Topic	Smart Clock I	
Class Description	**Students will learn how to calculate time using the RTC module.**	
Class	**PRO C265**	
Class time	**50 mins**	
Goal	● Understand how to calculate time internally with Arduino ● Understand the need for the RTC (Real-time clock) module ● Learn to use the RTC module 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:**	**ESR**: Hi, thanks! Yes, I am excited about it! Click on the slide show tab

● Greet the student. ● Revision of previous class activities. ● Quizzes.	and present the slides
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 how to calculate time internally with Arduino** ● **Understand the need for the RTC (Real Time Clock) module**	
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 do you read time?	**ESR**: Varied.

Yes. Correct. The time is read in hours, minutes, and seconds. After every 60 seconds, a minute is increased by a unit and every 60 minutes makes an hour. We have 24 hours a day. Can you tell me what time it is now? From where did you check? But how does the clock tell us the time? What if the batteries run out? Perfect. In the case of Arduino, it will store time and the code will keep on updating. Let's create a clock using Arduino. Are you excited?	**ESR**: Varied. **ESR**: Wall Clock/Watch. **ESR**: The hardware keeps rotating the three hands of hours, minutes, and seconds as per the calculation and shows the correct current time once we set the time. **ESR**: We change the batteries and update the clock to the current time. The batteries provide power to the hardware and it keeps updating and we see the correct time. **ESR**: Yes.
How do we get started with it?	**ESR**: Set the time first and then it keeps updating.

Do we manually set the time the very first time?

ESR: Yes.

Great. Let’s start.

Open the wokwi simulator and create a new Arduino Uno project.

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

1. We need variables to store the minutes, hours, and seconds.

```
byte seconds = 0;
byte minutes = 0;
byte hours = 0;
```

2. Let’s ask the user to enter the time. It should be done in the very beginning and once. So we write in **setup()**.

```
//manual setting time in the beginning
 Serial.print("Enter the hours passed : ");
 while (!Serial.available());
 hours = Serial.readString().toInt();
 Serial.println(hours);

 Serial.print("Enter the minutes passed : ");
 while (!Serial.available());
 minutes =
Serial.readString().toInt();
 Serial.println(minutes);
```

3. Also, we pass the value 9600 to the speed parameter. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second. That's 9600 binary ones or zeros per second, and is commonly called a **baud rate**.

```
Serial.begin(9600);
```

4. Now, we need a calculation for time. Every 60 seconds makes a minute and so on. To do so and validate the time, let's create a function.

```
void time_check(){
 if (seconds >= 60){
   seconds = 0;
   minutes++;
 }
 if (minutes >= 60){
   minutes = 0;
   hours++;
 }
 if (hours >= 24){
   hours = 0;
 }
}
```

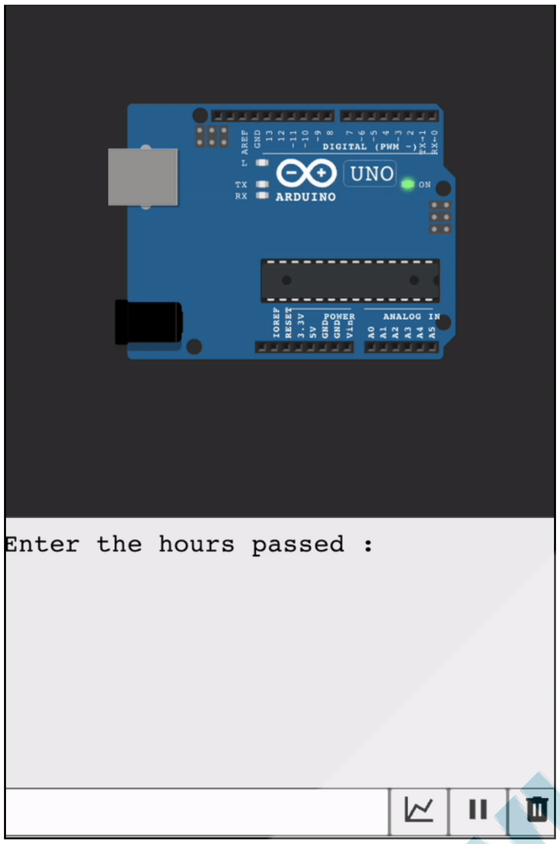
5. Call this function in the **loop()** to keep validating the time after updating.

```
time_check();
```

6. Next, we need to keep updating time after every second. To do so, we delay it for a second and print the current time to check on the console and increase the seconds variable by 1 unit.

```
delay(1000);
String time = "Time : " +
String(hours) + ":" + String(minutes)
+
               ":" + String(seconds);

   Serial.println(time);
 seconds++;
```

Reference output:

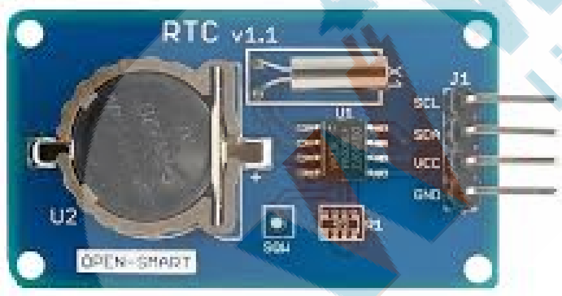
 Enter the hours passed : https://s3-whjr-curriculum-uploads.whjr.online/496f0b6b-4297-43ba-86d7-8a03cb97973a.gif Great. It's working. But does it seem good to enter the time again and again? Exactly. It shouldn't. How can we make it better? Arduino doesn't have the capability to store the time when it restarts. We need an external component to do so.	**ESR**: Nope. It shouldn't ask the time every time we run the program. **ESR:** Varied.

And that is **DS1307 RTC**.
Let’s understand what it is and how it works.

The first question that comes here is, why do we actually need a separate RTC for our Arduino Project when the Arduino itself has a built-in timekeeper?
Well, the point is that the RTC module runs on a battery and can keep track of the time even if we reprogram the microcontroller or disconnect the main power.

RTC is a Real-Time Clock. A real-time clock is a clock that keeps track of the current time and that can be used to program actions at a certain time.
The chip maintains seconds, minutes, hours, days, dates, months, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap years (valid up to 2100). The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator.
The **DS1307** incorporates a battery input and maintains accurate timekeeping when the main power to the device is interrupted.

Below is the diagram of RTC.



Pin Description:
SCL is the clock input for the I2C interface and is used to synchronize data movement on the serial interface.

SDA is the data input/output for the I2C serial interface.

VCC/5V pin supplies power for the module. It can be

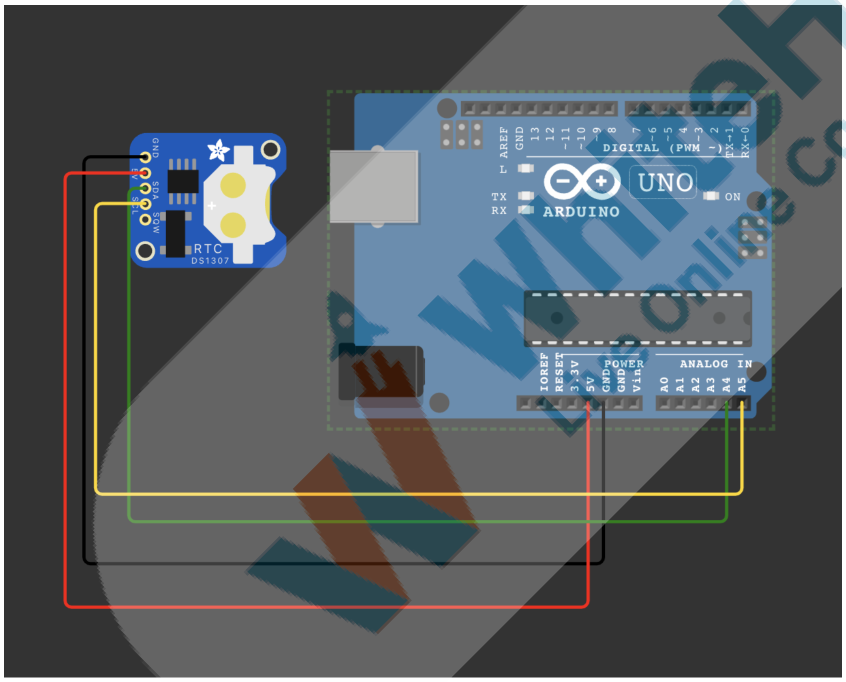
anywhere between 3.3V to 5.5V. **GND** is a ground pin.	
Superb! Now, let's learn to use RTC with Arduino. Are you excited to do this now?	**ESR**: Yes.
Teacher Stops Screen Share	
STUDENT-LED ACTIVITY- 15 mins	
• **Ask the student to press the ESC key to come back to the panel.** • **Guide the student to start Screen Share.** • **The teacher gets into Full Screen.**	
Student Initiates Screen Share	
ACTIVITY • **Learn to use the RTC module with Arduino.**	
Teacher Action	Student Action
Let's get started. First, we create the circuit diagram. **Step 1: Select the components:** • 1 x **Arduino Uno** • 1 x **DS1307 RTC** **Step 2: Make the connections:** The circuit of this project consists of an **Arduino** Controller and a **DS1307 RTC**.	Student opens wokwi simulator and creates a new **Arduino Uno** project.

DS1307 RTC	**Arduino PIN**
VCC/5V	5V

GND	GND
SDA	A4
SCL	A5

*Note: Wire color can be changed by **clicking on 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.*

Reference image:



Let's code now.

1. Add the library

```
#include <RTClib.h>
```

2. Create an object to work with the RTC module.

```
RTC_DS1307 rtc;
```

3. Check if it is initialized or not.

```
void setup(){
 Serial.begin(9600);
  if (!rtc.begin()){
   Serial.println("RTC not
initialized");
   while(true);
  }
 Serial.println("RTC found");
}
```

4. Create the variables to store the date and time.

```
String rtc_date = "";
String rtc_time = "";
```

5. Create a function to get date.

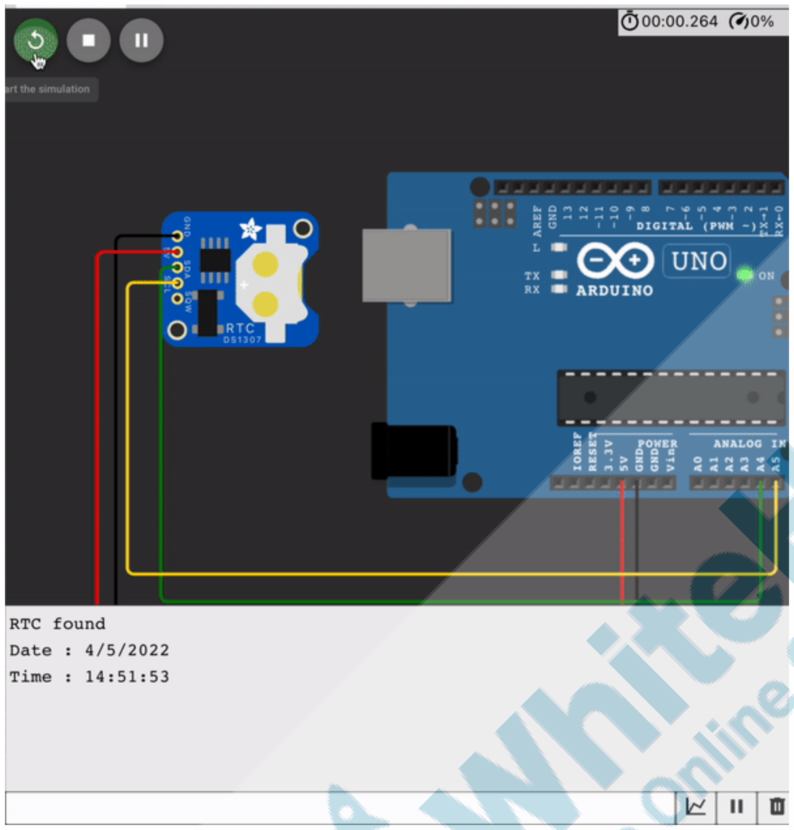
```
String get_date(DateTime current){

 int year = current.year();
 int month = current.month();
 int day = current.day();

 String current_date = "Date : " +
String(day) + "/" + String(month) +
               "/" + String(year);
  return current_date;
}
```

6. Similarly, we create a function to get time.

```
String get_time(DateTime current){

 int hour = current.hour();
 int minute = current.minute();
 int second = current.second();

 String current_time = "Time : " +
String(hour) + ":" + String(minute) +
                       ":" +
String(second);

 return current_time;
}
```

7. Call these functions in a loop to get date and time and save them in variables.
   rtc.now() returns the current date and time together. To extract them separately, we used hour(), minute(), second(), day(), month(), year() instructions above.

```
void loop(){
 DateTime dt = rtc.now();
 rtc_date = get_date(dt);
 rtc_time = get_time(dt);

 //  for better working of simulator
 delay(1000);
}
```

Reference output:

`RTC found`
`Date : 4/5/2022`
`Time : 14:51:53`
https://s3-whjr-curriculum-uploads.whjr.online/e6118a2e-da98-45c7-b394-118d34412c17.gif
 	
Our clock is ready.
Wouldn't it be great to make our clock smart?
 	
How can we do so?
 	
Yes, we will add these features in upcoming classes.
	ESR: Yes.
	
	ESR: Add Timer, Stopwatch, and Alarm clock.

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)

<u>FEEDBACK</u>

- **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 learn how to add encoders for selecting from multiple functionalities of a 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	

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Simulator	https://wokwi.com/
Teacher Activity 2	RTC	https://docs.wokwi.com/parts/wokwi-ds1307
Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/PRO-C265-TA-Reference-Code
Teacher Reference 2	Reference Code	https://github.com/procodingclass/PRO-C265-Clock-with-RTC
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/2f137aee-bf6a-4634-a46a-77b271d1abbb.pdf
Teacher Reference 4	Project Solution	https://github.com/procodingclass/PRO-C265-Project-Solution.git
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/69547b7b-e2f0-44ce-956b-125607ad564e.pdf
Student Activity 1	Simulator	https://wokwi.com/

Student Activity 2	RTC	https://docs.wokwi.com/parts/wokwi-ds1307