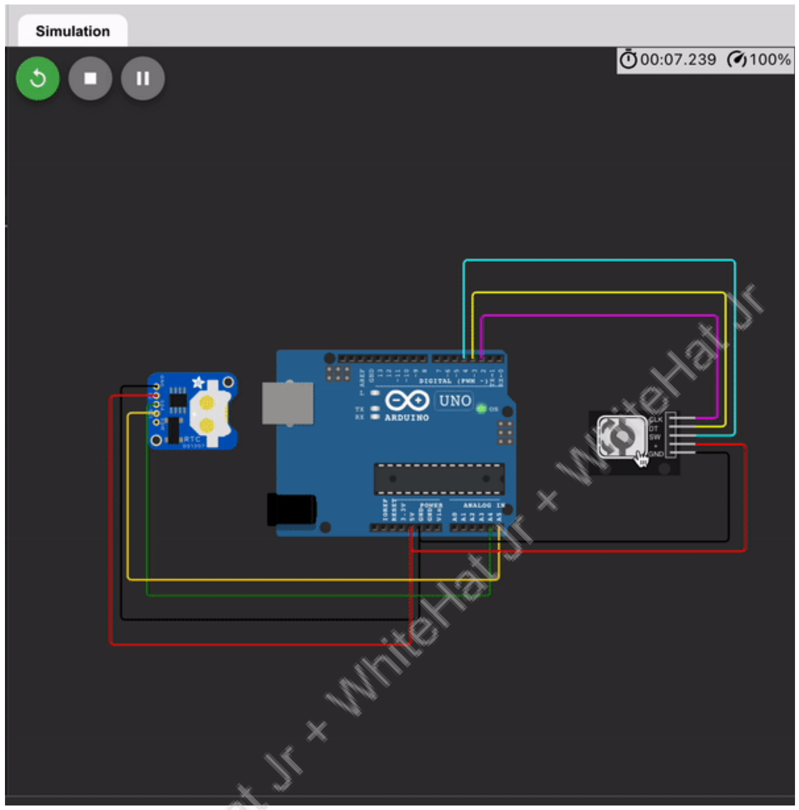
```
int flag = 0;
```

```
void encoder() {
 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

j. Next, we solved the issue with the first and last option being repeated again and again.
We used the **prev_counter** variable to store the previous value of the counter. So that when it increases/decreases even beyond the extremes, the previous value will help us solve it.

Reference code:

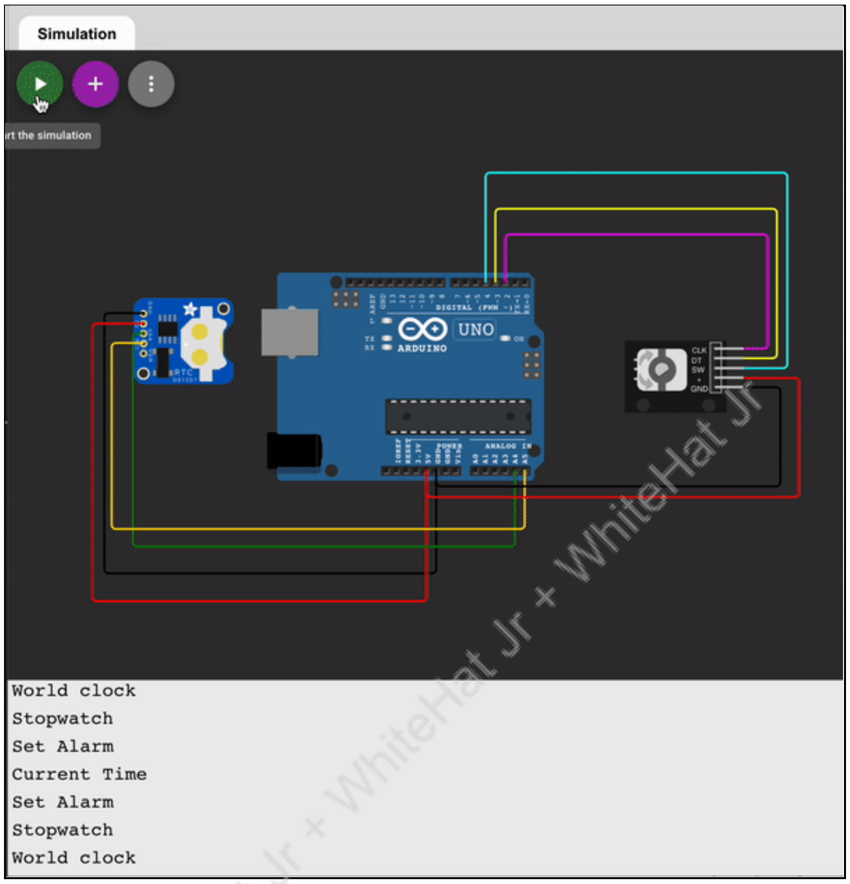
```
int prev_counter = 0;
```

```
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:



```
World clock
Stopwatch
Set Alarm
Current Time
Set Alarm
Stopwatch
World clock
```

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

k. We used **ezButton** to efficiently work with the push button on the encoder as the library provides in-built functionality.

```
#include<ezButton.h>
ezButton button(sw);
```

setup()

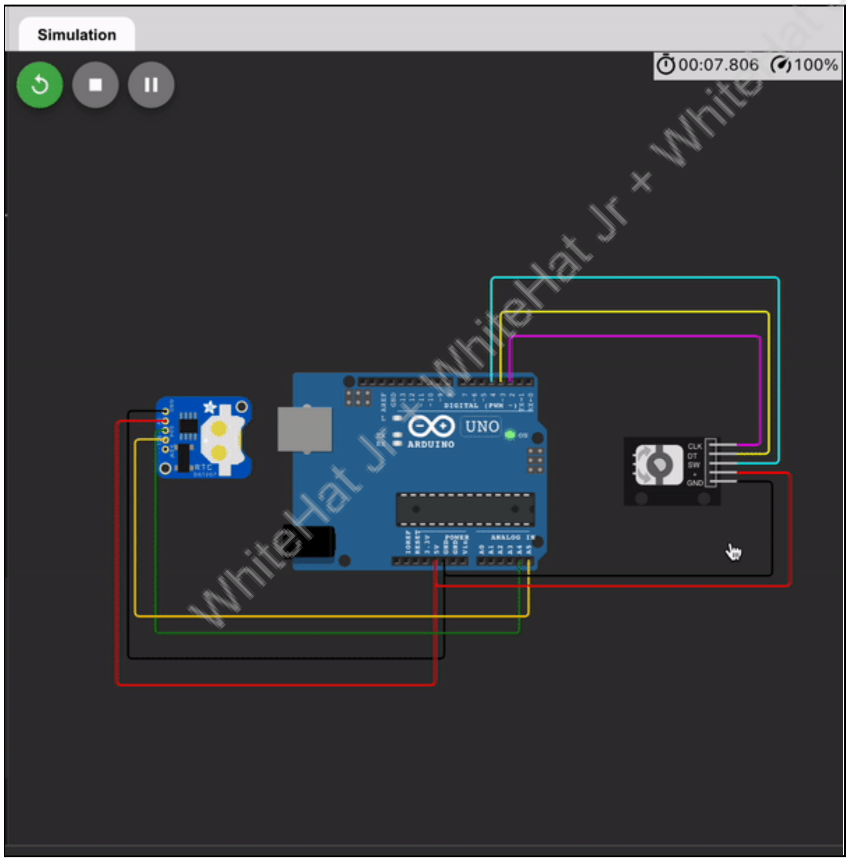
```
button.setDebounceTime(25);
```

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

What’s NEXT?

In the **next class**, we will learn about a new hardware to display and also implement code for smart clock features.

Expand Your Knowledge

Read more about the Rotary Encoder here.