Topic	Digital Book 1	
Class Description	**Students will learn how to interface an SD card with the Arduino. Using this knowledge, they will create a digital book.**	
Class	**PRO C273**	
Class time	**50 mins**	
Goal	• Understanding the basics of an SD card. • How to interface an SD card with the Arduino. • Reading data from the files which are there in the SD card.	

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 in more interest in students.	
TEACHER-LED ACTIVITY 15 mins	
Teacher Initiates Screen Share	
• **Understanding the basics of an SD card.** • **Interfacing SD card with the Arduino.** • **Initializing the SD card**	
Teacher Action	**Student Action**
Do you remember what we did in the last class? Great, Let's try to recall some of the important things that we covered in the last class, Which sensor did we use to control our **snake sprite**? Absolutely correct! Which component did we use to **display** our game? Perfect. It's great you remember all the things!	**ESR** : Varied **ESR** : Joystick **ESR** : 8X8 matrix

Now let's learn something new today. Can you tell me why you were able to recall the things that we learnt in the previous class, or which organ in your body is responsible for **storing** all the information or knowledge that you receive in the class? That's correct. To be more precise, the **memory** in your **brain** is responsible for storing all the information that you receive through your sense organs. Can you tell me the types of **memory** that our human brain has? Our human brain has **3** types of **memory** : a) **Sensory memory :** • **Sensory memory** stores the information that we receive from our **5 senses, touch, taste, smell, hearing, vision**. • The information in **sensory memory** retains for less than **4 seconds**. • **Examples** : Seeing a dog, smelling noodles etc. b) **Short term memory :** • **Short term memory** holds the information that a person is currently thinking about or is aware of. • The information in **short term memory** retains for less than **30 seconds**. • **Examples** : Remembering a phone number which is just recited to you, what you had for lunch yesterday, etc.	**ESR** : Brain **ESR** : Varied

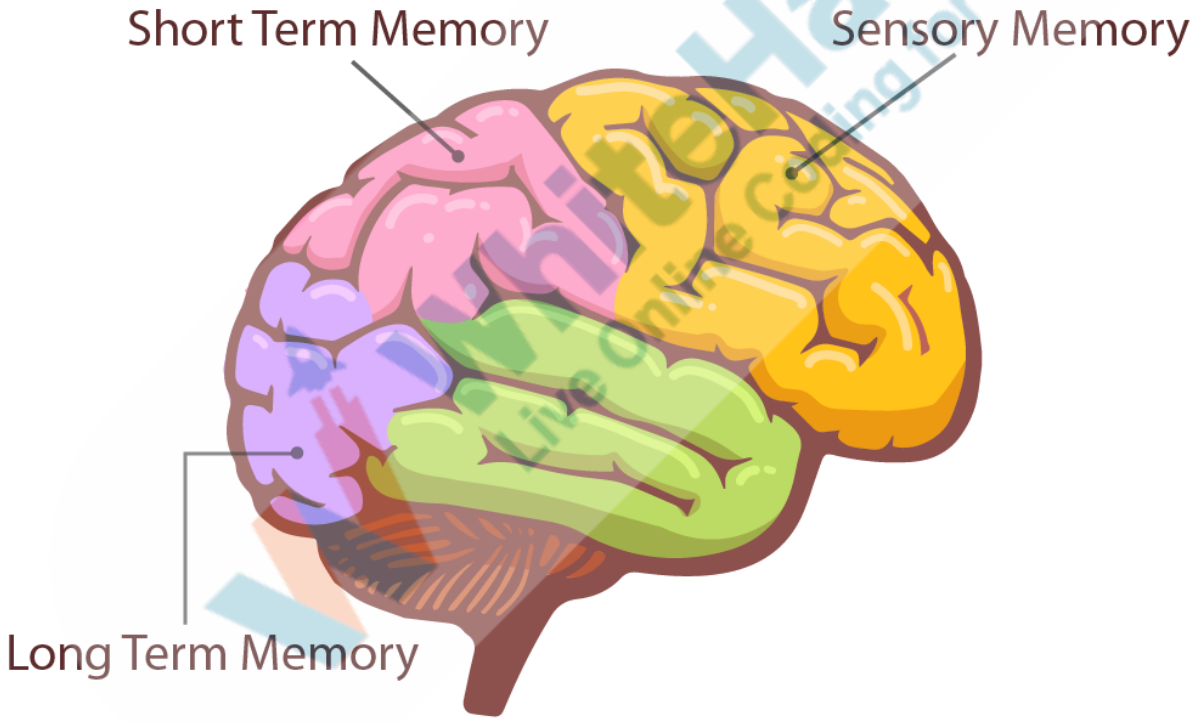
c) **Long term memory :**

- **Long term memory** holds the information or knowledge for a very long duration.
- **Examples :** Your birth date, first job, Your name etc.

Great, can you tell the amount of information or data that your brain can store?

Scientifically, your brain can hold **2.5 petabytes** or **2.5 million gigabytes**.

ESR : Varied



Now just like a human brain has memory, the **electronic brains** like **Arduino, ESP32** etc have **memory** too.

Can you tell me the types of **memory** that an **Arduino** has?

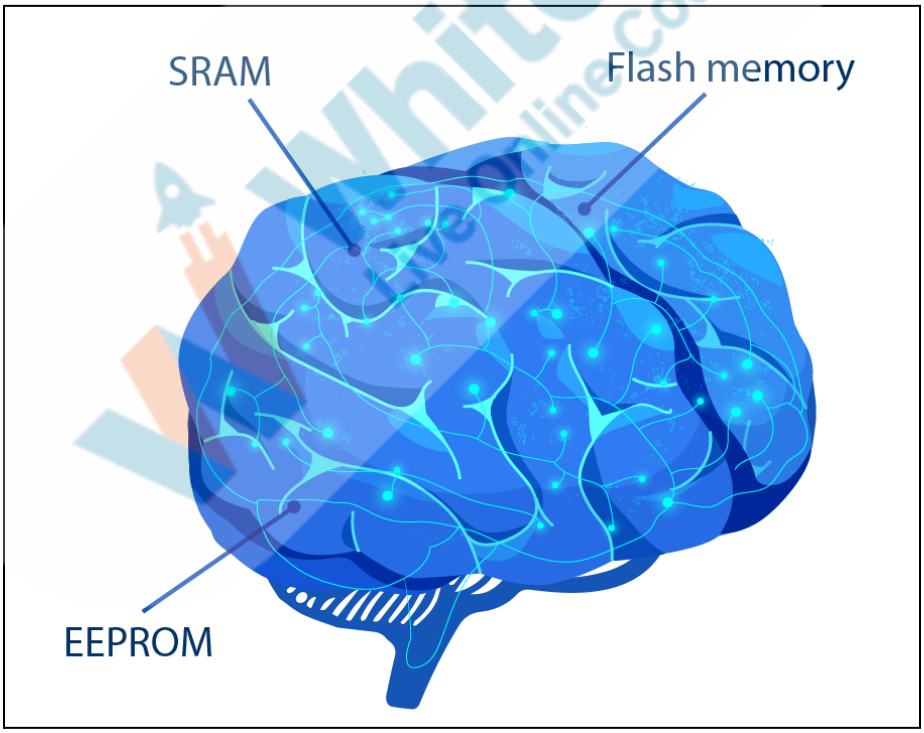
The **Arduino** has **3** types of memory.

a) **Flash memory or the program memory :** The space where your sketch or program is stored.

b) **Static Random access memory (SRAM):** It's the space where the Arduino creates and manipulates variables.

c) **EEPROM :** It's the long term memory of Arduino, which can be used by us to hold information for a very long duration.

ESR : Varied



Can you tell me the amount of memory the Arduino has? The arduino has **32 kb (flash memory) + 2 kb (SRAM) + 1kb (EEPROM) = 34 kb memory.** Let's do a simple calculation. **34 kb** is **34 * 1024 bytes** which is **34816 bytes.** 34 x 1,024 bytes = 34,816 bytes Now **1 character** takes **1 byte** of space, which means your Arduino can roughly store **34816 characters.** That's very less as compared to the human brain, right? So what if we want to process a very large amount of data with our Arduino? Will it be possible to store all that data in its memory? Can you tell me what can be done to solve this challenge? What do we do when we want to store excessive data in our phone? Correct, we use an external storage device like an **SD card** with our phone. What does **SD** stand for? It stands for **"Secured Digital" card.** Now just like we can use it with our phone to store	**ESR** : Varied **ESR :** Yes **ESR** : No

excessive data, similarly, we can interface it with Arduino as well. For that, a) Open the wokwi simulator and create a new Arduino project. b) Drag out the following components in the workspace. • **Arduino board X 1.** • **Micro SD card X 1.** c) Before making the connections, let's have a quick look at the pinout of the micro SD card module. It has the following pins : • **CD (Card detect) :** This pin should be connected to **ground** if there is no SD card in the module, but in case of wokwi simulator, there is always a card within the module, so you don't need to worry about this pin. • **DO/MISO (Data out / Master in slave out) :** The **SD card** or the **slave** device uses this pin to **send** data towards the **Arduino** or the **master** device. • **GND (Ground) :** This pin is used to provide **0 volts** to our **SD card**. • **SCK (Serial clock line) :** This pin is used to generate **clock signals** so that the communication or the data exchange between the **Arduino (master)** and the **SD card (slave)** is **synchronous**. • **VCC (Power line) :** This pin is used to	**ESR** : We use external storage devices like SD cards, Hard disks etc. **ESR** : Varied

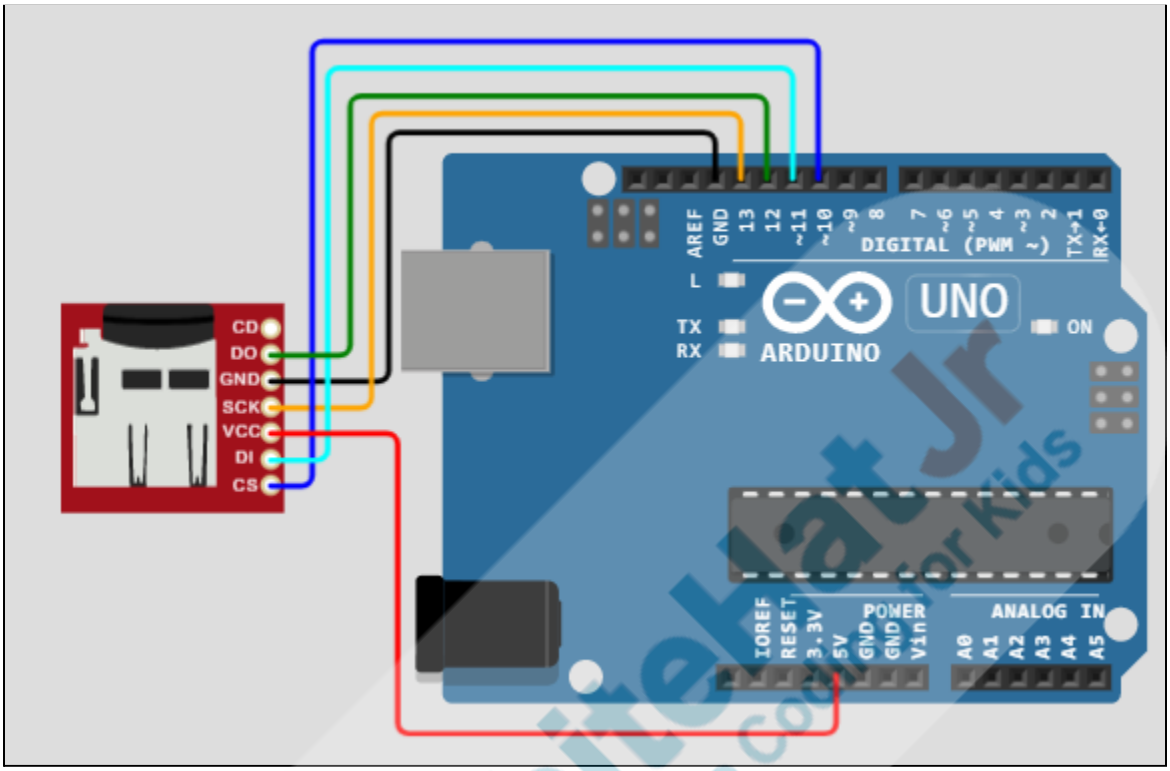
provide **5 volts** to the **SD card**.

- **DI/MOSI (Data input / Master out slave in) :** The **SD card (slave)** uses this pin to **receive** data from the **Arduino (master)**.
- **CS (Chip select / Slave select) :** This pin is used to decide, with which **slave** the **master** will communicate. (In case, there are multiple slaves)

Now let's connect the **SD card** with the **Arduino** using the instructions given below.

SD card	Arduino
CD **(Card detect)**	-
DO **(Data output)**	12
GND **(Ground)**	GND **(0 volts)**
SCK **(Serial clock line)**	13
VCC **(Power pin)**	5V **(5 Volts)**
DI **(Data input)**	11
CS **(Chip select)**	10

The circuit diagram will look as,



After making the connections,

- Open the Teacher Activity 2
- Download both **harry.txt** and **think.txt** files.
- Upload these files into your project, so that your **project menu bar** appears as,

The **think.txt** file contains the first chapter of the book, **"Think and grow rich"** and **harry.txt** contains the first chapter of the **"Harry potter"** book.

Note: Whenever you start the simulation, the wokwi simulator will copy both harry.txt and think.txt files into your SD card.

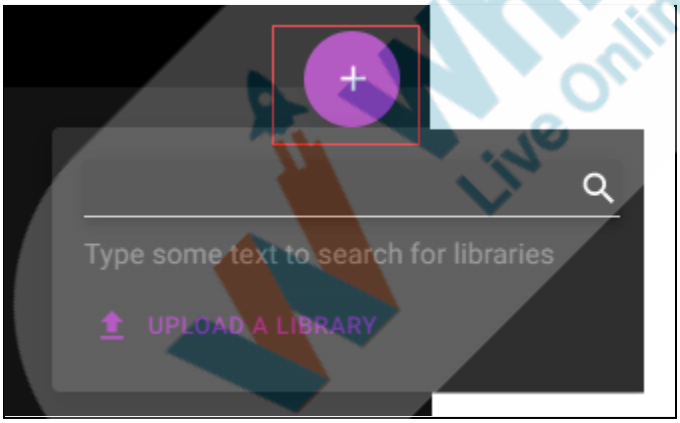
Great, it's time to code now. Let's define the **process** or the **flow** of our program as,

1) **Open the file in read mode.**
2) **Read all the data from it.**
3) **Display it on the Serial monitor.**

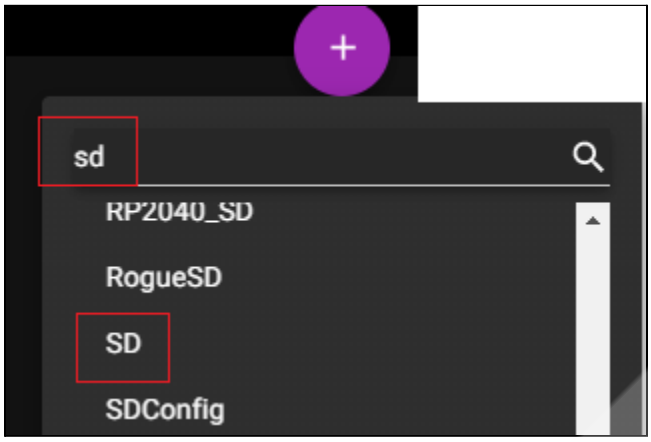
We will use **<SD.h>** header file in this project. For that we need to add it using the library manager first.

1. Go to the **library manager** tab.

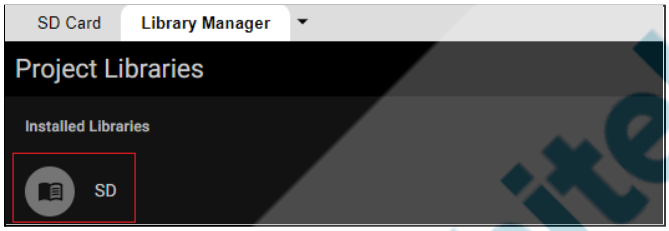
2. Click on the **Add library button (+ sign),** so that you can add a library in your program.



3. Search **'SD'** in the search bar. Scroll down and look for the SD library. Once you find it, add it.



4. SD library should be installed after this step.



Let's start defining each of the processes that we had listed. The first step is to **open** the file in **read mode**. For that,
In the sketch.ino file, **Import** the **SD card module** using the command,

#include <SD.h>

```
#include <SD.h>
```

From the SD card, we will be reading a file. To store this file, we need an object. Create a **Global file object** using the command,

File file;

```
File file;
```

In the **setup()** method, let's first initialize the serial communication using the **.begin()** method as,

Serial.begin(9600);

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

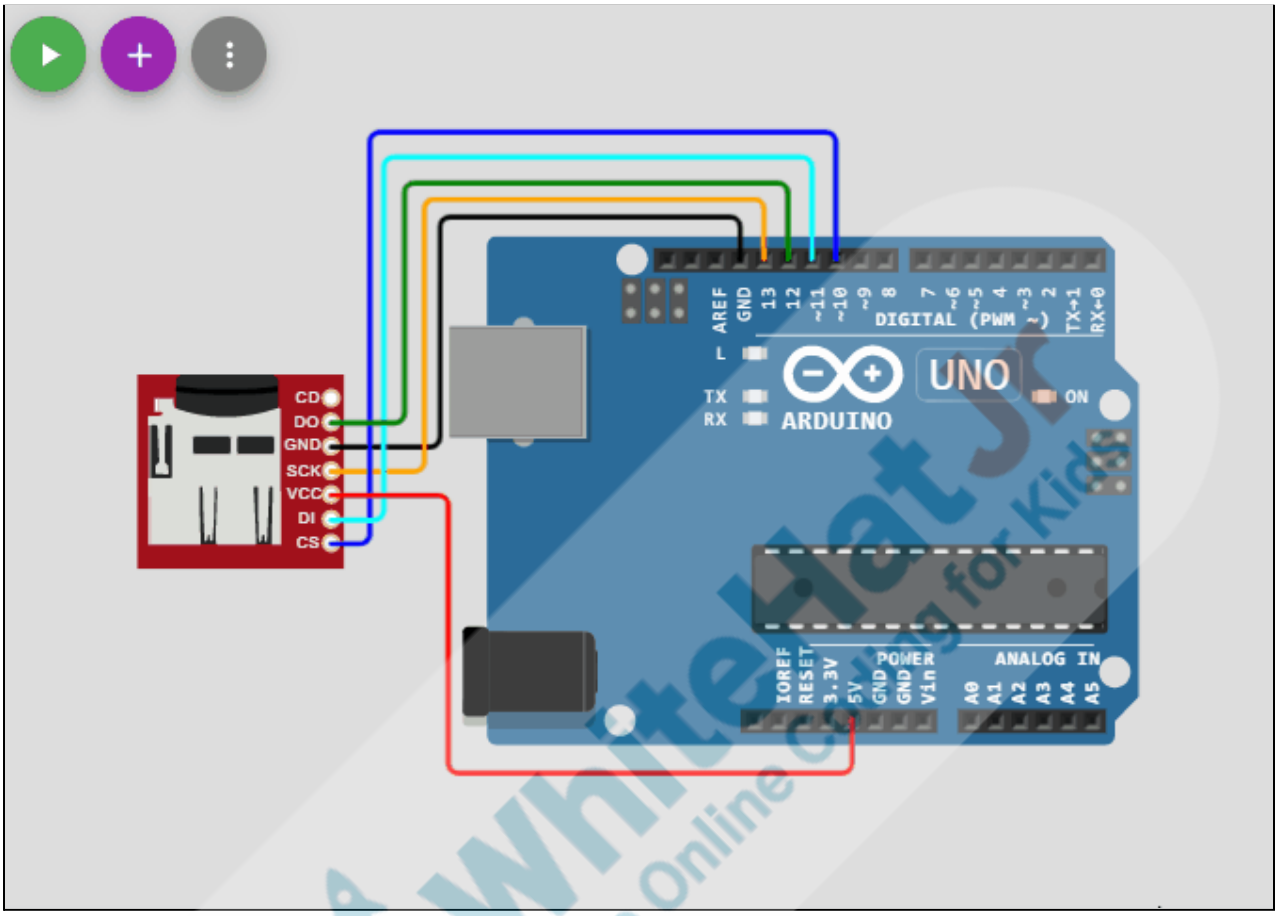
Initialize the SD card using the **.begin()** method as,

if (!SD.begin())Serial.println("SD card not detected");
else Serial.println("SD card detected");

The above piece of code will run the **SD.begin()** command, and if it does not return a valid output, we will see a message printed on the **serial terminal** which says, **"SD card not detected"**, otherwise **"SD card detected"** will be printed.

```
if (!SD.begin())Serial.println("SD card not detected");
else Serial.println("Sd card detected");
```

The output would look as.



Click here to view the reference video.

Great, your SD card is working perfectly.

We have one more class challenge for you.
Can you solve it?

Let's try. I will guide you through it.

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 • **Reading data from the SD card.** • **Printing it on the serial terminal.** • **Using button to navigate through different books**	
Teacher Action	Student Action
Open the wokwi simulator link and, • Create a new Arduino project. • Delete all the previously existing files **(sketch.ino, diagram.json)** from your project. • Open the Student Activity 2 and download all the files from it. • Upload all the files that you have just downloaded in the previous step into your project. • Run the simulation once and you should see the message **"SD card detected"** printed on the **Serial terminal**. Once the card is detected, let's open the **harry.txt** file in **read** mode, by writing the following command in the **setup()** function, **file = open("harry.txt" , FILE_READ);** **FILE_READ** makes sure the file is opened in read mode. The above line of code will return a **file object "file"**, which will point towards the **harry.txt** file.	

```
file = SD.open("harry.txt" , FILE_READ);
```

Now let's check how many characters are there in our file. For that, let's use the **.size()** method.

```
Serial.print("Total characters : ");
Serial.println(file.size());
```

The output for the following code will show the total number of characters in our **harry.txt** file, which is **29324 characters**.

```
Total characters : 29324
```

Now let's **read** the characters from our file and print them on the **serial monitor** using the following piece of code as,

```
if (file) {
  Serial.println("HARRY POTTER : ");
  delay(2000);
  while (file.available()) {
    char data = file.read();
    Serial.print(data);
  }
  Serial.println();
  file.close();
}
Serial.println("File read successfully");
```

The above piece of code will check,

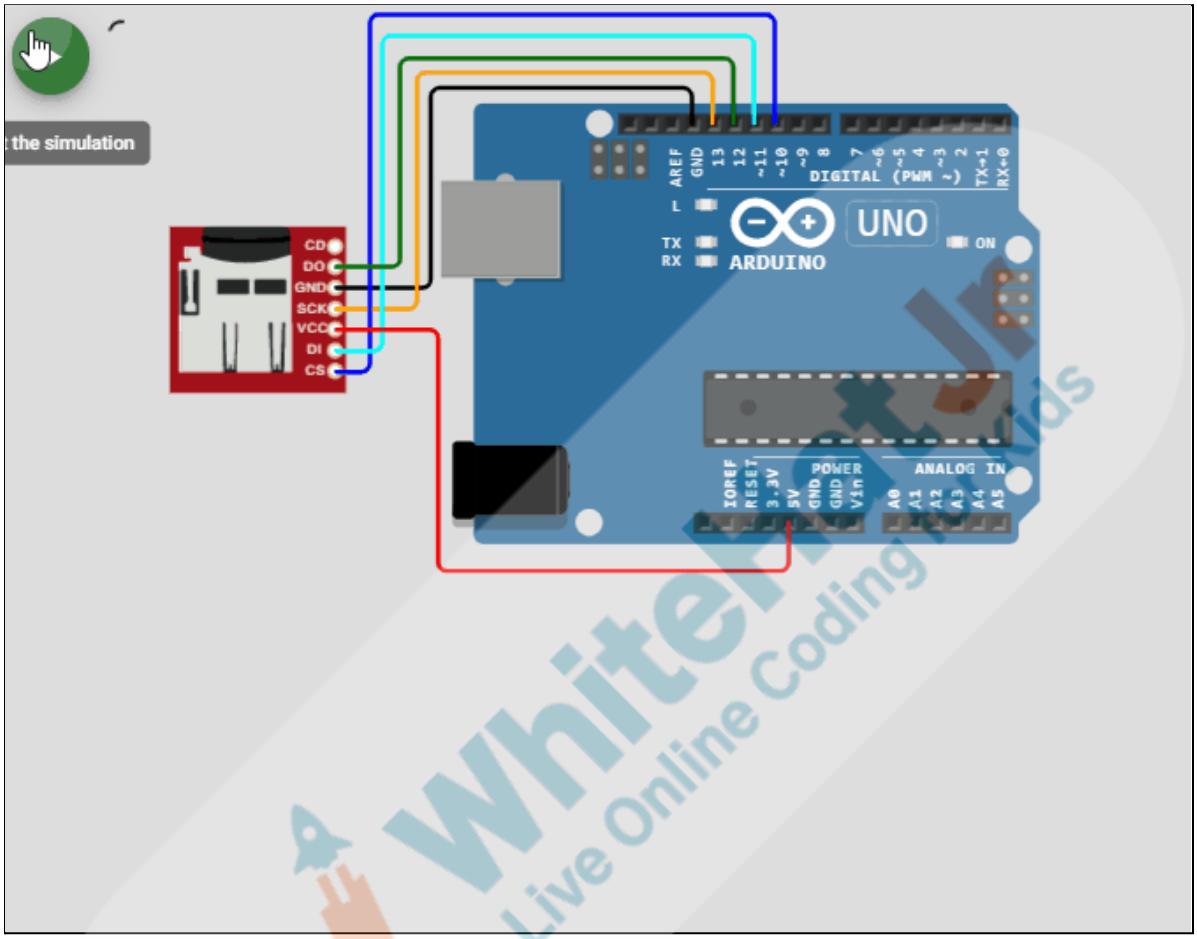
- If there is a valid file object or not.
- If the file object is valid, print **"HARRY POTTER : "** on the **serial terminal**.

- Wait for **2 seconds**.
- Check **continuously**, if there are characters **available** to **read** in the file or not.
- Read **one byte** or **one character** at a time.
- **Print** that **character** over the serial monitor.
- Once all the **characters** are **extracted** from the **file** and printed, the **loop** will **break**, and we will print an empty new line.
- **Close** the file.
- Print **"File is read successfully"**.

Your **setup()** method should look like this-

```
void setup() {

  Serial.begin(9600);
  SD.begin();

  file = SD.open("harry.txt", FILE_READ);
  Serial.print("Total characters :");
  Serial.println(file.size());

  if (file) {
    Serial.println("HARRY POTTER : ");
    delay(2000);
    while (file.available()) {
      char data = file.read();
      Serial.print(data);
    }
    Serial.println();
    file.close();
  }
  Serial.println("File read successfully");

}
```

The output should look like the following-



Click here to view the reference video.

Now we can see that our **harry.txt** file has **29324 characters.**

If we take the **think.txt** file into consideration, it will have approximately a similar number of characters. So in total close to **60000 characters,** and the arduino has a memory of **34 kb (~34816 characters),** which means that it's not possible to **hard code** or **store** all this data in the **program memory** of the Arduino. An external storage is always helpful.

Also, keep the **loop()** method empty as we don't need to repeat the above code again and again.

Great work. This all looks good.

This only reads one book, what if we want to read the other book?

ESR: We can write a method which we can call when we need to read a book.

Great! Let's do that. Also, let's comment out the section of code which prints the file data. After we have written the new function, we can remove this section of code.

```
void setup() {
  Serial.begin(9600);
  SD.begin();

  next.setDebounceTime(50);
  prev.setDebounceTime(50);
  select_book.setDebounceTime(50);
  open_book.setDebounceTime(50);

  /*file = SD.open("harry.txt", FILE_READ);
  Serial.print("Total characters :");
  Serial.println(file.size());

  if (file) {
    Serial.println("HARRY POTTER : ");
    delay(2000);
    while (file.available()) {
      char data = file.read();
      Serial.print(data);
    }
    Serial.println();
    file.close();
  }
  Serial.println("File read successfully");*/
}
```

But let's add another functionality as well. As we have two

books, what if we want to choose which book we want to read?

Exactly!

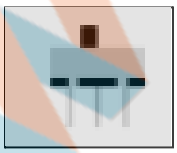
Let's integrate the buttons in the circuit first.

1. Add two push buttons and a slide switch in the circuit.
2. Connect them as per the following table.

Push Button (btn.1)	Arduino Uno board
1.l	2
2.r	GND

Push Button (btn.2)	Arduino Uno board
1.l	3
2.r	GND

3. We will also integrate a **slide switch** component which will help us to open or close a book.This component almost works like a light bulb switch.



It has three pins -

Left terminal, ground and **right terminal.**

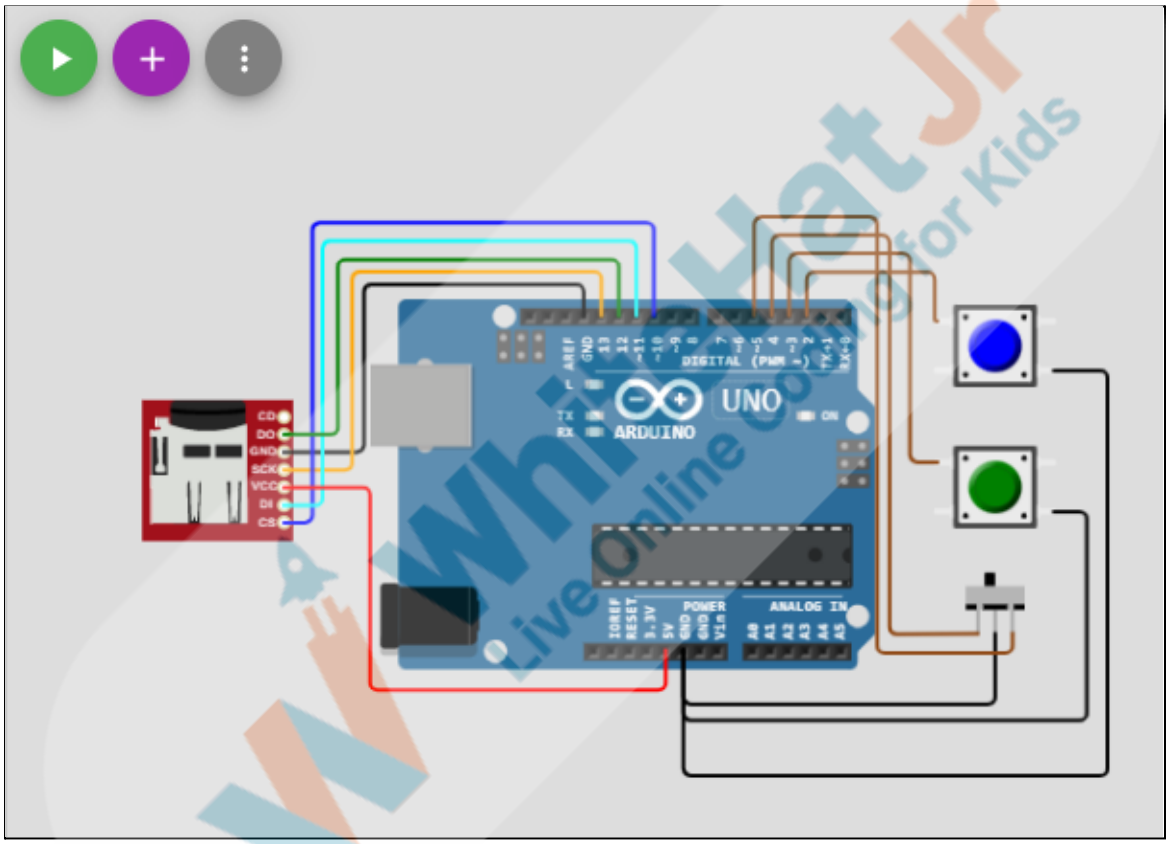
4. Connect the **slide switch** component

Slide Switch	Arduino Uno board

ESR: We could add buttons to choose the book.

sw1 :1	4
sw2 :2	GND
sw3 :3	5

Reference circuit:



Code:

1. Let's initiate a new variable named **book_num** to 0.

```
int book_num = 0;
int book_state = 0;
```

2. Include the **<ezButton.h>** header file

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

3. Initiate all 4 buttons.

```
ezButton next(2);
ezButton prev(3);
ezButton select_book(4);
ezButton open_book(5);
```

4. In the **setup()** method, call **setDebounceTime()** method for all the buttons.

```
next.setDebounceTime(50);
prev.setDebounceTime(50);
select_book.setDebounceTime(50);
open_book.setDebounceTime(50);
```

5. In the **loop()** method,

```
void loop(){

  next.loop();
  prev.loop();
  select_book.loop();
  open_book.loop();

}
```

6. If the **next** button is pressed, the **book_num** variable will increase and if the **prev** button is pressed the **book_num** variable will decrease.

This should be written in the **loop()** method.

```
if (next.isPressed()){
  book_num++;
}
else if (prev.isPressed()){
 book_num--;
}
```

7. As we have two books only, use the **constrain()** method so that the **book_num** cannot go beyond 0 and 1.

```
book_num = constrain(book_num, 0, 1);
```

8. Now, let's write a **check_book()** method, which will assign the **book_name** according to the selected **book_num**.

```
void check_book(){

    if (book_num)
      Serial.print("Think and Grow Rich");
    else
      Serial.print("Harry Potter");

}
```

9. Call **check_book()** method in the **loop()** method.
10. Now, when we run the code we can see that the

name of the selected book is being printed multiple times. This is happening because it is running for every **loop(),** but we do not want it. We want it to run only when the book is changed.

Let's resolve this problem.

Define a variable named **prev_book_num** to -1.

```
int prev_book_num = -1;
```

11. In the **check_book()** method, add an if condition which compares the **prev_book_num** with the **book_num** variable, if they are not equal to each other then only print the book name.

    Also, set **prev_book_num** equals to **book_num** whenever this condition is met.

```
void check_book(){
  if (prev_book_num  !=  book_num){
    prev_book_num = book_num;
    lcd.clear();
    if (book_num)
      Serial.println("Think and Grow Rich");
    else
      Serial.println("Harry Potter");
  }
}
```

12. Now, we will use the **slide switch** to open the selected book. According to our definition, the **select_book** button on the **slide switch** will make the **button_state** to 0.

    The **open_book** button on the **slide switch** will

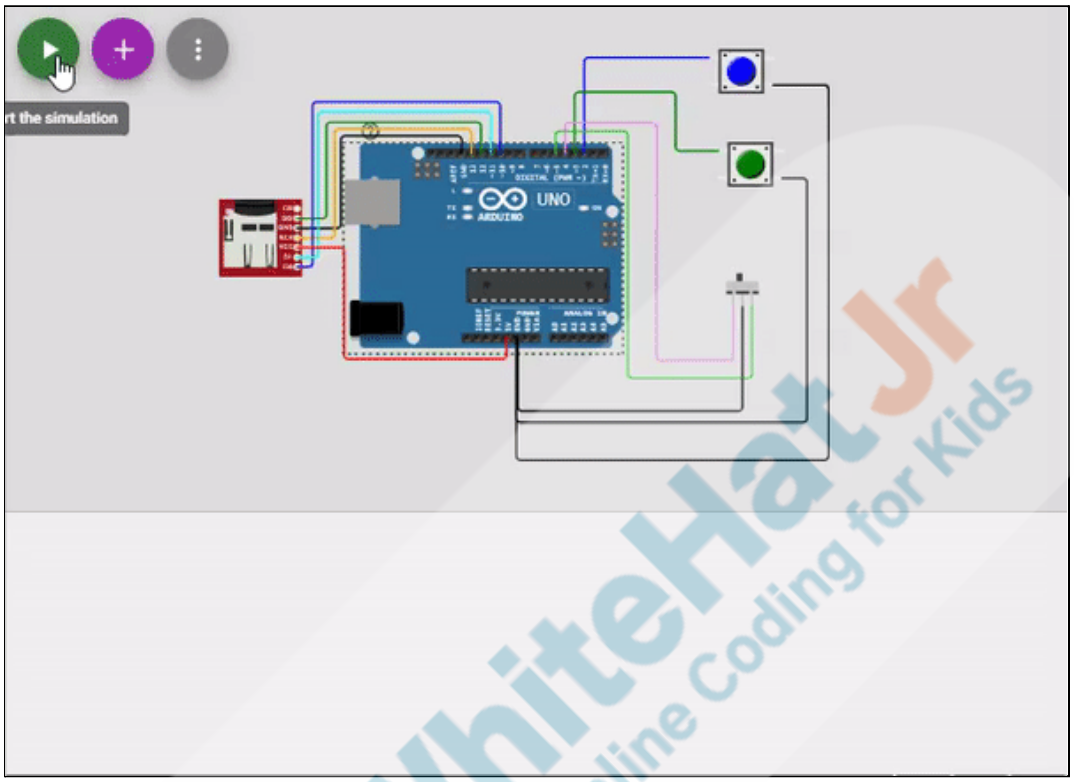
change the **button_state** to 1.

```
if (next.isPressed()) {
  book_num++;
}
else if (prev.isPressed()) {
  book_num--;
}
else if (select_book.isPressed()) {
  book_state = 0;
  prev_book_num = -1;
}
else if (open_book.isPressed())
  book_state = 1;
```

13. Now, let's define the **read_book()** method and print the book only when the **book_state** is 1.

```
void read_book() {
  if (book_state) { //  if book is opened
    if (book_num)
      file = SD.open("think.txt", FILE_READ);
    else
      file = SD.open("harry.txt", FILE_READ);

    String display_text = "";
    if (file) {
      while (file.available()) {
        char data = file.read();
        Serial.print(data);
      }
    }
  }
  file.close();
}
```

Reference output:



Click here to view the reference output video.

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

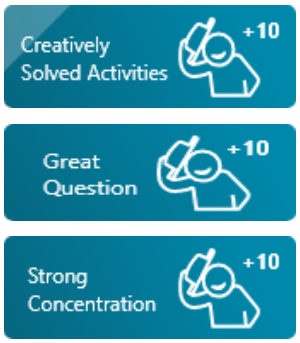
Activity Details

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 create a full-fledged reading device.	*Make sure you have given at least 2 hats-off during the class for:* 

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class

ADDITIONAL ACTIVITIES

(Optional)

Additional Activities	

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Simulator	wokwi
Teacher Activity 2	Github link with text files	https://github.com/procodingclass/PRO-C273-Teacher-Activity.git
Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/PRO-C273-Student-Activity.git
Teacher Reference 2	Student Activity Reference Code	https://github.com/procodingclass/PRO-273-Reference-Code.git
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/f1a8b759-c071-4c14-b6d5-2a83078ed716.pdf
Teacher Reference 4	Project Solution	https://github.com/procodingclass/PRO-C273-Project-Solution
Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/f69ff953-f396-4887-ad1d-7843a0249e2a.pdf
Student Activity 1	Simulator	wokwi
Student Activity 2	Boilerplate link	https://github.com/procodingclass/PRO-C273-Student-Activity.git
Student Activity 3	SD - Arduino reference	https://www.arduino.cc/reference/en/libraries/sd/